الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ÉCOLE NATIONALE POLYTECHNIQUE
CENTRE DE RECHERCHE SUR
L'INFORMATION SCIENTIFIQUE ET
TECHNIQUE

Département d'Électronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme :
Ingénieur d'état en Électronique

Navigation visuelle autonome de drone en milieu fermé

Réalisé par :
Mohamed Nabil Tchoulak

Présenté et Soutenu le 10 Septembre 2020 devant le jury composé de :

Président	Cherif	LARBÈS	Prof.	ENP, Alger
Encadrant	Sid-Ahmed	BERRANI	MCA.	ENP, Alger
Encadrant	Saïd	YAHIAOUI	MR.	CERIST, Alger
Examinateur	Mourad	ADNANE	Prof.	ENP, Alger

ENP 2020

الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ÉCOLE NATIONALE POLYTECHNIQUE
CENTRE DE RECHERCHE SUR
L'INFORMATION SCIENTIFIQUE ET
TECHNIQUE

Département d'Électronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme :
Ingénieur d'état en Électronique

Navigation visuelle autonome de drone en milieu fermé

Réalisé par :
Mohamed Nabil Tchoulak

Présenté et Soutenu le 10 Septembre 2020 devant le jury composé de :

Président	Cherif	LARBÈS	Prof.	ENP, Alger
Encadrant	Sid-Ahmed	BERRANI	MCA.	ENP, Alger
Encadrant	Saïd	YAHIAOUI	MR.	CERIST, Alger
Examinateur	Mourad	ADNANE	Prof.	ENP, Alger

ENP 2020

Dédicace

À mes très chers parents, pour tous leurs sacrifices, leur dévouement, et leur soutien tout au long de mes études. Que Dieu les protège.

À Nassim, Lydia
à Hamza, Merouane, Yacine, Sofiane, Daoud, Abdellah, Tinders,
et à tous ceux que j'aime,

Je dédie ce mémoire

Mohamed Nabil TCHOULAK

Remerciements

Ce travail a été réalisé au Centre de Recherche sur l'Information Scientifique et technique CERIST sous la direction de Monsieur *Said Yahiaoui*, maitre de recherche au CERIST, et Monsieur *Sid Ahmed Berrani*, enseignant chercheur à l'École Nationale Polytechnique.

Toute ma reconnaissance et mes remerciements à Monsieur *Said Yahiaoui*, et Monsieur *Sid Ahmed Berrani*, pour avoir assuré la direction de ce mémoire. Merci pour leurs visions globales et leurs caractères humains et aussi pour l'ensemble des orientations et conseils qu'ils m'ont prodigués tout au long de ce travail.

Je tiens à exprimer mes remerciements aux membres du jury, pour avoir consacré une partie de leurs temps à la lecture de ce mémoire et qui ont acceptés d'évaluer mon travail.
Mes remerciements les plus sincères et respectueux à Monsieur Cherif Larbès.Professeur à l'Ecole Nationale Polytechnique pour l'honneur qu'il ma fait de présider ce jury.
J'adresse mes sincères gratitudes à Monsieur *Mourad Adnane*, professeur à l'Ecole Nationale Polytechnique pour avoir accepté d'examiner ce travail et de faire partie du jury.

J'exprime également un grand merci à mes parents qui sans eux je n'aurais pas pu atteindre ce stade, à mes parents, mes proches, mes amis et ma sœur qui m'ont encouragés.

Mohamed Nabil TCHOULAK

ملخص يقدم التنقل المرئي محور بحث وتطوير نشط للغاية في مجال الروبوتات الجوية ، بدافع من الابتكارات العظيمة التي لوحظت في رؤية الكمبيوتر والمزايا التي تقدمها الطائرات بدون طيار ، والتي يتم استخدامها أكثر فأكثر ، مثل التدخل في الكوارث الطبيعية والذي يمكن اعتباره تحديًا جيدًا للملاحة يتميز بظروف قاسية. نقدم حلاً للتنقل المستقل في البيئات المغلقة ، يتميز بإجراء استكشاف يحسن المسافة المقطوعة باستخدام مبدأ ءفضل رؤية تاليةْ. وهو بدوره يستند إلى تقدير الموضع والتمثيل البيئي الذي توفره خوارزمية إدراك مرئي *SVO* باستخدام كاميرا وإشارات *IMU* . وسوف يستخدم تدفق هذه الكاميرا لاكتشاف الأشكال البشرية باستخدام شبكة عصبية تلافيفية، للإبلاغ عن وضع الناجين بعد كارثة طبيعية. وقد نُفذت و قيمت الأجزاء الثلاثة من النهج لتقديم دليل على الحل المقترح.

كلمات مفتاحية ـ الطائرات بدون طيار المستقلة، التصورات البصرية، استكشاف البيئات غير المعروفة، البحث والإنقاذ، الشبكات العصبية التلافيفية.

Abstract — Visual navigation presents an active research and development axis in aerial robotics, motivated by great innovations observed in computer vision and advantages offered by UAVs which are increasingly used in countless applications, such as intervention in natural disasters, which can be seen as a navigation challenge characterized by extreme conditions. We present a solution for autonomous navigation in indoor environments, characterized by an exploration procedure that optimizes the distance using the *Next Best View* principle. It is based on position estimation and representation of the environment provided by *SVO*, an visual perception algorithm, using a monocular camera and an *IMU* sensor. The stream of this camera is used to detect and locate human forms by means of a convolutional neural network, which allows to report the position of survivors after a natural disaster. The three parts constituting the approach were implemented and assessed in order to provide a proof of concept of the proposed solution.

Index-terms : Autonomous UAV, Visual perception, Exploration of unknown environments, Search and rescue, Convolutional Neural Networks.

Résumé — La navigation visuelle présente un axe de recherche et développement très actif dans la robotique aérienne, motivé par les grandes innovations observées dans la vision par ordinateur et les avantages qu'offrent les drones.Ces derniers se voient de plus en plus utilisés dans d'innombrables applications, comme l'intervention dans les catastrophes naturelles qui peut être vue comme un bon défi de navigation caractérisé par des conditions extrêmes. Nous présentons une solution pour la navigation autonome dans des milieux fermés, caractérisée par une procédure d'exploration qui optimise la distance parcourue à l'aide du principe *Next Best View*. Elle repose sur l'estimation de la position et la modélisation de l'environnement fournies par un algorithme de perception visuelle *SVO*, en utilisant une caméra monoculaire et une centrale inertielle *IMU*. Le flux de cette caméra sert aussi à détecter et localiser les formes humaines à l'aide d'un réseau de neurones convolutifs, ce qui permet de rapporter la position des survivants après une catastrophe naturelle. Les trois parties qui constituent l'approche ont été implémentées et évaluées afin de fournir une preuve de concept de la solution proposée.

Mots-clés : Drones autonomes, Perception visuelle, Exploration des milieux inconnues, Recherche et sauvetage, Réseaux de neurones convolutifs.

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

BiFPN : Bi-directional Feature Pyramid Network.

CICR : Center for Intelligent Computing and Robotics.

CNN : Convolutional Neural Network.

CPU : Central Processing Unit.

CUDA : Compute Unified Device Architecture.

cuDNN : CUDA Deep Neural Network.

CVPR : Conference on Computer Vision and Pattern Recognition.

FAIR : Facebook AI Research.

FPN : Feature Pyramid Networks.

FPS : Frame Per Second.

GPS : Global Positioning System.

GPU : Graphics Processing Unit.

IBVS : Image Based Visual Servoing.

IFRC : International Federation of Red Cross.

IMU : Inertial Measurement Unit.

LSD-SLAM : Large Scale Direct-Simultaneous Localization And Mapping.

mAP : mean Average Precision.

MEMS : Micro Electro-Mechanical Systems.

NAS : Neural Architecture Search.

NBV : Next Best Vue.

PAN (PANet) : Path Aggregation Network.

RCNN : Region Based Convolutional Neural Networks.

RFCN : Region-based Fully Convolutional Networks.

RGB : Red Green Blue.

RNN : Recurrent Neural Network.

ROS : Robot Operating System.

RPG : Robotics and Perception Group.

RRT : Rapidly-exploring Random Trees.

SIFT : Scale-Invariant Feature Transform.

SLAM : Simultaneous Localization And Mapping.

SPP : Spatial Pyramid Pooling.

SSD : Single Shot MultiBox Detector.

SURF : Speeded Up Robust Features.

SVO : Semi-Direct Visual Odometry.

UAV : Unmanned Aerial Vehicle.

UNDRR : United Nations Office for Disaster Risk Reduction.

V-SLAM : Visual-Simultaneous Localization And Mapping.

VO : Visual Odometry.

YOLO : You Only Look Once.

Introduction générale

Depuis son apparition, la robotique est un domaine qui connait une notoriété de plus en plus grandissante. En particulier, les robots volants tels que les drones à voilure fixe ou tournante se voient d'être d'une grande utilité et apportent de nouvelles solutions dans plusieurs domaines. On retrouve des applications qui vont de l'exploration et l'inspection aérienne à la livraison en passant par la reconstruction 3D d'environnements difficiles d'accès.

De ce fait, il existe un fort intérêt scientifique à développer des solutions permettant aux drones (UAV : *Unmaned Aerial Vehicels* en anglais) de voler de façon autonome, tout en ajoutant de nouvelles fonctionnalités pour exploiter les possibilités qu'offrent ces systèmes à savoir :

- Une bonne reconnaissance de l'environnement à travers les différents capteurs possibles (laser, caméras, capteurs ultrasoniques, etc.).
- Capacité à effectuer des traitements qui peuvent dépasser les performances d'humains qualifiés.
- Accessibilité aisée à des endroits difficiles d'accès ou jugés à haut risque.
- Possibilité d'embarquer plusieurs systèmes technologiques (antennes 5G, outils de télécommunications, actionneurs, etc).

Les méthodes de navigation pour la robotique mobile se focalisent, entre autres, sur les scanners laser, l'ultrason, et les caméras pour détecter les obstacles et suivre une trajectoire. De plus, l'utilisation de la vision permet d'élargir le champ d'action du drone en collectant des données plus explicites sur l'environnement. En effet, à l'inverse du GPS (*Global Positioning System*), IMU (*Inertial Measurement Unit*) et capteurs de distance (sonar, laser, infrarouge), les caméras produisent des données riches en caractéristiques utilisables lors de la modélisation 3D et la navigation (couleur, texture, flux optique, détection d'objets ...). Exploiter ces données est moins explicite que dans le cas d'un GPS ou d'un scanner laser mais bien plus éloquent [1].

Parallèlement, les drones, et les robots de manière générale, sont appelés à l'avenir à être déployés dans des environnements dynamiques et non structurés, fournissant une assistance dans de nombreux domaines allant des applications militaires aux secours en situations dangereuses. Il sera donc difficile de pré-programmer toutes les tâches et tous les scénarios possibles dans de tels robots. Les robots devront être capables de s'adapter aux environnements méconnues en tenant compte des signaux d'état internes uniquement (comme par exemple, en cas d'absence ou de perte des signaux GPS).

L'apprentissage automatique sera nécessaire pour atteindre ce haut degré

d'autonomie et permettra aux robots de percevoir et d'agir adéquatement aux variations de son environnement. De toutes les techniques d'apprentissage automatique, l'apprentissage profond (*Deep Learning*) a récemment retenu l'intérêt des chercheurs, en particulier dans le domaine de la vision par ordinateur [2, 3, 4].

Un des domaines de recherche actifs dans l'utilisation de ces technologies est l'assistance dans la gestion des catastrophes naturelles. Cet axe de recherche et développement propose des défis qui fournissent à la fois inspiration, impact et validation pour les développements à travers des exigences strictes en terme de fiabilité et d'efficacité. Il est motivé par l'énorme impact que peut avoir une assistance dans la gestion d'une telle crise sur la vie des personnes.

Objectif du travail

Les travaux présentés dans ce mémoire ont pour objectif :

- L'étude, le développement et la validation d'un système autonome embarqué sur un drone pour la navigation dans des milieux fermés, basé sur les techniques de perception visuelles. Ce système doit permettre l'exploration du milieu et la détection des humains dans le cas d'un scénario d'assistance dans la gestion de catastrophe naturelle.

Contribution

La contribution apportée dans ce travail consiste en la mise en place d'une architecture basée sur la navigation visuelle et les techniques d'apprentissage profond, permettant à un drone d'explorer un environnement fermée et d'en extraire des informations utiles.

Organisation du mémoire

Ce document est organisé de la manière suivante :

Le premier chapitre servira à décrire la problématique de la gestion des catastrophes naturelles, les défis qu'elle représente ainsi que les axes où nous pouvons apporter une contribution.

Dans le second chapitre, nous exposerons l'état de l'art des architectures proposées pour l'aide à la gestion de catastrophes naturelles en utilisant les drones. Nous verrons trois aspects principaux liés à la navigation des drones à savoir :

- la perception visuelle pour à la fois l'estimation de l'état et la reconstruction de l'environnement ;
- la procédure autonome d'exploration des milieux inconnues ;
- l'algorithme de détection des objets.

Une configuration qui sera reconduite dans l'élaboration de l'architecture de la solution proposée, qui est présentée à la fin du chapitre.

Dans le troisième chapitre, nous poserons le modèle cinématique du drone qui sera utilisé comme référence pour la suite du mémoire. Par la suite, nous décrirons la technique de perception visuelle qui permet l'estimation de l'état, les concepts adoptés dans l'algorithme de perception (*structure from motion*, géométrie épipolaire, calibration de la caméra, etc.), ainsi que les différents composants de l'algorithme de perception.

Le quatrième chapitre décrit la méthode adoptée pour la procédure d'exploration des milieux inconnues. Nous verrons en détails l'approche suivie et les différents critères qui sont pris en considération lors de l'exploration.

Enfin, le cinquième chapitre fera le lien entre les différents composants de la solution proposée à travers l'architecture globale. Nous présenterons les tests effectués sur les différents composants de l'architecture et la configuration adoptée.

Chapitre 1

Problématique

1.1 Introduction

Les catastrophes naturelles ont causé la mort de 1.3 millions de personnes dans le monde entre 1998 et 2017, soit 0.1% des décès mondiaux durant la dernière décennie. Les tremblement de terres en sont les plus dangereux et totalisent 260 milles morts entre 2010 et 2015 [1].
La Figure 1.1 montre les dégâts observés lors d'un séisme qui a touché l'Algérie en 2003, causant 2 266 morts et 10 261 blessés [5].

FIGURE *1.1 – Photo prise après le tremblement de terre de Boumerdès (2003)*

Dans le but de permettre une réaction utile avant, pendant et après une catastrophe naturelle, de nombreux efforts sont entamés pour maitriser la gestion de ces phénomènes. Notamment en évaluant rapidement et efficacement les dégâts, remédier aux pannes et restaurer la normalité.

1. Selon les données fournies par : https://ourworldindata.org/natural-disasters

Les robots en général et les drones en particulier sont des outils efficaces dans la gestion des catastrophes naturelles, ils peuvent pénétrer à travers des zones à hauts risques sans se mettre en danger et sans subir les effets des émotions humaines. De plus les drones peuvent être dotés de capteurs (caméras, lidars, etc.), qui puissent rapporter exactement ce qui se passe dans les environnements visités.

Dans ce chapitre nous présentons la problématique d'intervention des drones dans les catastrophes naturelles (sur les volets exploration, analyse, navigation), en focalisant sur les défis à relever et les critères à respecter par ces derniers.

1.2 Gestion des catastrophes naturelles

La "Fédération Internationale des Sociétés de la Croix-Rouge et du Croissant-Rouge" définit la gestion des catastrophes comme étant l'organisation et la gestion des ressources et des responsabilités pour le traitement des aspects humanitaires dans les situations d'urgence, en particulier la préparation, la réponse et le rétablissement afin de réduire l'impact des catastrophes naturelles.

Elle se divise donc en quatre étapes distinctes :

1. **La prévention :** L'UNDRR[2] voit la prévention comme étant le concept de s'engager dans des activités qui visent à prévenir ou éviter des risques potentiels à travers des actions prises à l'avance. Ces actions comptent l'élaboration d'un bon plan de gestion de risques, des plans d'évacuations, des standards de construction et de design, etc.

2. **La préparation :** Selon le CICR[3] la préparation fait référence aux mesures prises pour se préparer et réduire les effets des catastrophes, qu'elles soient naturelles ou causées par l'homme. Ceci est réalisé par la recherche et la planification afin d'essayer de prévoir les zones ou les régions qui pourraient être exposées à un risque de catastrophe et, si possible, de les empêcher de se produire.

3. **La réaction et les secours :** Elle est définie comme étant la fourniture de services d'urgence et d'assistance publique pendant ou immédiatement après une catastrophe dans le but de :
   - sauver des vies,
   - réduire les effets sur la santé,
   - assurer la sécurité publique et répondre aux besoins de subsistance de base des personnes touchées.

4. **La reprise :** La reprise fait référence aux programmes qui sont appliqués après la fourniture des secours immédiats pour aider ceux qui ont subi les conséquences de la catastrophe. Ces programmes doivent comporter la reconstruction des infrastructures, les soins de santé et de réadaptation, le renforcement du corps

2. United Nations Office for Disaster Risk Réduction : https://www.undrr.org/
3. Comité International de la Croix Rouge : https://www.ifrc.org/en/--/

sanitaire, ainsi que les politiques de développement visant à éviter ou atténuer des situations similaires à l'avenir.

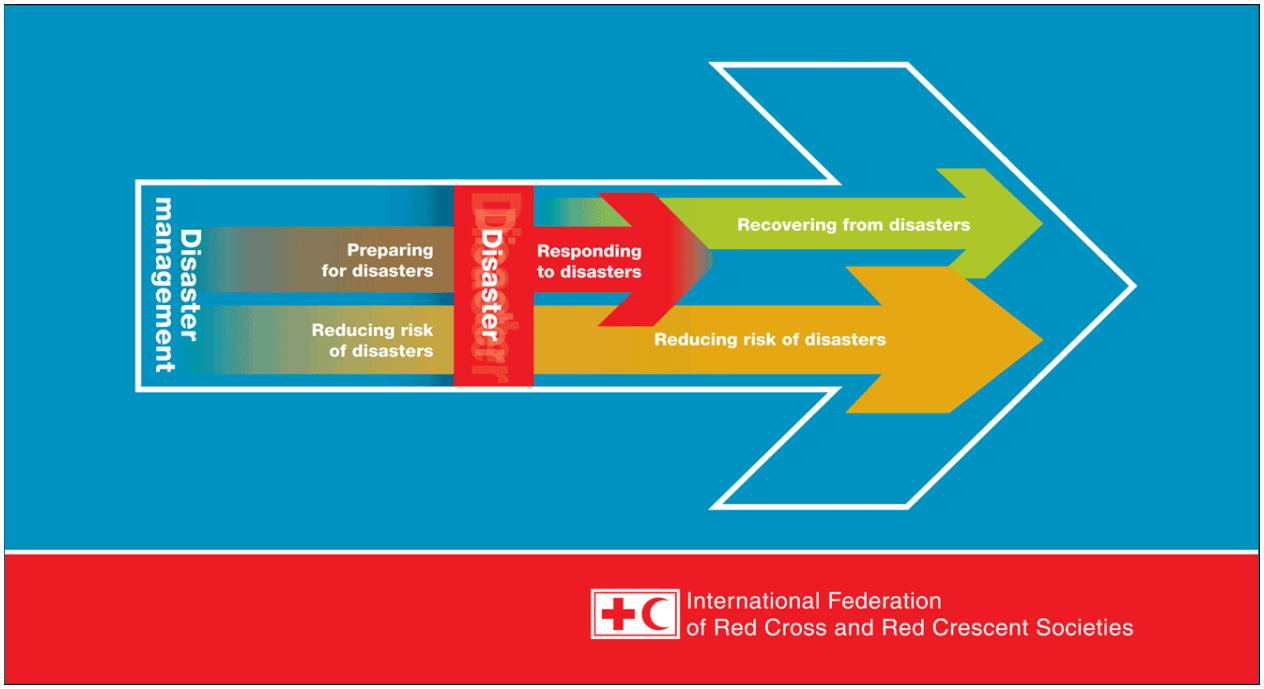


FIGURE 1.2 – *Schéma de la gestion des catastrophes naturelles d'après l'IFRC*[4].

1.3 Les défis et les critères d'opérations des drones dans un scénario de catastrophes naturelles

En réaction à une catastrophe naturelle, les premières 72 heures (qui constituent la période des recherches et sauvetages) sont considérés les plus importantes et les plus critiques dans la procédure de sauvetage des vies humaines qui doivent se dérouler de manière rapide et efficace. En effet, l'emploi des volontaires pour ces missions constitue dans plusieurs cas un désavantage aux équipes de sauvetages à cause du manque d'expérience et d'outils nécessaires chez ces volontaires [6].

Le problème principal que rencontre les équipes de sauvetages durant une catastrophe naturelle est le manque de communication, ainsi que la méconnaissance de l'environnement dans lequel se trouve les victimes, qui pousse parfois les secouristes à improviser et ainsi diminuer de l'efficacité de la mission.

Un drone qui contribue dans ces démarches va donc opérer dans des milieux fermés (communication par satellites impossible), sans informations fournies au préalable sur l'environnement, pour une période de temps limitée. De plus il devra répondre à des critères de déploiement dans le but d'assurer ses performances.

4. Plus de détails sur : https://www.ifrc.org/fr/introduction/gestion-de-catastrophes/about-disaster-management/

1.3.1 Défis

Les défis rencontrés dans les catastrophes naturelles sont principalement liés à la préservation des vies humaines, ce qui peut comporter :

- **Interventions directes :** comme l'illustre la figure 1.1, les endroits touchés par les désastres naturels sont caractérisés par des passages étroits et des environnements clos ; l'intervention dans ces cas présente à la fois une grande difficulté et un énorme risque pour les secouristes.
  Les plateformes mobiles peuvent avoir des caractéristiques physiques et mécaniques qui leur permettent de pénétrer facilement à travers des zones que les humains ne peuvent pas franchir.

- **Mesure des dégâts :** avant d'envoyer des équipes de secourisme sur les lieux, il est important de mesurer l'état des endroits touchés afin de garantir la sécurité des intervenants et de repérer les endroits qui peuvent contenir des survivants.

De plus, [7] présente une liste des défis et problèmes non résolus en ce qui concerne l'utilisation des drones dans les mission de sauvetage et de recherche :

- **Gestion des pannes de drones :** pour assurer un fonctionnement à sécurité intégrée du système, un opérateur humain doit être présent pour superviser l'état du drone. Il pourra réinitialiser ou arrêter le véhicule en enclenchant un interrupteur d'arrêt ou en outrepassant manuellement la commande du drone. Des procédures automatisées plus avancées de traitement des défaillances devraient être envisagées et mises en œuvre.

- **La diffusion des données sur un réseau de relais :** les images collectées par les drones présentent un aperçu de la situation. Cependant, les personnes touchées pourraient également utiliser les réseaux sociaux ou transmettre des messages textuels ou des images via un réseau de relais qui peut être éventuellement formé par le/les drones de sauvetage. Cette activité offre des informations précises en temps réel.

1.3.2 Critères

Un drone destiné à servir dans une procédure de recherche et sauvetage durant une catastrophe naturelle devrait remplir les conditions suivantes :

- **Efficacité de positionnement :** le drone doit connaître sa position de manière précise à chaque instant pour lier les observations aux lieux ou tout simplement pour naviguer. En effet dans de tels circonstances on peut facilement se retrouver dans des milieux dépourvues du signal GPS, ce qui rend l'utilisation d'un système de positionnement indépendant des signaux satellitaires une tâche primordiale.

- **Efficacité énergétique :** la mission d'un drone dans un scénario de recherche et sauvetage doit être fortement optimisé, certaines commandes sous-optimales limitant les mouvements donnent, en effet, de meilleures résultats dans les

longues missions.

- **Navigation agile :** dans les missions de recherche, on se retrouve souvent dans des environnements non convenablement structurés (murs penchés, fissures, débris, etc.). Tout système pénétrant dans ces environnements doit réussir à éviter les obstacles de manière efficace et agile.

- **Fiabilité de l'information :** l'information rapportée par le drone sur la présence d'une personne ou d'un objet lors de la recherche doit avoir un grand taux de fiabilité, vu l'importance de l'information sur les décisions prises.

1.4 Conclusion

Les drones peuvent avoir une grande contribution dans les missions de recherche et sauvetages. Ils peuvent ainsi contribuer à sauver un nombre considérable de vies pendant l'étape des secours.

Cette mission se veut d'être au croisement de plusieurs sous-problématiques. Elle peut être vue comme un problème de navigation et d'exploration, en plus du recueil et d'exploitation des informations à partir de l'environnement. En respectant les contraintes de précision, fiabilité et durabilité.

Dans le prochain chapitre, nous allons présenter l'état de l'art sur la problématique globale qui traite l'intervention des drones après les catastrophes naturelles ; ainsi que sur les sous problématiques qui englobent :
- la navigation et la perception visuelles,
- l'exploration des milieux inconnues à travers les drones,
- la détection des objets dans une image.

Chapitre 2

État de l'art

2.1 Introduction

Avec l'avancée technologique des robots, l'utilisation de ces derniers dans la gestion des catastrophes naturelles est un sujet d'actualité qui fait l'objet de plusieurs travaux durant les dernières années. L'embarcation de plusieurs capteurs sur un robot rend la cartographie des environnements touchés par des tremblement de terre possible [8]. La facilité de déplacement des drones leur permet d'être un outil très efficace pour la collecte d'information lors d'une catastrophe naturelle [9]. Les robots sous-marins aussi contribuent aux recherches, comme c'était le cas pendant le séisme de la côte Pacifique du Tōhoku au Japon [10].

Plus généralement, la navigation robotique à grande vitesse dans des environnements encombrés et inconnus est actuellement un domaine de recherche très actif [4, 11, 12, 13, 14, 15]. De nombreuses études récentes se basent sur les drones pour assister les campagnes de recherche et secourisme dans : le recueil d'informations [9], la livraison des kits de soin médicaux et des médicaments [16], l'exploration des environnements inconnus [17] et la détection des objets (humains, voitures, animaux , etc.) [18, 19].

La problématique de drones intervenants au cours d'une mission de recherche et de secourisme, telle que nous la concevons, est subdivisible en quatre sous-problématiques :

1. **L'estimation de l'état :** Position et orientation du drone par rapport à un repère fixe.

2. **Reconstruction de l'environnement :** Prise de connaissance de l'environnement de navigation pour permettre l'évitement d'obstacles.

3. **Exploration d'environnements :** Procédure optimale qui permet de parcourir et d'explorer un environnement méconnu.

4. **Détection intelligente des objets :** Détection d'objets présents (humains ou objets cibles) à travers un flux d'images enregistré durant l'exploration.

Dans ce chapitre, nous explorerons l'état de l'art de chacune de ces sous-problématiques et nous verrons les approches déjà adoptées dans la littérature. Nous identifierons ainsi les solutions qui nous conviennent le mieux.

2.2 Travaux antérieurs sur la contribution des drones après les catastrophes naturelles

Plusieurs travaux ont été réalisés dans le cadre de l'assistance des équipes de recherche et d'intervention lors des missions de secourisme.

L'une des approches les plus adaptées consiste en la détection des humains et/ou des véhicules dans de larges zones à l'extérieur utilisant une caméra thermique et une caméra monoculaire sur des drones à voilure fixe couvrants de larges zones [20, 17].
Un autre aspect est traité dans [8], qui consiste en la reconstruction en 3D d'immeubles touchés par un tremblement de terre, utilisant plusieurs robots aériens en collaboration qui pénètrent à l'intérieur des immeubles en question.

Par ailleurs l'un des problèmes fréquents rencontrés dans les catastrophes naturelles est la perte du réseau de communication mobile, ce qui rend la demande d'aide des victimes difficile et complique la coordination entre équipes de secourisme. Les articles [21] et [7] proposent de déployer un groupement de robots qui va lier le réseau de la zone endommagée avec le réseau global pour permettre une bonne communication.

Cependant ces solutions supposent le cas des milieux extérieurs caractérisés par une bonne couverture du signal GPS pour se positionner. Ce qui n'est pas adapté à tous les scénarios de catastrophes naturelles, où le principal défi consiste à pénétrer à l'intérieur des endroits (dépourvue de signal GPS) pour repérer et secourir les blessés. Par conséquent, le travail effectué dans [22] introduit une architecture complète d'un drone dont le but est d'explorer des milieux inconnus, et qui utilise un réseau de neurones convolutifs (*CNN*) pour repérer les objets et un algorithme d'apprentissage renforcé pour se positionner au dessus de l'objet détecté (en apprenant à partir des techniques IBVS : *Image Based Visual Servoing*). Dans une autre perspective, l'article [23] propose une architecture légère fonctionnant en 2D uniquement pour explorer un milieu inconnu en se basant sur des méthodes de navigation visuelle.

2.3 Estimation de l'état et perception de l'environnement

De façon générale, ils comprennent l'obtention de données sur le véhicule et son environnement ainsi que l'extraction d'informations utiles à partir de ces données.
Dans le contexte de la vision par ordinateur, la perception vise à extraire des éléments pertinents dans un flux d'images. L'estimation est le processus d'extraction d'informations sur des variables d'intérêt à partir de mesures bruitées et qui peuvent être liées à ces variables par des modèles mathématiques complexes.

Les drones ont besoin de précision et de faible temps de latence dans leur estimation d'état afin d'obtenir un vol stable et robuste. Cependant, en raison des contraintes de puissance et de charge utile des plates-formes aériennes, les algorithmes d'estimation d'état doivent fournir ces qualités sous les contraintes de calcul du matériel embarqué.

Dans cette section nous allons d'abord voir les différents capteurs employés pour

accomplir l'estimation de l'état et la perception de l'environnement, ensuite nous verrons les différentes techniques d'accomplir cette tâche à l'aide de ces capteurs.

2.3.1 Capteurs utilisés

Les techniques d'estimation de l'état des robots mobiles se basent sur l'utilisation de différents types de capteurs, notamment les capteurs qui donnent des informations sur la distance par rapport à l'environnement (capteurs lasers, capteurs ultrasoniques), l'orientation et la vitesse selon les 3 axes d'un référentiel cartésien 3D (centrale inertielle) ou encore des caméras qui permettent de collecter plusieurs informations à la fois.

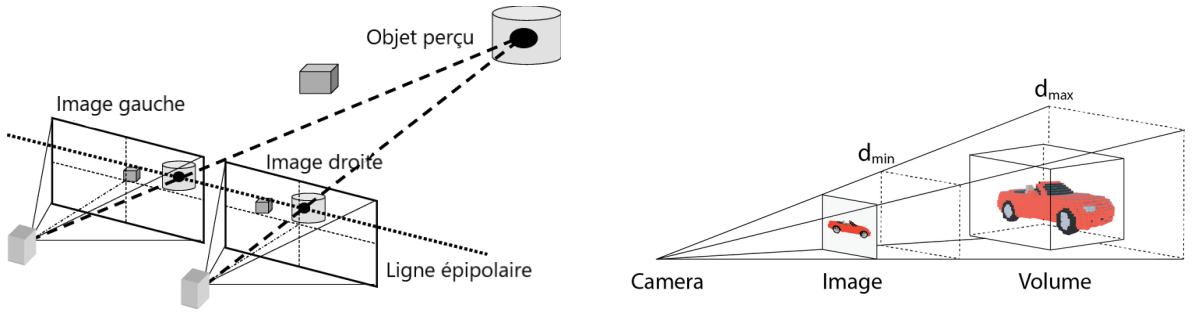
1. **Caméras**
   Ces caméras peuvent être classées en deux catégories selon le modèle mathématique qui décrit la transformation entre le repère de la caméra et les points observés dans la scène :

   (a) **Caméras en perspective :** cette classe regroupe l'ensemble des caméras obéissant au modèle mathématique d'une caméra à sténopé idéale qui suit la projection en perspective. Cependant, elle se divise en deux types de vision :

      i. **Vision monoculaire :** qui utilise une seule caméra à sténopé pour la perception ; l'information rapportée consiste en la projection des objets observés sur le plan 2D de la caméra.

      ii. **Vision stéréoscopique :** appelée aussi stéréo-vision. Elle consiste en l'utilisation de deux caméras pour reproduire la vision binoculaire des humains, en introduisant une donnée supplémentaire qui est la profondeur des objets perçus.

   La figure 2.1 illustre les deux types de vision ainsi que la différence de perception entre les deux.

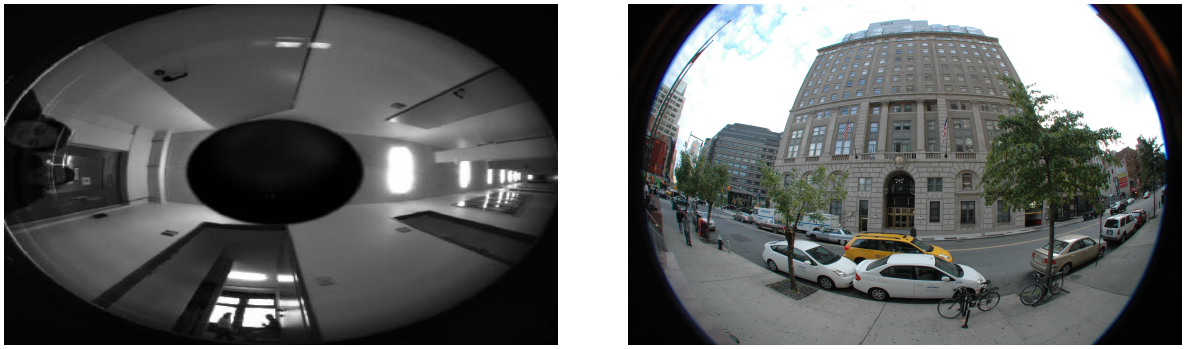


(a) Vision stéréoscopique. *(b) Vision monoculaire.*

FIGURE 2.1 – *Caméras obéissant au modèle de projection en perspective (pinhole).*

   (b) **Caméras omnidirectionnelles :** cette classe regroupe les caméras qui ont un champ de vision large, comme ceux utilisant l'objectif "*fisheye*" [24] avec un angle de vue allant jusqu'à plus de 180°, ou bien les caméras 360°utilisant des formes paraboliques de miroirs pour observer la lumière à partir de tous les sens sur un plan horizontal.

Ce type de caméras est largement utilisé dans les systèmes de navigation autonomes [25].
Dans la figure 2.2 nous pouvons voir les images obtenues des deux types de caméras omnidirectionnelles.



(a) Modèle 360° *(b) Image obtenue utilisant un objectif fisheye*

FIGURE 2.2 – *Caméras ayant un angle de vue large.*

(c) **Caméras événementielles :** un troisième type de caméras devient de plus en plus utilisé dans la navigation visuelle qui est la caméra événementielle. Il est caractérisé par sa capacité à détecter uniquement les changements de luminosité au niveau de chaque pixel, en d'autres termes il permet de capturer les mouvements uniquement ce qui permet d'éliminer le problème du flou cinétique tout en diminuant la consommation d'énergie et en augmentant considérablement la résolution temporelle. Une comparaison entre le rendu d'une caméra classique et une caméra événementielle est illustrée dans la figure 2.3.
Ce type de caméras non-conventionnelles bio-inspirés des grenouilles qui détectent le mouvements des mouches avant de les capturer, présente des résultats très prometteurs dans le domaine de la vision par ordinateur à condition de gérer la grande quantité de données générées [26, 27].

