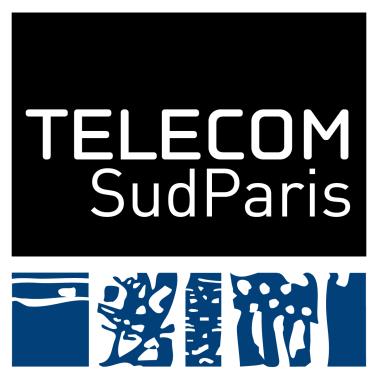


CASSIOPÉE - Rapport de synthèse

Romain AMÉDÉE
Yassine BOUSSAFIR

Conduite Autonome de Drone (CAD)

Référents :
Ghalid ABIB
Hossam AFIFI



Contents

1	Introduction	3
1.1	Contexte	3
1.2	But du projet	4
2	État de l'art	5
2.1	Bebop Power 2	5
2.2	YOLO	6
3	Notre travail	7
3.1	Détection de personnes	7
3.2	Flux vidéo du drone	7
3.3	Réaction du drone	8
4	Résultats et performances	10
4.1	Résultats	10
4.2	Limites	12
4.3	Pistes d'amélioration	12
5	Conclusion	12

1 Introduction

1.1 Contexte

Le développement des technologies de conduite autonome, en particulier pour les drones, revêt une importance capitale dans notre société moderne. Les drones autonomes offrent une multitude d'applications et de possibilités dans divers domaines tels que la logistique, la surveillance, la cartographie, la livraison de colis ainsi que des usages militaires. Grâce à leur capacité à opérer de manière autonome, ces drones peuvent accomplir des tâches plus efficacement, réduisant ainsi les coûts et le besoin en ressources humaines. De plus, ces derniers peuvent accéder à des zones difficiles d'accès pour les humains, ce qui en fait des outils précieux dans des situations d'urgence, telles que les opérations de sauvetage ou les inspections d'infrastructures critiques (les ponts sont un exemple d'infrastructure difficile d'accès). En outre, ces technologies peuvent contribuer à la réduction des émissions de carbone, les drones pouvant être alimentés par des sources d'énergie plus propres. L'importance du développement continu de technologies de conduite autonome pour les drones réside donc dans leur potentiel à transformer différents secteurs de notre société, en améliorant l'efficacité, la sécurité et la durabilité de nos opérations.

En outre, les technologies de conduite autonome pour les véhicules jouent également un rôle essentiel dans la quête pour améliorer la sécurité routière et réduire les accidents de la circulation. Les systèmes avancés d'assistance à la conduite (ADAS) intégrant des fonctionnalités telles que le freinage d'urgence, le maintien de voie et la détection des angles morts contribuent déjà à prévenir de nombreux accidents. En développant des véhicules entièrement autonomes, capables de prendre des décisions de conduite en temps réel, nous pourrions réduire considérablement les erreurs humaines responsables de la majorité des accidents de la route. Enfin, les véhicules autonomes promettent d'optimiser la fluidité du trafic, de réduire la congestion et d'améliorer l'efficacité énergétique, contribuant ainsi à une mobilité plus durable et respectueuse de l'environnement.



Figure 1: Point de vue de la voiture autonome

1.2 But du projet

Notre projet s'inscrit ainsi dans ce contexte et a pour but de permettre la conduite autonome d'un drone. Le drone devra à terme être capable d'effectuer des mouvements simples pour se positionner par rapport à une cible (une personne par exemple), et être capable de suivre cette même cible en mouvement, le tout en toute autonomie.

Pour cela, nous disposons d'un drone Bebop Power 2 de la marque Parrot et nous proposons d'utiliser YOLOv4, un outil d'analyse d'image, afin de gérer la détection de la cible (une personne dans notre cas) sur le flux vidéo du drone.

Parrot®



Figure 2: Bebop Power 2 (Parrot)



Figure 3: YOLO

2 État de l'art

2.1 Bebop Power 2

Pour pouvoir mettre en place ce projet, nous avons fait le choix de joindre plusieurs technologies existantes, tout d'abord le squelette et le cœur du projet : le drone.

Le drone que nous utilisons est un modèle de drone de la marque Parrot, le Bebop Power 2. Le drone possède une batterie permettant une autonomie de vol d'environ 30 minutes, ses différents capteurs en font un exemple de stabilité dans le domaine et il dispose de plusieurs fonctionnalités de vol intelligentes tels qu'un suivi GPS et des vols automatique préprogrammés.

