Détection automatique d'objets à partir d'un drone

Thesis · May 2022

DOI: 10.13140/RG.2.2.16741.68328

1 author:

Loqmane Hassane
University of Upper Alsace
3 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Université de Haute Alsace

Faculté des Sciences et Technique

Projet de Master 1 AII

Réalisé par :

Loqmane HASSANE

Imad KHAMMAR

Encadré par :

Jean-Philippe URBAN

Thème :

Détection automatique d'objets à partir d'un drone

Année universitaire 2021 - 2022

Remerciements

Nous voudrions exprimer nos sincères remerciements à notre tuteur, Monsieur Jean-Philippe Urban, pour ses précieux conseils et son aide durant toute la période de travail.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre travail, en acceptant de l'examiner et de l'enrichir par leurs propositions.

Nous remercions aussi tous les enseignants qui ont contribué à notre formation, et à qui nous exprimons notre profonde gratitude.

Enfin, nous tenons à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Table des matières

Chapitre III : Stratégies mises en œuvre pour la detection et le suivi d'objets par drone

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Ces dernières années, il y a eu un intérêt croissant pour les drones autonomes et leurs applications telles que la reconnaissance, la surveillance et l'inspection des infrastructures. La détection visuelle d'objets est un élément important dans de telles applications de drones et est essentielle pour développer des systèmes entièrement autonomes.

La technologie de détection d'objet est une méthode qui traite de la détection d'instances d'objet sémantique d'une certaine classe à l'intérieur d'une image, indiquant la position de l'objet et la classe d'objet via une description de boîte englobante.

La détection d'objets en temps réel est cruciale pour de nombreuses applications de drone. Au cours des dernières années, les réseaux de neurones convolutifs (CNN) sont devenus une puissante classe de modèles pour la reconnaissance du contenu des images et sont largement considérés dans la communauté de la vision par ordinateur comme l'approche standard pour la plupart des problèmes.

Depuis 2015, le réseau de neurones profond est devenu un cadre courant utilisé pour la détection et le suivi d'objets par drone, permettant d'obtenir de très bonnes performances.

L'objet de notre projet consiste, dans un premier temps, à appliquer l'algorithme de détection d'objets YOLO utilisant des réseaux de neurones convolutifs (CNN) pour la détection par drone, afin que ce dernier parvienne à détecter les différents objets répartis dans la scène à partir de l'image obtenue par sa caméra.

Dans un second temps on exploite les informations obtenues par YOLO, telles que la position d'objet dans l'image, afin d'effectuer le suivi par drone d'un ou des objets souhaités.

Dans un dernier temps, on réalise un recalage par rapport à des cibles collées aux murs de façon que le drone puisse avoir l'information concernant son emplacement et son inclinaison due à ses éventuelles dérives, pour finalement le combiner avec le déplacement du drone recherchant les objets.

De ce fait, le projet est constitué de trois chapitres :

- Le premier chapitre comporte une brève revue de la littérature sur les drones.
- Le second chapitre est consacré aux explications des différentes méthodes de détection d'objets.
- Le troisième chapitre est dédié aux stratégies mises en œuvre pour la détection et le suivi d'objets.
- Nous terminons notre travail par une conclusion générale ainsi que des perspectives futures.

Chapitre I

Généralités sur les drones

I.1. Introduction

Les véhicules aériens sans pilote (UAV), mieux connus sous le nom de drones, font partie des développements technologiques majeurs d'aujourd'hui. Les drones sont des aéronefs capables de voler sans pilote ni passagers à bord. Le contrôle des drones s'effectue à distance par ondes radio ou de manière autonome (avec un parcours prédéterminé) [1].

I.2. Classification des drones

Les drones peuvent être classé en fonction de la vitesse de vol, de la stabilisation, du type de vol (stationnaire, fixe ou flânant) et de l'environnement de vol. Plusieurs types de drones existent et diffèrent selon leurs caractéristiques (Figure I.1) [2].

- Les drones à rotor unique peuvent transporter de lourdes charges mais leur complexité mécanique entraîne des coûts élevés [2].
- Les drone multi rotor quant à eux sont des types courant car leur utilisation s'applique à la fois aux professionnels et aux non professionnels. Ces drones peuvent planer ou se déplacer le long de la cible donnée [2].
- Les drones à voilure fixe ont une vitesse de vol élevée et peuvent transporter de lourdes charges, néanmoins ils ont besoin d'une piste pour décoller et atterrir [2].
- Les drones hybrides sont des versions améliorées des drones à voilure fixe, mais ils sont encore en cours de développement [2].

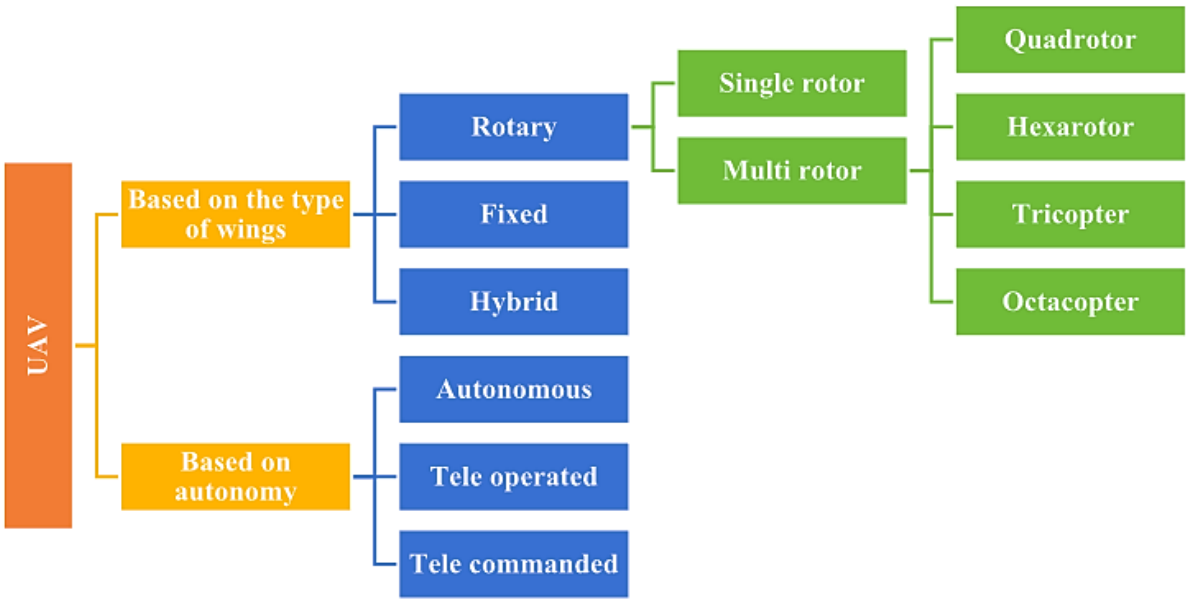

***Figure I.1** : Classification des drones* [2].

I.3. Composants de drone

Le drone est composé de deux grands systèmes :

- Système de mouvement.
- Système de contrôle.

I.3.1. Système de mouvement

I.3.1.1. Châssis

Le châssis est la base ou le squelette du drone où les différents composants reposent. Il est différent selon le nombre de bras, et peut ainsi avoir différentes formes et caractéristiques selon son utilisation. Les matériaux utilisés afin de constituer le châssis ont aussi leur importance. De manière générale, il s'agit de bois, plastique, métaux/alliages ou fibre de carbone [3].

Comme illustré sur la *figure I.2*, les drones peuvent être classifié selon le nombre de bras et des moteurs de la façon suivante :

- *Bicoptères* – deux moteurs
- *Tricoptères* – trois moteurs
- *Quadricoptères* – quatre moteurs
- *Hexacoptères* – six moteurs
- *Octocoptères* – huit moteurs

I.3.1.2. Hélices et moteurs

Les hélices et les moteurs constituent le système de propulsion principal d'un drone qui lui permettent de se propulser dans les airs. Les hélices changent un couple (dérivé du moteur) contre un travail utilisé pour soulever le drone dans les airs. En raison du système d'hélice, on peut avoir différentes structures [4] :

« **+** » : L'une est l'hélice principale (au moins quatre hélices).

« **X** » : La construction la plus courante, dans laquelle deux hélices sont en tête (avec un nombre pair d'hélices).

« **Y** » : Trois bras empilés dans le Y, où un ou deux bras peuvent être en tête,

« **V** » : Arrangement très rare dans lequel deux hélices mènent à des bras tendus,

« **H** » : Un arrangement très rare où la construction est basée sur la forme en H avec deux hélices en tête.