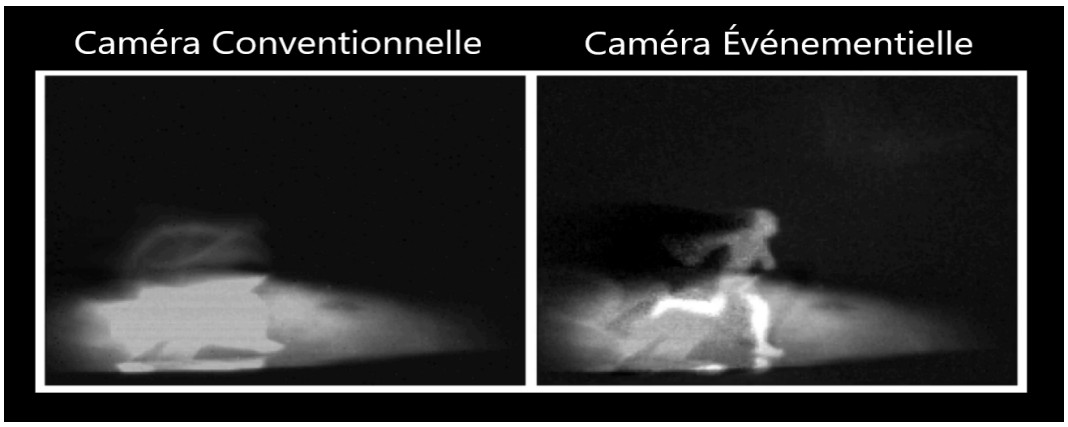


FIGURE 2.3 – *Comparaison entre une caméra événementielle et une caméra classique.*

2. **Centrale inertielle (IMU)**
   La centrale inertielle (communément appelée IMU : *Inertial Measurement Unit*) est un équipement de navigation qui comporte en général trois accéléromètres et trois gyroscopes, pour mesurer les mouvements relatifs par rapport à chaque axe d'un repère orthonormé à trois dimensions comme le montre la figure 2.4.
   Le rôle des accéléromètres est de mesurer les accélérations linéaires pour permettre l'estimation de la vitesse linéaire. Le gyroscope quant à lui donne l'orientation selon les axes de rotation comme indiqué sur la figure suivante :

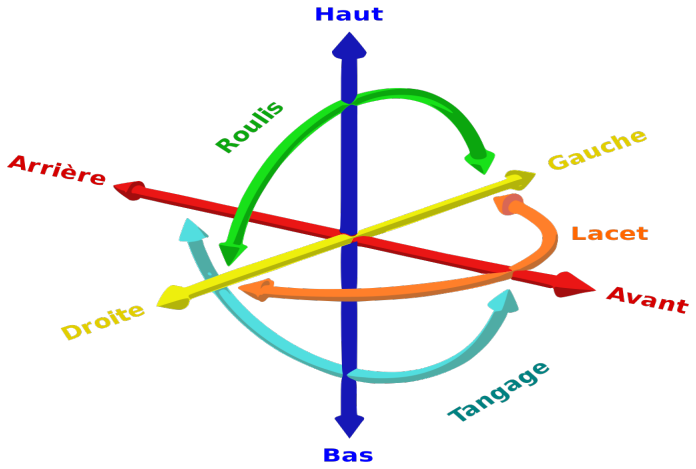


FIGURE 2.4 – *Angles de rotation.*

   Les centrales inertielles sont largement utilisées dans l'aviation civile et militaire, pour leur précision qui dépasse de loin les systèmes de positionnement par satellites et leur indépendance par rapport aux éléments extérieurs et aux systèmes de brouillage.

2.3.2 Localisation et cartographie visuelles simultanées

Le but de l'algorithme de localisation et de cartographie simultanées (SLAM : *Simultaneous localization and mapping*) en général, et du V-SLAM (*Visual-SLAM*) en particulier, est d'obtenir une estimation globale et cohérente de la trajectoire. Cela implique de garder une trace de la cartographie de l'environnement pour voir quand le robot revient à une zone déjà visitée [28] (c'est ce qu'on appelle la fermeture de boucle. Lorsqu'une fermeture de boucle est détectée, cette information est utilisée pour réduire la dérive de la carte et du trajet de la caméra).

La Figure 2.5 montre un véhicule muni d'une caméra se déplaçant dans l'environnement et capturant les caractéristiques inconnues de l'image.

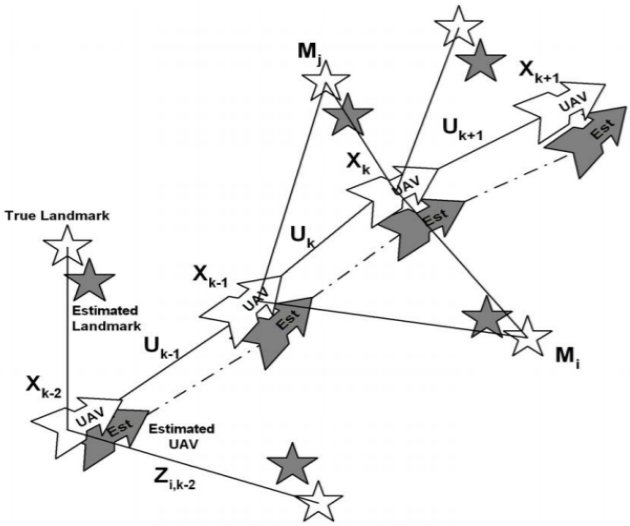


FIGURE 2.5 – *Localisation et Cartographie Simultanées.*

Avec :

- X_k l'état du véhicule à l'instant k.
- U_k le vecteur de commande du véhicule pour passer de $k-1$ à k.
- M_i vecteur d'emplacement de la $i^{\text{ème}}$ caractéristique d'image (l'hypothèse est que cette caractéristique est statique).
- Z_{ik} à l'instant k est la $i^{\text{ème}}$ image observée à partir de la caméra du véhicule.

Les étapes du SLAM :

1. Supposons que la position initiale du véhicule X_{k-1} est le point de départ avec certaines caractéristiques préexistantes de l'image sur la carte avec une grande incertitude de la position du véhicule.
2. Lorsque le véhicule commence à se déplacer de X_{k-1} à X_k, le modèle de mouvement fournit de nouvelles estimations de sa nouvelle position et l'incertitude de son emplacement augmente.
3. Pendant le déplacement du véhicule, ajouter de nouvelles caractéristiques M_i à la carte et mettre à jour la mesure des caractéristiques existantes.
4. Calculer la différence entre l'observation prévue et l'observation réelle.
5. Mettre à jour l'estimation de l'état actuel du véhicule ainsi que la carte.

MonoSLAM [29] est le premier système SLAM de vision monoculaire en temps réel. Il a été conçu pour étendre le filtre de Kalman dans la mise à jour continue de la carte estimée, en suivant des points de caractéristiques très clairsemés. Le LSD-SLAM (SLAM monoculaire direct à grande échelle) [30] est un algorithme basé sur des caractéristiques et des méthodes directes. Il applique la méthode directe au SLAM monoculaire semi-dense, qui peut réaliser une reconstruction de scène semi-dense sur un CPU. ORB-SLAM2 [31] proposent un système SLAM inspiré de LSD-SLAM et qui a pour innovation l'utilisation des caractéristiques ORB [32], et peut être appliqué à toutes les scènes en temps réel. L'algorithme est divisé en quatre modules : suivi, construction, déplacement et détection en boucle fermée. Avec l'avènement de la nouvelle caméra événementielle, de nombreuses études SLAM basées sur ce type de caméras ont vu le jour ces dernières années [33, 34]. Cependant, la caméra événementielle est coûteuse et possède une faible résolution spatiale, ce qui limite les performances de l'application.

Au bout de plusieurs décennies de développement, les méthodes de SLAM 2D avec scanner laser sont considérées comme matures. Les algorithmes modernes comme Hector SLAM [35] sont suffisamment robustes pour les applications sur un drone. En contrepartie, le SLAM visuel sur drone reste un problème ouvert, principalement parce que les algorithmes sont fragiles au mouvement agile du drone et à l'incertitude inconnue de l'environnement. Des algorithmes récents, dont ORB-SLAM2 et LSD-SLAM, sont des candidats potentiels qui peuvent être appliqués aux drones [36].

2.3.3 Odométrie visuelle

Dans la robotique en général, l'odométrie repose sur la mesure individuelle des déplacements relatifs à chaque pas pour reconstituer le mouvement global du robot. En partant d'une position initiale connue et en intégrant les déplacements mesurés, on peut ainsi calculer à chaque instant la position courante du véhicule.

L'odométrie visuelle (VO : *Visual odometry*) est une procédure d'estimation incrémentale de la position et de l'orientation du véhicule de navigation en analysant la disparité que le mouvement induit sur la séquence d'images capturées par une ou plusieurs caméras embarquées [37].

Cette technique calcule la position actuelle du véhicule en mettant en correspondance de façon incrémentale les points caractéristiques repérés sur les différents images acquises lors de la navigation. La figure 2.6 illustre le déplacement d'un système doté d'une caméra stéréoscopique. Les référentiels représentent les positions du système à chaque pas où une image est prise $(k-1, k, k+1)$. $T_{k,k-1}$ représente la matrice de transformation entre les positions aux moments respectives $k-1$ et k (equation 2.1) qui est estimée en analysant le déplacement des points perçus à travers les différents pas. C_k quant à lui représente la transformation par rapport à un repère initial pris comme référence. C_k la positon du système, est obtenue en concaténant tout les petits déplacements depuis moment de départ jusqu'au moment k.

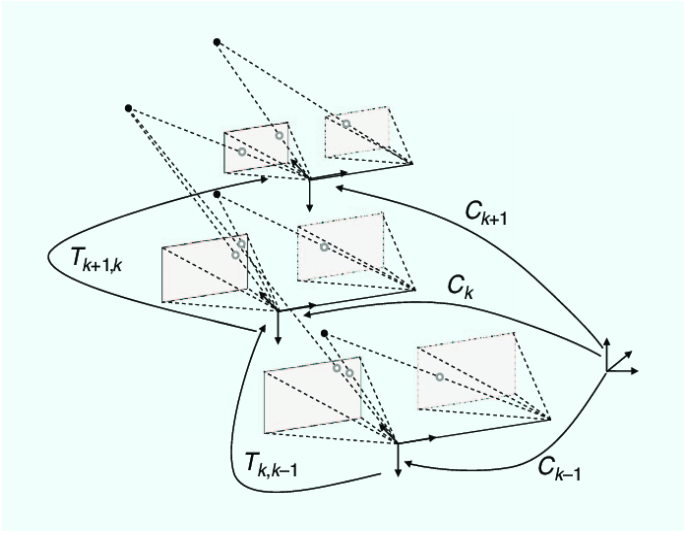


FIGURE 2.6 – *Illustration de l'odométrie visuelle.*

La matrice de transformation sur chaque pas est représentée donc comme suit :

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

Où, $R_{k,k-1}$ représente la matrice de rotation et $t_{k,k-1}$ représente le vecteur de translation. La pose (position et orientation) de la caméra $C_{0:n} = [C_0 ... C_n]$ est constituée par la concaténation des transformations de la caméra à partir de l'instant $k = 0$. La pose actuelle C_n contient donc toutes les transformations de la caméra $T_k, k = 1, ..., n$:

$$C_n = C_{n-1} T_n \tag{2.2}$$

L'objectif principal de cette technique est donc d'estimer la matrice de transformation T_k pour chaque pas.
L'algorithme rapide d'odométrie visuelle semi-directe (Semi-direct Visual Odometry : SVO) [38] est un exemple d'algorithme VO de pointe. Il est conçu pour être rapide et robuste alliant les avantages des techniques dites *directes* et *indirectes* de mise en correspondance des images (voire section 3.6). Il a été développé d'abord pour les caméras orientées vers le bas sur des drones miniatures [39], puis il a été généralisé pour une utilisation globale [1].

Un algorithme utilisant une caméra événementielle a été proposé par [40]. En plus de son efficacité, il est caractérisé par sa vitesse de traitement qui le rend apte à des utilisations de temps réel.

2.3.4 Estimation visuelle inertielle

Les solutions classiques se basant sur le flux optique d'images [41] sont sujettes à la dérive d'échelle et à une ambiguïté de l'échelle, ce qui engendre la nécessité d'une mesure supplémentaire pour y remédier. À travers une fusion avec les capteurs de distance (Lidar, Sonar) pour corriger l'échelle ou les informations d'une unité de mesure inertielle (IMU) pour créer un système de navigation visuelle inertielle (VI) [42].

L'article [43] présente une architecture pour estimer et suivre la pose d'une caméra relativement à une carte 3D de l'environnement. L'architecture suppose une connaissance au préalable d'un nuage de point qui représente l'environnement, le système navigant utilise une technique de *visual-inertial SLAM* pour optimiser la trajectoire de la caméra.

Les auteurs de [44] proposent un algorithme d'estimation de la position d'un drone en utilisant une approche qui prend en compte plusieurs hypothèses. Ces hypothèses sont générées à partir du flux d'images de la caméra en utilisant un algorithme de regroupement [45], et seront fusionné avec une mesure de confidence d'une centrale inertielle IMU via un filtre de Kalman pour générer une simple approximation d'une densité de probabilité.

2.3.5 Localisation visuelle par apprentissage profond

Inspirées par la performance exceptionnelle des réseaux de neurones convolutifs (CNN) dans une variété de tâches dans divers domaines et dans le but d'éliminer

1. Une démo de l'algorithme est disponible sur : https://www.youtube.com/watch?v=hR8uq1RTUfA

l'ingénierie manuelle des algorithmes pour la sélection de caractéristiques, les architectures CNN qui estiment directement la pose métrique en 6 degrés de liberté ont récemment été explorées [46] [47].

Dans [48] est introduite une méthode *End-to-End* pour l'apprentissage de l'odométrie visuelle à travers les données de l'image, basé sur l'apprentissage profond elle extrait le mouvement, la profondeur de la scène ainsi que l'information de l'odométrie à partir des images stéréo brutes. L'article [49] utilise un réseau de neurones récurrent (RNN : *Récurrent neural network*) pour former une estimation de la pose actuelle en prenant en compte les anciennes poses et les données de l'IMU, elle est combiné ensuite avec la sortie d'un système de localisation visuel utilisant un filtre de Kalman linéaire pour une estimation finale robuste, éliminant ainsi la nécessité d'un filtrage statistique pour résoudre le problème de la fusion de données.

Les auteurs de [50] ont proposé une architecture CNN multitâche entrainable de bout en bout (end-to-end) pour la localisation visuelle en 6 degrés de liberté et l'estimation d'odométrie à partir d'images monoculaires ultérieures, et a été l'un des premiers à combler l'écart de performance entre les méthodes de localisation locales basées sur les caractéristiques et celles basées sur un CNN.

Cependant, malgré leur capacité à gérer des conditions perceptuelles difficiles et à gérer efficacement de grands environnements, ils sont toujours incapables d'égaler les performances des méthodes de localisation basées sur les indices caractéristiques (features) les plus récentes. Ceci est dû en partie à leur incapacité à modéliser en interne les contraintes structurelles 3D de l'environnement tout en apprenant à partir d'une seule image monoculaire.

2.4 Techniques d'exploration

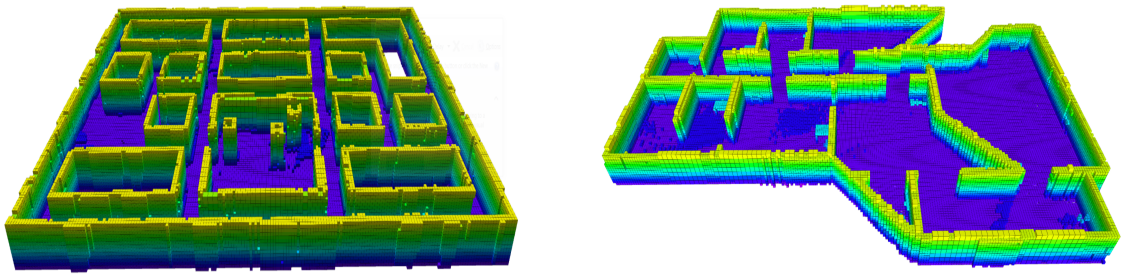


FIGURE 2.7 – *Résultats des environnements explorés avec l'algorithme proposé dans [15].*

Théoriquement, un robot ne peut apercevoir l'intégralité de l'environnement à partir d'un seul endroit. L'exploration consiste donc en la détection de l'espace libre ainsi que le choix de la prochaine cible à atteindre sur une zone déjà explorée partiellement [15]. Elle traite entre-autres le problème de la meilleure prochaine vue (NBV : *Next Best Vue*) qui a été étudiée pendant plusieurs décennies pour la reconstruction des objets en 3D à partir des scans de profondeurs [51, 52].

De plus l'exploration comprend aussi la navigation dans un environnement inconnu au préalable en recueillant le maximum d'informations utiles possibles.

Les techniques d'exploration essaient donc de satisfaire deux critères principaux :

1. Assurer une qualité et une cohérence géométrique de la carte de l'environnement exploré (influencer la trajectoire pour avoir une meilleure localisation et cartographie en diminuant l'erreur sur les points caractéristiques détectés dans les images).
2. Garantir un temps d'exploration minimale en balayant toutes les scènes possibles de l'environnement (couverture rapide de l'environnement exploré).

La figure 2.7 présente les résultats d'exploration d'une zone méconnue en utilisant l'algorithme proposé dans [15].

2.4.1 Techniques basées sur les frontières

Cette technique est une alternative aux techniques NBV. L'environement est discrétisé en une grille d'occupation 2D ou 3D (figure 4.2 page 68) où chaque case est étiquetée comme libre, occupé ou inconnue. Les cases qui désignent les frontières sont les cases libres qui sont adjacentes aux cases inconnues.
Elle se focalise sur l'acquisition du maximum d'informations à partir de l'environnement en un minimum de temps possible, en supposant que la position du système naviguant (robot, drone, sous-marins, etc.) soit déjà connue.

Les fondements de cette technique ont été pris à partir d'une approche proposée par [53], et qui consiste en la sélection du plus court chemin vers la frontière la plus proche dans la perspective de franchir la région méconnu la plus proche.
Soit t^{NF} (NF : *Next Frontier*) la cellule sélectionnée comme cible, elle obéit à l'équation :

$$t^{NF} = \arg\min_{a \in F} L(a) \tag{2.3}$$

où $L(a)$ représente la longueur du chemin (à minimiser) qui mène vers la cellule $a \in R^3$ appartenant à l'ensemble F et qui représente les points de frontières susceptibles d'être sélectionnées.
En pratique l'algorithme de Dijkstra est utilisé pour l'estimation du chemin le plus court et le choix de la cible.

On retrouve plusieurs travaux dans la littérature récente qui se basent sur l'approche de Yamauchi. Un critère d'utilité lors la sélection des cibles a été intégré dans [54] en rajoutant le terme suivant dans la fonction coût :

$$U(a) = \frac{\text{Unex}\,(a, R_s)}{\pi R_s^2} \tag{2.4}$$

où $\text{Unex}\,(a, R_s)$ est le nombre de cellules non-explorées dans la zone de porté du capteur à partir de la cellule a, R_s étant la porté maximale du capteur exprimée en unité de cellules. Cette technique considère donc la cible utile comme étant une cible qui offre le plus de chemins possibles une fois atteinte.

Une technique destinée pour les drones utilisant la grille d'occupation OctoMap [55] a été proposé dans [56]. Avant de sélectionner la prochaine frontière, un remblaiement est utilisé pour rejeter toutes les frontières ne pouvant être atteinte, ce qui augmente considérablement la vitesse de la technique.

De plus, une approche différente dans la représentation de l'environnement a été adoptée dans [57]. Seul l'espace occupé est représenté au lieu d'avoir les cellules libres et inconnues. Les particules d'environnement sont prélevées dans l'espace libre connu. Ils seront ensuite utilisées pour simuler un gaz contenu dans l'espace occupé connu, et les frontières sont identifiées comme étant les endroits où le gaz se dilate.

Une autre approche revisitant le problème du "*next-best-view*" a été proposé dans [58]. Les trajectoires possibles sont analysées à l'aide de l'algorithme RRT [2] sur un graphe aléatoire calculé en ligne.
Le critère du choix de la meilleure branche se manifeste par la quantité d'espace non cartographiée pouvant être explorée sur cette dernière. Seul le premier bord des branches est exécuté à chaque étape de la planification.

2.4.2 Méthodes intégrées

Certaines techniques se concentrent sur l'optimisation de la carte de l'environnement afin de garantir une précision dans la représentation de la carte. En effet, la représentation de l'environnement présente toujours une certaine incertitude qui peut résulter en une carte géométrique non utile pour les fins de navigation. Le travail effectué dans [43] est un exemple des techniques de navigation qui dépendent fortement d'une carte de l'environnement précise.

Une approche utilisant SLAM pour l'estimation de l'état est proposé dans [60, 61]. Elle consiste à intégrer une mesure d'incertitude (sous forme d'un critère d'utilité lié au cibles candidates) sur la position du robot dans le prochain pas, ainsi que sur les positions des indices caractéristiques qui sont utilisés pour la création de la carte. Ces incertitudes vont donc influencer sur le choix de la prochaine cible à explorer.

Une autre technique met en importance la fermeture de boucle du SLAM pour ajuster la représentation de la carte et la localisation durant l'exploration, ceci est accomplit en explorant l'environnement sur plusieurs trajets répétés favorisant à chaque fois le retour aux zones déjà explorées ou à la position de départ [62, 63].

L'adoption des techniques d'apprentissage machine a été proposé dans [64, 65]. Dans ce cas la planification de la trajectoire est subdivisée en deux parties : la sélection des *waypoints* appropriés au recueil d'informations, et la détermination des actions pour atteindre l'ensemble des *waypoints* en minimisant l'incertitude postérieure du SLAM. Utilisant le principe du *reinforcement learning*, l'espace des états englobe la position actuelle du robot, la carte et les lieux de détection ; l'espace des actions contient les paramètres qui résultent en une trajectoire lisse ; la fonction de récompense est une mesure d'incertitude postérieure. Le résultat étant donc une politique de contrôle qui

2. *Rapidly exploring random trees* proposé dans [59]

est apprise pour chaque distribution d'état, qui une fois entrainé, peut être utilisé en temps réel.

2.4.3 Méthodes collaboratives

La tâche de l'exploration et de la reconstruction de la carte peut être partagée à travers plusieurs robots en même temps, soit d'une façon centralisée ou bien distribuée indépendamment sur les robots (navigation collaborative ou coopérative). De plus, les robots navigant peuvent se servir de la carte assemblée pour la navigation.
Les méthodes collaboratives incluent à la fois les méthodes classiques qui se focalisent sur le temps d'exploration, et les méthodes intégrés qui mettent en évidence la construction de la carte. Des exemples sur ces techniques sont présentés dans [66, 67, 68].

2.5 Détection des objets dans une image

FIGURE 2.8 – *Cadrage et segmentation des classes détectées sur une image.*

Le développement des algorithmes d'apprentissage machine en général et des réseaux convolutifs (*CNN*) en particuliers à permis de voir une ascendance remarquable dans l'efficacité atteinte par les algorithmes de classifications d'images, et ce depuis la compétition ImageNet 2012 [69] où l'algorithme AlexNet [70] à permis d'obtenir pour la première fois une meilleure efficacité que les algorithmes de classification classiques.

La détection des objets est l'un des sujets les plus étudiés dans les domaines de la vision par machine et de l'intelligence artificielle. Il s'agit d'identifier et de localiser des classes prédéfinies tels que les personnes, voitures, bus, panneaux ,etc. lorsque ces dernières sont présentes dans l'image.
Cela peut être réalisé en dessinant un cadre (*bounding boxes*) autour des classes cibles spécifiques (ou bien de segmenter les différents classes détectées dans l'image), comme illustré dans la figure 2.8.

2.5.1 Classification des images

La classification est un problème fondamental de la vision par ordinateur qui désigne la catégorisation des images en classes selon le contenu, en d'autres termes elle consiste à attribuer des étiquettes aux images en fonction de leur type (classes). Les architectures de classification d'images constituent l'élément principal dans une architecture d'un détecteur d'objets.

Dans notre cas on traite le cas des réseaux de neurones convolutifs comme classificateurs d'images qui sont un type de réseaux de neurones qui peuvent extraire plusieurs caractéristiques à travers les différents couches de l'architecture.

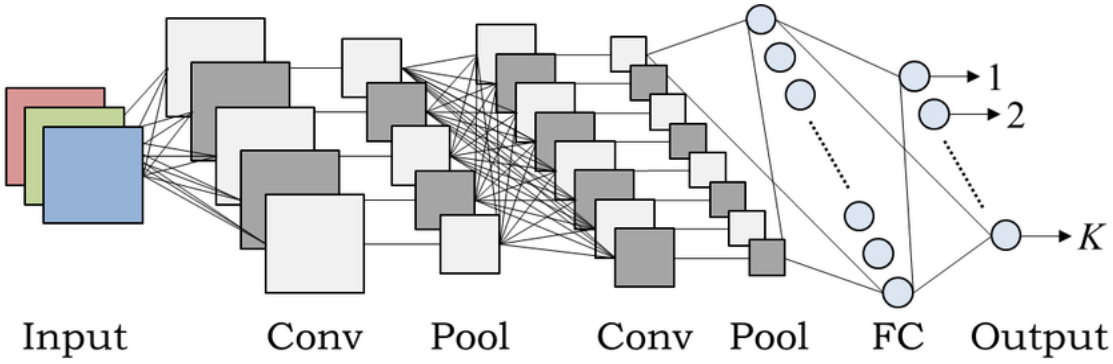


FIGURE 2.9 – *Architecture d'un CNN classique.*

La figure 2.9 montre une illustration d'un réseau de neurones convolutifs, plus de détail sur la constitution de ces réseaux est donné en annexe A.

2.5.2 Types de détecteurs d'objets

Les algorithmes de détection d'objets actuels se divisent principalement en deux catégories (figure 2.10) :

- **Détecteurs en deux étapes (*Two-stages*)** [3] : se basent sur une approche classique qui consiste à parcourir l'image à la recherche d'objets potentiels et ainsi déterminer des zones d'intérêts, ces zones vont passer ensuite à travers le classificateur d'objet pour déterminer sa nature. Ceci est illustré sur la figure 2.10, où ce type de détecteur utilise la *prédiction éparse* à partir de la carte de caractéristiques (sortie du bloc *Neck*) en deux étapes (proposition de régions puis classification).
- **Détecteurs en une seule étape (*One-stage*)** [4] : appelés aussi *single shot detectors* ils établissent la détection des objets et la classification en une seule étape directement à partir de la carte des caractéristiques à l'aide du bloc de la prédiction dense.

3. Pour plus de détails sur ce type de détecteurs : Medium: Region based detectors (fast-RCNN, RFCN)
4. Pour plus de détails sur ce type de détecteurs : Medium: Single shots detectors (SSD, YOLO, FPN)

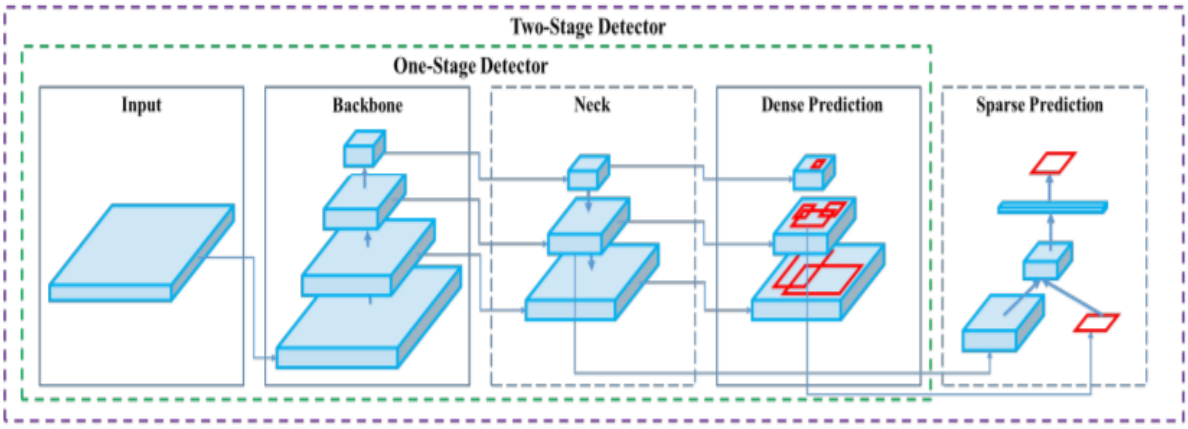


FIGURE 2.10 – *Architecture d'un détecteur d'objets basé sur un réseau de neurones convolutifs.*

Dans ce qui suit, nous allons énumérer quelques détecteurs populaires et récents du type *single shot detectors*, en tenant compte de trois critères principaux : efficacité, rapidité et ressources de calcul (possibilité d'exécution dans un système embarqué).

2.5.3 Critère de mesure d'efficacité des algorithmes

Le critère d'efficacité dans le cas d'un détecteur d'objet se traduit par l'habilitée de algorithme à produire le maximum de prédictions justes. On observe dans la littérature deux sous-critères d'évaluation de l'efficacité des algorithmes, à savoir la précision et le rappel.

Précision : un algorithme précis cherche à obtenir une fiabilité maximale, il peut dans une image donnée ne pas détecter l'objet en question mais s'il détecte un objet il ne doit pas se tromper. On peut mesurer la précision à travers la formule :

$$Précision = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ positifs}$$

Rappel : mesure l'habilitée de l'algorithme à trouver les Vrais positifs, l'algorithme peut faire de fausses prédictions mais il assure de trouver le maximum de vrais positifs. La formule du rappel est donc :

$$Rappel = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ négatifs}$$

Le critère principal utilisé dans l'évaluation est l'*Average Precision* (**AP**), on l'obtient en calculant l'intégrale suivante [71] :

$$\mathrm{AP} = \int_0^1 p(r)dr \tag{2.5}$$

Où $p(r)$ représente la précision en fonction du rappel.

Soit un exemple d'utilisation[5], où cinq images de pommes sont présentes dans un ensemble de données. Après chaque prédiction effectuée par un algorithme de test on note le résultat obtenu jusqu'à détecter toutes les cinq images, pour obtenir le tableau suivant :

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

TABLE 2.1 – *Tableau représentant les prédictions effectuées à l'aide d'un algorithme de test.*

On note par exemple dans la quatrième ligne un rappel qui vaut $2/5 = 0.4\%$ (deux des pommes ont été détectés uniquement). La précision quant à elle vaut $2/4 = 0.5\%$ (deux prédictions justes sur 4 prédictions faites).

Ceci nous donne la représentation graphique suivante qui illustre l'évolution de la précision en fonction du rappel lors du parcours d'un ensemble de données :

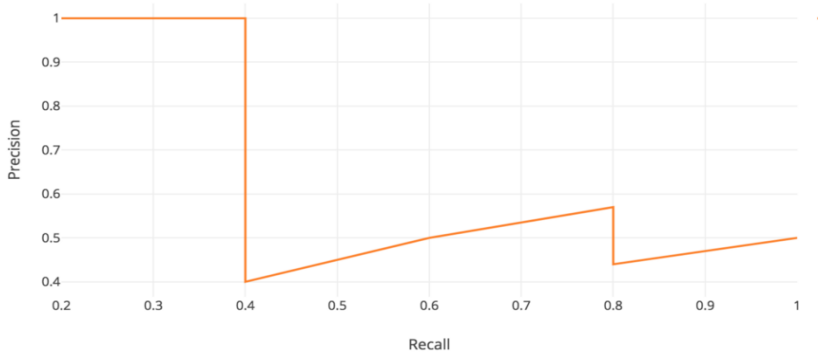


FIGURE 2.11 – *Précision en fonction du rappel*[6].

Afin de calculer l'**AP** un lissage de courbe est effectué selon la formule suivante :

$$p_{\text{interp}}(r) = \max_{\widehat{r} \geqslant r} p(\widetilde{r}) \tag{2.6}$$

On obtient donc le graphe suivant :

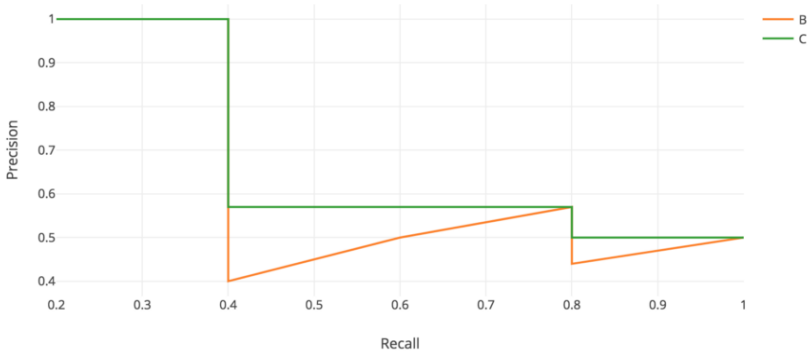


FIGURE 2.12 – *Précision en fonction du rappel (courbe lissée)*[6].

Par la suite on emploie une technique pour estimer l'air sous la nouvelle courbe pour trouver l'*Average précision*. Cette mesure est adopté pour évaluer les performances d'un détecteur d'objet sur une seule classe uniquement, pour les détecteurs d'objets qui peuvent détecter plusieurs classes dans une image une moyenne sur les **AP**s de toutes les classes détectables est calculée et elle est notée **mAP**(*Mean Average Precision*).

2.5.4 Algorithme *YOLO (You Only Look Once)*

Yolo est un algorithme qui se base sur l'architecture Darknet[7] en appliquant le principe du *single shot*. Il est caractérisé par sa grande vitesse de calcul qui le rend utilisable dans les scénarios temps-réel. Ceci veut dire qu'il n'a pas besoin d'une étape de détermination des régions d'intérêts dans d'image mais divise l'image en cellules, où chaque cellule prédit k cadres d'objets avec un score de confiance pour chacun reflétant la probabilité de présence d'un objet et la précision de la prédiction [72].
Ce qui veut dire que l'algorithme *YOLO* reçoit en entrée une image et donne en sortie un ensemble d'éléments, chaque élément contient les attributs suivants :

- Les coordonnés du centre de l'objet (x, y).
- La largeur et la hauteur respectives du cadre (w, h).
- la probabilité sur la classe détectée (score de confiance).

La deuxième version de l'algorithme [73] réalise une efficacité comparable à [74] en gardant le caractère rapide de l'algorithme. Les principales améliorations apportées sont :

- Détection les objets sous différents configurations ou proportions, grâce à l'entrainement multi-échelle.
- L'amélioration de la précision et de la localisation (rappel) sur les petits objets jusqu'au même niveau que le SSD300, comme conséquence de l'augmentation du nombre de cellules testées (13*13 contre 8*8 dans YOLOv1).

La table suivante montre les améliorations apportées dans chaque étape avant d'obtenir la version 2 de YOLO. Les résultats reflètent l'efficacité (par la métrique mAP [*mean average precision*]) de la détection sur le dataset VOC2007[8].

6. Prise à partir de l'exemple : Medium : Mean average precision for object detection
7. Développée en langage C par Joseph Redmond : https://pjreddie.com/darknet/
8. Disponible sur : http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html

	YOLO	1.1	1.2	1.3	1.4	1.5	1.6	1.7	YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP (%)	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

TABLE 2.2 – *Chemin de YOLO vers YOLOv2 : Tableau comparatif montrant l'évolution de l'efficacité des versions de l'algorithme en utilisant la métrique d'efficacité mAP (mean Average Precision) en %.*

En utilisant la nouvelle architecture performante Darknet-53, la troisième version [75] quant à elle a apporté une efficacité comparable aux détecteurs à deux étapes basés sur le parcours des régions d'intérêt de l'image. Presque au niveau d'efficacité de RetinaNet [76], YOLOv3 se voit 4 fois plus rapide par rapport aux techniques de l'état de l'art.
Ceci dit, la précision sur les classes détectées et le rappel des objets ont considérablement augmentées sur cette version, ainsi que la détection des objets sous différentes conditions grâce à la pyramide des caractéristiques introduites dans cette version [9].

La dernière version se voit d'être à ce jour l'algorithme de détection le plus rapide ce qui favorise son utilisation dans les applications qui nécessitent l'aspect temps-réel [77]. L'architecture est basée sur la dernière ossature CSPDarknet [78] (*Backbone*), une combinaison entre les deux architectures SPP [79] et PAN [80] comme réseau de caractéristiques, en gardant le même réseau de prédiction des classes que YOLOv3 sur le dernier bloc de la détection.