Si notre choix s'est porté sur ce drone, c'est pour la portée de contrôle qu'il autorise, environ 2 kilomètres en champ ouvert, via une connexion WI-FI ainsi que l'existence de librairies Python (pyparrot) permettant son contrôle à partir d'un ordinateur qui nous sert de station plutôt que d'un smartphone (grâce à l'application FreeFlight Pro de Parrot). Sa caméra embarqué haute résolution (1080p) avec stabilisation numérique en font également un choix pour notre projet.



Figure 4: Bebop Power 2

2.2 YOLO

YOLO (You Only Look Once) est une méthode de Deep Learning décrite dans l'article "You Only Look Once: Unified, Real-Time Object Detection" par Joseph Redmon, Santosh Divvala, Ross Girshick et Ali Farhadi. Elle est utilisée dans le cadre de la détection d'objets en temps réel au sein d'une image. Contrairement aux approches traditionnelles basées sur les régions d'intérêts, YOLO effectue la détection directement sur l'image globale.

Pour ce faire, YOLO divise une image en grille régulière et prédit les boîtes englobantes et les probabilités de classe pour chaque cellule de cette grille. Cette méthode que nous utilisons dans notre projet est efficace et rapide car elle effectue la détection d'objets et la classification simultanément, en utilisant un seul réseau de neurones convolutifs. Cela permet d'éviter les répétitions chronophages liées à l'analyse d'une image ayant plusieurs régions d'intérêt.

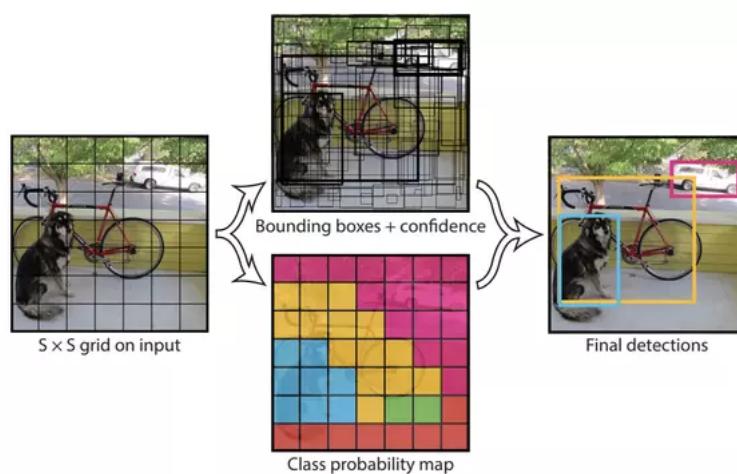


Figure 5: Principe de YOLO

YOLO est une technologie ayant connu de nombreuses améliorations, de YOLO à YOLOv4 que nous utilisons dans notre projet. Les différentes versions de YOLO ont apporté des améliorations en termes de précision et de performances, en utilisant des architectures de réseau plus profondes et des astuces d'optimisation. YOLO a été largement utilisée dans des domaines tels que la surveillance vidéo, les véhicules autonomes, la réalité augmentée, et bien d'autres, en raison de sa capacité à détecter et localiser efficacement des objets dans une scène en temps réel, ce qui en fait un choix idéal dans le cadre de notre projet de conduite autonome de drone.