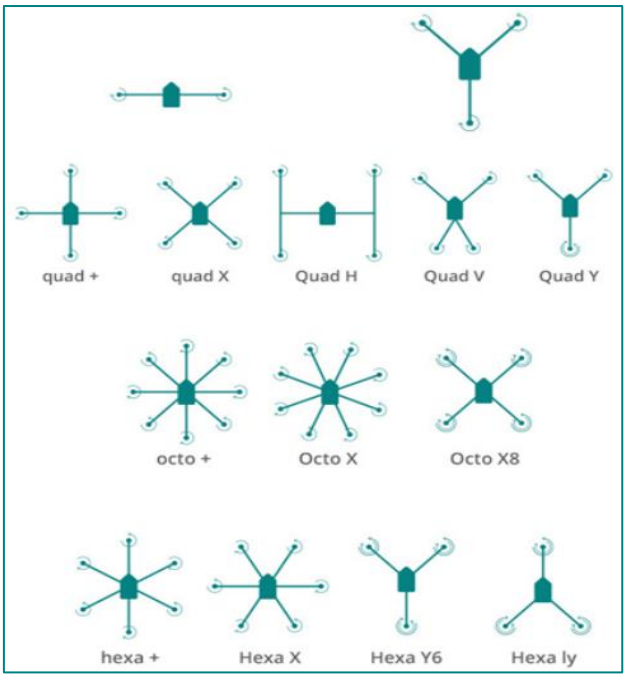

***Figure I.2** : Différentes structure de châssis* [4].

Les hélices sont en fibre de carbone, en plastique ou en aluminium, et sont attachées les unes aux autres par laminage, ce qui assure une performance optimale entre le poids de l'ensemble de la construction et de la durabilité mécanique. Il est important d'équilibrer chaque hélice avant utilisation pour minimiser les vibrations générées par le fonctionnement inégal du système [4].

I.3.2. Système électronique de contrôle et de communication

Le système de contrôle est responsable du vol du drone et de ses déplacements ; il est important pour la réaction aux forces émergentes et pour la stabilité du drone. La plupart des systèmes de contrôle sont équipés du même jeu de capteurs à la différence de la vitesse des calculs et des algorithmes utilisés. Le système de contrôle se compose de [4] :

- Un contrôleur de vol, responsable des capacités de contrôle de la machine,
- Un ESC (contrôle électronique de vitesse) qui est l'unité responsable du régime moteur,
- Un système d'alimentation qui fournit l'électricité à tous les autres systèmes du drone. Il est composé d'une batterie de plusieurs cellules,
- Un système de communication qui est un ensemble d'échange de messages entre les différents systèmes du drone,
- Un système anticollision (par exemple une caméra de proximité).

I.4. Le drone Tello

Tout au long de notre projet nous avons utilisé le drone Tello. Ce drone est un petit quadricoptère doté d'un système de positionnement visuel et d'une caméra embarquée. Grâce à son système de positionnement visuel et à son contrôleur de vol avancé, il peut voler sur place. Il est également adapté pour voler à l'intérieur. Tello capture des photos de 5 mégapixels et diffuse des vidéos en direct 720p. Son temps de vol maximal est d'environ 13 minutes, le temps de vol maximum a été testé dans des conditions sans vent, à une vitesse constante de 15 km/h, et sa distance de vol maximale est de 100 m [5].

***Figure I.3** : Tello drone EDU* [6].

Tello EDU est un drone programmable parfait pour un usage éducatif comme son nom "EDU" l'indique. Il est fabriqué par « Shenzhen Ryze Technology » et intègre la technologie de contrôle de vol DJI et les processeurs Intel [5].

La protection « Failsafe » permet à Tello d'atterrir en toute sécurité même si l'on perd la connexion ; ses protections d'hélice peuvent être utilisées pour améliorer la sécurité [5].

I.4.1 Composants du drone tello

- Les principaux composants du drone tello sont les suivants [5] :

1. Hélices
2. Moteurs
3. Indicateur d'état de l'avion
4. Caméra
5. Bouton d'alimentation
6. Antennes
7. Système de positionnement visuel
8. Batterie de vol
9. Port micro-USB
10. Protecteurs d'hélice

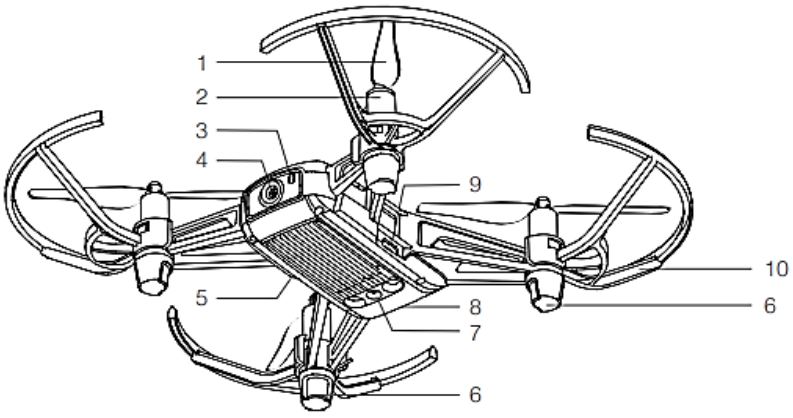

***Figure I.4** : différents composants du drone Tello EDU* [5].

I.4.2. Caractéristiques techniques du drone tello

Le tableau ci-dessous représente les différentes caractéristiques du drone tello [6] :

Poids	87 g
Dimension	98×92.5×41 mm
Hélice	3 pouces
Fonctions intégrées	Capteur télémétrique
	Baromètre
	LED
	Système de vision
	Wi-Fi 2.4 GHz 802.11n
	Diffusion en temps réel720p
Port	Port de charge de la batterie USB
Distance maximale de vol	100 m
Vitesse maximale	8 m/s
Temps de vol maximum	13 min
Hauteur de vol maximale	30 m

I.4.3. Système d'exploitation et langage de programmation

Pour programmer le drone tello, il est possible d'utiliser différents systèmes d'exploitation (Windows, Linux, MAC ...) ; pour ce projet nous avons opté pour une programmation sous Windows.

Il est possible de programmer ce drone dans différents langages tel que Java, C++, Python, Scratch, etc.). Nous avons choisi dans le cadre de notre projet une programmation sous python qui est l'un des langages de programmation les plus accessibles car il a une syntaxe simplifiée, et c'est surtout le langage du Deep Learning ce qui nous facilitera l'association entre le réseau de neurones et tello drone.

Chapitre II

Méthodes de détection d’objets

II. Introduction à la détection d'objets

Ces dernières années, la détection d'objets attire de plus en plus l'attention en raison des avancées technologiques récentes et de son large éventail d'applications tel que la vidéosurveillance, la conduite autonome, la surveillance des transports, la vision robotique et l'analyse de scènes de drones.

Parmi les nombreux facteurs qui contribuent à l'évolution rapide des techniques de détection d'objets, on note principalement le développement des réseaux de neurones convolutifs profonds ainsi que la puissance de calcul des GPU.

La plupart des détecteurs d'objets à la pointe de la technologie utilisent des réseaux d'apprentissage profonds comme épine dorsale pour extraire les caractéristiques des images (ou vidéos) permettant la classification et la localisation des objets.

II.1. Reconnaissance d'objets

La reconnaissance d'objets est la technique d'identification de l'objet présent dans les images et les vidéos. C'est l'une des applications les plus importantes de l'apprentissage automatique (machine learning) et de l'apprentissage profond (deep learning). L'objectif de ce domaine est d'apprendre aux machines à comprendre (reconnaître) le contenu d'une image comme le font les humains [7].

II.2. Classification d’images

La tâche la plus fondamentale pour les réseaux de neurones convolutifs est la classification d'images qui est une tâche d'apprentissage automatique, permettant de déterminer quels objets se trouvent dans une image ou une vidéo. Elle fait référence à la formation de modèles d'apprentissage automatique dans le but de découvrir quelles classes sont présentes tout en représentant la probabilité que l'image appartienne à l'une de plusieurs classes. La classification est utile pour décider si une image contient un objet/une anomalie ou non [8].

II.3. Localisation d'objet

L'algorithme de localisation d'objet localise la présence d'un objet dans l'image et le représente avec une boîte englobante. Il prend une image en entrée et affiche l'emplacement de la boîte englobante sous la forme de (position, hauteur et largeur) [9].