9. Plus de détails sur : Medium : detection with YoloV2

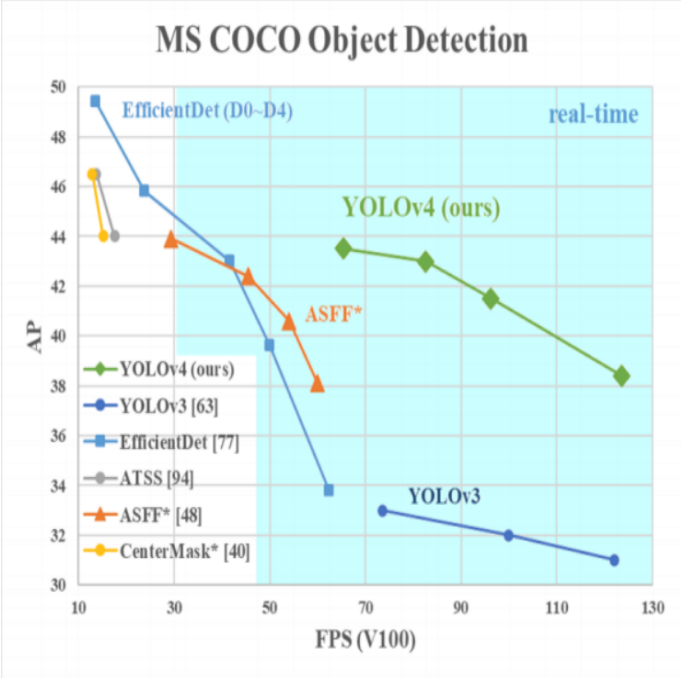


FIGURE 2.13 – *Compromis vitesse/efficacité des différents algorithmes de détection des objets de l'état de l'art sur l'ensemble de données COCO [77].*

La figure 2.13 montre que la vitesse a été améliorée par 12% et l'efficacité par 10% par rapport à la version précédente de YOLO, tout en diminuant la taille de l'architecture. Les résultats suivants ont été obtenus à l'aide de l'ensemble de données COCO, sur une carte graphique Nvidia TESLA V100.

2.5.5 Architecture *RetinaNet*

Développé par le groupe Facebook AI Research (FAIR) en 2018, l'algorithme consiste en une architecture de détection en une seule étape. L'algorithme est constitué de deux parties essentielles :

1. Une ossature qui comportent un CNN (ResNet50/ResNet101) et un réseau de caractéristiques FPN [81] pour améliorer la détection des objets sous différents proportions.
2. Un réseau classe/boîte qui reçoit en entrée une image des caractéristiques. Il est constitué de deux sous-réseaux dont la tâche est de cadrer les objets et prédire leur classes.

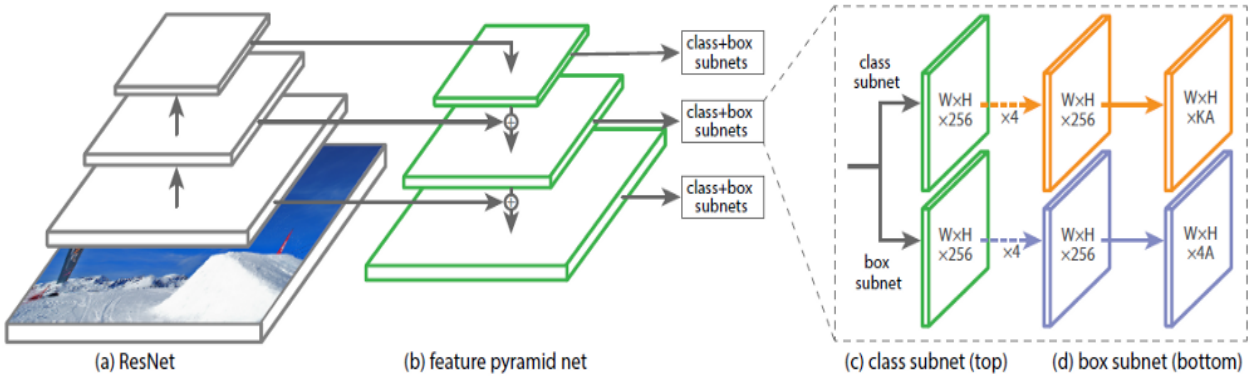


FIGURE 2.14 – *Architecture RetinaNet [76].*

L'algorithme de RetinaNet a été longtemps pris comme référence en terme de performance sur l'efficacité. Une version améliorée plus légère et plus rapide avec moins de temps d'inférence a été proposée dans [82]. Elle est Obtenue en utilisant des techniques d'élagage sur le réseau de neurone de l'ossature, et une quantification des numéros en passant de la représentation en virgule flottante sur 32 bits à une représentation d'entier sur 8 bits.

2.5.6 Algorithme *EfficientDet*

EfficientDet est un algorithme de détection des objets bâtit autour du réseau de classification EfficientNet [10]. Proposé par Google Research Brain Team dans la conférence CVPR 2020 [83] [11], l'algorithme se base sur l'architecture EfficientNet [84] et en intégrant un réseau de caractéristiques bidirectionnel (BiFPN). L'algorithme EfficientDet fournit une efficacité semblable à celle de l'état de l'art en étant 9 fois plus léger (en terme de nombre de paramètres) ce qui lui permet d'utiliser moins de ressources de calcul [83].

L'architecture de l'algorithme se divise en trois parties :

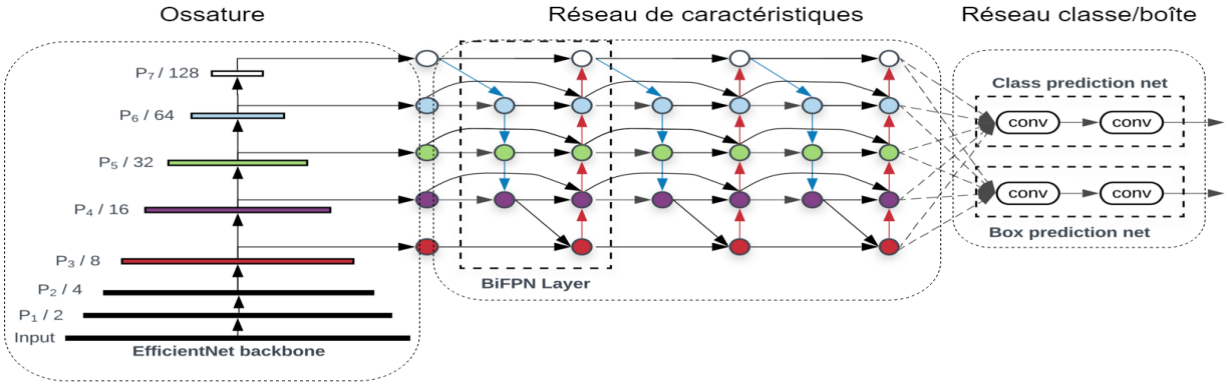


FIGURE 2.15 – *Architecture EfficientDet [83].*

10. Réseau de neurones convolutifs d'ossature, plus de détails seront donnés plus loin dans cette section sur son architecture.

11. Conference on Computer Vision and Pattern Recognition organisé en virtuel en juin 2020 par IEEE Computer Society

1. **Ossature (EfficientNet) :** il est caractérisé par une configuration légère qui n'est pas gourmande en terme de ressources de calcul. En remplaçant ResNet-50 dans RetinaNet par EfficientNet-B6 on obtient 3% de plus en efficacité en diminuant les calculs de 20%.
2. **Réseau de caractéristiques :** la contribution principale d'EfficientDet se trouve dans la configuration proposée pour le réseau de caractéristiques BiFPN, qui introduit la fusion des caractéristiques multi-niveaux à partir d'autres réseaux de caractéristiques connues comme FPN [81] (utilisé sur YOLOv3), PANet [80] et NAS-FPN [85] qui permettent aux informations de circuler dans les directions *top-down* et *bottom-up*. Cette couche permet d'améliorer l'efficacité de 4% en diminuant le coût de calcul par 50% [12]
3. **Réseau classe/boite :** partie responsable de la prédiction des classes et de leurs dimensions à partir des zones caractéristiques dans l'image (déterminé par le réseau de caractéristique).
   Il s'agit de deux architectures de réseaux de neurones convolutifs *CNN*, un qui prédit le type de classe et un autre prédisant les dimensions de l'objet dans l'image (cadrage).

Il a été démontré dans [83] que l'algorithme EfficientDet peut être modifié pour effectuer la segmentation des objets qui donne des résultats comparables à l'efficacité de l'état de l'art en effectuant 10 fois moins de calculs.

Architecture d'ossature *EfficientNet*

Une architecture a été proposée dans l'article [84] répondant au problème de la mise en échelle d'un réseau de neurones convolutifs pour améliorer les performances et l'adapter aux différents critères d'utilisation, en jouant principalement sur trois paramètres qui sont :

1. La profondeur du réseau.
2. La largeur de l'entrée.
3. La résolution des caractéristiques détectées de l'image.

La figure 2.16 illustre ce concept :

12. Plus de détails sur le réseau BiFPN sont donnés en annexe B.

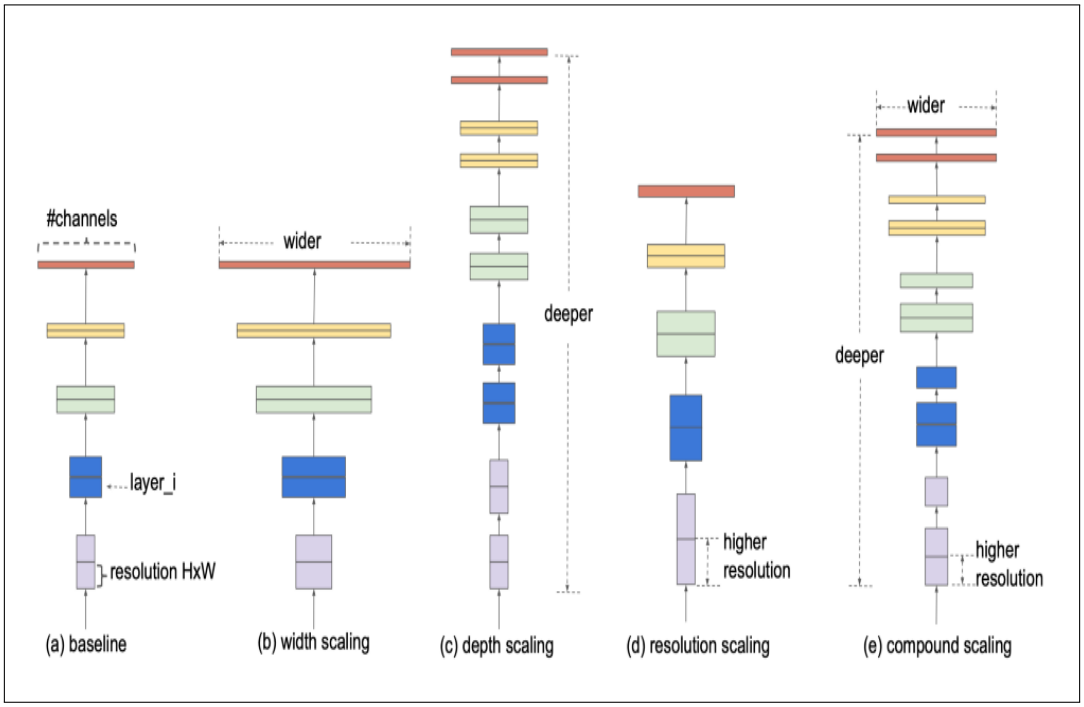


FIGURE 2.16 – *Différents paramètres de mise en échelle d'un réseau de neurones.*

En effet, cette approche a été motivée par l'effet positif qu'a l'augmentation de ces paramètres individuellement sur la précision de l'algorithme, ceci va engendrer en contre partie un besoin plus grand en ressources de calcul et introduit un problème de gradient disparaissant (*vanishing gradient problem* en anglais) qui empêche les poids des neurones d'évoluer lorsque ces derniers sont très petit limitant ainsi l'effet de la rétropropagation. Rajoutée à cela la dépendance qui existe entre ces paramètres et qui a été démontrée empiriquement.

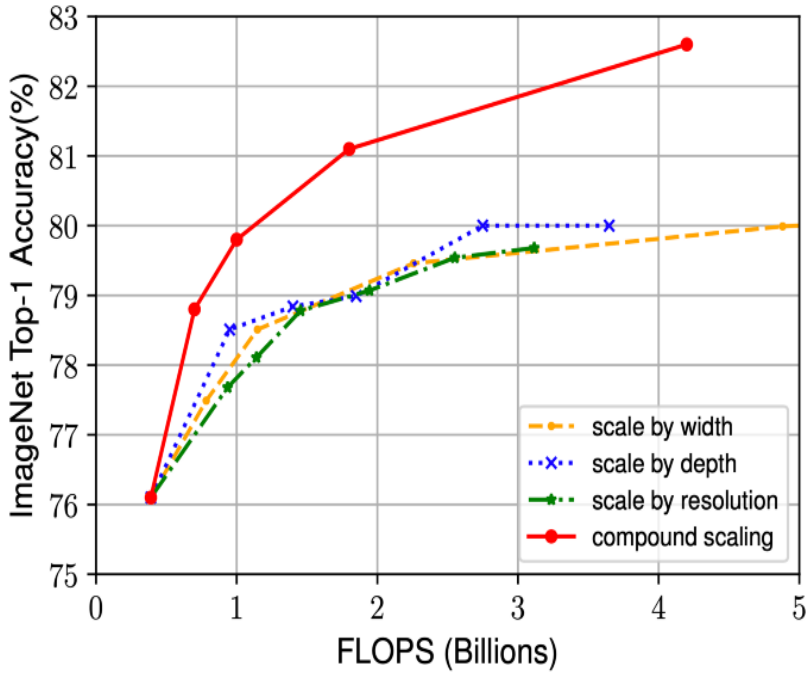


FIGURE 2.17 – *Précision en fonction des opérations effectué (ressources de calcul), en essayant d'améliorer l'architecture EfficientNet-B0 de [84] par différents méthodes.*

Comme le montre cette figure la mise en échelle composée (*compund scaling*) proposée dans [84], obtient de meilleures résultats en précision que les techniques de modification unidimensionnels.

la solution proposée à ce problème est une approche intuitive, elle consiste en l'augmentation de chaque paramètre à l'aide d'un facteur constant. Le modèle proposé en est le suivant :

$$\begin{aligned} depth : d &= \alpha^{\phi} \\ width : w &= \beta^{\phi} \\ resolution : r &= \gamma^{\phi} \\ t.q \quad \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\ \alpha \geq 1, \beta \geq 1, \gamma &\geq 1 \end{aligned} \tag{2.7}$$

ϕ est le paramètre commun qui va agir sur la proportion du réseau en contrôlant la quantité de ressources allouées au modèle, (α, β, γ) définissent quant à eux la proportion de l'allocation de ces ressources entre la largeur du réseau, sa profondeur et la résolution respectivement.

La structure de base (EfficientNet-B0) a été obtenue grâce à un algorithme d'apprentissage renforcé appelé NAS (Neural Architecture Search) à partir d'une fonction objectif, la configuration obtenue est la suivante :

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

TABLE 2.3 – *Composition d'EfficientNet-B0.*

A partir de cette architecture, en fixant $\phi = 1$ dans le modèle 2.7 les paramètres (α, β, γ) ont été déterminé et les résultats présentés sont : $\alpha = 1.2$, $\beta = 1.1$ et $\gamma = 1.15$. En fixant (α, β, γ) et en augmentant ϕ la famille EfficientNet à été obtenue de B0 à B7. Les résultats de performance sont montrés dans la figure 2.18

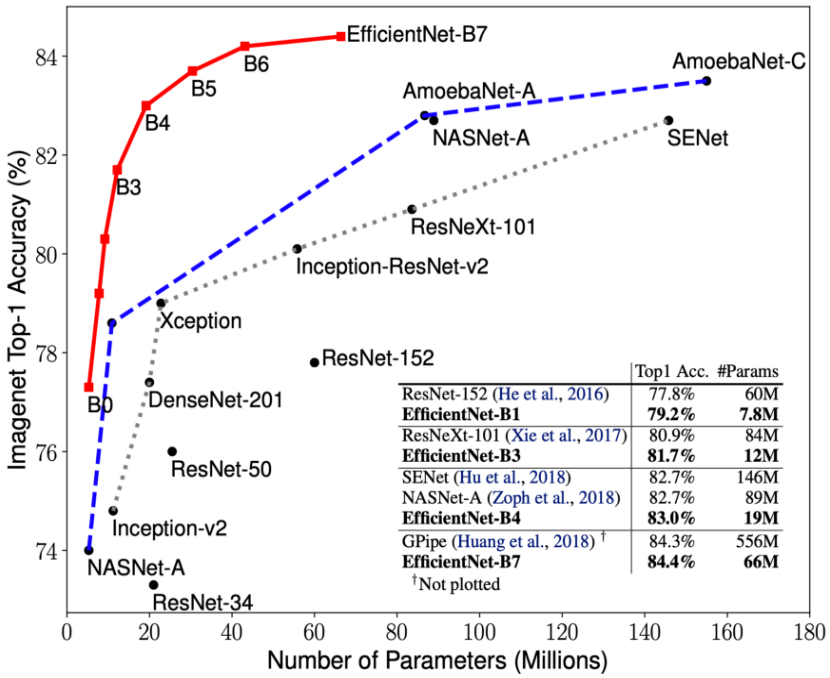


FIGURE 2.18 – *Comparaison de l'efficacité des classificateurs d'images, les courbes représente la précision en fonction du nombre de paramètres de l'architectur (résultats obtenues sur la base de données ImageNet)* **??***).*

Il est à noter que l'algorithme peut détecter 1000 classes, qui sont le nombre de classes dans la base de données d'ImageNet.

Mise en échelle

La famille des détecteurs EfficientDet D0-D7 suit la même intuition que [84] dans la mise en échelle globale de l'architecture, il s'agit d'utiliser un seul paramètre ϕ pour mettre en échelle toute l'architecture.

1. **Réseau d'ossature :** la famille B0-B6 d'EfficientNet est utilisée.
2. **Réseau BiFPN :** le même modèle adopté dans [84] a été reconduit pour la largeur du réseau(*channels*), quant à la profondeur (nombre de fois que le réseau est répété) elle augmente obéissant à un modèle linéaire. Les formules utilisées sont donc :
$$W_{bifpn} = 64 \cdot \left(1.35^{\phi}\right), \quad D_{bifpn} = 3 + \phi$$
3. **Réseau classe/cadre :** la largeur est maintenue la même que celle du réseau BiFPN et la profondeur s'accroît linéairement :
$$D_{box} = D_{\text{class}} = 3 + \lfloor \phi/3 \rfloor$$
4. **Résolution d'entrée :** la résolution de l'entrée est subdivisble par $2^7 = 128$ car les niveaux 3-7 sont utilisés, le modèle de mise en échelle est donc :
$$R_{\text{input}} = 512 + \phi \cdot 128$$

On obtient donc le tableau 2.4 qui résume les paramètres de chaque version de la famille des détecteurs EfficientDet. Il est à remarquer que D7 est le même réseau que D6 mais avec une augmentation dans la résolution d'entrée de l'architecture.

	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	2	3
D1 ($\phi = 1$)	640	B1	88	3	3
D2 ($\phi = 2$)	768	B2	112	4	3
D3 ($\phi = 3$)	896	B3	160	5	4
D4 ($\phi = 4$)	1024	B4	224	6	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1408	B6	384	8	5
D7	1536	B6	384	8	5

TABLE 2.4 – *Différentes configurations de la famille des détecteurs d'objets EfficientDet avec ϕ le facteur d'échelle.*

2.6 Conclusion

La contribution des drones lors des missions de recherche et sauvetage est un domaine de recherche déjà convoité. On y trouve de plus en plus de techniques qui intègrent l'apprentissage automatique et la vision par ordinateur.

Dans ce chapitre nous avons présenté l'état de l'art en matière de l'assistance dans les opérations de recherche et sauvetage, ainsi que dans les axes de la vision par ordinateur et l'intelligence artificielle qui peuvent avoir une incidence sur ce domaine.

Grâce à l'état de l'art dressé, nous proposons de combiner les techniques de perception visuelle et d'exploration comme approche pour palier les problèmes de la navigation et de la cartographie en n'ayant aucune information au préalable sur l'environnement cible. De plus un algorithme intelligent basé sur les réseaux de neurones convolutifs est intégré pour extraire d'autres informations utiles à partir du milieu exploré.

Chapitre 3

Estimation de l'état et perception de l'environnement

3.1 Introduction

L'estimation de l'état d'un système (position et orientation) est l'une des problématiques fondamentales de la robotique en général et des systèmes de navigation en particulier. La perception de l'environnement est l'outil essentiel qui va nous permettre de répondre à la problématique d'intervention des drones après les désastres naturels, en fournissant deux informations essentielles qui sont la localisation et une représentation de l'espace exploré par le drone.

L'instrumentation omniprésente transportée par tout véhicule aérien est une unité de mesure inertielle (IMU : *Inertial Measurement Unit*) pour mesurer l'orientation souvent, complétée par une forme de mesure de la hauteur, acoustique, infrarouge, barométrique ou laser.

Dans le cas des catastrophes naturelles on se retrouve fréquemment dans des environnements encombrés où les signaux GPS ne sont pas disponibles. De ce fait, ces dernières années avec le développement des algorithmes de vision par ordinateur, un des choix repose sur l'utilisation d'une caméra pour estimer l'attitude (position et orientation) du drone grâce à l'odométrie visuelle (VO : Visual Odometry), la localisation et la cartographie visuelles simultanées (V-SLAM : Visual-Simultaneous localization and mapping) ou bien les techniques qui allient à la fois vision et mesures inertielles [1].

Nous présenterons dans ce chapitre la solution que nous avons retenue, à savoir un système de perception visuelle qui rapporte en temps réel la position du drone et une modélisation de l'espace qu'il perçoit à travers l'algorithme *Semi-Direct Visual Odometry (SVO)* [2] [39]. En utilisant la configuration matérielle suivante :

- Une caméra monoculaire montée sur le front du drone pour rapporter le flux d'images en temps réel.

1. Certains systèmes de capture de mouvement tels que VICON, ou Optitrack, sont utilisés pour des applications industrielles (réalité virtuelle, jeux de simulations, etc.).

2. L'algorithme nous a été fournit par les auteurs de [38]. Une ancienne version limitée est disponible en Open Source sur : `https://github.com/uzh-rpg/rpg_svo`.

- Une centrale inertielle (IMU : *Inertial Measurement Unit*) pour mesurer l'orientation du drone et l'accélération linéaire suivant tous les axes de mouvement.

Cette partie constitue la brique principale de la contribution apportée par ce travail, vu que c'est le processus responsable de l'interaction de notre système avec le monde extérieur. De ce fait, l'approche entreprise dans cette partie doit répondre à deux critères posés dans la problématique, à savoir la fiabilité de l'information et l'efficacité de positionnement.

3.2 Modèle cinématique du système

Les hypothèses de départ vont nous permettre de bien décrire la navigation ainsi que la perception. Nous considérons un repère terrestre W avec sa base orthonormale $\{\mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W\}$, et un autre repère B dont l'origine coïncide avec le centre de gravité du drone, sa propre base orthonormale est noté $\{\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B\}$. La figure 3.1 illustre ce système de coordonnées :

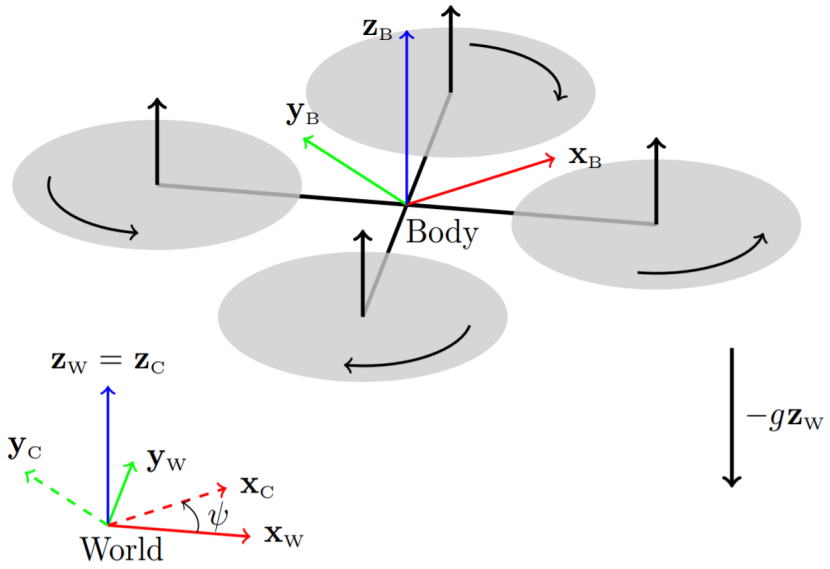


FIGURE 3.1 – *Systèmes de repères*[3].

On suppose dans ce qui suit que le repère W représente la position initiale du drone lors de sa navigation vers une orientation donnée, ce qui induit que tout déplacement du drone, ainsi que la représentation de l'environnement seront estimés et représentés par rapport à ce repère initial.

Matrice de transformation et de rotation

Nous noterons la matrice T_{WB} de transformation qui permet le passage du repère B au repère W. Elle est composée d'une rotation $\mathbf{R}_{WB}$ et d'une translation $\mathbf{t}_{WB}$.
Pour convertir un point $\mathbf{p}$ défini par rapport au repère B noté $\mathbf{p}_B$ à un point $\mathbf{p}_W$ définie

3. Par soucis de simplification, nous supposons dans la suite du document que le drone en question est un quadrirotor. En réalité notre architecture ne dépend pas du type du drone mais uniquement du type des capteurs embarqués.

par rapport à W, on utilise l'équation suivante :

$$\mathbf{p}_W = T_{WB} \cdot \mathbf{p}_B \tag{3.1}$$

Où T_{WB} s'écrit sous la forme :

$$T_{WB} = \begin{bmatrix} \mathbf{R}_{WB} & \mathbf{t}_{WB} \\ 0 & 1 \end{bmatrix} \tag{3.2}$$

Nous utilisons la norme *rpy* pour représenter les rotations (Figure 2.4 page 27) en suivant la convention $Z - Y - X$. La rotation entre W et B est définie donc par l'ordre suivant :

1. Angle de lacet ψ autour de $\mathbf{z}_B$ (yaw).
2. Angle de tangage θ autour du nouveau $\mathbf{y}_B$ (pitch).
3. Angle de lacet ϕ autour du nouveau $\mathbf{x}_B$ (roll).

On a aussi :

$$\mathbf{R}_{WB} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \tag{3.3}$$

Où,

$$\mathbf{R}_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{3.5}$$

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{3.6}$$

3.3 Centrale inertielle IMU

La récente prolifération de la technologie micro-électro-mécanique (MEMS : Micro Electro-Mechanical Systems) a conduit au développement d'une gamme d'unités de mesure inertielle optimisant à la fois le coût, le poids et les dimensions du capteur. Cela permet de combiner trois types de données différentes, à savoir :

- accélérations linéaires (accéléromètre),
- vitesses angulaires (gyroscope),
- Points cardinaux et orientation par rapport au repère terrestre grâce au magnétomètre.

La figure 3.2 montre la disposition des différentes mesures prises par la centrale inertielle par rapport au repère du système navigant.

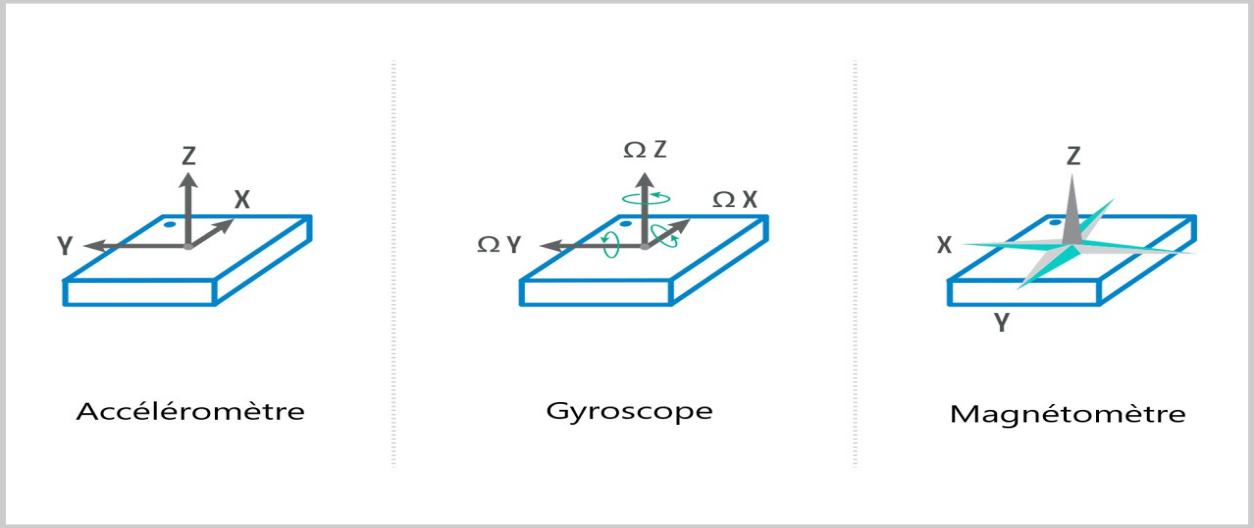


FIGURE 3.2 – *les différentes mesures de l'IMU suivant le référentiel du capteur.*

3.3.1 Accéléromètre

L'accéléromètre donne l'accélération du repère B par rapport au repère B exprimé dans W. Cette mesure est toujours prise par rapport à la mesure au repos. Elle doit être égale à la force gravitationnelle suivant l'axe z.
Soit $\dot{v}$ la mesure réelle de l'accélération du repère B par rapport à W, exprimée dans W ; g_0 est la mesure de référence exprimée dans le repère W.

Le modèle mathématique de la mesure est formulé par l'équation suivante :

$$\hat{a}_{imu} = (\dot{v} - g_0) + b_{imu} + n_{imu} \in \boldsymbol{R}^3 \tag{3.7}$$

où b_{imu} et n_{imu} représentent le biais constant (ou lentement variable) et le bruit de mesure (paragraphe 3.3.4).

3.3.2 Gyroscope

Le gyroscope mesure la vitesse angulaire du repère B par rapport au repère W exprimé dans le repère B. Son modèle de mesure est le suivant :

$$\hat{\omega}_g = \omega_g + b_g + n_g \in \boldsymbol{R}^3 \tag{3.8}$$

Avec b_g et n_g le biais et le bruit additif de la mesure.
Il permet aussi d'obtenir l'orientation avec précision en intégrant les vitesses angulaires obtenues. Ces mesures doivent d'abord être transformées pour être exprimées dans le repère W.
Cependant, ce résultat est entaché d'un grand bruit et ne peut être fiable, pour réussir à estimer l'orientation avec une précision, on utilise une fusion de données avec les données de l'accéléromètre (filtre complémentaire, filtre de Kalman, etc.).

3.3.3 Magnétomètre

La combinaison du gyroscope et de l'accéléromètre ne peut pas estimer avec précision l'angle de lacet car l'accéléromètre se base sur l'équilibre des forces, en gardant toujours

la résultante vers le bas nulle (ce qui veut dire qu'une rotation sur un plan horizontal qui influence uniquement sur l'angle de lacet ne sera pas captée par l'IMU). La valeur du *yaw* estimée sera donc peu précise.
La solution à ce problème consiste à équiper l'IMU avec un magnétomètre (compas) pour mesurer l'orientation du drone par rapport au repère terrestre.

3.3.4 Modèle du bruit

Les signaux de l'accéléromètre et du gyroscope sont toutefois soumis à des niveaux de bruits élevés ainsi qu'à des biais variant dans le temps, Ils doivent donc passer par une étape de traitement ou sont combinés les mesures des différents capteurs de l'IMU pour arriver à reconstruire l'estimation de l'orientation et les mesures des vitesses angulaires. Pour une seule mesure $w(t)$ le signal reçue est entaché de deux types de bruits, un bruit blanc additif $n(t)$ variant à une haute fréquence et un biais $b(t)$ variant à une basse fréquence, ce qui conduit à l'équation suivante pour modéliser cette mesure [86, 87] :

$$\hat{a}(t) = a(t) + n(t) + b(t) \tag{3.9}$$

Le même principe est reconduit indépendamment dans les trois axes de rotations et les deux capteurs (accéléromètre, gyroscope).

Bruit additif

Le bruit n(t) variant à une haute fréquence est modélisé par un bruit blanc gaussien de moyenne nulle et d'intensité σ. On aura donc :

$$E[n(t)] = 0, \quad E[n(t_1)n(t_2)] = \sigma^2\delta(t_1 - t_2) \tag{3.10}$$

En discrétisant le signal on aura :

$$n_d[k] = \sigma_{gd} w[k] \tag{3.11}$$

Avec

$$w[k] \sim \mathcal{N}(0,1), \quad \sigma_{gd} = \sigma_g \frac{1}{\sqrt{\Delta t}} \tag{3.12}$$

L'étape de discrétisation représente une étape préliminaire de traitement, équivalent à un passage par un filtre passe bas avec comme fréquence de coupure $f = \frac{1}{2\Delta t}$, où Δt représente le temps d'échantillonage.

Biais

Les variations lentes dans les mesures sont supposées comme *mouvement brownien* [88], et sont donc discrétisées à l'aide d'un *processus de Wiener*. Ce processus peut être exprimé au cours d'un temps discret par l'équation :

$$b_d[k] = b_d[k-1] + \sigma_{bgd} w[k] \tag{3.13}$$

Avec,

$$w[k] \sim \mathcal{N}(0,1) \quad \sigma_{bgd} = \sigma_{bg}\sqrt{\Delta t} \tag{3.14}$$

Le bruit de l'IMU est caractérisé donc par 5 paramètres (comme dans [89]) :
- Intensité du bruit blanc de l'accéléromètre et du gyroscope : σ_a (m/s^2) et σ_g (rad/s).
- Intensité du biais de l'accéléromètre et du gyroscope : σ_{ba} (m/s^3) et σ_{bg} (rad/s^2).
- Temps d'échantillonnage Δt.

3.4 Notions de la vision par ordinateur

Dans ce qui suit nous présentons le cas des appareils photographiques obéissant au modèle mathématique de projection en perspective (paragraphe 2.3.1). Le plan de l'image est un plan $2D$ où est projeté tout point présent dans la scène perçue.

Modèle de la caméra

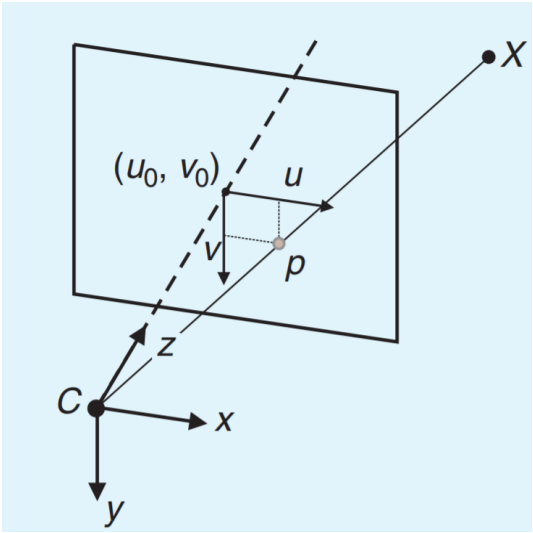


FIGURE 3.3 – *Modèle géométrique d'une caméra*

Afin de projeter la scène perçue sur le plan de la caméra, le modèle de projection de la caméra va fournir une transformation entre les points $3D$ et leur représentation sur le plan de l'image. La projection en perspective est le modèle le plus utilisé sur les caméras à sténopé (*Pinhole camera* en Anglais), où l'image est formée par l'intersection des rayons lumineux des objets à travers le centre de la lentille (centre de projection), avec le plan focal (Figure 3.3).

Soit $X = [x, y, z]^T$ un point dans la scène du champs de vision de la caméra exprimé dans le repère C de la figure 3.3 et $p = [u, v]^T$ sa projection sur le plan image mesurée en pixels.
La correspondance entre le monde 3-D et l'image 2-D est donnée par l'équation suivante :

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.15}$$

où λ est le facteur d'échelle, α_u et α_v sont les distances focales et u_0, v_0 les coordonnées images du centre de projection. Ces paramètres sont appelés *paramètres intrinsèques* de la

caméra.
Quand l'angle de vue d'une caméra est supérieure à 45° les effets de la distorsion radiale peuvent devenir visibles. Ils peuvent être modélisés cependant par un polynôme de second -(ou plus)- degré.