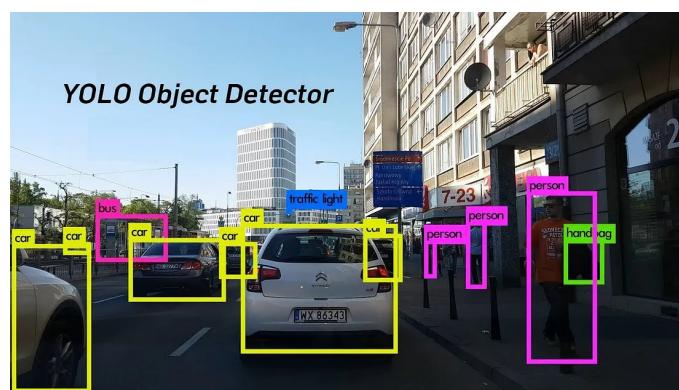


Figure 6: Exemple en sortie de YOLO

3 Notre travail

3.1 Détection de personnes

Pour notre solution, nous avons procédé par étape. Pour que notre drone puisse suivre une personne, il faut tout d'abord que notre programme détecte la personne en question. Comme expliqué plus tôt, nous utilisons la solution YOLO pour faire cela. À l'origine, YOLO est conçue pour détecter une multitude de classes (une vingtaine en tout parmi lesquelles : voitures, vélos, chiens, chats, ...). Il a donc fallu dans un premier temps adapter ce programme pour le restreindre à la détection d'individus. En effet, la détection de plusieurs objets, en plus d'entraver le bon fonctionnement du drone, ralentirait fortement le programme. Il est aussi nécessaire que le programme ne détecte qu'une personne à la fois (le drone ne sait pas réagir à plusieurs personnes à l'écran). Pour gérer cela, nous laissons l'algorithme détecter plusieurs personnes mais le drone ne suit que la première personne qu'il a détecté et ignore les autres. Si une personne sort du champ de vision du drone, ce dernier change de cible et se met à suivre la seconde personne détectée, et ainsi de suite.

Nous n'exécutons pas YOLO sur le drone car ce dernier ne dispose pas d'une puissance de calcul suffisante. On passe alors par un serveur en WiFi entre notre machine et le drone.

3.2 Flux vidéo du drone

L'analyse se base sur le champ de vision du drone. Il est ainsi nécessaire de récupérer les images que la caméra du drone capte. Au démarrage, le drone établit son propre environnement auquel nous pouvons accéder grâce au Wi-fi. Nous établissons ensuite la connection au drone avec les méthodes de connection de la librairie Python `pyparrot`.



Figure 7: Caméra du drone Bebop Power 2

Une fois connecté au serveur, nos machines récupère en temps réel le flux vidéo capté par la caméra du drone et transmise via le protocole RTP. Afin d'optimiser les performances (nos machines ne peuvent pas supporter le traitement d'un très grand flux vidéo) nous avons expérimentalement fixé la fréquence de réception à 5 images par seconde.

Le protocole RTP (Real-time Transport Protocol) est un protocole de transport utilisé pour la transmission en temps réel de données audio et vidéo sur les réseaux IP. Il a été spécialement conçu pour répondre aux besoins de la communication en temps réel, offrant des fonctionnalités telles que la latence minimale, la synchronisation et la correction d'erreurs. RTP découpe les données en paquets et les envoie de manière séquentielle. Chaque paquet est doté d'un numéro de séquence,

permettant au récepteur de les réorganiser correctement et de détecter les pertes éventuelles. De plus, chaque paquet est marqué d'une estampille temporelle qui permet de synchroniser les flux de données entre l'émetteur et le récepteur, garantissant ainsi une présentation cohérente des données.

Il a ainsi fallu adapter notre programme à la réception spécifique de ce protocole, notamment en utilisant la suite ffmpeg. Il s'agit d'une suite de logiciels et une bibliothèque logicielle open source utilisée pour manipuler, convertir et lire des fichiers multimédias. C'est un outil polyvalent et puissant largement utilisé dans l'industrie du multimédia. Il permet de convertir des fichiers d'un format à un autre, d'extraire des flux audio ou vidéo à partir de fichiers multimédias et de compresser ou décompresser des fichiers pour l'encodage et le décodage.

3.3 Réaction du drone

Dès qu'une image est reçue, elle est alors injectée dans YOLO qui indique la position des personnes présentes et stocke les coordonnées des rectangles encadrant celles-ci dans un tableau trié par ordre de détection. On choisit arbitrairement que le drone suivra la première personne qu'il détecte. De plus, afin de suivre et de comprendre la vision du drone, on affiche le flux vidéo en parallèle au traitement des images par YOLO. Ainsi la personne détectée est encadrée en rouge.

À partir de ces informations, le programme calcule le barycentre du rectangle qui encadre la personne, puis vérifie sa position sur l'image. Le principe du suivi automatique du drone est que ce dernier cherche à centrer le barycentre de la cible dans son champ de vision. Ce centre est défini à une zone de tolérance près correspondant à un carré de 40 pixels par 40 pixels autour du centre réel de l'image.



Figure 8: Vue du drone. Le grand rectangle rouge correspond à la détection de l'individu. Le petit carré est la zone de tolérance au centre du champ de vision du drone. Le point bleu correspond au barycentre de l'individu. On cherche à conserver le point bleu dans le petit carré.