II.4. Détection d'objets

Les algorithmes de détection d'objets agissent comme une combinaison de classification d'images et de localisation d'objets. Il prend une image en entrée et produit une ou plusieurs boîtes englobantes avec l'étiquette de classe attachée à chaque boîte englobante. Ces algorithmes sont suffisamment capables de traiter la classification et la localisation multi-classes ainsi que de traiter les objets à occurrences multiples [9].

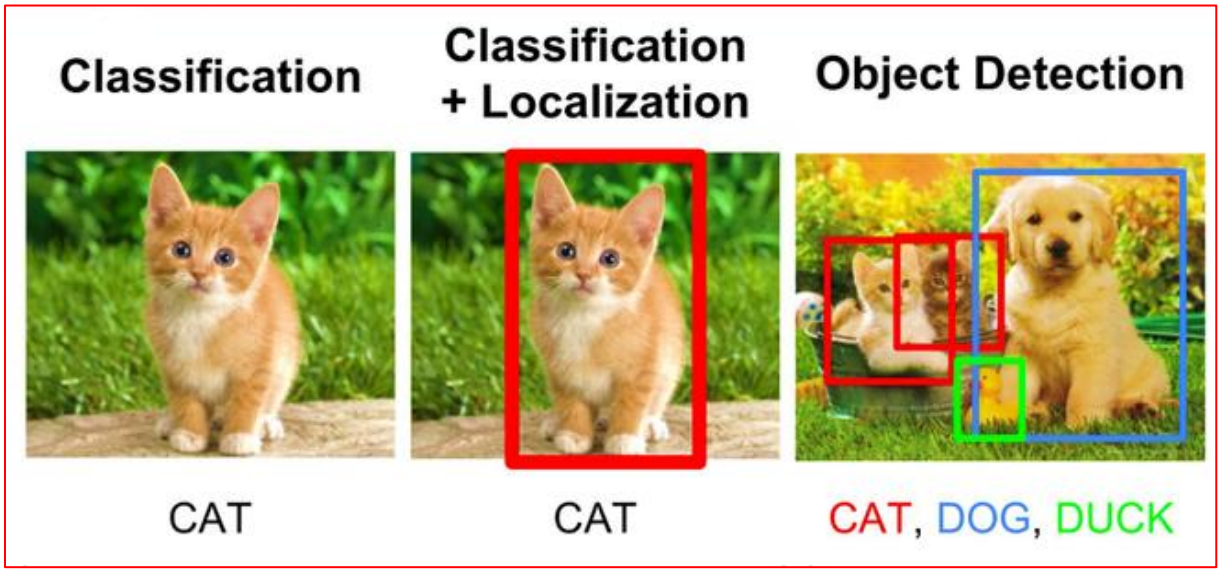

***Figure II.1** : Classification, localisation et détections d'objet* [9]

II.4.1. Défis de la détection d'objets

- Dans la détection d'objets, les boîtes englobantes sont toujours rectangulaires. Ainsi, cela n'aide pas à déterminer la forme des objets si l'objet contient une partie de courbure [9].
- La détection d'objet ne peut pas estimer avec précision certaines mesures telles que la surface et le périmètre d'un objet à partir d'une image [9].

II.5. Segmentation d'images

La segmentation d'image est une extension supplémentaire de la détection d'objet dans laquelle nous marquons la présence d'un objet à travers des masques pixel par pixel générés pour chaque objet dans l'image. Cette technique est plus granulaire que la génération de boîtes englobantes car cela peut nous aider à déterminer la forme de chaque objet présent dans l'image. Cette granularité nous aide dans divers domaines tels que le traitement d'images médicales, l'imagerie satellitaire, etc [9].

Il existe principalement deux types de segmentation :

- **Segmentation sémantique :** C'est une technique qui détecte, pour chaque pixel, la catégorie d'objets à laquelle il appartient. Toutes les catégories d'objets (étiquettes) doivent être connues du modèle [10].
- **Segmentation d'instance :** Elle est identique à la segmentation sémantique, mais en approfondissant un peu, elle identifie, pour chaque pixel, l'instance d'objet à laquelle elle appartient. La principale différence par rapport à la segmentation sémantique c'est qu'elle différencie deux objets avec les mêmes étiquettes [10].

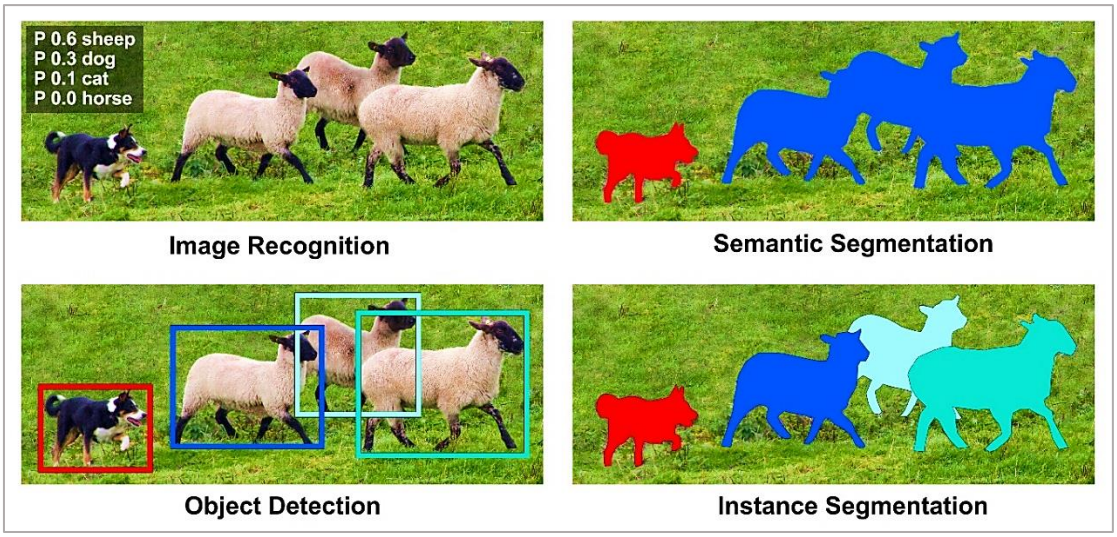

***Figure II.2** : Exemple de classification, de détection et de segmentation d'objets* [11].

Finalement, nous pouvons voir que la reconnaissance d'objets fait référence à une suite de tâches de vision par ordinateur, ce qui est illustré par la figure ci-dessous :

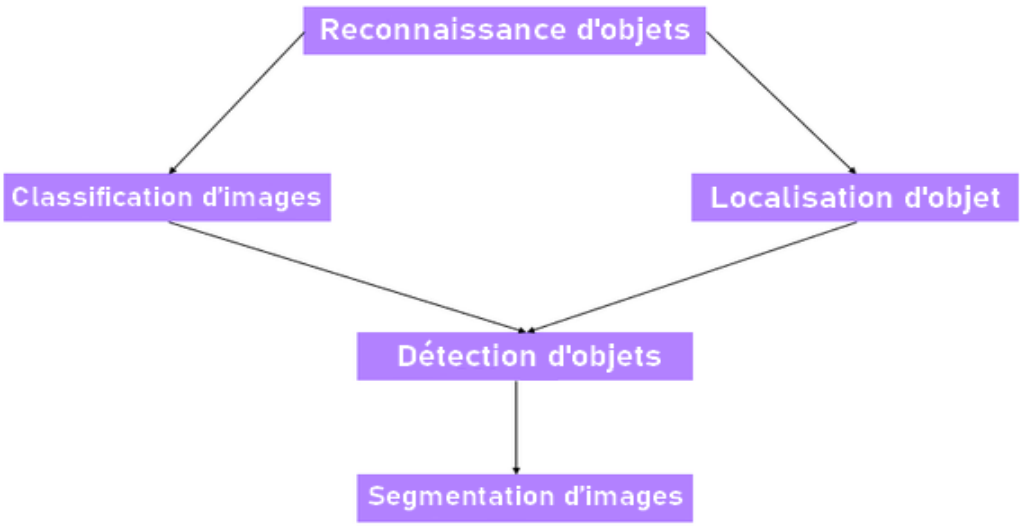

***Figure II.3** : Présentation des tâches de vision par ordinateur pour la reconnaissance d'objets*

II.6. Algorithmes de détection d'objet

Dans un cas de détection d'objet, il pourrait y avoir de nombreuses boîtes englobantes représentant différents objets d'intérêt dans l'image, et nous ne pourrons pas savoir combien il y en a à l'avance.