Il est impossible de retrouver la profondeur d'un point 3-D connaissant seulement sa projection dans un seul plan de l'image, ce qui rend la vision monoculaire incapable d'estimer la structure de la scène avec les proportions réels. Or, supposons que nous avons la projection sur deux caméras d'un même point X exprimé dans le repère terrestre, connaissant les paramètres intrinsèques $(\alpha_{u,i}, \alpha_{v,i}, u_{0,i}, v_{0,i})$, extrinsèques (rotation R_i et translation T_i) de ces deux caméras, ainsi que la transformation entre celles ci on peut écrire :

$$\lambda_1 \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = K_1 (R_1 X + T_1), \qquad \lambda_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = K_2 (R_2 X + T_2) \tag{3.16}$$

Il est alors possible de retrouver la position de X par rapport au repère de la caméra C. Ce processus est connu sous le nom de *Triangulation* (qui sera expliqué dans la section 3.4).

Calibration de la caméra

La calibration de la caméra est le processus qui nous permet de déterminer les paramètres intrinsèques et extrinsèques de la caméra et qui vont nous permettre par la suite d'établir le modèle mathématique de la caméra (équation 3.15).
Il est à noter que l'indice de profondeur n'est disponible que dans le cas des caméras stéréoscopiques.

Un autre type de paramètres est déterminé lors de la calibration, qui est le modèle de distorsion de l'image. En effet, chaque image présente une distorsion radiale et une autre tangentielle comme indiqué dans [90]. Ce modèle de distorsion va permettre de corriger les problèmes de distorsion qui sont présentés dans la figure 3.4 et reconstruire l'image en rapportant les formes réelles.

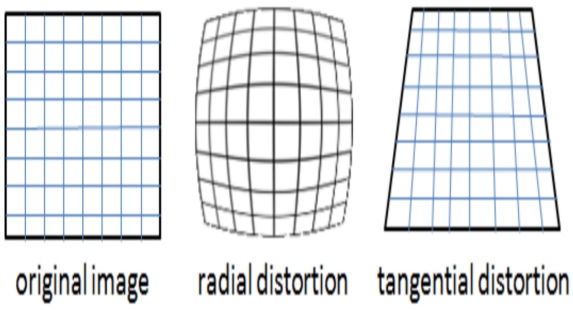


FIGURE 3.4 – *Types de distorsions dans une image.*

Un package ROS utilisant le modèle cité dans [90] est disponible pour déterminer les paramètres intrinsèques, extrinsèques, ainsi que les paramètres de distorsion d'une caméra [4]

4. Tutoriel de calibration des caméras : http://wiki.ros.org/.

Structure from Motion

Dans le domaine de la vision par ordinateur, la *Structure from motion* est utilisée pour déterminer la structure 3*D* d'un objet à partir des séquences d'images couplées à des signaux indiquant le mouvement de la caméra. Ce concept est bâtit sur la capacité des humains à déterminer la structure d'un objet ou d'une scène à partir d'un champ de déplacement.

L'adoption de cette technique à permis l'utilisation des systèmes de vision monoculaire dans toutes les applications de la vision par ordinateur qui nécessitent la connaissance de la profondeur de la scène (où la vision stéréscopique était exclusivement utilisée), notamment dans l'exploration des environnement inconnues [43], les véhicules autonome (pilote automatique de TESLA [5]), la topographie naturelle [91], etc. Comme illustré par la figure 3.5, l'estimation de la forme 3D de l'objet est estimé à partir de la séquence d'images qui comporte différents perspectives de l'objet. Ceci est accompli grâce à la connaissance du positionnement relatif entre les différents endroits où les images sont prises.

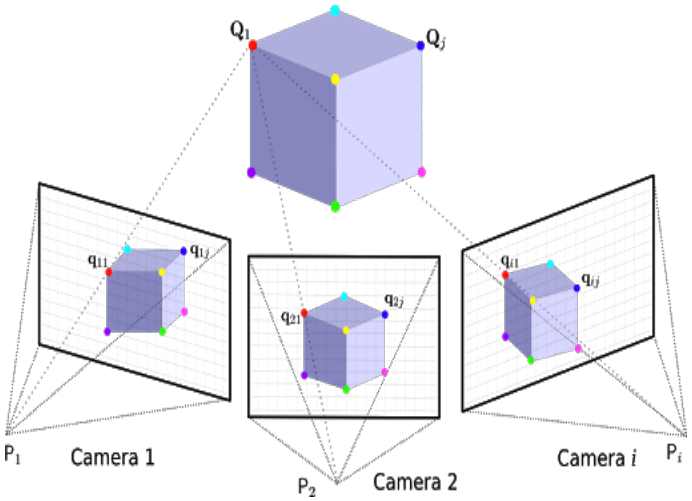


FIGURE 3.5 – *Exemple de SfM où la structure est déterminée après plusieurs prises de différents côtés de l'objet.*

Géométrie épiploaire

La géométrie épipolaire est appliquée dans la vision stéréoscopique en particulier pour la reconstruction des scènes 3*D* [92]. Elle se base sur le fait que pour deux caméras dans deux positions distinctes apercevant la même scène, certaines propriétés géométriques nous permettent d'établir des contraintes sur la projection de la scène dans les deux images.

5. Présentation du pilote automatique de TESLA 2019 : https://www.youtube.com/watch?v=HM23sjhtk4Q.

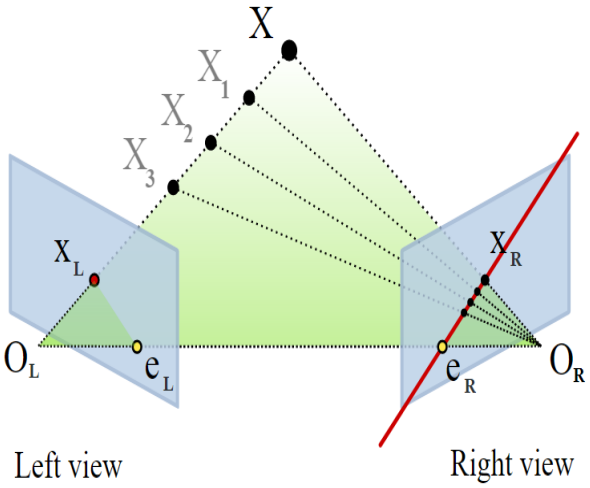


FIGURE 3.6 – *Modèle de la géométrie épipolaire.*

La figure 3.6 montre le principe de la géométrie épipolaire. La ligne rouge représente la ligne épipolaire et le triangle vert le plan épipolaire.

Soit un point X dans l'espace dont la projection est X_L dans la caméra de gauche et X_R dans la caméra de droite, O_L et O_R représentant les centres des deux caméras la ligne entre eux est nommée **ligne de référence**.
Le **plan épipolaire** est le plan défini par le point X et les centres des deux caméras. L'intersection entre le plan épipolaire et les plans de l'images sont appelés **lignes épipolaires**.
Ne connaissant pas la position 3D de X, on a sa projection sur l'image de gauche ainsi que la translation et la rotation relatifs entre les deux caméras ainsi que le modèle mathématique de chaque caméra. Avec ces informations uniquement on peut déterminer le plan épipolaire et les lignes épipolaires. Ceci dit, on saura que le point X_R sera situé automatiquement sur la ligne épipolaire de la vue de droite avant même d'avoir sa projection, ce qui nous donne une contrainte géométrique solide entre la paire d'images sans connaître au préalable la structure de la scène en 3D.

Triangulation

La triangulation consiste à déterminer la profondeur d'un point dans l'espace 3D, sachant sa projection sur au moins deux images dont les positions relatifs mutuelles sont connues.
La position du point par rapport à un référentiel terrestre pourra ensuite être déterminée si l'on connaît la transformation entre le référentiel 3D de la caméra dans l'une des positions et le référentiel terrestre.
Les points $3D$ sont déterminés à travers les raies générées par la reprojection à partir des images $2D$ en connaissant le modèle mathématique de la caméra. Dans les conditions parfaites, ces raies vont se croiser dans le point $3D$ souhaité (Figure 3.6).
À cause du bruit de la caméra, les erreurs dans le modèle de la caméra, l'erreur dans la calibration et l'incertitude sur la correspondance des indices caractéristiques dans les images ; les raies générées par la reprojection peuvent ne pas se croiser. Le point situé à une distance minimale entre ces raies est pris dans ce cas comme estimé.

Erreur de reprojection

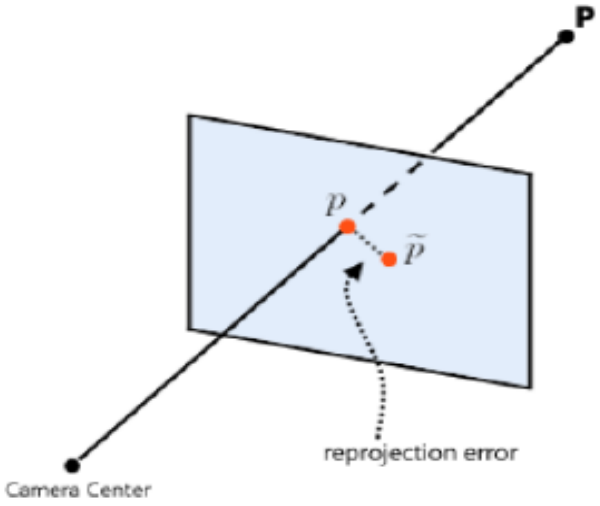


FIGURE 3.7 – *L'erreur de reprojection représentée par la distance entre* p *et* p̂.

L'erreur de reprojection est une distance géométrique entre la projection sur l'image (à travers le modèle de la caméra) d'un point $3D$ estimé au préalable et la projection réelle du point dans l'image même. En d'autres termes, c'est la distance entre un point mesuré et son estimation sur le plan de l'image. Elle fournit une mesure qualitative de la précision sur l'estimation d'un point 3D qui crée la projection réelle du point.

Soit un point $P \in \boldsymbol{R}^3$ dans le champ de vision de la caméra dont la profondeur est estimé au préalable, on note π la matrice projection qui permet de passer du point $3D$ à sa projection dans le plan $2D$ de la caméra à travers l'équation :

$$\hat{p} = \pi P \tag{3.17}$$

L'erreur de projection sur P est définie par la distance euclidienne $d(p, \hat{p})$ entre les points représentés par p et $\hat{p}$ comme illustré dans la figure 3.7.

Bundle adjustment

Le *bundle adjustment* consiste à affiner conjointement un ensemble d'estimations initiales des paramètres de la caméra et de la structure pour trouver l'ensemble de paramètres qui prédit le plus précisément les emplacements des points observés dans l'ensemble des images disponibles.
Supposons que nous avons n points $3D$ dans m *vues* et soit $\mathbf{x}_{ij}$ la projection du point i sur l'image j. Soit v_{ij} une variable binaire égale à 1 si le point i est présent dans l'image j et 0 ailleurs. Supposons que chaque caméra j est paramétrée par un vecteur a_j et chaque vecteur $3D$ i par un vecteur b_i. Le *bundle adjustment* minimise l'erreur totale de reprojection par rapport à tous les points 3D, en particulier :

$$\min_{\mathbf{a}_j, \mathbf{b}_i} \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij}\, d(\mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i),\ \mathbf{x}_{ij})^2 \tag{3.18}$$

Où $\mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i)$ est la projection prédite du point i sur l'image j et $d(\mathbf{x}, \mathbf{y})$ indique la distance euclidienne entre les points d'image représentés par les vecteurs $\mathbf{x}$ et $\mathbf{y}$.

3.5 Techniques d'estimation du déplacement dans l'odométrie visuelle

Dans les algorithme utilisant l'odométrie visuelle, il existe trois méthodes pour estimer le déplacement entre deux images I_{k-1} et I_k à partir des deux ensembles de points caractéristiques respectivement f_{k-1} et f_k [93], selon que ces points soient exprimés dans un repère 2D où 3D et on aura donc :

- **2D vers 2D :** Dans ce cas les deux ensembles f_{k-1} et f_k sont pris dans le repère 2D sur le plan de l'image.
- **3D vers 3D :** Dans ce cas f_{k-1} et f_k sont exprimés dans le repère 3D. Pour que cela soit possible l'information de la profondeur doit être connue pour chaque pas, ce qui veut dire qu'on aura besoin d'une vision stéréoscopique pour utiliser cette méthode. Cette méthode présente plus d'incertitude sur les points estimés puisqu'elle ne prends pas en compte les erreurs de reprojection comme le fait la troisième méthode.
- **3D vers 2D :** Les points de l'ensemble f_{k-1} sont des points 3D et les points f_k sont leur projection sur l'image I_k, dans le cas de la vision monoculaire on aura besoin de trois prises pour estimer le mouvement au lieu de deux, les points $3D$ doivent être estimé à partir des deux vues précédentes à savoir I_{k-1} et I_{k-2}, pour être ensuite comparé avec les projections sur le plan $2D$ de l'image à l'instant k.

La dernière méthode se voit d'être plus précise que les deux autres, puisqu'elle offre la possibilité de diminuer en même temps l'erreur de reprojection, ce qui va mener à une meilleure estimation du mouvement.

L'estimation du mouvement en utilisant la technique 3D vers 2D

Les méthodes d'estimation qui font la correspondance des points 3D vers les points 2D sont plus précise que les autres méthodes (3D-3D ou 2D-2D), puisqu'elle minimise l'erreur de reprojection. La transformation $\mathrm{T_k}$ sera estimée donc à travers la mise en correspondance entre X_{k-1} et p_k, où les points X_{k-1} sont obtenus soit grâce à une vision stéréoscopique ou triangulés à partir des points p_{k-2} et p_{k-1} dans le cas monoculaire. La formule générale est donc d'optimiser l'équation suivante :

$$\mathrm{T}_k^\star = \arg\min_{T_k} \sum_i \left\| p_k^i - \hat{p}_{k-1}^i \right\|^2 \tag{3.19}$$

Avec $\hat{p}_{k-1}^i$ la reprojection du point 3D X_{k-1}^i sur le plan de l'image I_k suivant la transformation $\mathrm{T_k}$. Ce problème est connue sous le nom *perspective from n points* (PnP)et il existe plusieurs algorithmes pour le résoudre [94].

3.6 Éléments de mise en correspondance entre les images

L'objectif de la mise en correspondance est de détecter les changements observés entre deux ou plusieurs images, permettant d'avoir une mesure de disparité qui peut être exploitée pour :

- Estimer le mouvement des objets dans l'image entre les différents prises.

- Aligner différents images pour la reconstruction des scènes [95].
- Estimer le mouvement de la caméra entre les différents prises [96].
- Recherche par similarité à travers plusieurs images [97, 98].

Il existe principalement deux méthodes de mise en correspondance entre images, qualifiées comme directe/indirecte selon l'utilisation des pixels de l'image.

3.6.1 Extraction des descripteurs de l'image (correspondance indirecte)

Cette technique agît sur des descripteurs des points caractéristiques extraits de l'image pour mettre en correspondance deux ou plusieurs images, Le critère de sélection de ces points caractéristiques varie selon le cas d'utilisation.
L'extraction de caractéristiques visuelles (ou *visual features extraction* en anglais) consiste en des transformations mathématiques calculées sur les pixels d'une image numérique, ce qui donne la possibilité d'exploiter certaines propriétés visuelles de l'image, utilisées ensuite pour la mise en correspondance qui va nous permettre d'inférer le déplacement de la caméra selon le déplacement de ces indices. Nous nous intéresserons aux descripteurs locaux qui sont mesurés autour des points d'intérêt.

Sélection et détection

Dans cette étape l'image est parcourue pour retrouver les points saillant qui seront reconnaissables dans d'autres images. Chaque point caractéristique est un motif d'image qui doit être différent des points qui l'entourent (coins, contour, blobs). Un bon détecteur de caractéristiques doit assurer à la fois la précision de la localisation, la répétabilité (l'habilitée à détecter le même indice dans d'autres vues), l'efficacité de calcul (en temps et en ressources) ainsi que la robustesse et l'invariance par rapport aux bruit, flou, rotation, changement d'échelle [99].

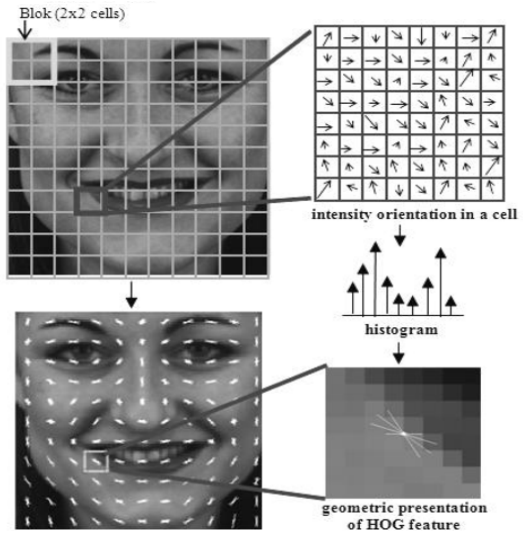


FIGURE *3.8 – Caractéristiques d'un visage.*

Afin de reconnaitre l'indice caractéristique, on doit lui attribuer un descripteur formé à partir des pixels qui entourent l'indice. L'un des descripteurs les plus utilisés est l'histogramme de gradient orienté qui a été adopté dans SIFT [100] et SURF [101] (Figure 3.8).

Mise en correspondance et suivi

L'étape de mise en correspondance consiste à prendre l'ensemble des descripteurs obtenus dans la première image et chercher leur correspondance dans l'image suivante. Ils sont comparés en utilisant des mesures de similarités entre les descripteurs (*corrélation croisée*, *somme des erreurs quadratiques*, distance euclidienne (SIFT).
Pour diminuer le temps de calcul des correspondances, une limitation dans la zone de recherche est mise en place, cette limitation est calculée à partir de l'estimation du mouvement de la caméra grâce à des capteurs additionnels comme l'IMU.

3.6.2 Correspondance des pixels (correspondance directe)

Les méthodes qui se basent sur les pixels cherchent à produire des disparités entre une image de référence et une deuxième image pour les pixels non occultés. Le critère d'appariement est basé sur une mesure de similarité de la fonction d'illumination locale [102], pour arriver à estimer le déplacement entre les deux images.
Ces techniques s'appuient sur l'hypothèse suivante : soit deux images prises dans des lieux proches, l'intensité d'un point physique donné reste inchangé entre les deux images. Ce concept introduit la contrainte d'illumination qui suppose que pour deux images $I(x,y)$ et $J(x,y)$:

$$J(x,y) = I(x + u(x,y), y + v(x,y)) \tag{3.20}$$

où (x,y) sont les coordonnées du pixel dans l'image, et (u,v) le déplacement du pixel entre les deux images. Pour un petit déplacement (u,v), en linéarisant I autour de (x,y), on obtient la contrainte d'illumination [95] :

$$I_x u + I_y v + I_t = 0 \tag{3.21}$$

avec la paire (I_x, I_y) représentant les dérivés spatiales de l'intensité de l'image et

$$I_t = I - J \tag{3.22}$$

Le but étant de trouver la paire (u,v), d'autres contraintes représentants le modèle globale du mouvement seront ajoutées. Ce modèle comporte deux ensembles de paramètres, un ensemble de paramètres globaux liés aux mouvement de la caméra et un autre ensemble de paramètres locaux (un pour chaque pixel) liés à la structure 3D. Un exemple de ce modèle est présenté dans [95] exprimé à travers l'ensemble d'équations :

$$\begin{aligned} u &= -xy\Omega_X + \left(1 + x^2\right)\Omega_Y - y\Omega_Z + \left(T_X - T_Z x\right)/Z \\ v &= -\left(1 + y^2\right)\Omega_X + xy\Omega_Y + x\Omega_Z + \left(T_Y - T_Z y\right)/Z \end{aligned} \tag{3.23}$$

Où $(\Omega_X, \Omega_Y, \Omega_Z)$ et (T_X, T_Y, T_Z) représentent respectivement les rotations et les translations de la caméra (paramètres globaux), et Z la profondeur de la scène pour chaque pixel (paramètre local).

3.6.3 Discussion sur les deux méthodes

Le but principal de cette section est de se familiariser avec les deux concepts fondamentaux de la mise en correspondance entre images. Les équations et les détails

d'implémentations quant à eux peuvent varier d'une technique à une autre [95, 99].

Le principal avantage de la méthode de correspondance directe c'est qu'elle permet d'exploiter l'information à partir des pixels directement sans passer par une étape d'association de l'information entre descripteurs et points détectés. Cependant, l'optimisation conjointe dans le cas des méthodes denses [6] entre la structure et le mouvement en temps réel reste un problème ouvert.

Un des inconvénients intuitifs des méthodes directes est le manque de garantie vis-à-vis de l'optimalité et de la consistance de l'estimation [103]; la disparité entre les images ne doit pas être importante ce qui introduit des erreurs d'estimation, vu la distance considérablement petite entre les images (section 3.7.1).

De plus, les méthodes directes se voient dépendre d'une grande puissance de calcul vu qu'elles nécessitent une représentation dense ou semi-dense de l'environnement, tandis que le coût de temps dominant dans les méthodes indirectes représente l'extraction des descripteurs pour les points saillants.

3.7 Algorithme d'Odométrie Visuelle Semi-directe SVO

L'algorithme SVO [38] s'appuie sur une technique de mise en correspondance hybride qui regroupe à la fois une méthode directe pour le suivi et la triangulation des pixels, et une méthode indirecte pour accomplir l'optimisation conjointe de la structure et du mouvement estimés. Il adopte le principe du *structure from motion* pour estimer la profondeur des points 3D et ainsi inférer la reconstruction de l'environnement.

6. Une méthode de mise en correspondance entre deux images est dite "dense" lorsque cette dernière emploie tous les pixels des images pour étudier la similarité.

3.7.1 Architecture

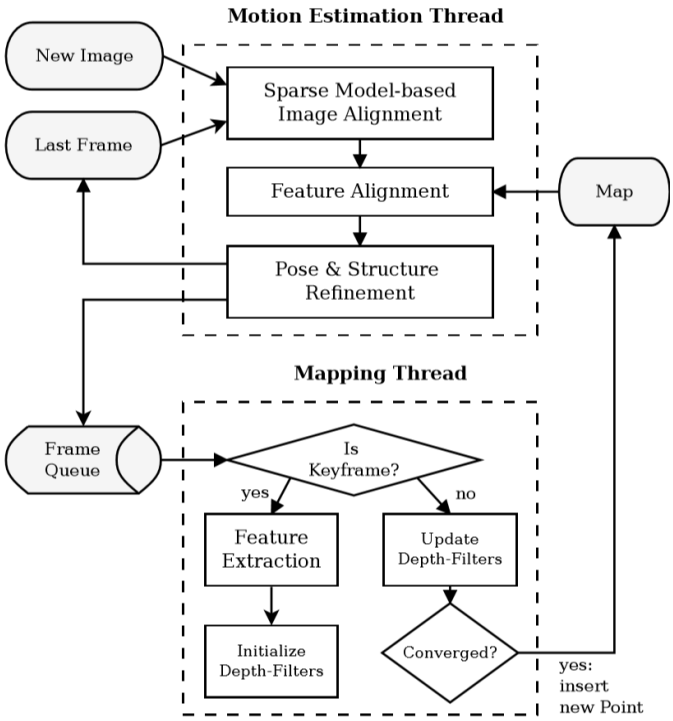


FIGURE *3.9 – Pipeline de l'architecture SVO.*

Comme illustré dans la figure 3.9, l'algorithme se divise en deux processus (*threads*) qui s'exécutent en parallèle pendant la navigation. Un pour estimer le déplacement de la caméra (odométrie visuelle) et un autre pour la construction de l'environnement (*structure from motion*).
L'approche semi-directe est implémentée pour estimer le mouvement dans le premier processus, qui se décompose en trois parties principales :

1. Un alignement éparse d'images pour estimer le déplacement par rapport à l'étape précédente, en minimisant la différence d'intensité entre les deux images et en tenant compte d'un ensemble de points caractéristiques qui correspondent aux projections dans les deux images des mêmes points 3D.
2. Une correspondance indirecte (à l'aide des descripteurs de l'image) avec la séquence la plus ancienne prise possible à partir de la carte (*Feature Alignment*).
3. Un affinement conjoint de la position et de la structure en tenant compte de plusieurs prises à travers un ajustement d'ensemble (*Bundle Adjustment*).

Dans le deuxième processus, un estimateur de profondeur probabiliste est initialisé pour chaque descripteur détecté (ce dernier va être associé à un point 3D estimé). Un nouveau estimateur est initialisé avec une grande incertitude dans la profondeur à chaque fois qu'un nouveau "*keyframe*" est sélectionné. Une mise à jour récursive bayésienne est ensuite effectuée avec les nouvelles prises jusqu'à ce que l'incertitude soit assez petite. Un nouveau point est inséré dans la carte et va contribuer ensuite dans l'estimation du mouvement.

Rôle de la mise en correspondance des descripteurs : dans une procédure de triangulation à travers plusieurs images (comme expliqué dans la section 3.4), si la distance entre les deux images est petite, on aura donc une plus grande incertitude sur les points 3D triangulés.
Pour palier ce problème, une solution consiste à mettre en correspondance ce descripteur avec une ancienne prise pour augmenter la ligne de référence (la distance entre les deux lieux où sont prises les images). Cette prise de référence nommée "*keyframe*" correspond à la première vue ou ce point a été observé.

Annotations

- L'intensité de l'image enregistrée à partir d'une caméra C à un instant k est notée $I_k^c : \Omega^c \subset \mathbb{R}^2 \mapsto \mathbb{R}$, où Ω^c représente le domaine de l'image.

- Pour simplifier les équations, on suppose que le repère de la caméra C concorde avec le repère B du drone (Figure 3.1).
- $\mathcal{R}_k^c \subseteq \Omega$ représente les pixels dont la profondeur est déjà connue dans une caméra C à un instant k.

3.7.2 Estimation du mouvement

La tâche de ce processus est d'estimer correctement le mouvement entre l'étape précédente et l'étape actuelle. On suppose que la position des points $3D$ détectés dans les anciennes prises est connue avec précision et on cherche à trouver la transition T_{kk-1} illustré dans la figure 3.10 en combinant trois techniques, à savoir :

1. L'alignement éparse d'images ;
2. L'alignement des points caractéristiques ;
3. Le *Bundle Adjustment*.

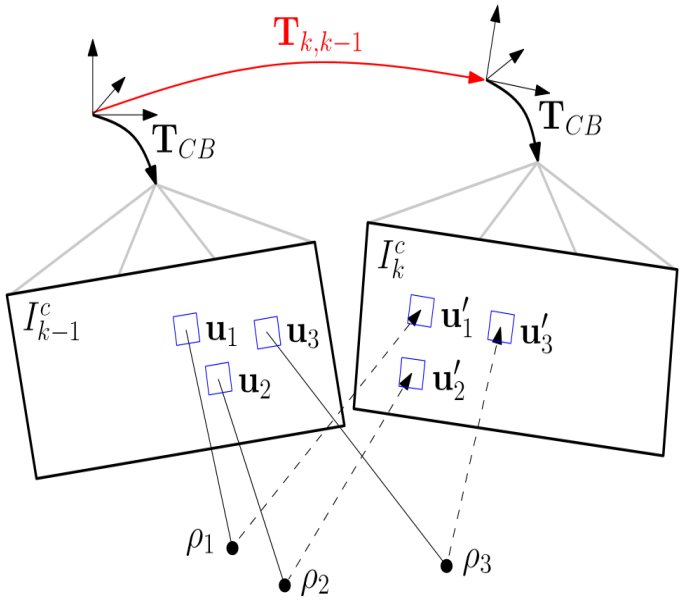


FIGURE 3.10 – *Mouvement et matrice de transformation*

Alignement éparse d'images

La première étape de l'estimation du mouvement est une technique directe de mise en correspondance des images en minimisant la différence d'intensité des pixels qui observent les mêmes points $3D$. Le but est donc d'estimer la matrice T_{kk-1} qui représente la matrice de transformation entre l'étape $k-1$ et l'étape k (figure 3.10) en minimisant l'*erreur photométrique* :

$$\mathrm{T}^{\star}_{kk-1} = \arg\min_{\mathrm{T}_{kk-1}} \sum_{\mathbf{u}\in\mathcal{R}^{\mathrm{C}}_{k-1}} \frac{1}{2} \left\| \mathbf{r}_{\mathrm{I}^{\mathrm{C}}_{\mathbf{u}}} \left(\mathrm{T}_{kk-1} \right) \right\|^2_{\Sigma_{\mathrm{I}}}, \tag{3.24}$$

- $\mathcal{R}^{\mathrm{C}}_{k-1}$ l'ensemble des pixels (points caractéristiques) dont la profondeur était déjà connue à l'étape $k-1$.
- Le résidu photométrique $\mathbf{r}_{\mathrm{I}^{\mathrm{C}}_{\mathbf{u}}} \left(\mathrm{T}_{kk-1} \right)$ est défini par la différence d'intensité entre les images $\mathrm{I}^{\mathrm{c}}_{k}$ et $\mathrm{I}^{\mathrm{c}}_{k-1}$ qui observent les mêmes points 3D notés $\boldsymbol{\rho}_{\mathbf{u}}$:

$$\mathbf{r}_{\mathrm{I}^{\mathrm{C}}_{\mathbf{u}}} \left(\mathrm{T}_{kk-1} \right) \doteq \mathrm{I}^{\mathrm{C}}_{k} \left(\pi \left(\mathrm{T}_{kk-1} \boldsymbol{\rho}_{\mathbf{u}} \right) \right) - \mathrm{I}^{\mathrm{C}}_{k-1} \left(\pi \left(\boldsymbol{\rho}_{\mathbf{u}} \right) \right), \quad \forall \mathbf{u} \in \mathcal{R}^{\mathrm{c}}_{k-1} \tag{3.25}$$

  L'image $\mathrm{I}^{\mathrm{C}}_{k-1} \left(\pi \left(\boldsymbol{\rho}_{\mathbf{u}} \right) \right)$ représente le résultat d'une opération de masquage que subis l'image initiale à l'étape $k-1$ pour garder uniquement les points 3D $\rho_{\mathbf{u}}$ dont la profondeur est connue. Les coordonnées de la projection de ces poins dans l'image est obtenu à travers le modèle de projection $\mathbf{u} = \pi(\boldsymbol{\rho})$.
  Par ailleurs l'opération $\pi \left(\mathrm{T}_{kk-1} \boldsymbol{\rho}_{\mathbf{u}} \right)$ va donner la projection de ces points sur l'image de la caméra à l'étape k à travers la transformation T_{kk-1} qui devra engendrer le minimum de différence entre ces images.
- Σ_{I} étant la covariance établie selon l'incertitude de la matrice image.

Il est à noter que l'optimisation de l'équation 3.24 ne prend en compte qu'un sous ensemble de pixels $\overline{\mathcal{R}}^{c}_{k-1} \subseteq \mathcal{R}^{c}_{k-1}$. Ces pixels sont caractérisés par le fait qu'ils sont visibles dans l'image $\mathrm{I}^{\mathrm{c}}_{k-1}$ et l'image $\mathrm{I}^{\mathrm{c}}_{k}$:

$$\overline{\mathcal{R}}^{c}_{k-1} = \left\{ \mathbf{u} | \mathbf{u} \in \mathcal{R}^{c}_{k-1} \wedge \pi \left(\mathrm{T}_{kk-1} \pi^{-1}_{\rho}(\mathbf{u}) \right) \in \Omega^{\mathrm{c}} \right\} \tag{3.26}$$

Pour améliorer la robustesse de la technique, l'erreur photométrique est estimée par rapport à des patchs au lieu de se contenter des points qui appartiennent à l'ensemble $\overline{\mathcal{R}}^{c}_{k-1}$. Ces patchs sont construits en supposant que les pixels entourant le point $\mathbf{u}$ ont la même profondeur. L'équation 3.24 se voit optimisée par rapport à des patchs dont le centre correspond à un point de profondeur connue.

Feature Alignment

La technique de mise en correspondance à travers les indices caractéristiques est employée pour diminuer le *drift* qui résulte de l'algorithme éparse d'alignement. Pour ce faire il est nécessaire de mettre en correspondance l'image récente avec l'image la plus ancienne qui soit possible. Pour chaque patch donc, une correspondance dans le plan de l'image (en 2D) va se faire par rapport à un patch de référence qui correspond à l'emplacement où il a été repéré la première fois, ce qui va diminuer le drift et ajuster l'estimation.

L'algorithme consiste à apporter un ajustement dans la position des points u' dans l'image la plus récente (Figure 3.10), en minimisant la différence d'intensité entre un patch P centré par ce point et un patch de référence perçu dans l'image de référence r où l'indice correspondant a été détecté la première fois.

Néanmoins cette étape engendre une erreur de réprojection (section 3.4) entre la projection du point 3D et l'indice caractéristique aligné après l'ajustement et se voit violer la contrainte épipolaire. Néanmoins cette erreur reste toujours inférieure à 0.5 pixels.

Bundle Adjustment

Le bundle adjustement est la dernière étape de l'estimation du mouvement. Il corrige l'erreur engendrée par l'alignement des *features* en affinant à la fois la pose de la caméra et les positions des points 3D dont la position est connue $\mathcal{X} = \{T_{kw}, \rho_i\}$, en minimisant la somme quadratique des erreurs de réprojections à travers plusieurs images.

3.7.3 Construction de la carte

Le processus de la construction de l'environement suppose que le déplacement de caméra est connu à travers le premier processus. La profondeur dans un pixel est estimée à travers plusieurs observations à l'aide d'un *filtrage récursif bayésien* initié dans les coins et les petits contours.

Sélection des images clés et initialisation

Quand le nombre des *features* suivis précédemment (dont la profondeur est connue) descend en dessous d'un certain seuil, une nouvelle image clé (*keyframe* en anglais) de référence r est sélectionnée. L'algorithme extrait les *features* et leur associe la *keyframe* (pour être utilisé dans 3.7.2). Cette initialisation est caractérisée par une large incertitude sur le point estimé.

Mise à jour et validation des points

Par la suite sur l'ensemble des images où le point apparaît, le filtre de profondeur est mis à jour avec une estimation de la position en cherchant le patch qui a le maximum de corrélation avec l'estimation sur la ligne épipolaire pour chaque image comme indiqué sur la Figure 3.11.

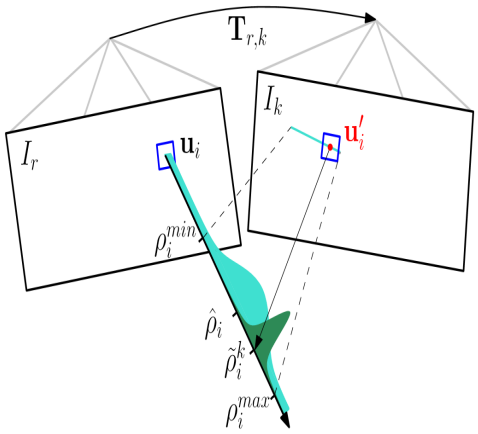


FIGURE *3.11 – Estimation probabiliste de la profondeur.*

Lorsque le filtre reçoit assez de mesures jusqu'à ce que l'incertitude sur la profondeur soit inférieure à un certain seuil, un nouveau point 3D est initialisé à la profondeur estimée, qui peut ensuite être utilisé pour l'estimation du mouvement.

Remarques

- Les *features* sont détectés à l'aide d'un détecteur FAST [104]. Un tableau de comparaison entre les détecteurs d'indices caractéristiques est fourni en annexe C.
- Lorsqu'il s'agit d'un cas d'utilisation où la caméra est pointée vers le bas, la mise à jour du filtre de profondeur (section 3.7.3) est effectuée uniquement avec les images qui arrivent après la *keyframe*. Tandis que si la caméra est pointée vers l'avant du système (*Forward Motion* en anglais), toutes les images sont prises en compte, même celles qui précédent la *keyframe*.