Dès lors, le drone doit prendre une décision selon différents cas de figures :

- La personne est excentrée horizontalement par rapport à la zone de tolérance: alors le drone effectue une rotation dans la direction qui va rapprocher le barycentre du centre de l'image, jusqu'à ce que celle-ci s'y trouve.
- La personne est excentrée verticalement par rapport à la zone de tolérance: si le barycentre se trouve en dessous de cette zone, c'est que la personne se rapproche, donc le drone recule. Si le barycentre se trouve au-dessus de cette zone, c'est que la personne recule, donc le drone avance.

Une condition supplémentaire doit cependant être prise en compte lors de la prise de décision du drone dans le cas où ce dernier est excentré verticalement. En effet, lorsque le drone se rapproche, la géométrie de l'espace change par rapport au plan 2D de l'image et il arrive que le rectangle qui encadre la personne change de forme : les jambes de l'individu sortent du champs de vision du drone, ce qui rétrécie grandement son rectangle de détection et ainsi rehausse son barycentre. Cela mène le drone à poursuivre sa trajectoire indéfiniment vers l'avant.



Figure 9: Le drone avance indéfiniment sans la condition supplémentaire. Le rectangle de détection se rétrécie par le bas à mesure que le drone avance ce qui réhausse continuellement son barycentre.

Notre solution pour pallier cela a été de considérer le rapport entre la longueur et la largeur du rectangle, en effectuant une approximation sur les proportions perçues par le drone d'un être humain. Ainsi, pour avancer, le barycentre doit se situer au-dessus de la zone de tolérance et le rapport longueur/largeur doit être supérieur à 2. Cette valeur a été obtenue expérimentalement, de sorte que le drone se stabilise à environ 1,5 mètres de la cible.

Algorithm 1 Algorithme de déplacement du drone

```

1: while reçoit une image do
2:    $bx \leftarrow barycentre_x$ 
3:    $by \leftarrow barycentre_y$ 
4:    $rapport \leftarrow largeur/longueur$ 
5:   if  $bx > 448$  then
6:     rotation de 6 degrés sens anti-horaire
7:   else if  $bx < 408$  then
8:     rotation de 6 degrés sens horaire
9:   else if  $rapport \geq 2$  ET  $by < 220$  then
10:    translation de 15 cm en avant
11:   else if  $by > 260$  then
12:    translation de 15 cm en arrière
13:   end if
14: end while

```

Figure 10: Algorithme de déplacement du drone

4 Résultats et performances

4.1 Résultats

En reliant les trois parties décrites précédemment, on obtient un résultat plutôt satisfaisant. Le drone s'élève à environ 2 mètres du sol et suit lentement la cible d'avant en arrière et pivote de droite à gauche pour s'aligner avec la cible. En cas de cible immobile, le drone se stabilise dans l'attente d'un prochain mouvement.



Figure 11: Translation du drone vers l'avant (vue du drone)