La principale raison pour laquelle nous ne pouvons pas résoudre ce problème en construisant un réseau convolutif standard suivi d'une couche entièrement connectée est que la longueur de la couche de sortie est variable, c'est parce que le nombre d'occurrences des objets d'intérêt est non fixé. Une approche pour résoudre ce problème serait de prendre différentes régions d'intérêt de l'image et d'utiliser un CNN (réseau neuronal convolutif) pour classer la présence de l'objet dans cette région. Le problème avec cette approche est que les objets d'intérêt peuvent avoir différents emplacements spatiaux dans l'image et différents rapports d'aspect. Par conséquent, nous devrons sélectionner un grand nombre de régions et cela pourrait faire beaucoup trop en termes de calcul. Par conséquent, des algorithmes comme R-CNN, YOLO, etc. ont été développés pour trouver ces occurrences rapidement [12].

II.6.1. R-CNN

Les réseaux de neurones convolutifs basés sur les régions (R-CNN) sont des techniques qui utilisent des recherches sélectives pour extraire 2000 régions de l'image appelées « propositions de régions ». De cette manière, plutôt que de tenter classer un grand nombre de régions, on peut simplement travailler avec 2000 régions qui sont générées à l'aide de l'algorithme de recherche sélective décrit ci-dessous [13].

Recherche sélective : [12]

1. Générer une sous-segmentation initiale pour obtenir de nombreuses régions candidates.
2. Utiliser un algorithme glouton pour combiner de manière récursive des régions similaires en régions plus grandes.
3. Utiliser les régions générées pour produire les propositions finales de régions candidates.

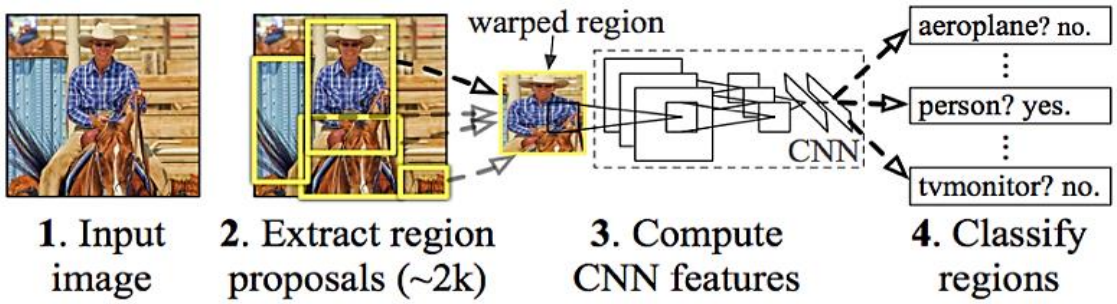

***Figure II.4** : Réseaux de neurones convolutifs basés sur les régions (R-CNN)* [13].

Les 2000 propositions de régions candidates sont déformées en carré et introduites dans un réseau neuronal convolutif qui produit un vecteur de caractéristiques de 4096 dimensions en sortie. En plus de prédire la présence d'un objet dans les propositions de région, l'algorithme prédit aussi quatre valeurs de décalage pour augmenter la précision de la boîte englobante [14].

Problèmes avec R-CNN :

- Il nécessite énormément de temps pour former le réseau car il faut classer 2000 propositions de région par image [14].
- Il ne peut pas être mis en œuvre en temps réel car il faut environ 47 secondes pour chaque image de test [14].
- L'algorithme de recherche sélective est un algorithme fixe. Par conséquent, aucun apprentissage ne se produit à ce stade. Cela pourrait conduire à la génération de mauvaises propositions de régions candidates [14].

II.6.2. Fast R-CNN

Certains des inconvénients de R-CNN ont été résolu permettant de créer un algorithme de détection d'objets plus rapide qui s'appelle Fast R-CNN.
L'approche est similaire à l'algorithme R-CNN, mais au lieu de transmettre les propositions de région au CNN, on lui transmet l'image d'entrée pour générer une carte de caractéristiques convolutives, à partir de de laquelle nous identifions la région des propositions et les déformons en carrés en utilisant une couche de regroupement (RoI). À partir du vecteur de caractéristiques, nous utilisons une couche softmax pour prédire la classe de la région proposée ainsi que les valeurs de décalage pour la boîte englobante [15].

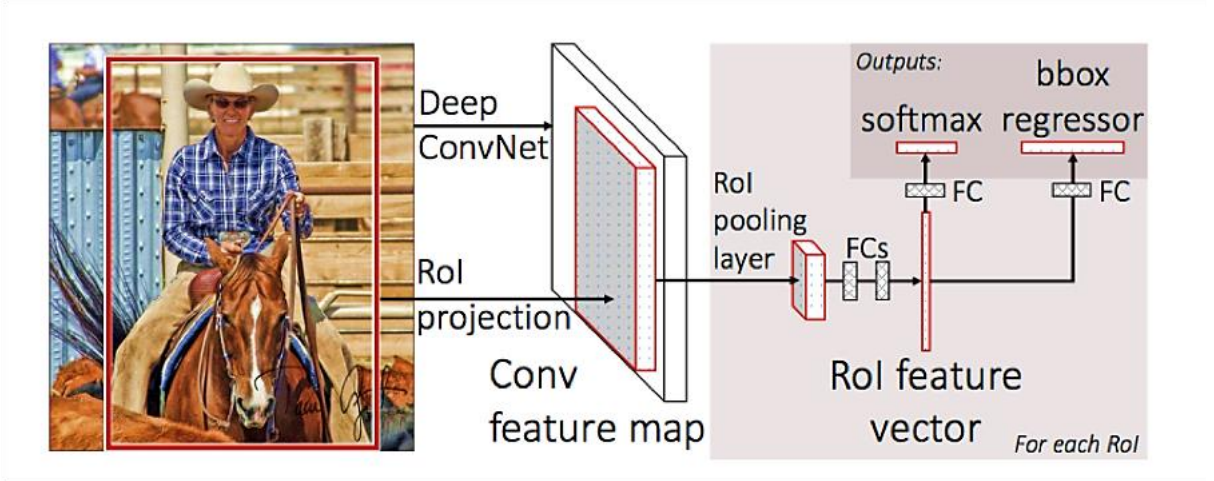

***Figure II.5** : R-CNN Rapide* [15].

La raison pour laquelle "Fast R-CNN" est plus rapide que R-CNN est que nous n'avons pas à fournir 2000 propositions de région au réseau neuronal convolutif à chaque fois. Au lieu de cela, l'opération de convolution n'est effectuée qu'une seule fois par image et une carte de caractéristiques est générée à partir de celle-ci [14].

II.6.3. Faster R-CNN

Les deux algorithmes précédents (R-CNN et Fast R-CNN) utilisent une recherche sélective pour découvrir les propositions de région. La recherche sélective est un processus lent et chronophage qui affecte les performances du réseau. Par conséquent, il a été proposé un algorithme de détection d'objet qui élimine l'algorithme de recherche sélective et permet au réseau d'apprendre les propositions de région [16].