3.7.4 *Motion prior*

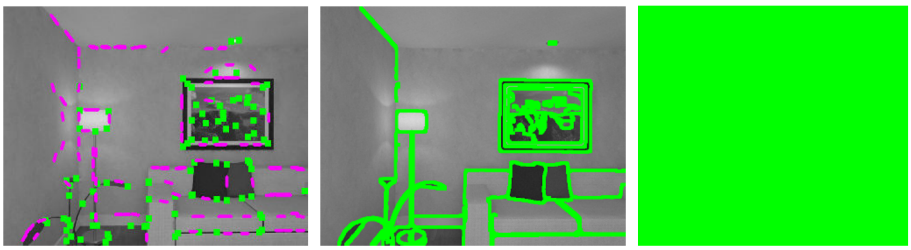
Les données de l'IMU (ou d'autres modèles d'estimation de mouvement *à-priori* additionnelles) peuvent être intégrées pour améliorer la performance de l'algorithme dans les cas d'un déplacement rapide ou des environnements à basses fréquences (structures qui continent peu de détails).

Un terme additionnel s'ajoute à l'équation 3.24 dans le but de pénaliser les mouvements non concordant avec les estimations de déplacement obtenues par les autres sources. Les sauts brusques dans l'estimation du mouvement ainsi que les points aberrants estimés (*outliers*) seront supprimés.
On suppose qu'une translation relative $\tilde{\mathbf{p}}_{kk-1}$ et une rotation relative $\tilde{\mathrm{R}}_{kk-1}$ sont disponibles au préalable (à partir d'une prédiction/estimation de vitesse linéaire et d'un gyroscope). Le *motion prior* est appliqué en ajoutant des termes en plus à l'équation 3.24

qui visent à minimiser la différence entre les mesures et les estimations :

$$\begin{aligned} \mathrm{T}^{\star}_{kk-1} = \arg\min_{\mathrm{T}_{kk-1}} \sum_{\mathbf{u}\in\overline{\mathcal{R}}^{c}_{k-1}} \frac{1}{2} \left\| \mathbf{r}_{\mathrm{I}^{C}_{\mathbf{u}}} \left(\mathrm{T}_{kk-1} \right) \right\|^{2}_{\Sigma_{\mathrm{I}}} \\ + \frac{1}{2} \left\| \mathbf{p}_{kk-1} - \tilde{\mathbf{p}}_{kk-1} \right\|^{2}_{\Sigma_{\mathrm{p}}} \\ + \frac{1}{2} \left\| \log \left(\tilde{\mathrm{R}}^{\mathrm{T}}_{kk-1} \mathrm{R}_{kk-1} \right)^{\mathrm{V}} \right\|^{2}_{\Sigma_{\mathrm{R}}} \end{aligned} \tag{3.27}$$

Représentation éparse



(a) Eparse (b) Semi-Dense (c) Dense

FIGURE *3.12 – Estimation probabiliste de la profondeur.*

Les techniques de construction de l'environnement se divisent en deux catégories :

- **Les techniques dites *denses* :** Utilisent tous les pixels de l'image et reprennent fidèlement l'environnement.
- **Les techniques dites *semi-denses* :** Utilisent uniquement les pixels caractérisés par un grand gradient d'intensité de l'image (contours).

L'architecture SVO reproduit une représentation éparse de l'environnement (Figure 3.12.a) n'utilisant que des pixels sélectionnés présent dans les coins et les petits contours. Cependant le même filtre de profondeur utilisé dans la construction de la carte à été utilisé dans une reconstruction dense de l'environnement dans [105].

3.7.5 Fermeture de la boucle

Le principe de la fermeture de la boucle est un principe utilisé fortement dans les algorithmes *SLAM* [106]. Il consiste à comparer les descripteurs de la dernière image avec les descripteurs des anciennes images où la carte est déjà connue ; si une grande ressemblance est trouvée la position actuelle est ajustée vers la position qui correspond à l'ancienne image pour éliminer les erreurs accumulées durant le trajet. Comme indiqué précédemment, notre approche garde en mémoire une file de quelques derniers *keyframes* pour les utiliser dans le processus de l'estimation du mouvement et les utiliser pour détecter les cas de fermeture de boucle en ajustant la position estimée [93].

Performances

Ce tableau fourni [38] nous donne des indications sur l'efficacité de calcul de l'algorithme *SVO* par rapport aux autres de *SLAM*.

	Mean	St.D.	CPU@20 fps
SVO Mono	2.53	0.42	55 ±10%
SVO Mono + Prior	**2.32**	**0.40**	**70 ± 8%**
SVO Mono + Prior + Edgelet	2.51	0.52	73 ± 7%
SVO Mono + Bundle Adjustment	5.25	10.89	72 ±13%
SVO Stereo	4.70	1.31	90 ± 6%
SVO Stereo + Prior	**3.86**	**0.86**	**90 ± 7%**
SVO Stereo + Prior + Edgelet	4.12	1.11	91 ± 7%
SVO Stereo + Bundle Adjustment	7.61	19.03	96 ±13%
ORB Mono SLAM (No loop closure)	29.81	5.67	187 ±32%
LSD Mono SLAM (No loop closure)	23.23	5.87	236 ±37%

TABLE *3.1 – La première et la deuxième colonne présentent la moyenne et la déviation standard du temps de traitement en millièmes de secondes sur un ordinateur portable doté d'un processeur Intel Core i7 (2,80 GHz). La troisième colonne indique la charge moyenne du CPU lors de la fourniture de nouvelles images à une fréquence constante de 20 Hz.*

Ce tableau est le résultat obtenu après avoir essayé ces algorithmes sur l'ensemble de données EUROC [107]. Nous pouvons en conclure que l'approche adoptée (SVO Mono + *Prior + Edgelets*) enregistre des performances meilleures que les algorithmes SLAM tout en gardant une basse consommation du CPU.
Ceci-dit, nous verrons dans le cinquième chapitre (conception et réalisation) les performances de cette approche pour notre cas d'utilisation.

3.8 Conclusion

Nous avons présenté dans ce chapitre la technologie fondamentale qui permet un fonctionnement efficace et performant aux plateformes mobiles en général et aux drones en particulier, à savoir le processus d'estimation de l'état avec une fonctionnalité de plus qui consiste en la représentation de l'environnement perçu.
Nous avons commencé par énumérer des notions fondamentales dans ce domaine qui permettent de comprendre le fonctionnement de l'algorithme d'estimation d'état, ainsi que sa composition et la façon avec laquelle la fusion de données entre les différents capteurs est effectuée. Ceci va nous servir pour adapter l'algorithme à la configuration matérielle et au cas d'utilisation.

L'algorithme retenue dans l'architecture globale est une implémentation Open-Source SVO en utilisant une approche qui concorde avec la configuration matérielle adoptée. L'un des principaux motifs pour choisir cet algorithme consiste en sa faible consommation en termes de ressources de calcul.

L'algorithme de perception va donc fusionner les données visuels et inertiels pour obtenir à la fois l'état du drone en plus d'une représentation 3D de la scène perçue, ce qui va permettre par la suite d'établir une cartographie de l'environnement parcourue.

Chapitre 4

Configuration de la procédure d'exploration

Dans cette partie, il s'agit de proposer une procédure qui nous permet de planifier un chemin dans un milieu méconnue que l'on découvre. Ceci a pour but d'optimiser le trajet effectué et de veiller à ce que le drone explore l'environnement cible en intégralité.

En effet, cette procédure repose sur une bonne représentation de l'environnement et une information fiable sur l'état du système en temps réel (position et orientation). Dans ce cadre, nous exploitons les données générées précédemment par l'algorithme de perception visuelle (à savoir la localisation du drone et la cartographie de l'espace), pour inférer une procédure d'exploration qui permet au drone de découvrir d'une manière autonome l'espace cible, en effectuant une navigation de prospection.

4.1 Architecture du système

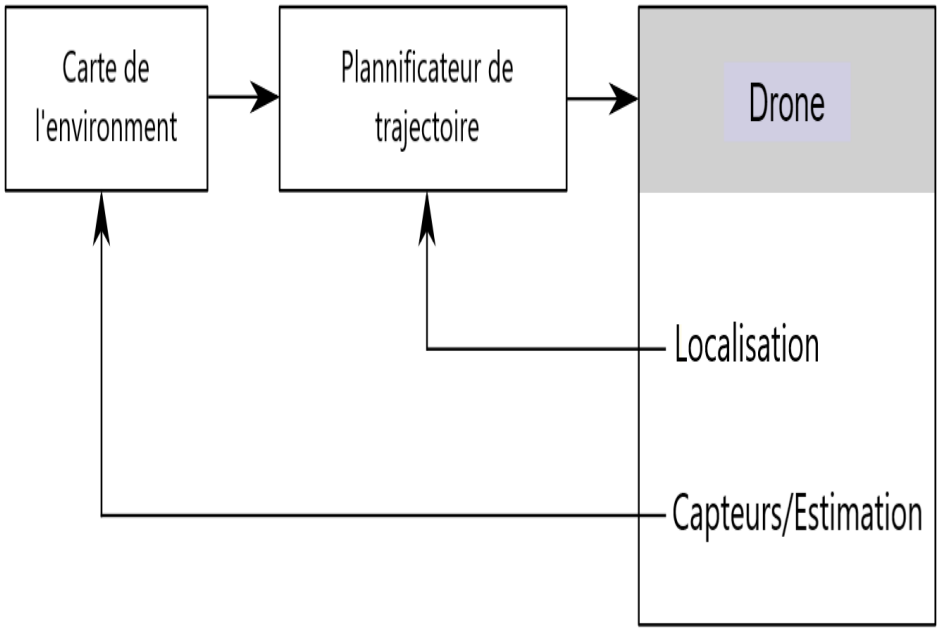


FIGURE 4.1 – *Diagramme représentant la configuration d'un système d'exploration.*

La figure 4.1 montre la configuration d'un drone d'exploration. Le planificateur de trajectoire reçoit l'état $\xi = (x, y, z, \psi)^T$ [1] à partir d'un système de localisation. Un capteur ou un algorithme de SfM (*structure from motion*) fournit une carte représentative de l'environnement perçu (grille d'occupation section 4.2) en temps réel. Le planificateur à partir de la cartographie de l'environnement, ainsi que la position actuelle du drone, décide du prochain pas (prochaine position et orientation) à atteindre pour continuer l'exploration jusqu'à la fin de la mission.

4.2 Cartographie de l'environnement

La carte de l'environnement peut être soit fournie à l'avance ou construite durant la navigation [108]. Elle peut servir pour guider le robot dans l'environnement [102], ajuster l'estimation de l'état (*fermeture de boucle* dans SLAM [109]) ou uniquement pour la représentation lors de l'exploration.
Ces cartes peuvent être sous différentes formes selon les besoins du système [102] :

- **Cartes métriques :** Ce sont les cartes classiques où les informations de longueur, distance, position sont explicitement définies dans un repère réel.
  Ce type de cartes nécessite une cohérence géométrique sur l'ensemble de la représentation et doit assurer donc la possibilité d'une fermeture de boucle lors de la navigation.
- **Cartes topologiques :** Ces cartes sont représentées par des graphes dont les sommets correspondent à des zones de l'environnement qui peuvent être associées à des actions (tourner, s'arrêter, traverser une porte, etc.), tandis que les arêtes indiquent le chemin à traverser entre ces nœuds.
- **Grilles d'occupation :** Ce type de carte constitue une représentation surfacique de l'environnement où l'espace est subdivisé en cellules. Chaque cellule contient un indice (probabilité, histogramme, etc.) indiquant l'occupation de l'espace.
- **Modèle de primitives :** Cette catégorie englobe les cartes qui se fient aux balises pour se repérer. Dans le cas de la vision par ordinateur, ces balises sont appelées primitives géométriques dans une image (*features* en anglais). Elles doivent avoir un pouvoir discriminant, un domaine de vision important, une stabilité stricte, ainsi qu'une invariance aux changements météorologiques et lumineuses.

La figure 4.2 montre une représentation visuelle des différentes cartes représentatives de l'environnement.

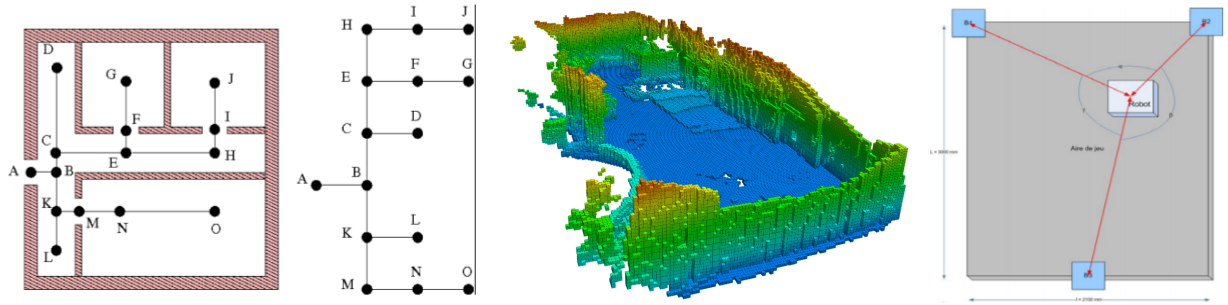


FIGURE 4.2 – *Les différents types de cartes utilisés dans la robotique respectivement : métrique-topologique, métrique, grilles d'occupation, modèle de primitives.*

1. ψ : Angle de lacet (yaw).

4.3 Formulation du problème

Le problème d'exploration des milieux inconnus est formulé comme suit :
Soit un environnement 3D clos noté $V \subset \mathbb{R}^3$. À l'état initial tout le volume est supposé méconnue, ce qui nous donne donc :

$$V_{inconnue} \overset{initial}{=} V$$

Lors de la navigation, deux autres sous-espaces sont considérés : l'espace libre $V_{libre} \subset V$ et l'espace occupé $V_{occ} \subset V$.
La procédure d'exploration subie des incertitudes liées aux limitations de capteurs employés pour la construction de la carte. Par conséquent, la perception en général ne prend pas en considération les espaces au delà des murs, les espaces creux, ainsi que les poches serrés [58]. Cet espace résiduel est noté V_{res}.

Le problème de navigation est considéré comme entièrement résolu quand on obtient :

$$V_{\text{libre}} \cup V_{\text{occ}} = V \backslash V_{\text{res}}$$

Le trajet est donc calculé en temps-réel à chaque pas qu'effectue le drone jusqu'à résoudre le problème.

4.4 Procédure adoptée

L'approche adoptée dans la procédure d'exploration consiste en un algorithme d'exploration qui essaie de répondre au problème de la meilleure vue prochaine ("*next-best-vue*" en anglais), en ayant comme critère de sélection la quantité d'espace pouvant être exploré sur la cible prochaine. Les cibles candidates appartiennent à l'espace V_{free}. Ils sont représentées par des nœuds sur un arbre aléatoire dont les arrêtes sont considérées comme le chemin qui mène vers ces nœuds. Cet algorithme a été proposé dans [58].

En faisant partie du schéma 4.1, le planificateur de trajectoires a pour objectif d'explorer l'intégralité de l'environnement cible en minimisant la distance d'exploration, selon la procédure suivante :
Soit $\xi_k \in \Xi$ l'état du drone à une étape k.
Un trajet élémentaire entre deux étapes $k-1$ et k est défini par la fonction $\sigma : \mathbb{R} \to \xi$. Cette fonction s'écrit sous la forme : $\sigma_{k-1}^{k}(s)$, avec :

$$\begin{aligned} & s \in [0,1] \ ; \\ & \sigma_{k-1}^{k}(0) \ = \ \xi_{k-1} \ ; \\ & \sigma_{k-1}^{k}(1) \ = \ \xi_{k} \ ; \end{aligned}$$

Ce trajet ne doit pas passer par un obstacle, et respecter les contraintes cinématiques et dynamiques du véhicule.

Dans un état ξ et pour une carte $\mathcal{M}$ de l'environnement, l'ensemble des voxels (pixels en 3D) visibles et non définies à cet état est noté $\textbf{Visible}(\mathcal{M}, \xi) \subset V_{inconnue}$. Il représente

l'espace derrières les voxels occupés visibles et l'espace qui dépasse l'angle de vision.

Un arbre géométrique $\mathbb{T}$ est construit d'une façon incrémentale à partir de la configuration ξ_0 utilisant l'algorithme RRT [110]. On obtient donc un arbre contenant $N_{\mathbb{T}}$ nœuds n et leurs chemins σ. À chaque nœud est associé un gain $\mathbf{Gain}(n)$ représentant le volume non défini qui pourrait être exploré à partir de ce nœud.
Le gain est estimé grâce à la formule :

$$\text{Gain}\,(n_k) = \text{Gain}\,(n_{k-1}) + \text{Visible}\,(\mathcal{M}, \xi_k)\, e^{-\lambda c\left(\sigma_{k-1}^{k}\right)} \tag{4.1}$$

Avec :

- $c(\sigma_{k-1}^{k})$ le coût du trajet entre le point actuel et le point suivant.

- λ facteur pénalisant du coût.

Le coût du trajet est défini comme étant la distance euclidienne entre l'état actuel et le nœud cible :

$$c\left(\sigma_{k-1}^{k}\right) = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2 + (z_k - z_{k-1})^2} \tag{4.2}$$

Le chemin vers le nœud n_{best} qui possède le meilleur gain est extrait qui va représenter le prochain trajet du drone. L'arbre qui résulte de la meilleure branche est gardé en mémoire pour initialiser l'arbre dans la prochaine étape. Les gains quant à eux seront ré-évalués avec la nouvelle représentation de la carte dans le prochain pas.

Si la construction de l'arbre atteint la limite (lorsque $N_{\mathbb{T}} = N_{\max}$) et le gain g_{best} reste nul, la construction continue jusqu'à ce qu'on obtient $g_{\text{best}} > 0$. Cependant si le nombre de nœuds dans l'arbre dépasse une valeur N_{TOL} de tolérance sans que le gain n'ait augmenté, le problème d'exploration est considéré comme étant résolu.

L'algorithme ci-dessous résume les étapes citées ci-dessus :

Algorithm 1: Algorithme du planificateur de trajectoire

$\xi_0 \leftarrow$ Etat actuel du drone
Initialiser $\mathbb{T}$ avec ξ_0 et la meilleure branche précédente
$g_{best} \leftarrow 0$
`// Initialiser le meilleur gain à 0`
$n_{best} \leftarrow n_0(\xi_0)$
`// Initialiser le meilleur nœud à l'origine`
$N_{\mathbb{T}} \leftarrow$ Nombre des noeuds initiaux dans la branche$\mathbb{T}$
while $N_{\mathbb{T}} < N_{\max}$ ***or*** $g_{best} = 0$ **do**
  Construire $\mathbb{T}$ en ajoutant $n_{new}(\xi_{new})$
  $N_{\mathbb{T}} \leftarrow N_{\mathbb{T}} + 1$
  if ***Gain***$(n_{new}) > g_{best}$ **then**
    $n_{best} \leftarrow n_{new}$
    $g_{best} \leftarrow$ **Gain**(n_{new})
  end
  if $N_{\mathbb{T}} > N_{TOL}$ **then**
    Terminer l'exploration
  end
end
$\sigma \leftarrow$ **ExtraireMeilleurChemin**(n_{best})
Supprimer $\mathbb{T}$
return σ

4.5 Conclusion

L'exploration d'un milieu inconnu nécessite une politique d'exploration pour à la fois optimiser la trajectoire et garantir la prospection de tout l'environnement cible.

L'environnement est représenté à travers la grille d'occupation, une cartographie qui sert à analyser la configuration des scènes perçues à chaque instant. L'arbre des cibles potentiels quant à elle, est une carte topologique fournissant l'information d'utilité des cibles au planificateur de trajectoire en ligne. Ceci dit, notre approche comporte l'utilisation d'une représentation hybride (topologique-grille d'occupation) de l'environnement.

La procédure d'exploration retenue dans l'architecture globale met en évidence le potentiel d'espace à explorer dans le choix du prochains pas. Une approche adaptée à un cas d'utilisation d'une mission de recherche et sauvetage lors d'une catastrophe naturelle où l'objectif principal est d'aller vers les zones non explorées à la recherche de survivants.

Chapitre 5

Conception et réalisation

5.1 Introduction

Les drones sont équipés de systèmes complexes composés de plusieurs sous-systèmes communicant et interdépendants pour : l'estimation, la planification, la navigation, le contrôle. Cela impose un parallélisme dans l'architecture du système. Aussi, les contraintes d'espace et les limitation des ressources matérielles pour le calcul à bord exigent des solutions légères et optimisées.

Dans les chapitres précédents nous avons décrit chaque algorithmes qui peuvent contribuer à la résolution des sous-problématiques énoncées dans le premier chapitre. Nous verrons dans ce qui suit comment vont être employés ces algorithmes dans une architecture qui vise à apporter une solution à la problématique principale, ainsi que les détails de la réalisation des différents systèmes constituant l'architecture globale. L'objectif étant de fournir un système d'exploration et de perception visuelle autonome, capable de détecter des postures humaines dans des images.

5.2 Solution proposée

La solution proposée est une architecture implémentée sur un drone équipé d'un contrôleur haut niveau et de deux capteurs uniquement : une caméra à sténopé (monoculaire ou stéréoscopique) et une centrale inertielle.

Dans un scénario de catastrophe naturelle, ce drone est utilisé dans les premières 72h dans les opérations de recherche et sauvetage. Il est envoyé à l'intérieur d'un immeuble démoli ou dans une zone à haut risque pour l'explorer, à la recherche de survivants. Le scénario d'utilisation est expliqué dans ce qui suit.

Scénario d'utilisation : Soit un endroit fermé partiellement détruit (ou une zone précaire) représenté par le plan dans la figure 5.1, le drone commence l'exploration à partir d'un point en ayant comme information sur l'environnement les dimensions de la zone à explorer. Il effectue la navigation selon la procédure d'exploration, en fournissant en continu sa position relative au point de départ et une mise à jour de la carte de l'environnement qui est enregistrée dans une station de suivi terrestre. Ceci nous permet de tracer une trajectoire en ayant une modélisation du milieu parcouru.

De plus, lors de la navigation un processus de détection des humains à partir des images s'exécute sur le drone pour détecter la présence d'humains. Dès qu'une personne est détectée un signal d'alarme est envoyé à la station de suivi avec la position du survivant et continue ensuite l'exploration jusqu'à couvrir la zone cible. Les secours auront donc la position des personnes à secourir avec tous les détails du chemin à emprunter.

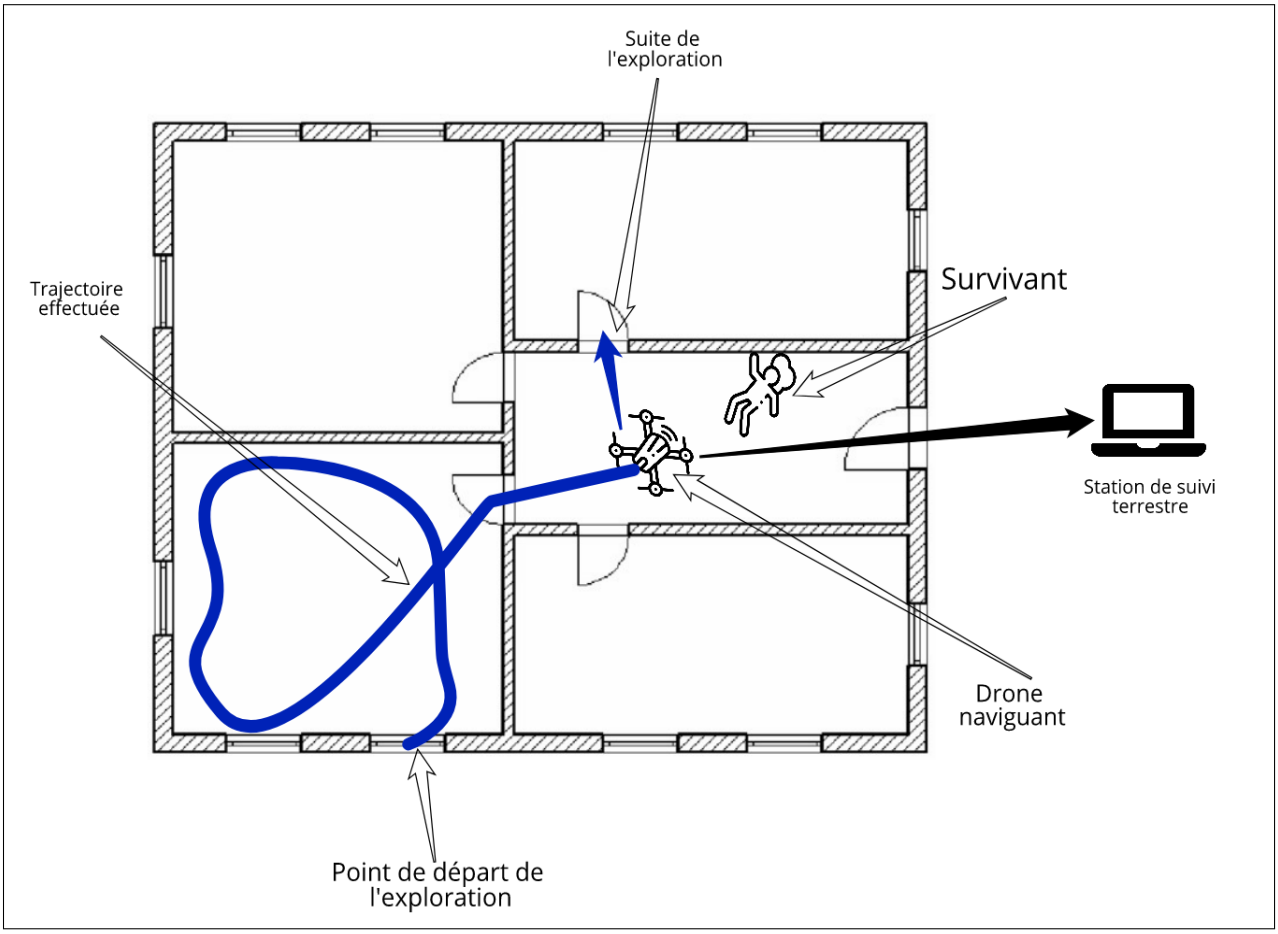


FIGURE *5.1 – Scénario d'utilisation de la solution proposée.*

Architecture globale

La figure 5.2 montre l'architecture de la solution proposée avec l'environnement nécessaire pour son exécution.

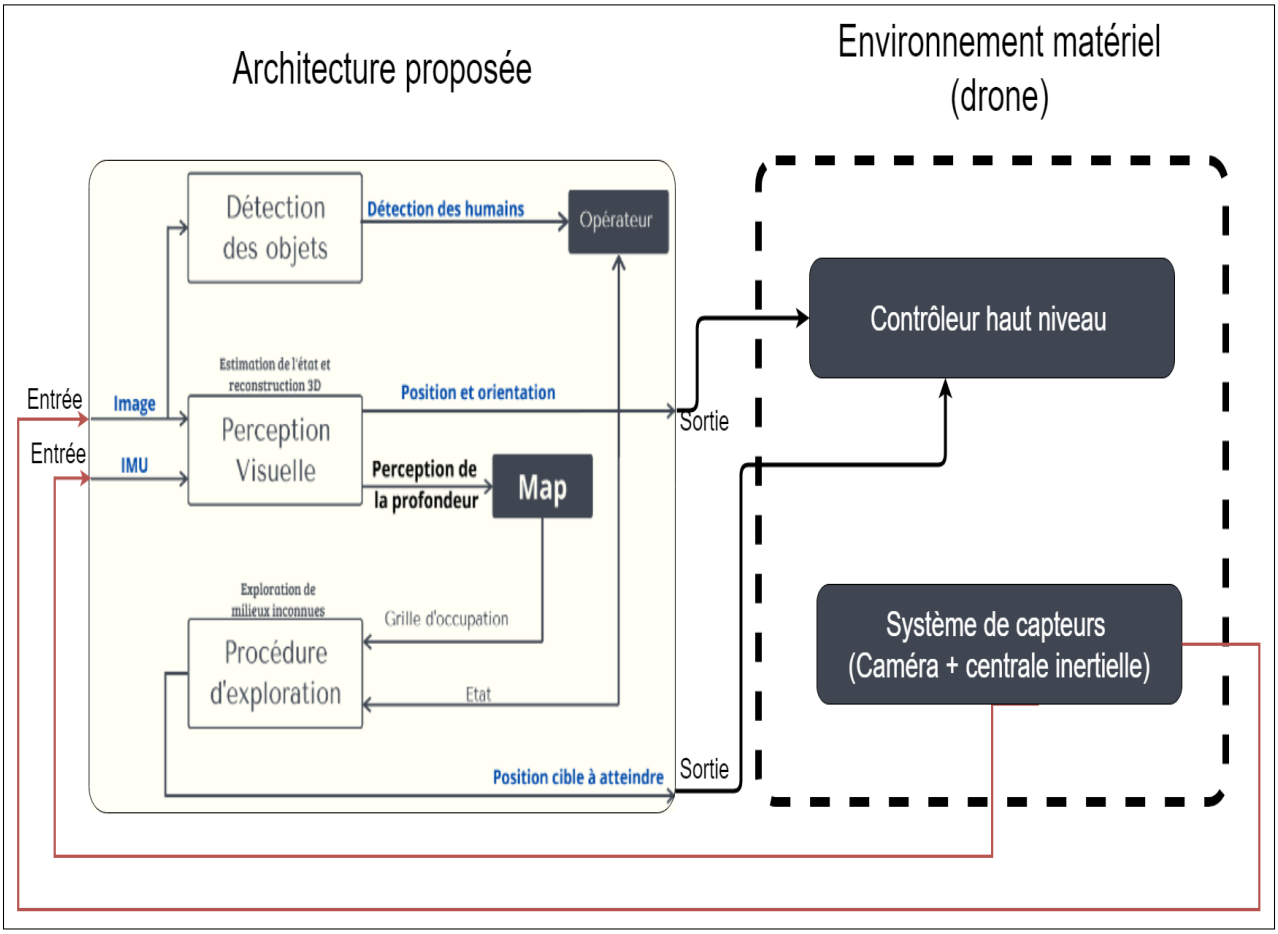


FIGURE 5.2 – *Architecture globale du système.*

Perception visuelle : bloc responsable sur l'identification de l'environnement et la connaissance de l'état actuel du drone (orientation et position). Dans ce bloc est exécuté l'algorithme présenté dans la section 3.7. Il reçoit en entrée les signaux issues du système de capteurs extéroceptifs à savoir : les images de la caméra monoculaire et les signaux de la centrale inertielle. En sortie, ce bloc fournit l'attitude (position et orientation) du système.

Procédure d'exploration : le système de planification est la procédure adoptée pour définir les prochains points que le système navigant doit atteindre sur un environnement partiellement connu afin de l'explorer. Il reçoit en entrée l'attitude du véhicule et l'environnement autour du système. En tenant compte de l'optimisation de la distance et du temps d'exploration, ce bloc calcule les *points de cheminement* qui vont déterminer la trajectoire parcourue.

Détection des objets : algorithme de détection se basant sur une architecture de réseaux de neurones convolutifs, responsable de la détection des objets. Il est utilisé pour détecter des humains mais peut détecter 1000 classes d'objets possibles dans son implémentation initiale, une information qui peut être exploitée pour un autre cas d'utilisation.

Environnement matériel : l'environnement matériel qui supporte notre architecture se comporte de deux parties principales :

1. **Contrôleur haut niveau :** un dispositif qui permet de monter en niveau d'abstraction dans la commande du déplacement d'un robot. Au lieu de commander le système en vitesse ou en position à travers un système asservi, on lui fournit la position à atteindre ainsi que la position actuelle et il établit la commande sur le système en utilisant des capteurs d'état embarqués dans ce dernier.
2. **Système de capteurs :** pour le fonctionnement de notre système nous avons besoin des signaux de l'image et de la centrale inertielle. Ces mesures doivent passer naturellement par un processus de calibration pour éliminer certains types de bruits et ajuster les signaux obtenus.

Remarque : les 3 modules principaux de la solution citée ci-dessus sont implémentés dans des conteneurs docker[1] qui peuvent s'exécuter en même temps[2]. Ceci rend l'application indépendante de la configuration logicielle de l'environnement d'exécution et nous offre un moyen facile de remplacer n'importe quelle brique présente dans l'architecture sans affecter les autres parties de la solution. Ajouté à cela la compatibilité sur les entrées/sorties des différentes parties de la solution.

5.3 Validation des algorithmes utilisés

5.3.1 Perception visuelle

Afin de valider l'algorithme de perception qui a été proposé dans la section 3.7 et voir les différentes possibilités qu'il offre, des tests sont effectués sur ce dernier en utilisant une version de l'algorithme fournit par les auteurs de [38] sur un équipement hardware accessible pour tout amateur de drones.

Outils logiciels

L'exécution de l'algorithme se fait sous l'environnement ROS (Robot Operating system)[3], elle repose sur les outils suivants pour son fonctionnement :

1. **Raspicam Node**[4] **:** un outils sous ROS qui permet de récupérer le signal image à partir d'une caméra montée sur une Raspberry PI et de le diffuser sur une queue de messages de type *"sensor_msgs/Image Message"* de ROS.
2. **Image transport**[5] **:** il permet de compresser le flux d'images obtenus à partir de la caméra pour accélérer une éventuelle communication entre différents systèmes indépendants communicants entre eux (via WIFI par exemple).

1. Plus d'information sur Docker dans : https://www.docker.com/.
2. Docker permet la communication entre plusieurs conteneurs qui s'exécutent en même temps en partageant la carte réseau de l'hôte, une autre solution qui garde la séparation des réseaux est disponible dans : https://gist.github.com/ruffsl/4a24c26a1aa2cc733c64.
3. Plus d'information sont fournies dans : https://www.ros.org/.
4. Code source : https://github.com/UbiquityRobotics/raspicam_node.
5. Code source et plus de détails sur : https://github.com/ros-perception/image_common.

3. **Rviz** [6] **:** outil de visualisation 3D de ROS, il permet de visualiser plusieurs formats standards, notamment les formes, les déplacements des systèmes en temps-réel (à l'aide du type de données TF2) ainsi que la représentation de l'environnement à partir des nuages de points fournies sous format OctoMap [55].
La figure 5.3 représente un exemple sur la visualisation fournie par l'outil Rviz, où on peut apercevoir : la position du repère du drone par rapport au repère terrestre fixe, les points représentants l'environnement, ainsi que la trajectoire effectuée par le système navigant. Pour l'algorithme SVO, les points violets représentent les petits contours tandis que les points verts représentent les coins détectés dans les images.

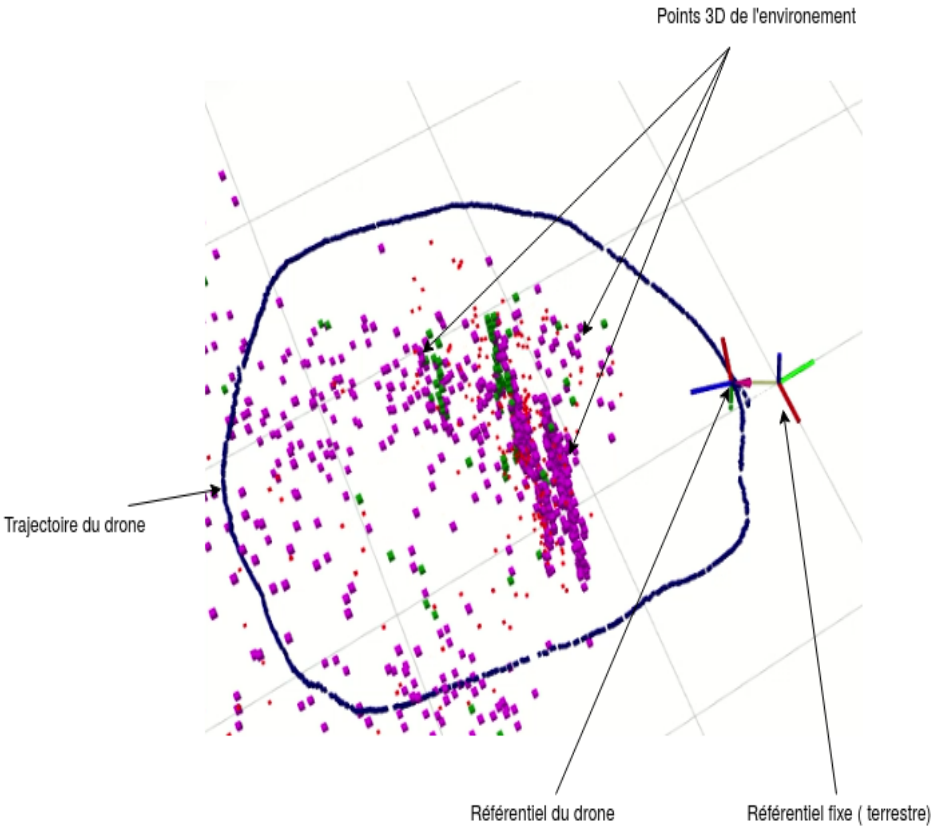


FIGURE *5.3 – Exemple de visualisation par Rviz.*

Le fonctionnement globale de l'algorithme se fait sous la configuration logicielle suivante :

- *Xenial Ubuntu 16.04.06* comme système d'exploitation
- *ROS KInetic Kame* comme environnement d'exécution, correspondant à la distribution Xenial de Ubuntu.