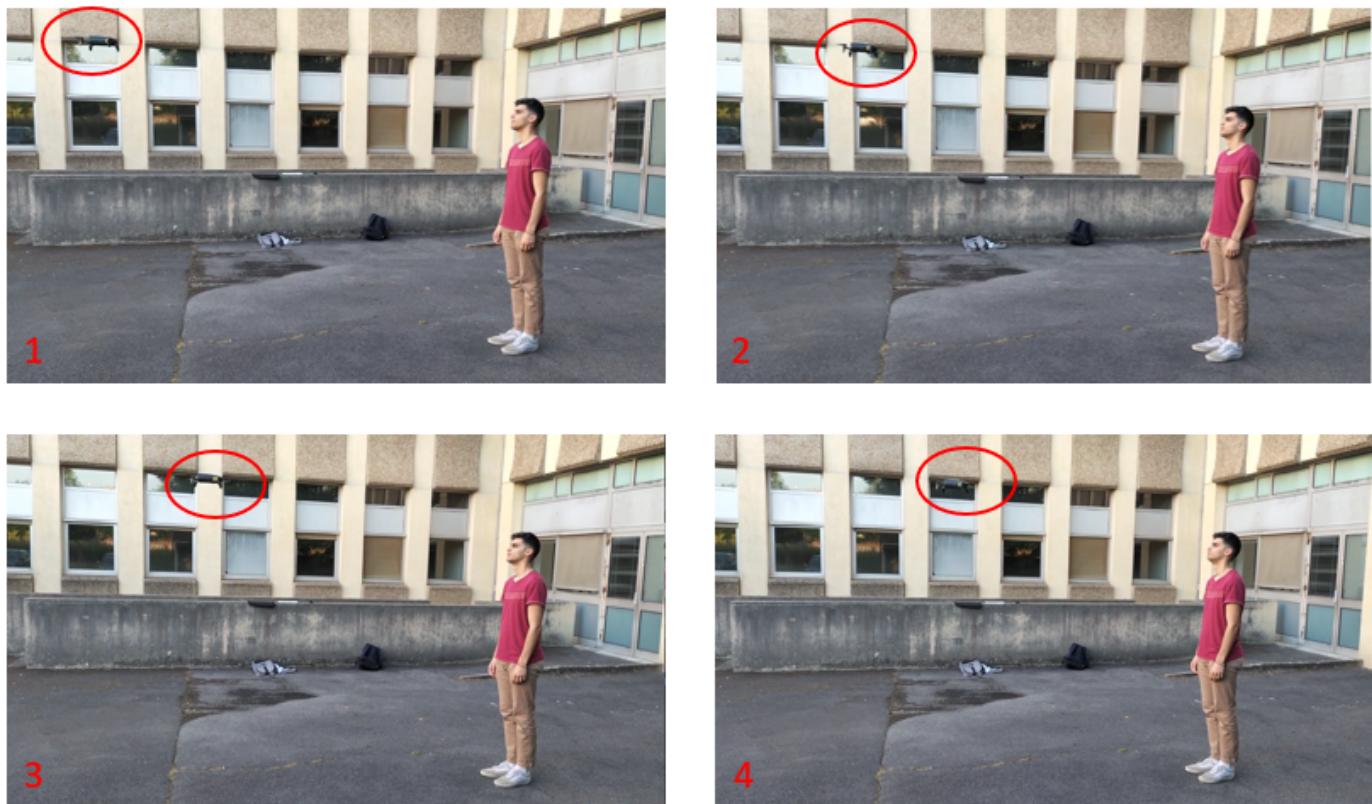


Figure 12: Translation du drone vers l'avant (vue externe)



Figure 13: Rotation du drone

4.2 Limites

Une première limite dans la réalisation de ce projet est le contrôle du drone. Le Bebop Power 2 est un drone propriétaire destiné au grand public pour une utilisation récréative grâce à l'application FreeFlight Pro. Le but premier du drone n'étant pas d'être contrôlé par un ordinateur via python, la marge de manœuvre n'est pas très large. Par exemple, nous n'avons pas la possibilité d'ajuster la vitesse du drone. Dans le cadre de notre prototype cela n'est pas très gênant mais cela représente un blocage pour des tests plus poussés de notre travail.

Une autre limite de notre solution est la puissance de nos machines. On supporte ici le traitement de cinq images par secondes car une fréquence plus élevée n'est pas possible. On peut imaginer un dispositif plus précis avec des machines plus puissantes que les nôtres.

Le drone étant récréatif, il n'est pas nécessairement très stable. Au cours d'une session son altitude peut varier de plusieurs dizaines de centimètres avec le vent ou les aléas du vol. Cela complique la mise en place de l'algorithme de déplacement, la hauteur du drone n'étant pas nécessairement constante.

4.3 Pistes d'amélioration

Une des améliorations principales envisageable dans ce projet serait de rendre le dispositif complètement intégré au drone. En lui ajoutant une petite machine (Raspberry Pi ou Jetson nano par exemple), on pourrait embarquer l'algorithme de déplacement directement sur le drone et ainsi le rendre réellement autonome.

Une autre amélioration possible serait de proposer un algorithme plus précis pour le déplacement et l'asservissement du drone, notamment en proposant des calculs de distances (difficile étant donné l'instabilité du drone) permettant d'adapter les déplacements en tenant compte de la distance focale de la caméra.

5 Conclusion

En conclusion, nous avons rempli la majorité des objectifs que nous nous étions fixés. Le drone peut suivre une cible et se stabiliser en cas d'immobilité ou de cible manquante. Cela reste évidemment un prototype et de nombreuses améliorations peuvent encore être faites afin de préciser les déplacements du drone. Ces améliorations feront peut-être l'objet d'un futur projet Cassiopée.