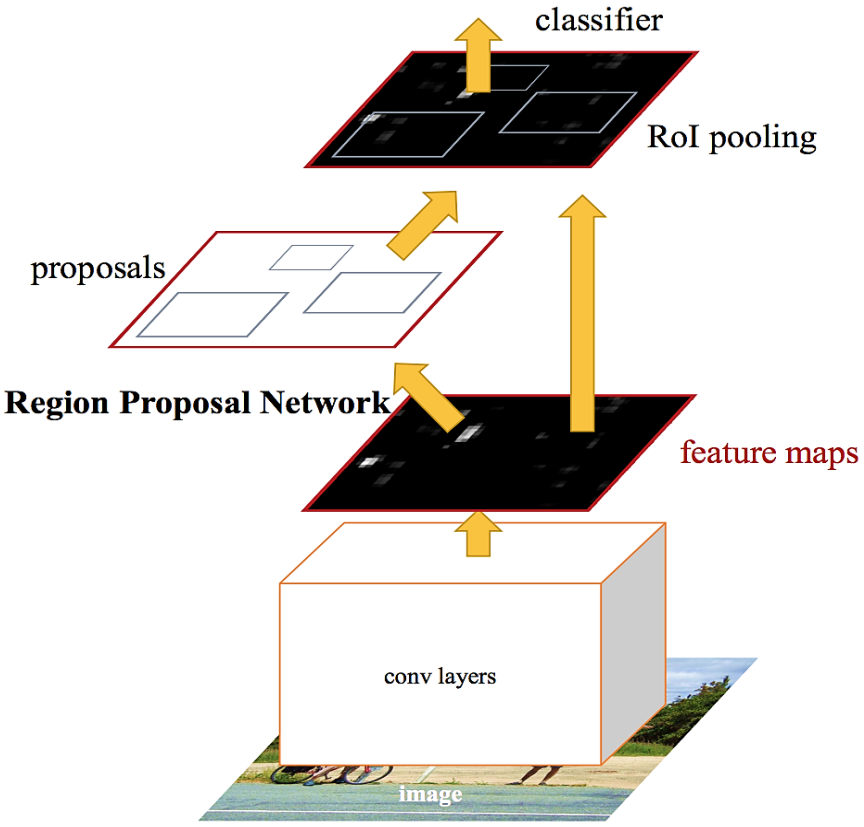

***Figure II.6** : R-CNN plus rapide* [16].

Semblable au Fast R-CNN, l'image est présentée en entrée d'un réseau convolutif qui fournit une carte de caractéristiques convolutives. Au lieu d'utiliser un algorithme de recherche sélective sur la carte des caractéristiques pour identifier les propositions de région, un réseau séparé est utilisé pour prédire les propositions de région. Les propositions de régions prédites sont ensuite remodelées à l'aide d'une couche de regroupement (RoI) qui est ensuite utilisée pour classer l'image dans la région proposée et prédire les valeurs de décalage pour les boîtes englobantes.

Dans le graphique ci-dessous, on peut voir que le Faster R-CNN est beaucoup plus rapide que ses prédécesseurs. Par conséquent, il peut même être utilisé pour la détection d'objets en temps réel [14].

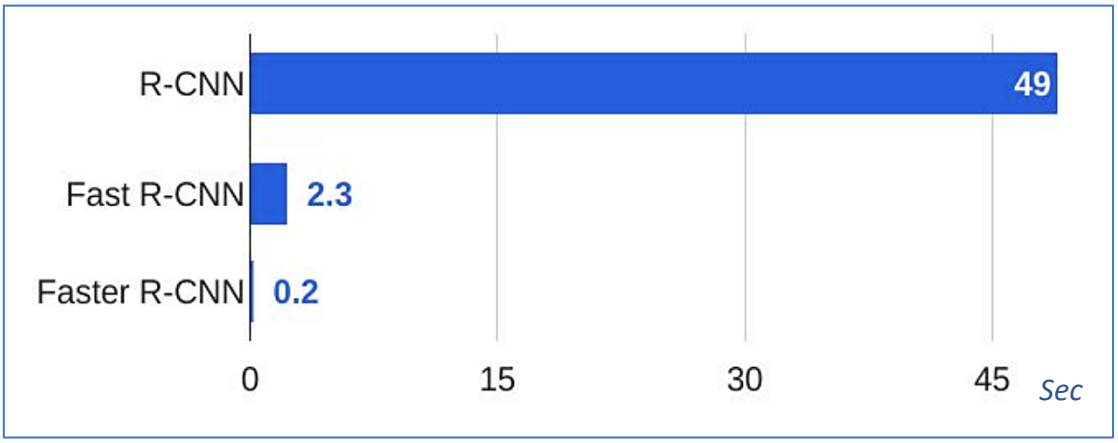

***Figure II.7** : Comparaison du temps de test des algorithmes de détection d'objets* [14].

II.6.4. YOLO

Tous les algorithmes de détection d'objets précédents utilisent des régions pour localiser l'objet dans l'image. Le réseau ne regarde pas l'image complète mais plutôt les parties de l'image qui ont de fortes probabilités de contenir l'objet [12].

YOLO ou You Only Look Once, est un algorithme de détection d'objet très différent des algorithmes basés sur la région vus ci-dessus. Dans YOLO, un seul réseau convolutif prédit les boîtes englobantes et les probabilités de classe pour ces boîtes [12].

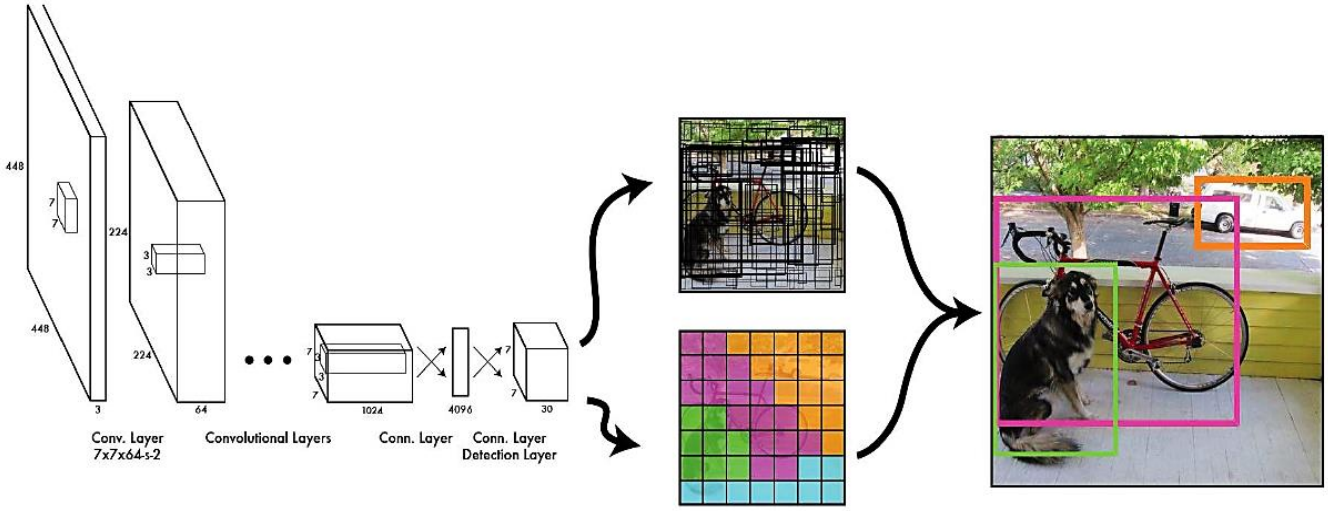

Figure II.8 *: Principe de l'algorithmes de détection d'objets « YOLO »* [17].

Le fonctionnement de YOLO se repose sur le fait que nous prenons une image et la divisons en une grille *SxS*. Dans chacune des grilles nous prenons *m* boîtes englobantes, et pour chacune de celles-ci, le réseau génère une probabilité de classe et des valeurs de décalage pour la boîte englobante. Les boîtes englobantes ayant la probabilité de classe supérieure à une valeur de seuil sont sélectionnées et utilisées pour localiser l'objet dans l'image [18].

YOLO est plus rapide que les autres algorithmes de détection d'objets ; selon la version, il peut atteindre plusieurs dizaines d'images par seconde.

Chapitre III

Stratégies mises en œuvre pour la détection et le suivi d'objets par drone

III.1. Reconnaissance d'objet avec drone

Lors de ce projet, nous avons eu comme objectif la détection en temps réel des objets. Pour ce faire, nous avons opté pour l'algorithme de ***YOLO*** (dont la dernière version est YOLOv5), qui, comme vu dans le chapitre précédent, permet de détecter des objets avec plusieurs images par seconde, cela bien sûr lorsqu'il n'y a pas de contraintes au niveau du processeur graphique (GPU).

III.2. Bases de données d'images

III.2.1. COCO

L'ensemble de données **COCO** est un ensemble de données publié par Microsoft avec lequel YOLO est entrainé à l'origine. La base **COCO** est largement utilisée pour comparer les modèles de détection et de segmentation d'objets ; il est composé d'images naturelles de scènes complexes contenant plusieurs objets. **COCO** contient 91 types d'objets avec plus d'un million d'instances étiquetées dans 328 000 images [19].

Cependant, malgré sa richesse de données, la base **COCO** ne contient pas les objets d'intérêt que l'on veut reconnaitre dans la scène.

III.2.2. HOLOI (Home & Office Library of Object Images)

Voulant détecter des objets spécifiques présents dans la scène, l'algorithme YOLO a été ré-entrainé avec la base d'images **HOLOI** qui contient plus de 500 objets dont chacun est référencé par un numéro unique. Cette base contient les images des objets que nous désirons détecter dans notre projet.

Figure III.1 *: Quelques objets de la base de données d'images HOLOI.*

III.3. Choix du modèle YOLOv5

L'algorithme de détection YOLOv5 est proposé en différentes versions qui correspondent à la taille du réseau (nombre de couches et de neurones). Les versions pré-entrainées sur Coco sont : *"YOLOv5n"*, *"YOLOv5s"*, *"YOLOv5m"*, *"YOLOv5l"* et *"YOLOv5x"*.