Le schéma donné par la figure 5.4 illustre la liaison entre les différents composants logiciels de l'algorithme de perception.

6. Détails et tutoriels sur l'outil sont disponibles dans : `http://wiki.ros.org/rviz`.

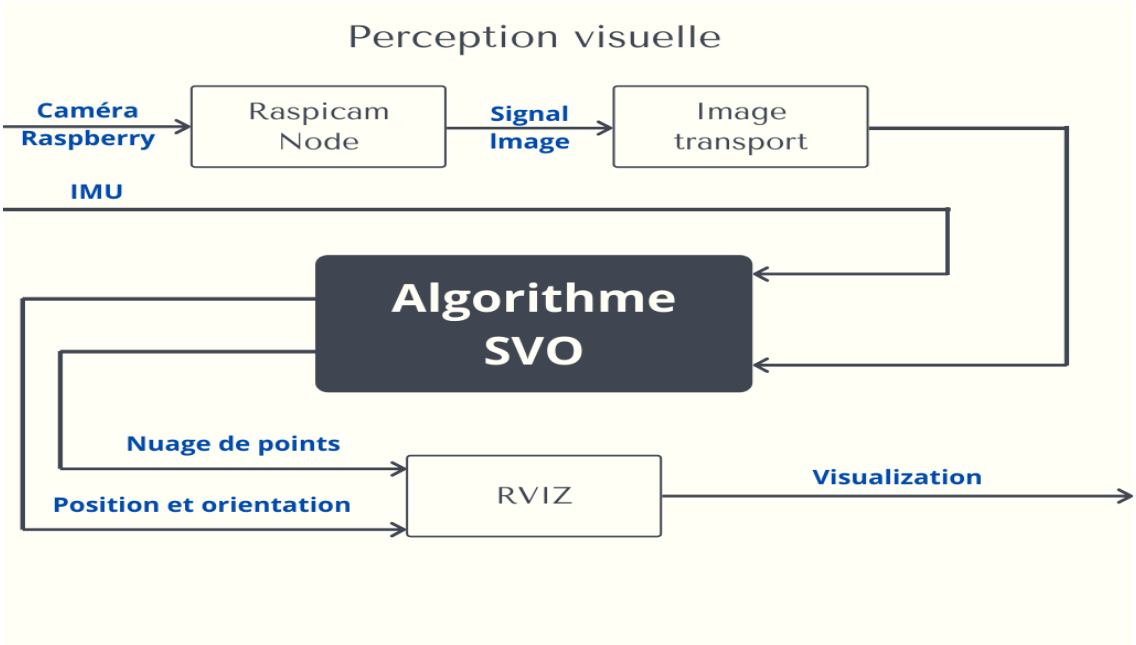


FIGURE 5.4 – *Configuration logicielle de l'algorithme SVO.*

Il est à noter que l'algorithme SVO fournit en sortie la position et l'orientation sous format standard TF2 de ROS. Quant à la reconstruction de l'environnement, elle se fait à l'aide du nuage de points qui représente les points caractéristiques sélectionnés dans les images (section 3.7). Ce nuage de points, est représenté sous format OctoMap visualisé par Rviz.

Équipement Hardware

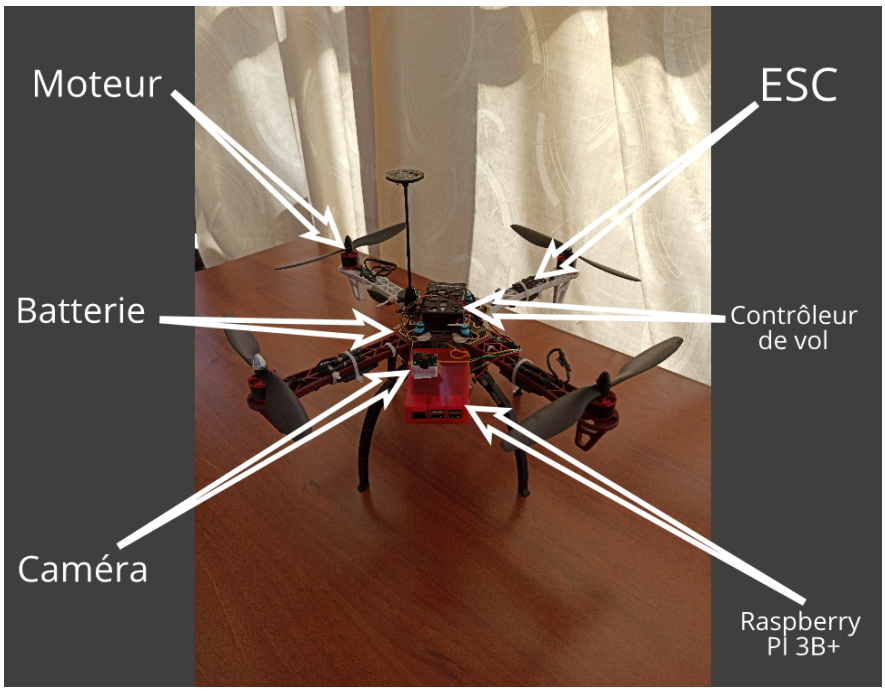


FIGURE 5.5 – *Drone de test.*

L'algorithme de perception a été testé sur un drone réel (figure 5.5) pour se mettre dans les conditions proches d'un cas d'utilisation réel. Il est équipé du matériel suivant :

1. **Caméra :** Raspberycam monoculaire RGB, fonctionnant à une fréquence d'images de 90Hz avec une résolution d'image de 640*480.
2. **Ordinateur de bord :** Raspberry Pi 3 B+ doté d'un processeur quad core Broadcom BCM2837B0, Cortex-A53 64-bit tournant à 1.4GhZ.
3. **Contrôleur de vol :** Pixhawk 3DR 2.4.8, qui assure la tâche de la commande des moteurs pour permettre un vol stable au drone, un capteur IMU y est intégré pour lui permettre d'établir la commande. Nous allons utiliser le signal de ce capteur comme centrale inertielle.

Calibration

1. **Calibration de la caméra :** comme indiqué dans la section 3.4, cette partie consiste en la détermination des paramètres intrinsèques de la caméra en plus des paramètres du modèle de distorsion de l'image adoptée.

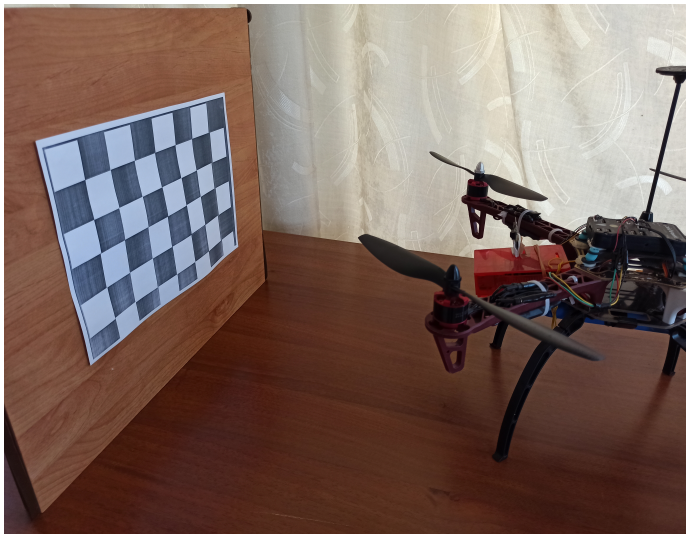


FIGURE 5.6 – *Prise d'image d'un échiquier pour calibration de la caméra embarqué sur le drone.*

La procédure de calibration consiste à prendre plusieurs images d'un échiquier dont on connais les dimensions à partir de plusieurs angles différents[7] pour estimer le modèle de distorsion adopté dans [90] et déterminer les paramètres intrinsèques. Nous utilisons le package *camera_calibration* pour déterminer les paramètres voulues, le résultats est le suivant :

7. Comme illustré dans la figure 5.6.

```
distortion_parameters :
    data: [0.069528, -0.232888, 0.003760, -0.000691]
  type: radial-tangential
image_height: 480
image_width: 640
intrinsics_parameters :
  data: [623.352262,  623.525006,319.067666, 240.139489]
```

2. **Calibration caméra-IMU :** dans cette partie nous déterminons la transformation de position entre la pose de la caméra et celle de l'IMU, en plus du déphasage dans le temps entre les deux signaux (image et IMU). Pour se faire, nous utilisons l'outil Kalibr[8] (inspiré des articles [111] et [112]) qui à partir d'un data-set (données caméra et IMU synchronisés) représentant la procédure de calibration. Cette procédure de calibration consiste en des mouvements du drone face à un échiquier cible avec des dimensions connues. Les mouvements doivent exciter tous les axes de translation et de rotation possibles, ce qui veut dire plusieurs translations et rotations successives pour obtenir une séquence exploitable pour la calibration[9].

Environnement de test

Afin de valider la brique de perception visuelle, nous procédons à des tests réels pour mesurer l'efficacité de la méthode adoptée et des possibilités qu'elle propose.
L'environement de test est illustré par le schéma suivant :

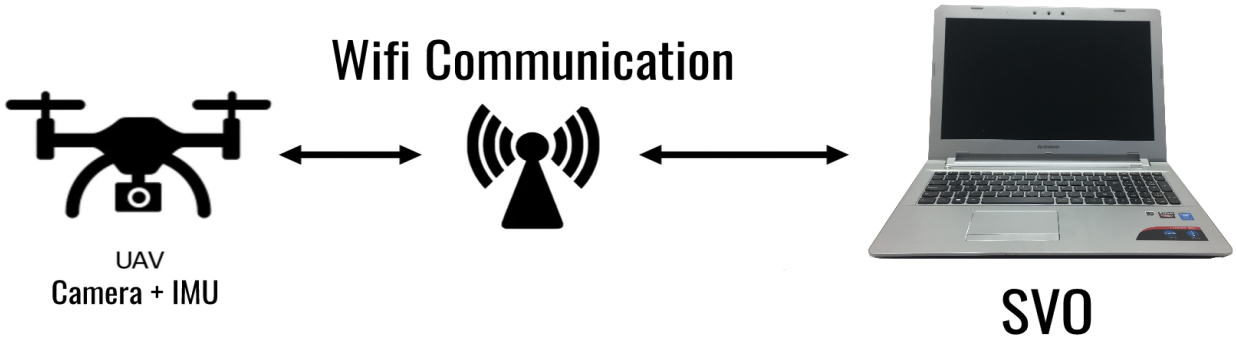


FIGURE 5.7 – *Environnement d'exécution (perception).*

Le drone de test effectue une exploration manuelle de l'environnement cible, les données des capteurs sont recueillies par l'ordinateur de bord et sont envoyées par

8. Code source et plus de détails dans : https://github.com/ethz-asl/kalibr.
9. La procédure de calibration est expliquée dans https://www.youtube.com/watch?v=puNXsnrYWTY&app=desktop.

protocole de communication WIFI à une machine distante qui exécute l'algorithme de perception à partir des signaux reçus.
Cette configuration nous a permis de faire abstraction du problème d'efficacité de calcul afin de tester les capacités de la technique en utilisant des capteurs génériques à coût faible.

Tests expérimentaux

Les tests effectués sous l'environnement cité précédemment ont pour but de valider les deux aspects principaux de l'algorithme, à savoir le positionnement par rapport à un référentiel fixe et la reconstruction efficace de l'environnement perçu. Comme nous utilisons une caméra monoculaire, la profondeur de la scène de départ doit être connue pour établir une bonne estimation. Nous nous positionnons donc à une distance précise d'un obstacle que voit la caméra.
Afin d'évaluer les résultats, nous nous sommes basés sur deux critères d'évaluation :

1. **Positionnement :** nous avons d'abord commencé par mesurer les erreurs de positionnement engendrés par la méthode. Pour ce faire, nous effectuons des trajectoires prédéfinies et nous posons comme mesures d'évaluation la forme de la trajectoire et les distances parcourues.
2. **Proportions de l'environnement recueillis :** ensuite pour mesurer la fidélité de l'environnement reconstruit, nous évaluons la modélisation obtenu à travers l'estimation sur les formes et les proportions des objets détectés.

La figure 5.8 montre la configuration d'exécution de l'algorithme de perception visuelle à l'aide de l'outil de visualisation **rqt_graph** de ROS (plus de détails sur cet outil sont fournies en annexe D). On peut voir les différents constituants de l'algorithme et la relation entre eux :

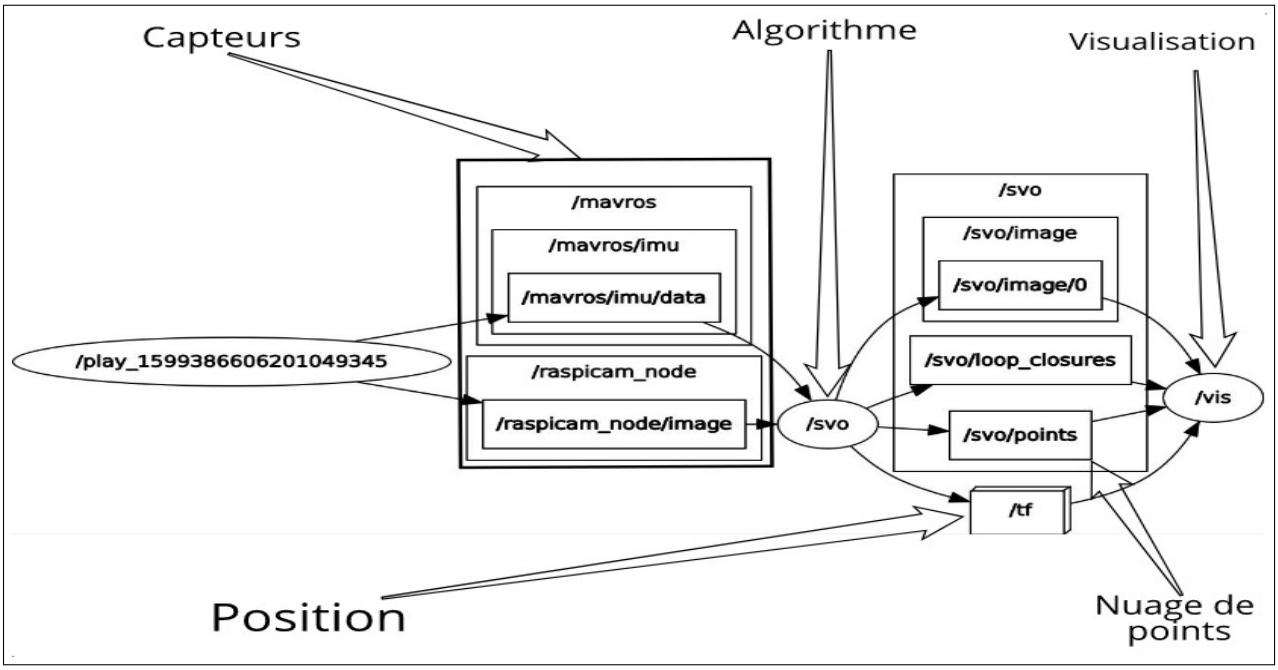


FIGURE 5.8 – *Configuration d'exécution de l'algorithme de perception.*

Trajectoire en zigzag : la trajectoire en zigzag comporte huit segments de 1.8m de long avec une distance entre point de départ et point d'arrivé mesurée à 10.24m où le drone reste toujours orienté vers la porte où se trouve le point de démarrage. Elle est illustré dans la figure 5.9.

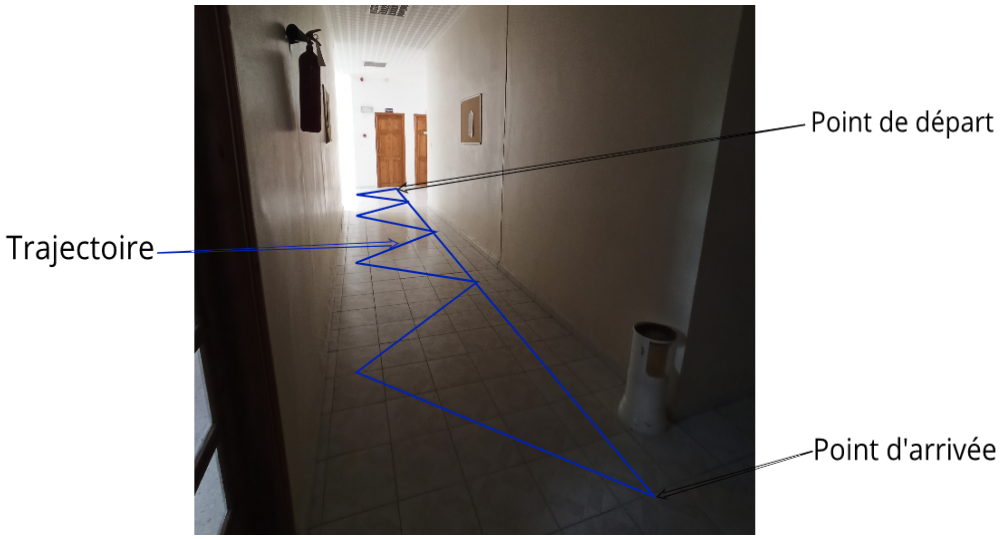


FIGURE 5.9 – *Trajectoire en zigzag.*

La figure 5.10 montre une visualisation du résultat de l'estimation à l'aide de l'algorithme SVO en utilisant l'outil de visualisation **Rviz**.

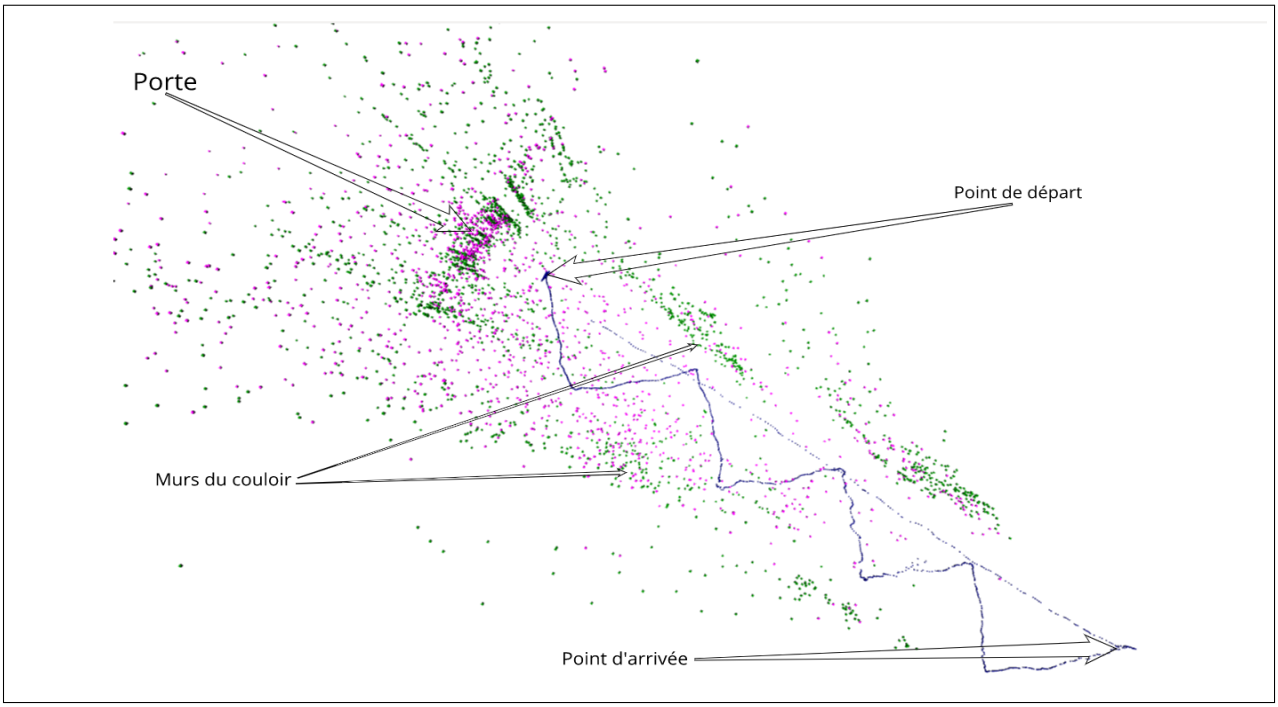


FIGURE 5.10 – *Visualisation de la trajectoire en zigzag estimée par notre système de perception visuelle.*

On peut voir à travers la correspondance établie entre le nuage de points qui représente les points 3D estimés par l'algorithme et les composants de l'environnement, que la perception visuelle génère une modélisation fiable de l'espace qu'elle perçoit. Les

dimensions de l'environnement sont mis en examen pour s'assurer de la justesse de l'estimation.

Dans un autre contexte nous avons évalué la position estimée durant cette expérience à travers l'étude de la trajectoire générée par le déplacement du drone. Nous constatons que la trajectoire estimée est proche de la trajectoire réelle avec des erreurs d'emplacement qui ne dépassent pas les 5%.
Les résultats sont rapportés dans le tableau :

Distance	Estimation de SVO (m)	Mesure réelle (m)
Point de départ - Point d'arrivée	10.18	10.24
Entre les murs du couloir	1.91	2
Moyenne d'erreur sur les segments	0.1	0

TABLE *5.1 – Comparaison entre les mesures estimées et les mesures réelles de la trajectoire en zigzag.*

Trajectoire en carré : elle consiste en un trajet sous forme de carré en faisant face à la porte qui est perçu au départ, comme le montre la figure 5.11.

FIGURE *5.11 – Trajectoire suivant une forme carrée.*

Le but principal de cet essai est d'étudier les dimensions des formes générées par l'estimation et les comparer aux mesures réelles. La figure 5.12 montre les résultats de l'estimation par l'algorithme de perception visuelle.

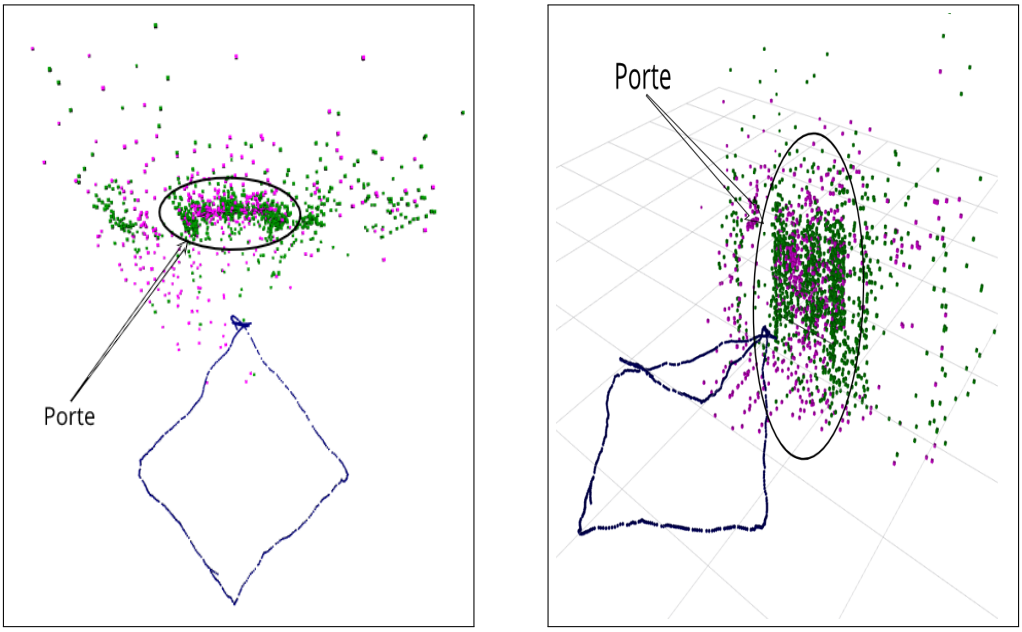


FIGURE 5.12 – *Résultats de la navigation suivant la trajectoire carré.*

Nous constatons une concordance dans l'estimation de la trajectoire et de la structure du nuage de point avec la réalité qui va jusqu'au détailles de la structure de la porte. Ceci est dû entre-autres à la richesse de l'image de la porte en points saillants (coins et petits contours) qui favorisent la détection des descripteurs et qui fournissent par conséquent une meilleure estimation.

Nous constatons que l'erreur commise sur les dimensions est tolérable pour notre cas d'utilisation. Cependant nous constatons un léger décalage de l'estimation lors du retour à la position initiale (décalage de 8cm, erreur accumulée de 1%). Le tableau 5.2 contient une comparaison entre les estimations et la mesure.

Distance	Estimation de SVO (m)	Mesure réelle (m)
Trajectoire	7.2	6.83
Largeur de la porte	0.89	0.91
Entre les portes	0.3	0.26
Entre la porte et le mur de gauche	0.68	0.62

TABLE 5.2 – *Comparaison entre les mesures estimées et les mesures réelles de la trajectoire en carré.*

Trajectoire circulaire : un parcours en cercle de circonférence de 8 mètres, qui revient à la position de départ.

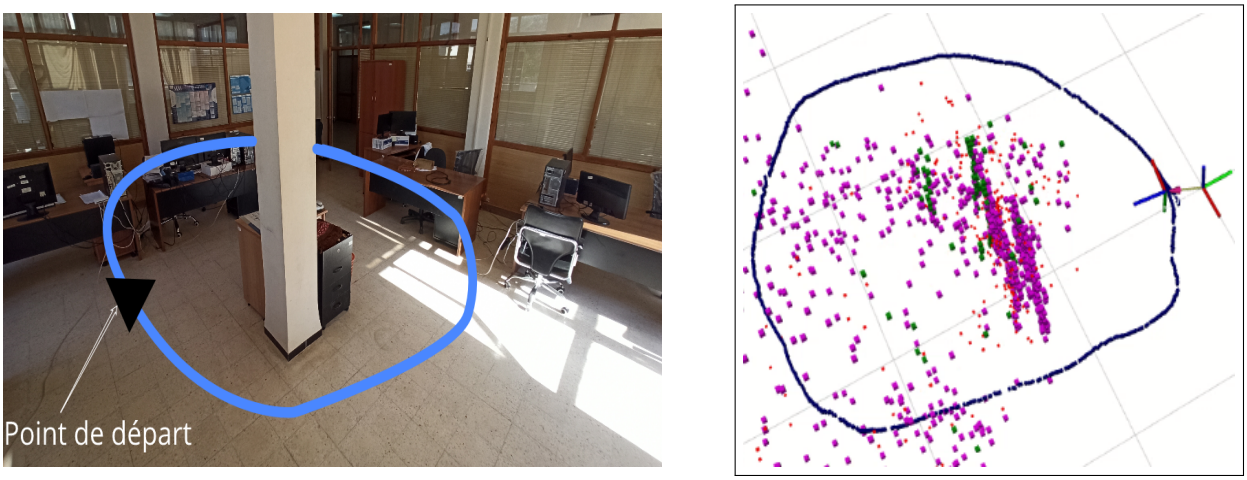


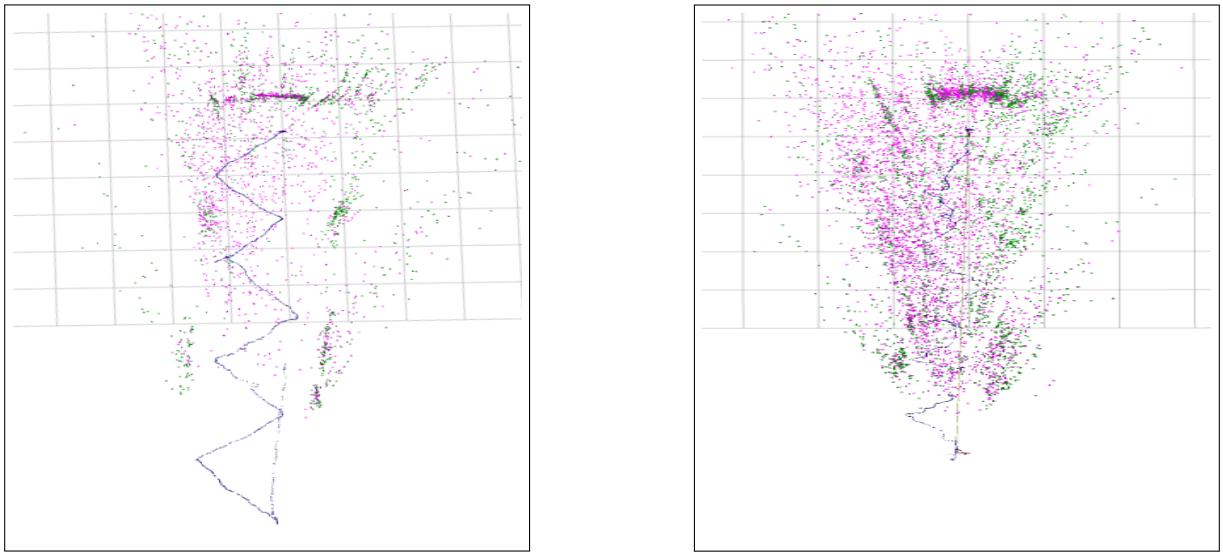
FIGURE 5.13 – *Trajectoire en cercle*

La trajectoire circulaire vise à étudier un autre aspect de l'estimation de la position, elle est caractérisée par le changement de vue durant le trajet. Ceci-dit, nous observons un décalage de positionnement lors du retour à la position de départ de 12cm par rapport à une distance parcourue d'environs 8m, ce qui nous donne un pourcentage d'erreur de 1.5% qui est un ratio acceptable pour notre cas d'utilisation.

Effet des paramètres sur l'estimation de la position : l'algorithme de perception dépend de plusieurs paramètres pour accomplir son estimation. Jusqu'à présent, nous avons travaillé avec un ensemble de paramètres qui correspondent le mieux avec notre environnement d'exécution qui ont été fixé de façon empirique.
Dans ce qui suit, nous modifions quelques paramètres pour observer leurs effets sur l'estimation obtenue lors de la navigation. Le but de ces expérimentations est de fournir une étude partielle de l'algorithme SVO sur les aspects qui peuvent intervenir dans notre cas d'utilisation.
Pour accomplir ces expériences, nous avons effectué des essais sur les trajectoires déjà cités et enregistré les données recueillis à partir des différents capteurs dans des fichiers de type *bag* de ROS[10] qu'il est possible d'exécuter exécuter après pour refaire jouer l'expérience. Ceci va nous permettre de garantir une simulation dans les conditions réelles.

1. ***Motion prior* :** le *Motion prior* est la partie qui permet la fusion des signaux de la centrale inertielle avec l'estimation de la perception visuelle. Comme indiqué dans la section 3.7.4, il vient apporter une correction à cette dernière.

10. Plus de détails sur : `http://wiki.ros.org/rosbag`.

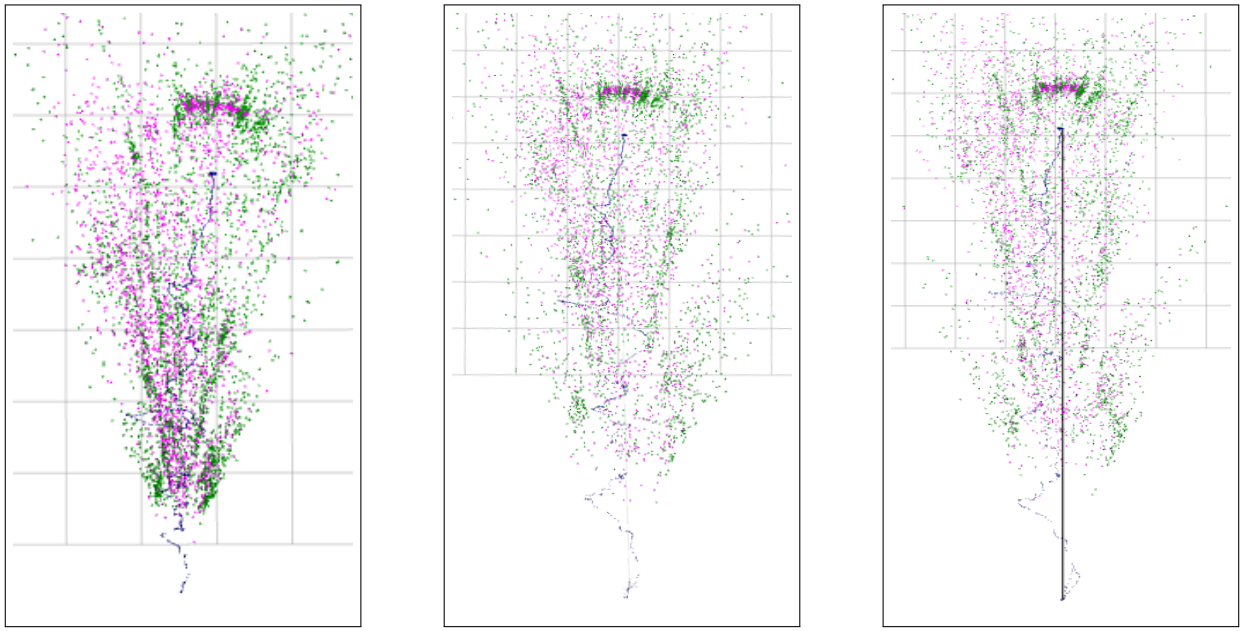


Motion prior intégré Sans *Motion prior*

FIGURE *5.14 – Étude de l'effet du motion prior sur l'estimation*

L'effet des données IMU sur le positionnement paraît clairement sur l'estimation présente dans la figure 5.14, la trajectoire n'est pas estimée correctement ce qui veut dire que la position estimée n'est pas fiable. Ceci peut être principalement dû à la faible résolution de la caméra et à la basse fréquence d'images.
L'erreur sur la profondeur engendre une déformation dans l'environnement reconstruit par les points 3D.