Ces différents modèles sont présentés sur la « *figure III.2* » à partir de laquelle on constate que plus le taille du réseau est grande (YOLOv5x ayant la plus grande taille), plus les résultats sont fiables et bons mais cela implique une durée d'apprentissage et une durée d''inférence plus grande. Et inversement pour les réseaux les plus petits (YOLOv5n étant le plus petit), les résultats ont une fiabilité moins bonne mais ils sont obtenus avec une durée d'apprentissage et une durée d'inférence plus courtes.

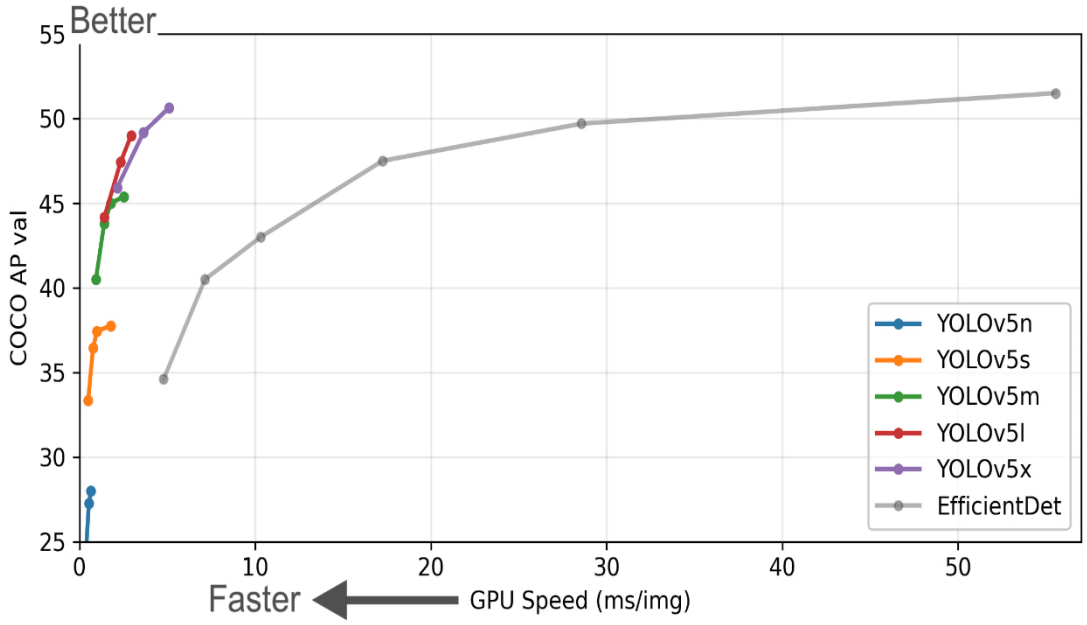

***Figure III.2**: Courbes représentatives des différents niveaux de pré-entrainement de YOLOv5* [20].

Nous avons réalisé des tests sur notre machine qui est équipée du processeur graphique NVIDIA GeForce **GTX 850m** d'une mémoire de 2GB. Ces tests sont effectués avec les différents modèles de YOLOv5 afin d'évaluer le temps qu'il faut à un modèle pour traiter les données et faire une prédiction (temps d'inférence), et de déterminer le nombre d'images par seconde (FPS).

Les résultats sont illustrés dans le tableau ci-dessous :

Modèle	YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
Temps d'inférence	0.1134 sec	0.1465 sec	0.2145 sec	0.3440 sec	0.5797 sec
FPS	8.822	6.825	4,661	2.907	1.725

Pour notre projet, étant donné que nous visons des résultats en temps réel, nous avons choisi d'utiliser la version YOLOv5n parce qu'elle offre un temps d'inférence plus court et donc un plus grand nombre d'images par seconde.

III.4. YOLOv5 sous Windows

Dans le cadre de notre projet, nous avons utilisé l'algorithme YOLO sous le système d'exploitation Windows via l'éditeur de code « Visual Studio Code » ; ce qui a nécessité :

- **Le répertoire YOLOv5** que l'on retrouve sur le site d'Ultralytics.
- **Une version récente de Python** (3.8 ou ultérieure).
- **PyTorch :** C'est est une bibliothèque Python optimisée pour les GPU Nvidia avec le langage CUDA. Elle fonctionne également avec les processeurs (CPU) avec des performances réduites.
- **CUDA** (Compute Unified Device Architecture) **:** C'est une technologie de GPGPU (General Purpose Computing on Graphics Processing Units), c'est-à-dire une technologie qui utilise un processeur graphique (GPU) pour exécuter des calculs généraux à la place du processeur central (CPU). Ainsi, CUDA permet d'utiliser toute la puissance de calcul du système [21].

III.5. Application de YOLOv5 pour la détection d'objet

Afin de détecter les objets via le drone, on charge l'algorithme YOLOv5n ré-entrainé sur la base de données d'images HOLOI.

L'algorithme YOLO permet d'extraire les coordonnées des objets détectés dans les images renvoyées par la caméra du drone : Position ***xmin***, position ***ymin***, position ***xmax***, position ***ymax***, degré de ***confiance***, ***classe*** et ***nom*** respectivement.

On exploite d'abord les informations de position de l'objet détecté pour dessiner un cadre le délimitant, puis son nom pour l'afficher au-dessus du rectangle.

Ainsi, en appliquant l'algorithme YOLOv5n dans notre scène, on obtient les résultats de la « *figure III.3* ».

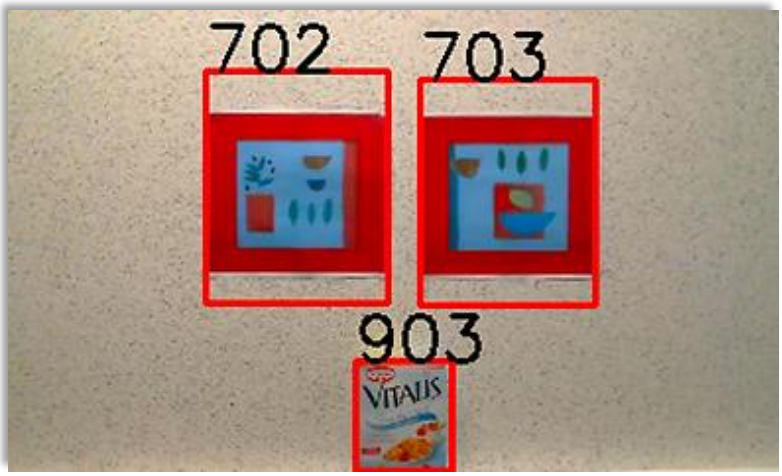

Figure III.3 *: Exemple pratique d'utilisation de l'algorithme YOLOv5n entrainé avec la base HOLOI.*

III.6. Suivi d'objets avec le drone tello

Apres avoir réalisé la détection d'objets en temps réel à partir des vidéos streaming obtenues par le drone, nous aimerions utiliser les informations extraites par l'algorithme YOLO afin que le drone puisse suivre les différents objets.

Pour ce faire, on calcul les deux composantes du centre de l'objet en fonction des postions des quatre points qui le délimitent, ***xmin***, ***xmax***, ***ymin*** et ***ymax :***

$$Centre_x = \frac{Xmax - Xmin}{2}$$

$$Centre_y = \frac{Ymax - Ymin}{2}$$

Ainsi, on doit faire en sorte que le centre de l'objet soit le plus proche possible du centre de la fenêtre affichée par la caméra du drone :