2. **Nombre de Keyframes à détecter :** comme indiqué dans 3.7.1, la sélection des *keyframes* affecte à la fois la position du système et la précision des points 3D estimés. Afin de voir concrètement l'effet de ce paramètre, nous effectuons des essais en tenant compte uniquement de la perception visuelle (sans prendre en considération les signaux IMU).
Cependant, le critère de sélection dépend entre-autres d'une mesure de disparité entre les images qui se suivent, une nouvelle *keyframe* est sélectionnée quand cette mesure atteint un seuil prédéfini (un grand seuil de disparité signifie moins de keyframes et vice versa). Nous modifions la valeur de ce seuil de disparité pour avoir les résultats de la Figure 5.15

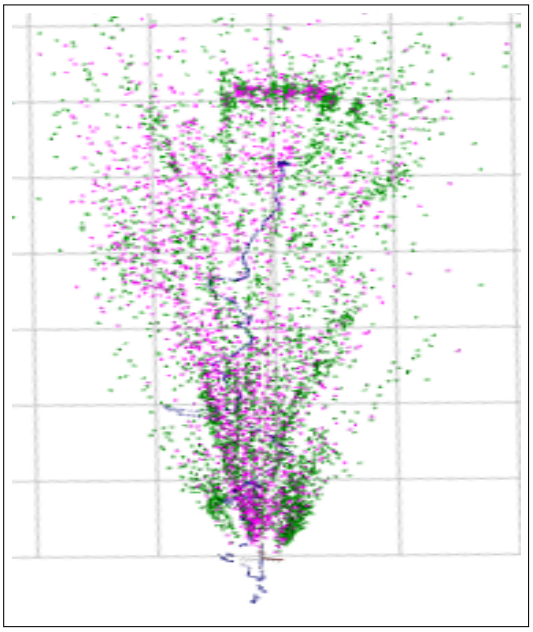


Critère de disparité élevée (plus de *keyframes*) | Critère de disparité moyen | Critère de disparité bas (moins de keyframes)

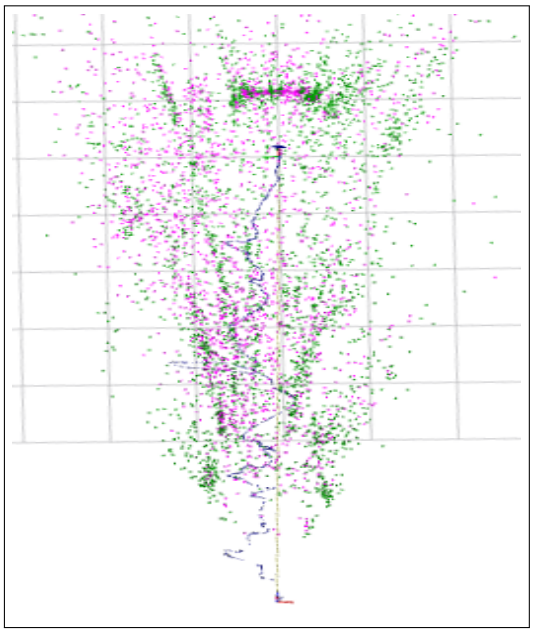
FIGURE *5.15 – Étude de l'effet du nombre des keyframes sur l'estimation.*

Nous constatons que la précision des points 3D estimés augmente avec la diminution du nombre de *keyframes*. Pour peu de *keyframes* nous obtenons des nuages de points moins denses mais plus précis.
Ce paramètre intervient lors de l'alignement des descripteurs dans l'algorithme. Un nombre élevé de *keyframes* engendre une distance plus petite entre eux lors de la mise en correspondance, et qui engendre par conséquent une erreur de reprojection plus grande.

3. **Seuil de convergence des points 3D estimés :** comme indiqué dans la section 3.7.3, un point est publié dans la carte quand son incertitude descend en dessous d'un certain seuil de convergence.
   Dans cette partie, nous verrons l'effet qu'engendre ce paramètre sur le fonctionnement de l'algorithme en désactivant le *motion prior*.



Seuil élevé (attitude peu sélective)

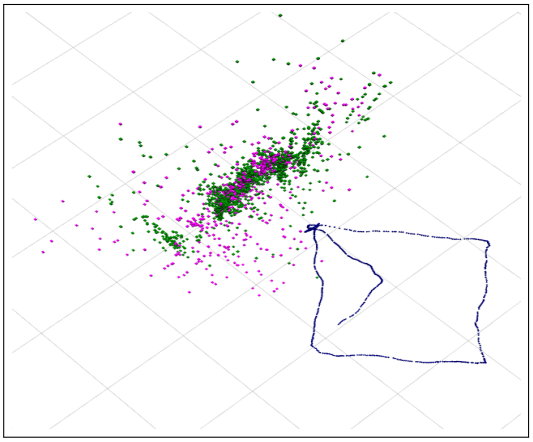


Seuil bas (attitude très sélective)

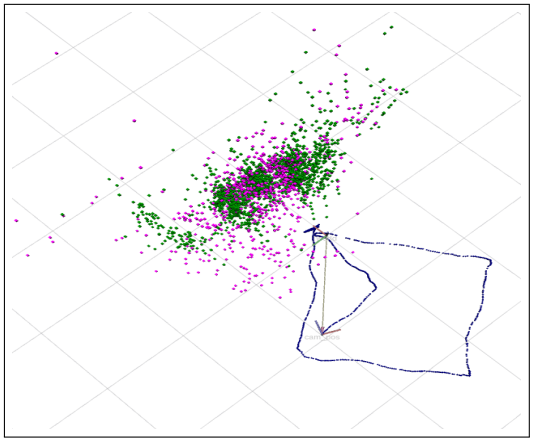
FIGURE 5.16 – *Étude de l'effet de la sélectivité des points 3D.*

Comme nous pouvons le constater dans la Figure 5.16, l'attitude très sélective engendre moins de points qui constituent l'environnement perçu mais améliore la précision de l'environnement engendré. Ce paramètre dépend de la résolution et la qualité de la caméra et son habilité à fournir des signaux qui permettent une estimation fiable. Il peut être considéré comme correcteur d'estimation pour les systèmes d'instrumentation non fiables.

4. **Mode normal et mode rapide :** deux paramètres permettent de prévoir un fonctionnement en mode rapide de l'algorithme. Ces paramètres sont le nombre maximum de features détectables par image et le nombre des derniers keyframes à garder en mémoire lors de la navigation (en diminuant ces valeurs on se retrouve dans le mode rapide).
   Nous avons effectué deux essais sur la trajectoire carrée, en activant le *motion prior*, pour observer l'effet de ces paramètres sur l'estimation des points 3D.



Mode normal



Mode rapide

FIGURE 5.17 – *Comparaison entre les modes de fonctionnement de l'algorithme SVO.*

Dans un mode rapide les images sont traitées à une fréquence plus élevée, ce qui veut dire que pour une même expérience on obtient plus de points 3D estimés mais ces points seront entaché d'une plus grande erreur.
La figure 5.17 illustre ce concept. Nous remarquons que le nuage de points dans le mode rapide est plus dense et volumineux que dans le mode normal, nous observons donc plus de points estimés mais qui sont entachés d'erreurs.
Ceci montre que le mode rapide est plus efficace en termes de calcul en dépit de la précision des points 3D estimés.

Discussion sur les performance de l'algorithme de perception

Plusieurs limites prévisibles de l'algorithme ont été observées durant les essais qui conduisent à des estimations erronées. L'un des principaux problèmes des algorithmes de perception est observé avec les images à faible gradient (dites lisses qui ne contiennent pas beaucoup de détails). Il se manifeste par l'impossibilité d'extraire des descripteurs à partir des ces images, ce qui engendre une plus grande erreur dans la mise en correspondance entre les images et entraine plus d'erreur dans l'estimation.
Par ailleurs, notre cas d'utilisation comporte une navigation dans des milieux endommagés, fissurés, non ordonnées et remplies de débris ; les images prises seront donc riches en détails. Ceci augmente la fiabilité de l'algorithme de perception et apporte une justification à notre choix d'utiliser cette approche.

De plus, à travers ces résultats nous constatons que l'algorithme de perception respecte deux critères posés dans la problématique de l'intervention des drones dans les catastrophes naturelles, qui sont l'efficacité de positionnement et la fiabilité de l'information. En plus des défies relevés dans le cadre des interventions directes et de la mesure des dégâts.

5.3.2 Procédure d'exploration

La procédure d'exploration sert à guider le drone durant l'exploration d'un milieu inconnu. Nous présentons une implémentation de l'algorithme **NBV Planner** proposée dans le chapitre 4.4, simulée à l'aide du simulateur de drones **RotorS** de **RPG** [11] [113], qui repose sur **Gazebo** [12] comme noyau de simulation.
Suivant le schéma indiqué dans la figure 4.1, l'algorithme d'exploration reçoit en entrée la map d'environnement sous format OctoMap, et l'attitude (position et orientation) sous format TF2 [13], qui sont les formats conventionnels de ROS. L'implémentation de l'algorithme d'exploration expliqué dans le chapitre 4 a été faite selon la configuration logicielle et matérielle suivante :

- *Ubuntu 14.04 trusty* comme système d'exploitation correspondant avec la distribution ROS.
- *ROS Indigo* selon la recommandation des développeurs du simulateur pour assurer une compatibilité avec les différents composants de ce dernier.

La figue 5.18 fournit une illustration de l'exécution de l'algorithme qui assure la procédure d'exploration à travers l'outil **rqt_graph** (annexe D) :

11. Robotics and Perception Group : http://rpg.ifi.uzh.ch/.
12. Logiciel de simulation des robots : http://gazebosim.org/.
13. Plus de détails dans : http://wiki.ros.org/tf.

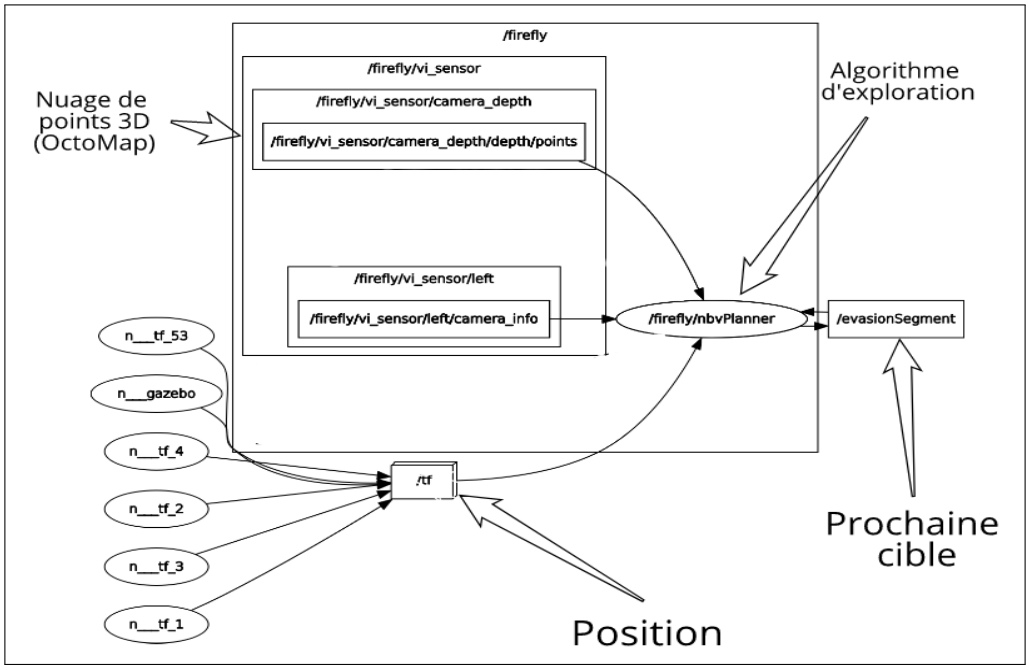
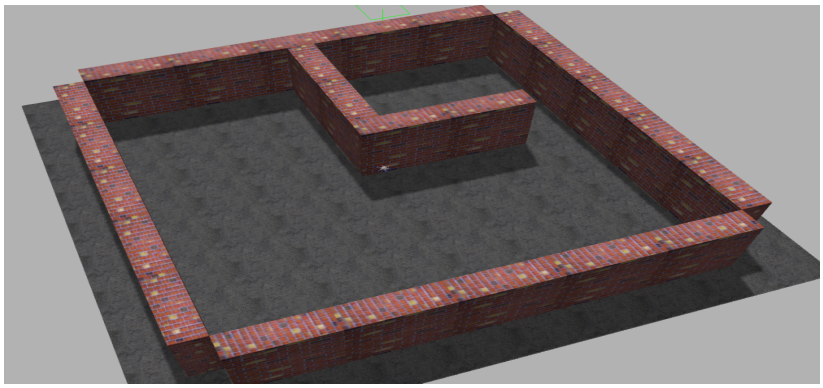
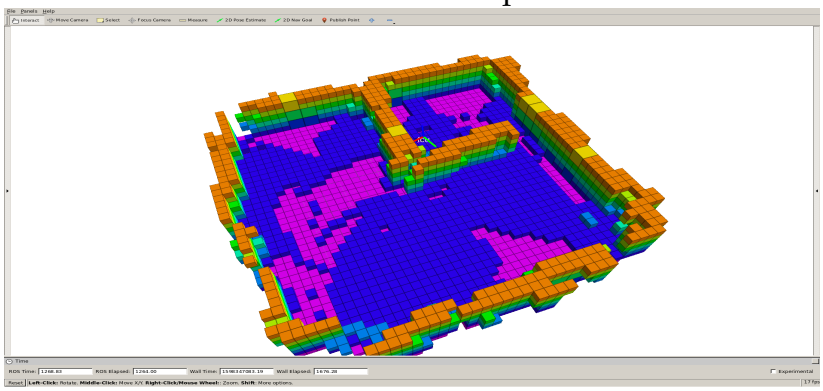


FIGURE *5.18 – Configuration d'exécution de la procédure d'exploration.*

La queue de messages **/evasionSegment** stocke les prochains points cibles à atteindre qui proviennent de l'algorithme. Ce dernier utilise la position fournie par la queue de messages **/tf** et le nuage de points qui constituent la carte de l'environnement pour générer les cibles. De plus, l'algorithme peut extraire des informations statiques sur la caméra (caractéristiques intrinsèques et extrinsèques et données de la calibration) à partir de la queue de messages **/camera_info**.



Environement à explorer



Résultat de l'exploration

FIGURE *5.19 – Simulation de la procédure d'exploration*

La figure 5.19 montre le résultat de la simulation de l'exploration d'une zone de dimensions de $16 * 16 * 3$ mètres en 6 : 14 minutes. Cette procédure d'exploration est caractérisé par un critère de sélection des prochaines cibles défini comme étant l'estimation du volume qui peut être exploré si le drone atteint le point.

Cet algorithme a été choisi pour sa capacité à garantir l'efficacité énergétique qui a été citée comme critère dans la problématique (premier chapitre). En effet, lors de l'exploration l'algorithme garde toujours une trace de la meilleur branche possible du graphe en tenant compte du critère du volume pouvant être exploré, ce qui lui permet d'optimiser la distance parcourue lors de l'exploration de l'intégralité de l'espace cible.

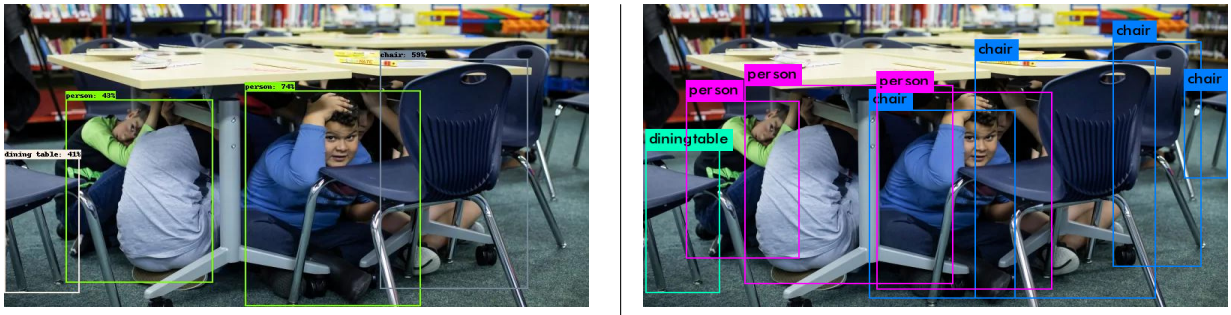
5.3.3 Détection des objets

Nous avons testé l'algorithme de détection des objets EfficientDet (présenté dans le chapitre 2.5.6) sur des images prises lors de séismes pour avoir une mesure qualitative de son efficacité à détecter les objets dans ces milieux.
Une implémentation de l'algorithme avec un modèle entrainé (version D0 de l'algorithme) sur la base de données ImageNet a été utilisée pour les essais [14], sous la configuration logicielle et matérielle suivante :

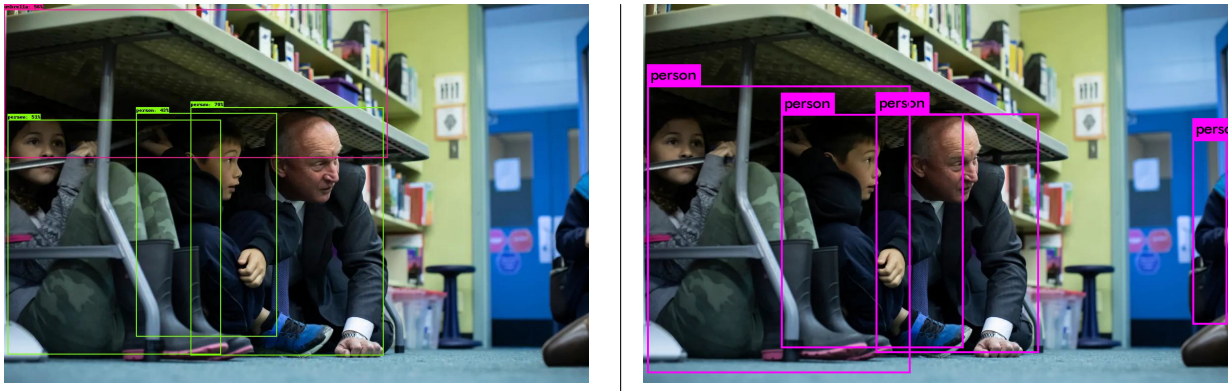
14. Disponible sur https://github.com/google/automl/tree/master/efficientdet.

- Une station GPU donnant accès à une carte graphique **Nvidia Geforce RTX 2080 Ti**.
- **Ubuntu 18.04 Bionic** comme système d'exploitation supportant les derniers drivers de la carte graphique.
- Librairies **CUDA 10.1** et **cuDNN 7** pour la gestion de la carte graphique.

Nous comparons les résultats obtenus avec des prédictions obtenues à l'aide de l'algorithme YOLOV3 :



EfficientDet-do | YoloV3



EfficientDet-do | YoloV3

EfficientDet-do | YoloV3

FIGURE 5.20 – *Résultats de la détection utilisant les deux algorithmes EfficientDet-D0 et YOLOV3.*

Comme indiqué dans la section 2.5, l'architecture *efficientdet-d0* accomplit des performances de précision proches de YOLOV3 tout en étant **28 fois** plus légère [83], ce qui la rend plus adaptée pour un système à ressources limitées comme notre contexte d'utilisation qui concerne l'emploi de drones avec des capacités de calcul et de traitement modérés. Ceci remplit le critère de fiabilité de l'information posé dans la problématique.

5.4 Conclusion

Fournir une solution pour un drone autonome nécessite une architecture où les différentes parties doivent fonctionner indépendamment en toute compatibilité. Cette configuration doit permettre de remplacer n'importe quelle partie du système sans influencer les autres, afin d'assurer la flexibilité du système.

La perception visuelle de l'environnement est la brique la plus importante dans cette architecture puisqu'elle assure l'interaction avec l'environnement externe, et c'est pour ça qu'elle doit être bien étudiée. Les essais ont révélés que cette perception dépend de plusieurs paramètres qui assurent son adaptation avec l'environnement matériel. Ceci dit, une bonne maîtrise des différents concepts de l'algorithme de perception est nécessaire si on veut l'adapter à un environnement d'exécution.

Dans ce chapitre, nous avons présenté l'architecture globale que nous avons proposée, ainsi que les aspects techniques de l'implémentation de notre solution pour la problématique d'assistance dans la recherche et le sauvetage lors de catastrophes naturelles. Aussi, nous avons présenté quantitativement les résultats et les performances des différents systèmes impliqués.

Chapitre 6

Conclusion générale

Motivé par l'engouement croissant autour de l'intelligence artificielle et la navigation visuelle des drones, nous avons tenté à travers ce travail d'apporter une contribution à un domaine d'une grande importance qui est la gestion des catastrophes naturelles.
En effet ce travail se divise en trois parties à savoir :

1. Présentation d'une problématique qui peut être résolue à travers des systèmes autonomes robotisés, suivie d'une étude de l'état de l'art et des accomplissement dans le domaine.
2. Étude détaillée des technologies utilisées et méthodes employées pour apporter une solution à la problématique.
3. Description de la solution proposée, suivi d'une validation de ses différents composants à travers des tests réels et simulations.

La solution proposée nécessite la présence de deux capteurs uniquement et peut être déployée dans n'importe quelle plateforme mobile (drone à voilure tournante, robot différentiel, drone à voilure fixe, etc.) sans exiger une grande capacité de calcul. Elle est constituée de trois parties :

1. Un algorithme de perception visuelle **SVO**, qui s'appuie à la fois sur l'odométrie visuelle pour estimer le mouvement et sur le concept de la *structure from motion* pour établir la cartographie.
2. Une procédure d'exploration appliquant le principe *next best view*
3. La toute toute dernière architecture de détection des objets de *Google AI* nommée *"EfficientDet-D0"*

Chaque partie de l'algorithme a été validée après des tests réels et des simulations afin d'assurer la compatibilité entre les différents processus.

Perspectives

Afin d'étendre ce travail, nous proposons :

- Établir une technique d'exploration collaborative qui permet de déployer plusieurs drones en coopération.
- L'utilisation d'un algorithme de planification de trajectoire à partir de la représentation de l'environnement pour déduire le meilleur chemin qui mène vers les victimes.

- L'utilisation d'une caméra événementielle [26] à la place de la caméra à sténopé pour plus de précision.
- Utilisation d'une caméra thermique dédié à la détection des humains à travers la chaleur qu'ils dégagent.
- Inférer une procédure d'exploration en tenant compte des informations fournies par le bloc de détection des objets.
- Utiliser un modèle de drone pliable [114] capable de passer dans différents petites ouvertures pour augmenter l'efficacité d'exploration.
- Utiliser une architecture de détection des objets optimisée [82] et entrainée sur des images de catastrophes naturelles.

Références

[1] Danica Kragic et Kostas Daniilidis. « 3-D Vision for Navigation and Graspin ». In : *Springer Handbook of Robotics* (2016), p. 811-824.

[2] Ankit Ravankar, Abhijeet Ravankar, Yukinori Kobayashi et Takanori Emaru. « Autonomous Mapping and Exploration with Unmanned Aerial Vehicles Using Low Cost Sensors ». In : *Proceedings* 4.1 (14 nov. 2018), p. 44.

[3] Rakesh Shrestha. « 3D visual-inertial odometry and autonomous mobile robot exploration with learned map prediction ». In : 2018.

[4] Sunggoo Jung, Sunyou Hwang, Heemin Shin et David Hyunchul Shim. « Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning ». In : *IEEE Robotics and Automation Letters* 3.3 (juil. 2018), p. 2539-2544.

[5] Y. Bouhadad, A. Nour, A. Slimani, N. Laouami et D. Belhai. « The Boumerdes (Algeria) earthquake of May 21, 2003 (Mw = 6.8) : Ground deformation and intensity ». In : *Journal of Seismology* 8.4 (1er oct. 2004), p. 497-506.

[6] Margarita Poteyeva, Megan Denver, Lauren E. Barsky et Benigno E. Aguirre. « Search and Rescue Activities in Disasters ». In : *Handbook of Disaster Research*. New York, NY : Springer New York, 2007, p. 200-216.

[7] Milan Erdelj, Enrico Natalizio, Kaushik R. Chowdhury et Ian F. Akyildiz. « Help from the Sky : Leveraging UAVs for Disaster Management ». In : *IEEE Pervasive Computing* 16.1 (jan. 2017), p. 24-32.

[8] Nathan Michael, Shaojie Shen, Kartik Mohta, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi et Satoshi Tadokoro. « Collaborative Mapping of an Earthquake Damaged Building via Ground and Aerial Robots ». In : *Field and Service Robotics : Results of the 8th International Conference*. Sous la dir. de Kazuya Yoshida et Satoshi Tadokoro. Springer Berlin Heidelberg, 2014, p. 33-47.

[9] Masahiko Onosato, Fumiaki Takemura, Kenzo Nonami, Kuniaki Kawabata, Kenjiro Miura et Hiroaki Nakanishi. « Aerial Robots for Quick Information Gathering in USAR ». In : *2006 SICE-ICASE International Joint Conference*. Oct. 2006, p. 3435-3438.

[10] Fumitoshi Matsuno, Noritaka Sato, Kazuyuki Kon, Hiroki Igarashi, Tetsuya Kimura et Robin Murphy. « Utilization of Robot Systems in Disaster Sites of the Great Eastern Japan Earthquake ». In : *Field and Service Robotics : Results of the 8th International Conference*. Sous la dir. de Kazuya Yoshida et Satoshi Tadokoro. Berlin, Heidelberg : Springer Berlin Heidelberg, 2014, p. 1-17.

[11] Kartik Mohta, Michael Watterson, Yash Mulgaonkar, Sikang Liu, Chao Qu, Anurag Makineni, Kelsey Saulnier, Ke Sun, Alex Zhu, Jeffrey Delmerico, Konstantinos Karydis, Nikolay Atanasov, Giuseppe Loianno, Davide Scaramuzza, Kostas Daniilidis, Camillo Jose Taylor et Vijay Kumar. « Fast, autonomous flight in GPS-denied and cluttered environments ». In : *Journal of Field Robotics* 35.1 (jan. 2018), p. 101-120.

[12] Charles Richter et Nicholas Roy. « Safe Visual Navigation via Deep Learning and Novelty Detection ». In : *Robotics : Science and Systems XIII*. Robotics : Science et Systems Foundation, 12 juil. 2017.

[13] Davide Falanga, Kevin Kleber et Davide Scaramuzza. « Dynamic obstacle avoidance for quadrotors with event cameras ». In : *Science Robotics* 5.40 (18 mar. 2020).

[14] Sertac Karaman et Emilio Frazzoli. « High-speed flight in an ergodic forest ». In : *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, p. 2899-2906.

[15] Titus Cieslewski, Elia Kaufmann et Davide Scaramuzza. « Rapid exploration with multi-rotors : A frontier selection method for high speed flight ». In : *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC : IEEE, sept. 2017, p. 2135-2142.

[16] Valentina Gatteschi, Fabrizio Lamberti, Gianluca Paravati, Andrea Sanna, Claudio Demartini, Alberto Lisanti et Giorgio Venezia. « New Frontiers of Delivery Services Using Drones : A Prototype System Exploiting a Quadcopter for Autonomous Drug Shipments ». In : *2015 IEEE 39th Annual Computer Software and Applications Conference*. T. 2. Juil. 2015, p. 920-927.

[17] Kalyani Singh, Isha Chaudhary, Archana Kumari, Bhojraj Singh, Monica Kathuria et UG Scholar. « Human Live Detection Robot used in Natural Disasters ». In : *International Journal of Engineering Science and Computing IJESC* 6.4 (2016), p. 4.

[18] Gagandeep Singh Khanuja. « A study of real time search in flood scenes from UAV videos using Deep Learning techniques ». Mémoire de Master. Department of Computer et Information Technology, West Lafayette, Indianna, déc. 2019, p. 98.

[19] Anna Gaszczak, Toby P. Breckon et Jiwan Han. « Real-time people and vehicle detection from UAV imagery ». In : *Intelligent Robots and Computer Vision XXVIII : Algorithms and Techniques*. Sous la dir. de Juha Röning, David P. Casasent et Ernest L. Hall. T. 7878. International Society for Optics et Photonics. SPIE, 2011, p. 71-83.

[20] Patrick Doherty et Piotr Rudol. « A UAV Search and Rescue Scenario with Human Body Detection and Geolocalization ». In : *AI 2007 : Advances in Artificial Intelligence*. Sous la dir. de Mehmet A. Orgun et John Thornton. T. 4830. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, p. 1-13.

[21] Gurkan Tuna, V. Cagri Gungor et Kayhan Gulez. « An autonomous wireless sensor network deployment system using mobile robots for human existence detection in case of disasters ». In : *Ad Hoc Networks* 13 (fév. 2014), p. 54-68.

[22] Carlos Sampedro, Alejandro Rodriguez-Ramos, Hriday Bavle, Adrian Carrio, Paloma de la Puente et Pascual Campoy. « A Fully-Autonomous Aerial Robot for Search and Rescue Applications in Indoor Environments using Learning-Based Techniques ». In : *Journal of Intelligent & Robotic Systems* 95.2 (août 2019), p. 601-627.

[23] Yingcai Bi, Menglu Lan, Jiaxin Li, Shupeng Lai et Ben M. Chen. « A lightweight autonomous MAV for indoor search and rescue ». In : *Asian Journal of Control* 21.4 (juil. 2019), p. 1732-1744.

[24] Ellen Schwalbe. « Geometric modelling and calibration of fisheye lens camera systems ». In : jan. 2005, p. 6.

[25] Shinko Yuanhsien Cheng et Mohan Manubhai Trivedi. « Lane Tracking with Omnidirectional Cameras : Algorithms and Evaluation ». In : *EURASIP Journal on Embedded Systems* 2007 (2007), p. 1-8.

[26] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis et Davide Scaramuzza. « Event-based Vision : A Survey ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* (août 2020), p. 1-1.

[27] Davide Scaramuzza, Agostino Martinelli et Roland Siegwart. « Tutorial on Event-based Cameras : » in : *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Beijing, China : IEEE, oct. 2006, p. 5695-5701.

[28] H. Durrant-Whyte et T. Bailey. « Simultaneous localization and mapping : part I ». In : *IEEE Robotics Automation Magazine* 13.2 (2006), p. 99-110.

[29] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton et Olivier Stasse. « MonoSLAM : Real-Time Single Camera SLAM ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (juin 2007), p. 1052-1067.

[30] Jakob Engel, Thomas Schöps et Daniel Cremers. « LSD-SLAM : Large-Scale Direct Monocular SLAM ». In : *Computer Vision – ECCV 2014*. Sous la dir. de David Fleet, Tomas Pajdla, Bernt Schiele et Tinne Tuytelaars. Lecture Notes in Computer Science. Cham : Springer International Publishing, 2014, p. 834-849.

[31] Raúl Mur-Artal et Juan D. Tardós. « ORB-SLAM2 : An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras ». In : *IEEE Transactions on Robotics* 33.5 (oct. 2017), p. 1255-1262.

[32] Ethan Rublee, Vincent Rabaud, Kurt Konolige et Gary Bradski. « ORB : An efficient alternative to SIFT or SURF ». In : *2011 International Conference on Computer Vision*. Barcelona, Spain : IEEE, nov. 2011, p. 2564-2571.

[33] Henri Rebecq, Timo Horstschaefer, Guillermo Gallego et Davide Scaramuzza. « EVO : A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time ». In : *IEEE Robotics and Automation Letters* 2.2 (avr. 2017), p. 593-600.

[34] David Weikersdorfer, David B. Adrian, Daniel Cremers et Jorg Conradt. « Event-based 3D SLAM with a depth-augmented dynamic vision sensor ». In : *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China : IEEE, mai 2014, p. 359-364.

[35] Stefan Kohlbrecher, Oskar Von Stryk, Technische Universität Darmstadt, Johannes Meyer et Uwe Klingauf. « A flexible and scalable slam system with full 3d motion estimation ». In : *in International Symposium on Safety, Security, and Rescue Robotics. IEEE*. 2011.

[36] Merouane Guettache et Yacine Ben Ameur. « Perception et Navigation visuelle d'un quadrirotor en utilisant des techniques d'apprentissage profond ». Mémoire de Master. Département d'Automatique, Ecole Nationale Polytechnique, Alger, juil. 2019, p. 104.

[37] David Nistér, Oleg Naroditsky et James Bergen. « Visual Odometry ». In : *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR'04)*. 2004.

[38] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger et Davide Scaramuzza. « SVO : Semidirect Visual Odometry for Monocular and Multicamera Systems ». In : *IEEE Transactions on Robotics* 33.2 (avr. 2017), p. 249-265.

[39] Christian Forster, Matia Pizzoli et Davide Scaramuzza. « SVO : Fast semi-direct monocular visual odometry ». In : *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China : IEEE, mai 2014, p. 15-22.

[40] Henri Rebecq, Timo Horstschaefer et Davide Scaramuzza. « Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization ». In : *Procedings of the British Machine Vision Conference 2017*. London, UK : British Machine Vision Association, 2017, p. 16.

[41] J L Barron et N A Thacker. « Tutorial : Computing 2D and 3D Optical Flow. » In : *Tina internal Memo* 2004-012 (2005), p. 12.

[42] Fendy Santoso, Matthew A. Garratt et Sreenatha G. Anavatti. « Visual–Inertial Navigation Systems for Aerial Robotics : Sensor Fusion and Technology ». In : *IEEE Transactions on Automation Science and Engineering* 14.1 (jan. 2017), p. 260-275.

[43] Simon Lynen, Torsten Sattler, Michael Bosse, Joel Hesch, Marc Pollefeys et Roland Siegwart. « Get Out of My Lab : Large-scale, Real-Time Visual-Inertial Localization ». In : *Robotics : Science and Systems XI*. Robotics : Science et Systems Foundation, 13 juil. 2015.

[44] Céline Teulière, Eric Marchand et Laurent Eck. « 3-D Model-Based Tracking for UAV Indoor Localization ». In : *IEEE Transactions on Cybernetics* 45.5 (mai 2015), p. 869-879.

[45] A.I. Comport, E. Marchand, M. Pressigout et F. Chaumette. « Real-time markerless tracking for augmented reality : the virtual visual servoing framework ». In : *IEEE Transactions on Visualization and Computer Graphics* 12.4 (juil. 2006), p. 615-628.

[46] Alex Kendall, Matthew Grimes et Roberto Cipolla. « PoseNet : A Convolutional Network for Real-Time 6-DOF Camera Relocalization ». In : *2015 IEEE International Conference on Computer Vision (ICCV)*. Déc. 2015, p. 2938-2946.

[47] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers et Thomas Brox. « FlowNet : Learning Optical Flow with Convolutional Networks ». In : *2015 IEEE International Conference on Computer Vision (ICCV)*. Santiago : IEEE, déc. 2015, p. 2758-2766.

[48] Kishore Konda et Roland Memisevic. « Learning Visual Odometry with a Convolutional Network : » in : *Proceedings of the 10th International Conference on Computer Vision Theory and Applications*. Berlin, Germany : SCITEPRESS - Science, 2015, p. 486-490.

[49] Jason R. Rambach, Aditya Tewari, Alain Pagani et Didier Stricker. « Learning to Fuse : A Deep Learning Approach to Visual-Inertial Camera Pose Estimation ». In : *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Merida, Yucatan, Mexico : IEEE, sept. 2016, p. 71-76.

[50] Abhinav Valada, Noha Radwan et Wolfram Burgard. « Deep Auxiliary Learning for Visual Localization and Odometry ». In : *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD : IEEE, mai 2018, p. 6939-6946.

[51] J. Maver et R. Bajcsy. « Occlusions as a guide for planning the next view ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.5 (mai 1993), p. 417-433.

[52] C. Connolly. « The determination of next best views ». In : *1985 IEEE International Conference on Robotics and Automation Proceedings*. T. 2. Mar. 1985, p. 432-435.

[53] Brian Yamauchi. « Frontier-based exploration using multiple robots ». In : *Proceedings of the second international conference on Autonomous agents - AGENTS '98*. Minneapolis, Minnesota, United States : ACM Press, 1998, p. 47-53.

[54] Héctor H. González-Baños et Jean-Claude Latombe. « Navigation Strategies for Exploring Indoor Environments : » in : *The International Journal of Robotics Research* (4 déc. 2016).

[55] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss et Wolfram Burgard. « OctoMap : an efficient probabilistic 3D mapping framework based on octrees ». In : *Autonomous Robots* 34.3 (avr. 2013), p. 189-206.

[56] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen et Marc Pollefeys. « Vision-based autonomous mapping and exploration using a quadrotor MAV ». In : *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2012, p. 4557-4564.

[57] Shaojie Shen, Nathan Michael et Vijay Kumar. « Stochastic differential equation-based exploration algorithm for autonomous indoor 3D exploration with a micro-aerial vehicle ». In : *The International Journal of Robotics Research* 31.12 (1er oct. 2012), p. 1431-1444.

[58] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova et Roland Siegwart. « Receding Horizon "Next-Best-View" Planner for 3D Exploration ». In : *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden : IEEE, mai 2016, p. 1462-1468.

[59] Sertac Karaman et Emilio Frazzoli. « Sampling-based algorithms for optimal motion planning ». In : *The International Journal of Robotics Research* 30.7 (juin 2011), p. 846-894.

[60] Alexei A. Makarenko, Stefan B. Williams, Frederic Bourgault et Hugh F. Durrant-Whyte. « An Experiment in Integrated Exploration ». In : *In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2002, p. 534-539.

[61] F. Bourgault, A.A. Makarenko, S.B. Williams, B. Grocholsky et H.F. Durrant-Whyte. « Information based adaptive robotic exploration ». In : *IEEE/RSJ International Conference on Intelligent Robots and Systems*. T. 1. Sept. 2002, 540-545 vol.1.