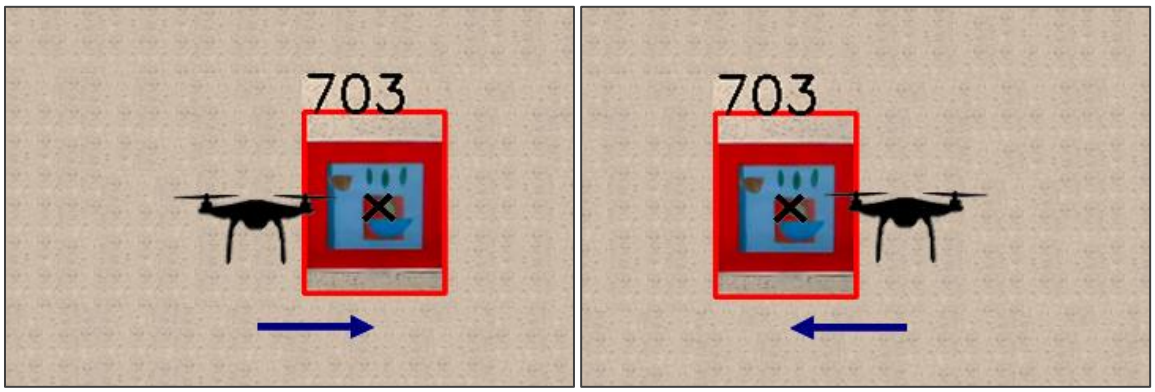

Figure III.4 *: Repositionnement du drone par rapport à l'axe « x ».*

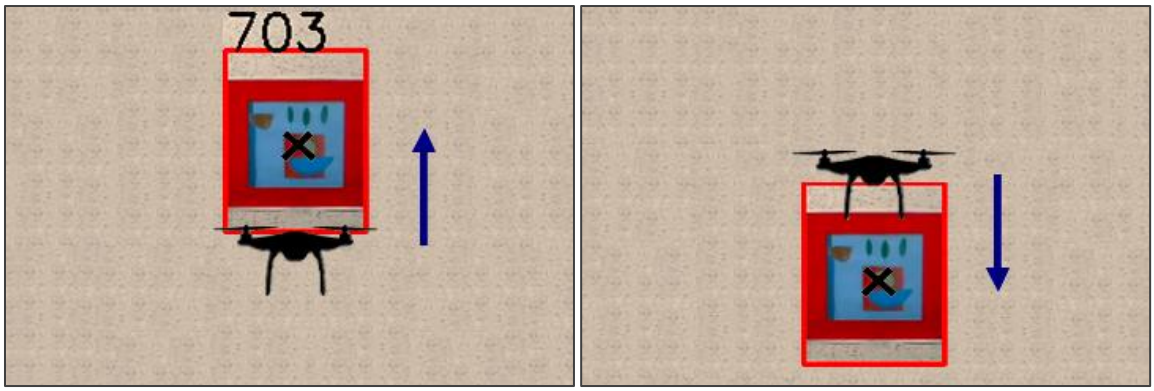

Figure III.5 *: Repositionnement du drone par rapport à l'axe « y ».*

Pour assurer cela, on calcul la distance (l'erreur) entre le centre de l'objet et celui de la fenêtre, ce qui va nous donner les valeurs des déplacements nécessaires pour centrer le drone selon les axes x et y, ceci après avoir multiplier l'erreur par un gain proportionnel qui applique une correction instantanée pour tout écart entre la mesure et la consigne : plus la perturbation est grande, plus la correction apportée est grande.

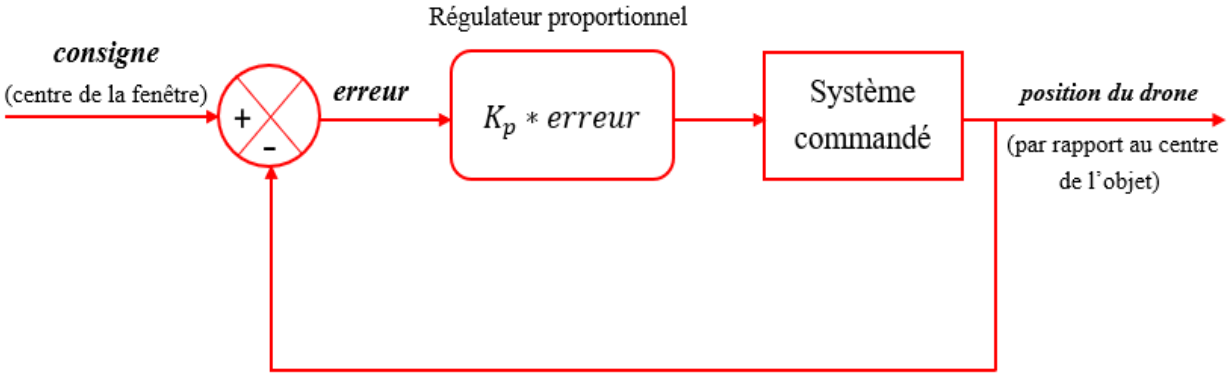

Figure III.6 *: Régulation proportionnelle appliquée pour centrer le drone en « x » et en « y » par rapport à l'objet.*

- Néanmoins, le fait de centrer le drone par rapport à l'objet ne suffit pas pour assurer son suivi ; la notion de distance drone-objet est indispensable pour que le drone parvienne à suivre les objets désirés.

A cet effet, nous définissions un intervalle de surfaces dans lequel l'aire de l'objet vu par le drone doit être.

Cet intervalle est donné par : **[SurfaceMin , SurfaceMax].**

- Si la surface de l'objet est inférieure à *SurfaceMin*, cela signifie que le drone est trop loin, on lui demande donc d'avancer avec une certaine vitesse.

- Si la surface de l'objet est supérieure à *SurfaceMax*, cela signifie que le drone est trop près, on lui demande donc de reculer avec une certaine vitesse.

- Si la surface de l'objet est présente dans l'intervalle [SurfaceMin , SurfaceMax], cela signifie que le drone est à la bonne distance de l'objet.

III.7. Recalage du drone dans la scène

Grace à la stratégie de suivi d'objets élaborée, le drone parvient à se centrer par rapport à l'objet tout en étant à la bonne distance de celui-ci. Cependant il ne connait pas le degré de perpendicularité par rapport à l'objet.

Pour y remédier, nous avons opté pour une méthode qui consiste à recaler le drone par rapport aux murs de manière à ce qu'il soit bien droit dans la scène.

Les murs n'étant pas des objets présents dans la base de données d'images, nous avons utilisé deux objets qui existent dans la base, de taille similaire, placés côte à côte, pour permettre le recalage du drone.

Pour ce faire, nous comparons à chaque instant les surfaces des deux objets :

- S'ils ont la même surface, le drone est perpendiculaire par rapport au mur, on ne lui demande donc pas d'effectuer un mouvement de rotation.
- S'ils ont des surfaces différentes, on calcul la différence ***SurfaceObjet2 – SurfaceObjet1*** pour obtenir la vitesse de rotation (yaw velocity) qui est multipliée à son tour par un régulateur proportionnel.

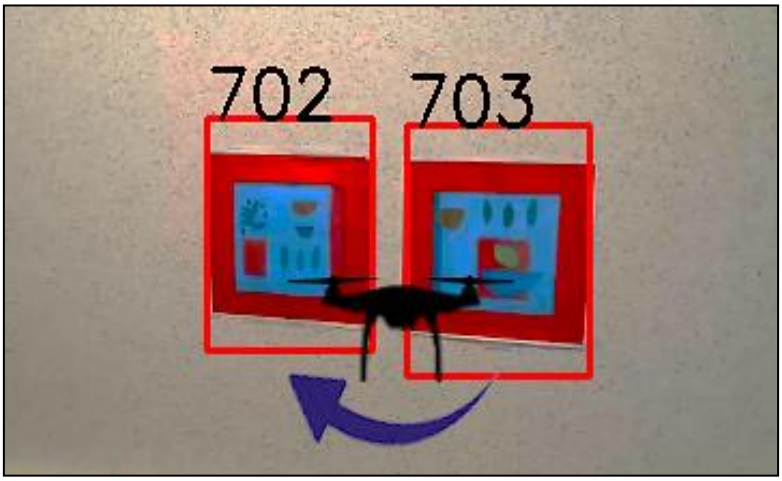

***Figure III.7** : Recalage du drone par rapport à deux cibles.*

III.8. Déplacement du drone combiné avec la technique de recalage

L'objectif ici consiste en la détection d'objets par le drone le long d'un labyrinthe. Une fois que le drone décolle, il longe successivement deux côtés du labyrinthe tout en chargeant l'algorithme de YOLO pour pouvoir détecter les différents objets.

Afin que le drone ait l'information de son positionnement et de sa perpendicularité par rapport au labyrinthe selon différents points de départ, on exploite la technique de recalage par cibles vue précédemment. Ainsi, une fois que le drone est bien droit et en face des deux cibles, il pivote de 180 degrés et se met face au labyrinthe où sont posés les objets. Suite à cela, le drone se déplace autour du labyrinthe et repère les objets présents.

Cependant, suite à ce déplacement du drone, on constate vite les limites de ce dernier concernant la précision des déplacements. En effet les erreurs s'accumulent lors des mouvements du drone ce qui fausse les résultats attendus.

III.9. Obstacles et défis rencontrés

Au cours de ce projet, nous avons été confrontés à quelques obstacles. Tout d'abord, l'utilisation du processeur graphique GTX 850m nous a poussé à choisir le modèle de YOLO ayant la plus petite taille afin d'avoir assez d'images par secondes ; cependant ce modèle YOLOv5n a montré qu'il n'était pas robuste quant à la détection des objets.

Ensuite, afin de mettre le drone en mouvement, nous avons utilisé des commandes bloquantes qui ne permettent aucune interaction lorsqu'elles s'exécutent. Nous avons remarqué que ces commandes bloquantes engendrent des dérives du drone provoquant des erreurs qui s'accumulent.

III.10. Conclusion

Dans ce chapitre nous avons abordé différentes stratégies utilisées pour le drone. Dans un premier temps, nous avons vu comment est faite la détection d'objets en temps réel grâce à l'algorithme YOLO ; puis dans un second temps nous avons présenté la technique de suivi d'objet par le drone Ryze Tello. Le principe du recalage et son utilité ont été ensuite illustrés.

Enfin, nous avons combiné entre le recalage et le déplacement du drone détectant les objets, ce qui nous a amené à constater que Ryze Tello n'est pas dépourvu de toute erreur comme nous l'avons pensé ; il faudrait en effet effectuer un recalage continuellement avec chaque paire d'objets détectée de manière à compenser toute dérives du drone.

CONCLUSION GENERALE

CONCLUSION GENERALE

Ce projet nous a permis d'approfondir nos connaissances quant aux techniques de détection d'objets avec les divers algorithmes dédiés qui utilisent des réseaux de neurones convolutifs : R-CNN, Fast R-CNN, Faster R-CNN et YOLO.

Désirant obtenir des résultats de détection en temps réel via la caméra du drone, nous nous sommes penchés vers l'algorithme YOLO qui permet, même avec un GPU limité, d'obtenir des résultats acceptables.

Nous avons par la suite exploité les informations de la détection d'objets obtenues pour faire un suivi d'objet par le drone DJI Ryze Tello.

Le recalage du drone par rapport au mur à partir de deux cibles de la même taille est nécessaire pour savoir si le drone est bien perpendiculaire dans la salle et pour compenser ses éventuelles dérives.

Nous avons ainsi essayé de combiner la technique de recalage avec le déplacement du drone autour du labyrinthe présent dans la salle, ce qui permet au drone d'être bien perpendiculaire à un mur puis de se mettre en face du labyrinthe, quelle que soit la position de départ du drone. A partir de là, Ryze Tello longe successivement deux côtés du labyrinthe pour détecter et localiser les objets qui s'y trouvent.
Cela a cependant montré quelques limites avec notamment des dérives du drone qui s'accumulent lorsqu'il se déplace le long du labyrinthe.

Le travail présenté dans ce projet ouvre des voies d'améliorations que l'on peut donner comme suit :

- Amélioration de la stratégie de déplacement du drone afin que celui-ci se recale en voyant chaque paire d'objet sur le labyrinthe.
- Mise au point d'un drone autonome qui puisse chercher tout seul les différents objets dans la salle.
- Réalisation d'une cartographie de la salle avec localisation des différents objets détectés.

BIBLIOGRAPHIE

Bibliographie

[1] G. P. Kumar et B. Sridevi, « Chapter 6 - Development of Efficient Swarm Intelligence Algorithm for Simulating Two-Dimensional Orthomosaic for Terrain Mapping Using Cooperative Unmanned Aerial Vehicles », in *The Cognitive Approach in Cloud Computing and Internet of Things Technologies for Surveillance Tracking Systems*, D. Peter, A. H. Alavi, B. Javadi, et S. L. Fernandes, Éd. Academic Press, 2020, p. 75-93. doi: https://doi.org/10.1016/B978-0-12-816385-6.00006-4.

[2] A. Ramchandran et A. Sangaiah, « A Review on Object Detection in Unmanned Aerial Vehicle Surveillance », *International Journal of Cognitive Computing in Engineering*, vol. 2, déc. 2021, doi: 10.1016/j.ijcce.2021.11.005.

[3] « De quoi est composé un drone ? », *studioSPORT*. https://www.studiosport.fr/guides/drones/de-quoi-est-compose-un-drone.html (consulté le 25 mai 2022).

[4] P. Kardasz et J. Doskocz, « Drones and Possibilities of Their Using », *Journal of Civil & Environmental Engineering*, vol. 6, janv. 2016, doi: 10.4172/2165-784X.1000233.

[5] « Manuel d'utilisation du drone RYZE Tello », *Manuals+*, 5 juillet 2021. https://fr.manuals.plus/ryze/tello-drone-manual (consulté le 25 mai 2022).

[6] J. Cavadas, « Using Drone Tello Edu for educational purposes », Master en Applications et Technologies pour les Systèmes Aéronautiques Sans Pilote (Drones), Université Polytechnique de Catalogne, 2019.

[7] F. Chollet, *Deep Learning with Python*. Manning, 2017.

[8] R. Airola et K. Hager, « Image Classification, Deep Learning and Convolutional Neural Networks », C-level thesis, Karlstad University, 2017.

[9] « Object Recognition vs Object Detection vs Image Segmentation | Data Science and Machine Learning ». https://www.kaggle.com/getting-started/169984 (consulté le 22 mai 2022).

[10] U. Sehar et M. L. Naseem, « How deep learning is empowering semantic segmentation: Traditional and deep learning techniques for semantic segmentation: A comparison », *Multimed Tools Appl*, avr. 2022, doi: 10.1007/s11042-022-12821-3.

[11] R. Tedrake, « Object detection and segmentation », 20 mai 2022. https://manipulation.csail.mit.edu/segmentation.html (consulté le 22 mai 2022).

[12] R. Gandhi, « R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms », *Medium*, 9 juillet 2018. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e (consulté le 22 mai 2022).

[13] R. B. Girshick, J. Donahue, T. Darrell, et J. Malik, « Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation », *2014 IEEE Conference on Computer Vision and Pattern Recognition*, p. 580-587, 2014, doi: 10.1109/CVPR.2014.81.

[14] H. Chehri, A. Chehri, L. Kiss, et A. Zimmermann, « Automatic Anode Rod Inspection in Aluminum Smelters using Deep-Learning Techniques: A Case Study », *Procedia Computer Science*, vol. 176, p. 3536-3544, janv. 2020, doi: 10.1016/j.procs.2020.09.033.

[15] R. Girshick, « Fast R-CNN », in *Proceedings of the IEEE international conference on computer vision*, 2015, p. 1440-1448.

[16] S. Ren, K. He, R. Girshick, et J. Sun, « Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, n° 6, p. 1137-1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[17] S.-H. Tsang, « Review: YOLOv1 — You Only Look Once (Object Detection) », *Medium*, 20 mars 2019. https://towardsdatascience.com/yolov1-you-only-look-once-object-detection-e1f3ffec8a89 (consulté le 22 mai 2022).

[18] J. Redmon, S. Divvala, R. Girshick, et A. Farhadi, « You Only Look Once: Unified, Real-Time Object Detection ». arXiv, 9 mai 2016. Consulté le : 22 mai 2022. [En ligne]. Disponible sur : http://arxiv.org/abs/1506.02640

[19] B. Al-Abudi et R. Hassan, « CLASSIFICATION COCO DATASET USING MACHINE LEARNING ALGORITHMS », janv. 2022.

[20] G. Jocher et al, *ultralytics/yolov5: v6.0 - YOLOv5n Nano models, Roboflow integration, TensorFlow export, OpenCV DNN support*. Zenodo, 2021. doi: 10.5281/zenodo.5563715.

[21] N. S. Artamonov et P. Y. Yakimov, « Towards Real-Time Traffic Sign Recognition via YOLO on a Mobile GPU », *J. Phys.: Conf. Ser.*, vol. 1096, p. 012086, sept. 2018, doi: 10.1088/1742-6596/1096/1/012086.

Abstract :

The objective of our project is the detection of objects with a drone using a neural network. To do this, we have developed a strategy in several separate steps. First, we perform object recognition by the drone with the YOLO algorithm, then we develop the object tracking technique that we improve in order to exploit it for the readjustment of the drone.

Following this, we combine the detection of objects with the readjustment of the drone so that it is positioned in the right direction and in the right position to detect the objects.

However, the Ryze Tello drone showed some limitations in accuracy, prompting us to consider in the future readjusting the drone from every object it sees.

Keywords : *Ryze Tello drone , object detection , neural network, , YOLO , object tracking.*