[62] Robert Sim et James J. Little. « Autonomous vision-based robotic exploration and mapping using hybrid maps and particle filters ». In : *Image and Vision Computing* 27.1 (1er jan. 2009), p. 167-177.

[63] R. Sim, G. Dudek et N. Roy. « Online control policy optimization for minimizing map uncertainty during exploration ». In : *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. T. 2. Avr. 2004, p. 1758-1763.

[64] Ruben Martinez-Cantin, Nando de Freitas, Eric Brochu, José Castellanos et Arnaud Doucet. « A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot ». In : *Autonomous Robots* 27.2 (1er août 2009), p. 93-103.

[65] Thomas Kollar et Nicholas Roy. « Trajectory Optimization using Reinforcement Learning for Map Exploration : » in : *The International Journal of Robotics Research* (1er fév. 2008).

[66] W. Burgard, M. Moors, D. Fox, R. Simmons et S. Thrun. « Collaborative multi-robot exploration ». In : *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*. T. 1. Avr. 2000, p. 476-481.

[67] W. Burgard, M. Moors, C. Stachniss et F.E. Schneider. « Coordinated multi-robot exploration ». In : *IEEE Transactions on Robotics* 21.3 (juin 2005), p. 376-386.

[68] Miguel Julia, Oscar Reinoso, Arturo Gil, Monica Ballesta et Luis Paya. « Behaviour Based Multi-Robot Integrated Exploration ». In : *International journal of innovative computing, information control : IJICIC* (sept. 2011), p. 21.

[69] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg et Li Fei-Fei. « ImageNet Large Scale Visual Recognition Challenge ». In : *International Journal of Computer Vision* 115.3 (1er déc. 2015), p. 211-252.

[70] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton. « ImageNet Classification with Deep Convolutional Neural Networks ». In : *Advances in Neural Information Processing Systems 25*. Sous la dir. de F. Pereira, C. J. C. Burges, L. Bottou et K. Q. Weinberger. Curran Associates, Inc., 2012, p. 1097-1105.

[71] Ethan Zhang et Yi Zhang. « Average Precision ». In : *Encyclopedia of Database Systems*. Sous la dir. de LING LIU et M. TAMER ÖZSU. Boston, MA : Springer US, 2009, p. 192-193.

[72] Joseph Redmon, Santosh Divvala, Ross Girshick et Ali Farhadi. « You Only Look Once : Unified, Real-Time Object Detection ». In : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (9 mai 2016).

[73] Joseph Redmon et Ali Farhadi. « YOLO9000 : Better, Faster, Stronger ». In : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (25 déc. 2016).

[74] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu et Alexander C. Berg. « SSD : Single Shot MultiBox Detector ». In : *Computer Vision – ECCV 2016*. Sous la dir. de Bastian Leibe, Jiri Matas, Nicu Sebe et Max Welling. Lecture Notes in Computer Science. Cham : Springer International Publishing, 2016, p. 21-37.

[75] Joseph Redmon et Ali Farhadi. « YOLOv3 : An Incremental Improvement ». In : *arXiv :1804.02767 [cs]* abs/1804.02767 (8 avr. 2018).

[76] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He et P. Dollár. « Focal Loss for Dense Object Detection ». In : *2017 IEEE International Conference on Computer Vision (ICCV)* (7 fév. 2018), p. 2999-3007.

[77] Alexey Bochkovskiy, Chien-Yao Wang et Hong-Yuan Mark Liao. « YOLOv4 : Optimal Speed and Accuracy of Object Detection ». In : *arXiv :2004.10934 [cs, eess]* (22 avr. 2020).

[78] Chien-Yao Wang, Hong-Yuan Mark Liao, I.-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen et Jun-Wei Hsieh. « CSPNet : A New Backbone that can Enhance Learning Capability of CNN ». In : *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (26 nov. 2019), p. 1571-1580.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. « Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8691 (2014), p. 346-361.

[80] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi et Jiaya Jia. « Path Aggregation Network for Instance Segmentation ». In : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (18 sept. 2018), p. 8759-8768.

[81] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan et Serge Belongie. « Feature Pyramid Networks for Object Detection ». In : *arXiv :1612.03144 [cs]* (19 avr. 2017).

[82] Abderrahim Benaouda et Anouar Laouichi. « Deep Neural Networks Optimization for Embedded Platforms ». Mémoire de Master. Département d'Electronique, Ecole Nationale Polytechnique, Alger, juil. 2020, p. 94.

[83] Mingxing Tan, Ruoming Pang et Quoc V. Le. « EfficientDet : Scalable and Efficient Object Detection ». In : *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (14 juin 2020), p. 10778-10787.

[84] Mingxing Tan et Quoc V. Le. « EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks ». In : *arXiv :1905.11946 [cs, stat]* (22 nov. 2019).

[85] Golnaz Ghiasi, Tsung-Yi Lin, Ruoming Pang et Quoc V. Le. « NAS-FPN : Learning Scalable Feature Pyramid Architecture for Object Detection ». In : *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (15 avr. 2019), p. 7029-7038.

[86] « IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros ». In : *IEEE Std 952-1997* (1998), p. 1-84.

[87] Nikolas Trawny et Stergios I Roumeliotis. « Indirect Kalman Filter for 3D Attitude Estimation ». Rapport technique. Dept. of Computer Science Engineering University of Minnesota, mar. 2005, p. 25.

[88] Jim Pitman et Marc Yor. « A guide to Brownian motion and related stochastic processes ». Cours. Dept. Statistics, University of California, 367 Evans Hall # 3860, Berkeley, CA 94720-3860, USA.

[89] John L Crassidis. « Sigma-Point Kalman Filtering for Integrated GPS and Inertial Navigation ». In : *American Institute of Aeronautics and Astronautics* (2006), p. 24.

[90] Frédéric Devernay et Olivier Faugeras. « Straight lines have to be straight ». In : *Machine Vision and Applications* 13.1 (1er août 2001), p. 14-24.

[91] L. Javernick, J. Brasington et B. Caruso. « Modeling the topography of shallow braided rivers using Structure-from-Motion photogrammetry ». In : *Geomorphology* 213 (mai 2014), p. 166-182.

[92] Kenji Hata et Silvio Savarese. « Epipolar Geometry ». Cours. Stanford University, p. 14.

[93] Davide Scaramuzza et Friedrich Fraundorfer. « Visual Odometry [Tutorial] ». In : *IEEE Robotics & Automation Magazine* 18.4 (déc. 2011), p. 80-92.

[94] Vincent Lepetit, Francesc Moreno-Noguer et Pascal Fua. « EPnP : An Accurate O(n) Solution to the PnP Problem ». In : *International Journal of Computer Vision* 81.2 (19 juil. 2008), p. 155.

[95] M. Irani et P. Anandan. « About Direct Methods ». In : *Vision Algorithms : Theory and Practice*. Sous la dir. de Bill Triggs, Andrew Zisserman et Richard Szeliski. Lecture Notes in Computer Science. Berlin, Heidelberg : Springer, 2000, p. 267-277.

[96] Francisco Bonin-Font, Alberto Ortiz et Gabriel Oliver. « Visual Navigation for Mobile Robots : A Survey ». In : *Journal of Intelligent and Robotic Systems* 53.3 (nov. 2008), p. 263-296.

[97] Stéphane Bres et Jean-Michel Jolion. « Detection of Interest Points for Image Indexation ». In : *Visual Information and Information Systems*. Sous la dir. de Dionysius P. Huijsmans et Arnold W. M. Smeulders. Lecture Notes in Computer Science. Berlin, Heidelberg : Springer, 1999, p. 427-435.

[98] Mark Cummins et Paul Newman. « FAB-MAP : Probabilistic Localization and Mapping in the Space of Appearance ». In : *The International Journal of Robotics Research* 27.6 (juin 2008), p. 647-665.

[99] Friedrich Fraundorfer et Davide Scaramuzza. « Visual Odometry : Part II : Matching, Robustness, Optimization, and Applications ». In : *IEEE Robotics Automation Magazine* 19.2 (juin 2012), p. 78-90.

[100] David G. Lowe. « Distinctive Image Features from Scale-Invariant Keypoints ». In : *International Journal of Computer Vision* 60.2 (nov. 2004), p. 91-110.

[101] Herbert Bay, Tinne Tuytelaars et Luc Van Gool. « SURF : Speeded Up Robust Features ». In : *Computer Vision – ECCV 2006*. Sous la dir. d'Aleš Leonardis, Horst Bischof et Axel Pinz. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, p. 404-417.

[102] Haythem Ghazouani. « Navigation visuelle de robots mobiles dans un environnement d'intérieur ». Thèse doctorale. Université Montpellier II - Sciences et Techniques du Languedoc, 12 déc. 2012, p. 121.

[103] Yaakov Bar-Shalom, X. Rong Li et Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation : Theory Algorithms and Software*. John Wiley & Sons, 5 avr. 2004. 584 p.

[104] E. Rosten, R. Porter et T. Drummond. « Faster and Better : A Machine Learning Approach to Corner Detection ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (jan. 2010), p. 105-119.

[105] Matia Pizzoli, Christian Forster et Davide Scaramuzza. « REMODE : Probabilistic, monocular dense reconstruction in real time ». In : *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Mai 2014, p. 2609-2616.

[106] T. Bailey et H. Durrant-Whyte. « Simultaneous localization and mapping (SLAM) : part II ». In : *IEEE Robotics Automation Magazine* 13.3 (sept. 2006), p. 108-117.

[107] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik et Roland Siegwart. « The EuRoC micro aerial vehicle datasets ». In : *The International Journal of Robotics Research* (2016).

[108] Yuncheng Lu, Zhucun Xue, Gui-Song Xia et Liangpei Zhang. « A survey on vision-based UAV navigation ». In : *Geo-spatial Information Science* 21.1 (2 jan. 2018), p. 21-32.

[109] Raúl Mur-Artal, J. M. M. Montiel et Juan D. Tardós. « ORB-SLAM : A Versatile and Accurate Monocular SLAM System ». In : *IEEE Transactions on Robotics* 31.5 (oct. 2015), p. 1147-1163.

[110] Steven M. LaValle. « Rapidly-exploring random trees : a new tool for path planning ». Cours. Department of Computer Science Iowa State University, 1998.

[111] Paul Furgale, Joern Rehder et Roland Siegwart. « Unified Temporal and Spatial Calibration for Multi-Sensor Systems ». In : *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, p. 1280-1286.

[112] Paul Furgale, Timothy D. Barfoot et Gabe Sibley. « Continuous-time batch estimation using temporal basis functions ». In : *2012 IEEE International Conference on Robotics and Automation*. St Paul, MN, USA : IEEE, mai 2012, p. 2088-2095.

[113] Fadri Burri, Michael Achtelik, Markus Siegwart et Roland. « RotorS—A Modular Gazebo MAV Simulator Framework ». In : *Robot Operating System (ROS) : The Complete Reference (Volume 1)*. Sous la dir. de Koubaa, Anis. Springer International Publishing, 2016, p. 595-625.

[114] Davide Falanga, Kevin Kleber, Stefano Mintchev, Dario Floreano et Davide Scaramuzza. « The Foldable Drone : A Morphing Quadrotor That Can Squeeze and Fly ». In : *IEEE Robotics and Automation Letters* 4.2 (avr. 2019), p. 209-216.

[115] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. « Deep Residual Learning for Image Recognition ». In : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA : IEEE, juin 2016, p. 770-778.

Annexe A

Notions sur les réseaux de neurones convolutifs

A.1 Architecture classique

Les réseaux de neurones convolutifs (en anglais Convolutional neural networks), aussi connus sous le nom de CNNs, sont un type spécifique de réseaux de neurones qui sont généralement composés des couches suivantes :

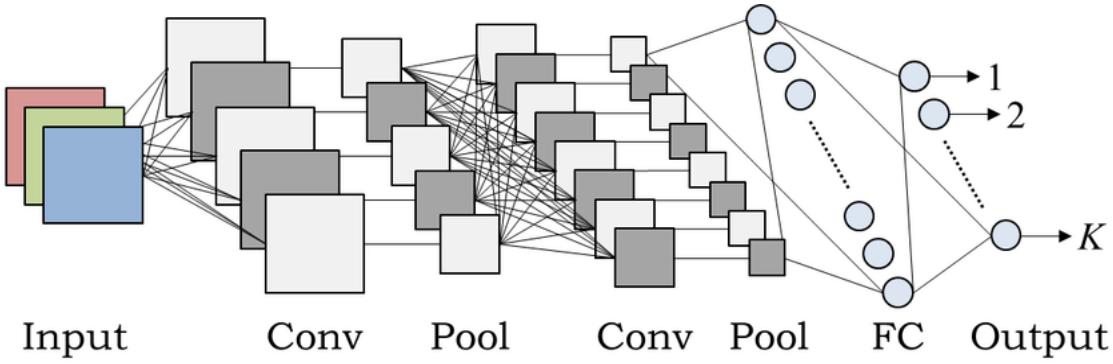


FIGURE *A.1 – Architecture d'un CNN classique.*

A.1.1 Couche convolutive (CONV)

La couche convolutive (en anglais convolution layer) (CONV) utilise des filtres qui *scannent* l'entrée I suivant ses dimensions en effectuant des opérations de convolution. Elle peut être réglée en ajustant la taille du filtre F et le *stride* S. La sortie O de cette opération est appelée *feature map* ou aussi *activation map*.

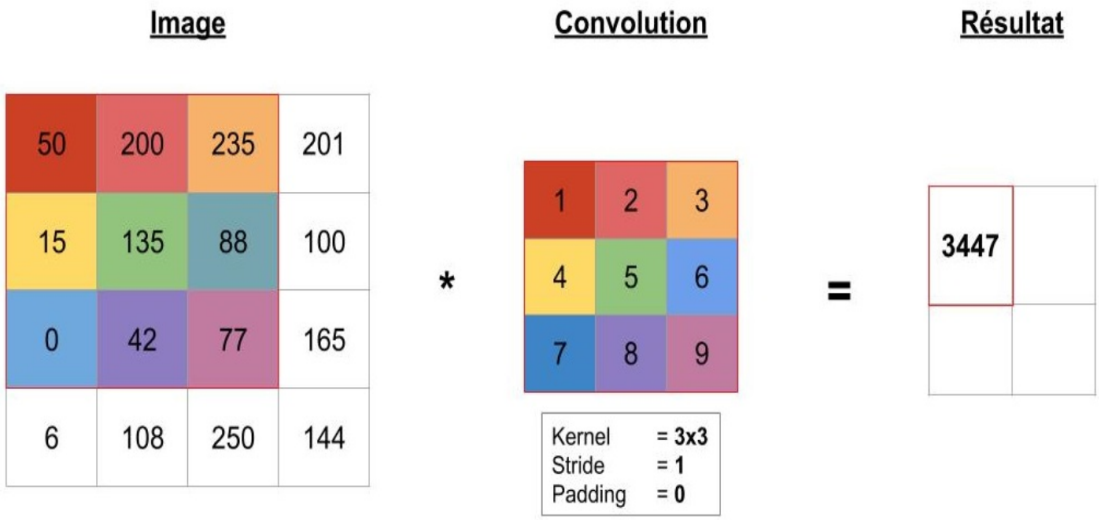


FIGURE *A.2 – Illustration de l'opération convolution 2D.*

La taille de l'image de sortie est donnée par : $O = \frac{I-K+2P}{S} + 1$ avec :
O : taille de l'image de sortie, I : taille de l'image d'entrée, F : taille du *kernel* (filtre) utilisé, S :*stride* appliqué et P : *padding* appliqué.

A.1.2 Couche d'activation

La couche d'activation consiste en l'application d'une fonction *non-linéaire* après une opération de convolution. Cette fonction d'activation g est utilisée sur tous les éléments du volume. Elle a pour but d'introduire des complexités non-linéaires au réseau. Ses variantes sont récapitulées dans le tableau suivant :

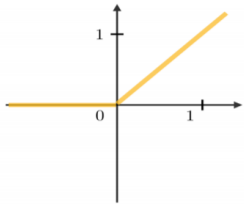
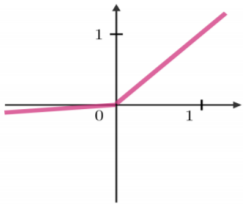
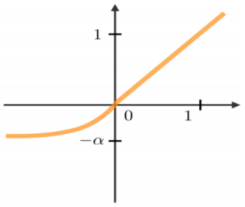
ReLU	**Leaky ReLU**	**ELU**
$g(z) = \max(0,z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$	$g(z) = \max(\alpha(e^z - 1), z)$ with $\alpha \ll 1$
		
Complexités non-linéaires interprétables d'un point de vue biologique	Répond au problème de *dying ReLU*	Dérivable partout

TABLE *A.1 – Fonctions d'activation couramment utilisées.*

A.1.3 Pooling

La couche de *pooling* est une opération de sous-échantillonnage typiquement appliquée après une couche convolutif. Elle consiste à remplacer un carré de pixels (généralement 2×2 ou 3×3) par une valeur unique. De cette manière, l'image diminue en taille et se retrouve simplifiée (lissée). En particulier, les types de pooling les plus

populaires sont le *max* et l'*average pooling*, où les valeurs maximales et moyennes sont prises, respectivement.

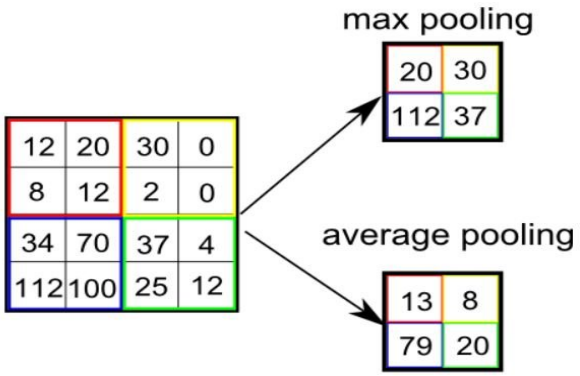


FIGURE *A.3 – Illustration des opérations de pooling "max" et "avg"*

A.1.4 La mise à plat (flattening)

le flattening consiste simplement à mettre bout à bout toutes les images (matrices) qui émanent des couches de convolutions précédentes pour en faire un (long) vecteur. Les pixels sont simplement récupérés ligne par ligne et ajoutés au vecteur final.

A.1.5 Couche entièrement connecté (*Fully Connected*)

La couche *fully connected* s'applique sur une entrée préalablement aplatie où chaque entrée est connectée à tous les neurones. Les couches *fully connected* sont typiquement présentes à la fin des architectures de CNN et peuvent être utilisées pour optimiser des objectifs tels que les scores de classe.

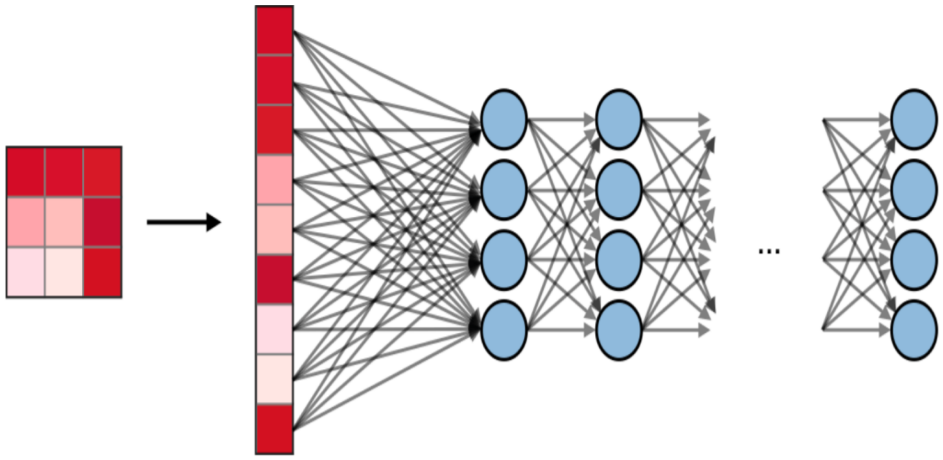


FIGURE *A.4 – Illustration d'une couche FC après aplatissement.*

A.1.6 Normalisation de lots

La normalisation de lot (en anglais *batch normalization*) est une étape qui normalise le lot x_i avec un choix de paramètres γ, β. En notant μ_B, σ_{B^2} la moyenne et la variance de ce

que l'on veut corriger du lot on a :

$$x_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_{B^2} + \epsilon}\beta} \tag{A.1}$$

Couche de sortie : la couche de sortie est une couche *fully connected* dont la fonction d'activation ne peut être spécifié arbitrairement puisque celle ci dépend du type de problème. Nous donnons à titre d'exemple le cas de figure suivants :

1. Régression dans R^m : La fonction identité $f(x) = x$.
2. Classification Binaire dans $[0\,1]^2$:la fonction sigmoïde $f(x) = (1 + e^{-x})^{-1}$
3. Classification dans $[0\,1]^m$: la fonction *softmax* définie pour un vecteur $z = [z_1\, z_2 \ldots z_k]$ comme suit : $\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ pour tout $j \in 1, .., k$.

A.2 Apprentissage des réseaux de neurones

A.2.1 Fonction objective

(ou *loss function* en anglais) est une fonction $L : (z, y) \in \mathbb{R} \times Y \rightarrow L(z, y) \in \mathbb{R}$ qui sert de critère pour déterminer la meilleure solution à un problème d'optimisation. Le but du problème d'optimisation devient alors une minimisation (ou maximisation) de cette fonction objectif. Les fonctions objectifs les plus courantes sont représenté dans la figure A.2 suivante :

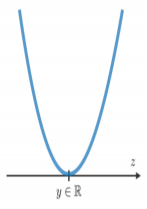
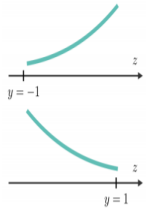
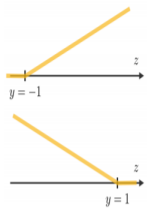
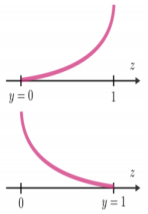
Moindres carrés	Logistique	Hinge loss	Cross-entropie
$\frac{1}{2}(y-z)^2$	$\log(1+\exp(-yz))$	$\max(0, 1-yz)$	$-\left[y\log(z) + (1-y)\log(1-z)\right]$
			
Régression	Classification	Classification	Classification

TABLE *A.2 – Exemples de fonctions objectifs*

Fonction de coût : la fonction de coût J est communément utilisée pour évaluer la performance d'un modèle (réseau de neurones), et est définie avec la fonction de loss L par :

$$J(\theta) = \sum_{i=1}^{m} L(h_\theta(x^{(i)}), y^{(i)}) \tag{A.2}$$

A.2.2 Algorithme du gradient

C'est un algorithme itératif simple et populaire qui permet de mettre à jour des paramètres θ en vue de minimiser une fonction de coût J. En notant $\alpha \in \mathbb{R}$ le taux d'apprentissage (en anglais *learning rate*), la règle de mise à jour de l'algorithme est :

$$\theta \leftarrow \theta - \alpha \nabla J(\theta) \tag{A.3}$$

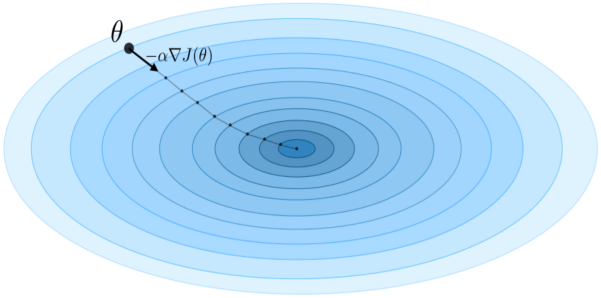


FIGURE *A.5 – Visualisation de l'algorithme du gradient.*

Le taux d'apprentissage α indique la vitesse à laquelle les coefficients sont mis à jour. Il peut être fixe ou variable. La méthode actuelle la plus populaire est appelée *Adam*, qui est une méthode faisant varier le taux d'apprentissage. Voici une liste non exhaustive des algorithmes d'adaptation du taux d'apprentissage les plus couramment utilisés :

Méthode	Explication	Mise à jour de w	Mise à jour de b
Momentum	- Amortit les oscillations - Amélioration par rapport à la méthode SGD - 2 paramètres à régler	$w - \alpha v_{dw}$	$b - \alpha v_{db}$
RMSprop	- Root Mean Square propagation - Accélère l'algorithme d'apprentissage en contrôlant les oscillations	$w - \alpha \frac{dw}{\sqrt{s_{dw}}}$	$b \longleftarrow b - \alpha \frac{db}{\sqrt{s_{db}}}$
Adam	- Adaptive Moment estimation - Méthode la plus populaire - 4 paramètres à régler	$w - \alpha \frac{v_{dw}}{\sqrt{s_{dw}} + \epsilon}$	$b \longleftarrow b - \alpha \frac{v_{db}}{\sqrt{s_{db}} + \epsilon}$

TABLE *A.3 – Principaux algorithmes d'adaptation du taux d'apprentissage.*

A.2.3 Rétropropagation du gradient

Rétropropagation (*Backpropagation* en anglais) est une méthode pour calculer le gradient de l'erreur pour chaque neurone d'un réseau de neurones, de la dernière couche vers la première. La dérivée par rapport à chaque coefficient (*poids*) du réseau est calculée en utilisant la règle de la chaîne.

A.3 Amélioration de l'apprentissage des réseaux de neurones

A.3.1 *Dropout*

Le dropout est une technique qui est destinée à empêcher le sur-ajustement (*overfitting*) sur les données de training en abandonnant des unités dans un réseau de neurones avec une probabilité $p > 0$. Cela force le modèle à éviter de trop s'appuyer sur un ensemble particulier de *features*.

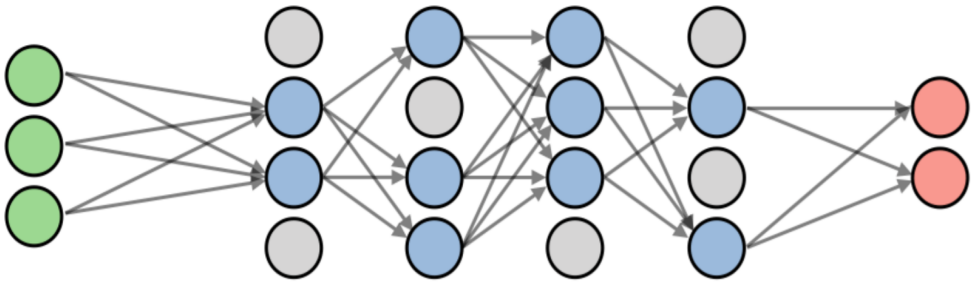


FIGURE *A.6 – Illustration du dropout.*

A.3.2 Régularisation des coefficients

Pour s'assurer que les coefficients ne sont pas trop grands et que le modèle ne sur-ajuste pas sur le training set, on utilise des techniques de régularisation sur les coefficients du modèle. Les techniques principales sont résumées dans le figure suivante :

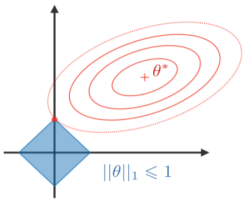
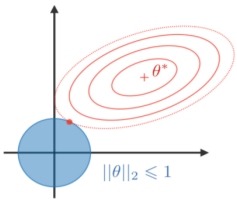
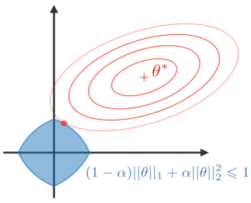
LASSO	**Ridge**	**Elastic Net**
- Réduit les coefficients à 0 - Bon pour la sélection de variables	Rend les coefficients plus petits	Compromis entre la sélection de variables et la réduction de la taille des coefficients
		
$\ldots + \lambda\|\|\theta\|\|_1$ $\lambda \in \mathbb{R}$	$\ldots + \lambda\|\|\theta\|\|_2^2$ $\lambda \in \mathbb{R}$	$\ldots + \lambda\left[(1-\alpha)\|\|\theta\|\|_1 + \alpha\|\|\theta\|\|_2^2\right]$ $\lambda \in \mathbb{R}, \alpha \in [0,1]$

TABLE *A.4 – Différents types de régularisation des coefficients.*

A.3.3 Arrêt prématuré

L'arrêt prématuré (en anglais *early stopping*) est une technique de régularisation qui consiste à stopper l'étape d'entraînement dès que le loss de validation stagne ou commence à augmenter.

A.3.4 Apprentissage par transfert

Entraîner un modèle d'apprentissage profond requière beaucoup de données et beaucoup de temps. Il est souvent utile de profiter de coefficients pré-entraînés sur des données énormes qui ont pris des jours/semaines pour être entraînés. l'apprentissage par transfert ou *fine-tuning* est une technique très utilisé pour l'entraînement des CNN, elle consiste à débloquer quelques-unes des dernières couches d'un réseau de base gelée déjà entraînée sur une grande quantité de données. Ce modèle servira donc à extraire les premières et principales caractéristiques et à former conjointement avec la partie ajoutée un nouveau modèle. C'est ce qu'on appelle le *fine-tuning* (réglage fin) parce qu'il ajuste légèrement les représentations plus abstraites du modèle réutilisé, afin de les rendre plus pertinentes pour le problème en question.

A.3.5 Réseau résiduel

L'architecture du réseau résiduel (en anglais Residual Network), aussi appelé ResNet, utilise des blocs résiduels avec un nombre élevé de couches et a pour but de réduire l'erreur de training. Son implémentions est relativement simple : chaque couche s'alimente dans la couche suivante et directement dans 2-3 couches plus loin.

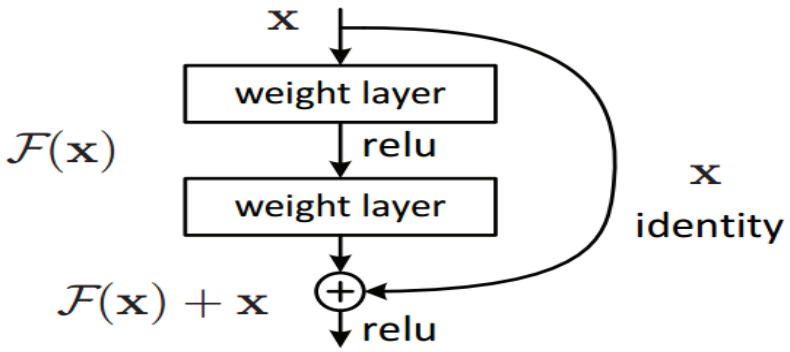


FIGURE *A.7 – Implémentation typique d'un réseau résiduel.*

Les auteurs de [115] ont d'abord démontrés le problème de profondeur des réseaux de neurones et ont proposé une solution remarquable qui a depuis lors permis l'entraînement de plus de 2000 couches avec une précision croissante. En appliquant ce principe, les auteurs ont gagné l'édition 2015 de Imagenet [69] et ont atteint un nouvel état de l'art sur tous les benchmarks standard de vision par ordinateur.

Annexe B

Réseau de caractéristiques BiFPN

Le réseau de caractéristiques répond au problème de la fusion des caractéristiques à travers plusieurs niveaux et différentes résolutions. Soit $\vec{P^{in}} = \left(P^{in}_{l_1}, P^{\text{lin}}_{l_2}, \ldots\right)$ un vecteur qui regroupes les sorties des couches de convolution sur différents niveaux, où $P^{in}_{l_i}$ représenta la caractéristique au niveau l_i. Le but est de trouver une transformation f qui peut combiner d'une manière efficace et donner en sortie de nouvelles caractéristiques $\vec{P}^{\text{out}} = f\left(\vec{P}^{in}\right)$.

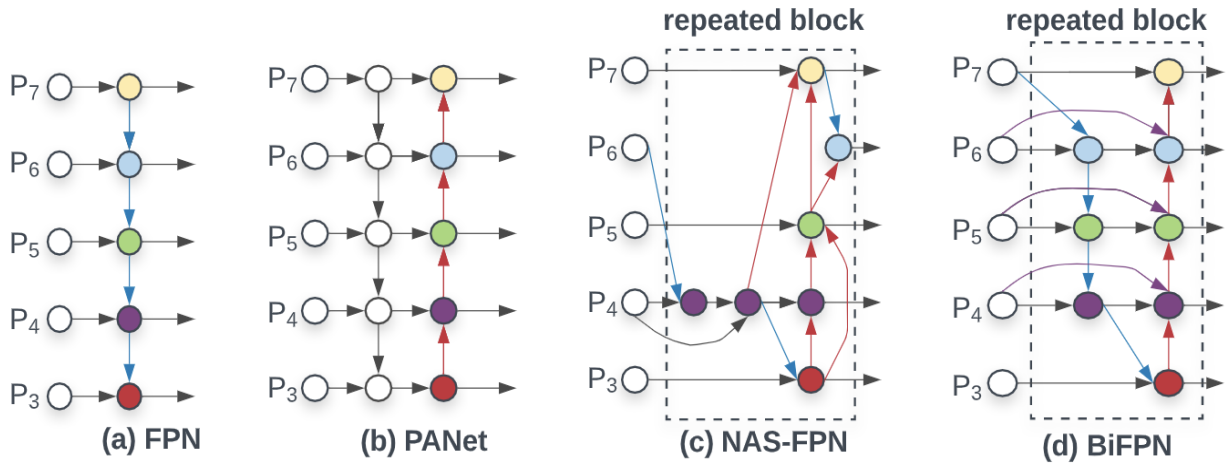


FIGURE *B.1 – Comparaison entre l'architecture BiFPN et les autres réseaux de caractéristiques présents dans la littérature.*

La figure B.1 montre les différents réseaux caractéristiques qui sont utilisés. Le réseau FPN montre un réseau conventionnel "*top-down*" où $\vec{P^{in}} = \left(P^{in}_3, \ldots P^{in}_7\right)$, $\vec{P^{in}}$ représentant un niveau de caractéristiques avec une résolution $1/2^i$ de l'image d'entrée. Ce réseau va s'écrire donc : $P^{\text{out}}_7 = \text{Conv}\left(P^{\text{in}}_7\right)$ $P^{\text{out}}_6 = \text{Conv}\left(P^{\text{in}}_6 + \text{Resize}\left(P^{\text{out}}_7\right)\right)$ $\cdots$ $P^{\text{out}}_3 = \text{Conv}\left(P^{in}_3 + \text{Resize}\left(P^{\text{out}}_4\right)\right)$ Avec Resize un opérateur de sur-échantillonnage ou sous échantillonnage, qui sert dans la mise en correspondance des résolutions à travers les différnts niveaux.

B.1 Configuration du réseau de caractéristiques BiFPN

L'architecture BiFPN a été inspirée à partir de PANet, qui offre une meilleure précision que les deux autres architectures (FPN et NAS-FPN) mais avec beacoup plus de paramètres et de calculs. Cependant pour y remédier, des transformations sont faits, d'abord les nœuds qui n'ont en entrée qu'un seul noeud de périphérie (le premier et le dernier nœud) sont enlevés et leur connections sont redirigées vers le nœuds suivant dans le même niveau, ensuite une connexion est effectuée entre le nœud d'entrée et de sortie de chaque niveau. A la fin, ce réseau est répété plusieurs fois pour permettre une meilleure fusion des caractéristiques haut-niveau.

Cette fusion des caractéristiques est effectuée en adressant un poids à chaque entrée qui va être déterminé durant la phase d'entrainement, la formule de fusion dans chaque nœud est la suivante :

$$O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i$$

- Les poids w_i pouvant être scalaires, vecteurs ou multi-dimensionels, en appliquant une fonction Relu à chaque poids pour assurer le signe positif.
- Le paramètre $\epsilon = 0.0001$ est mis pour éviter la division sur zéro.

Au final pour un niveau 6 par exemple les transformations entre caractéristiques seront :

$$P_6^{td} = \text{Conv}\left(\frac{w_1 \cdot P_6^{in} + w_2 \cdot \text{Resize}\left(P_7^{in}\right)}{w_1 + w_2 + \epsilon}\right)$$

$$P_6^{\text{out}} = \text{Conv}\left(\frac{w_1' \cdot P_6^{in} + w_2' \cdot P_6^{td} + w_3' \cdot \text{Resize}\left(P_5^{\text{out}}\right)}{w_1' + w_2' + w_3' + \epsilon}\right)$$

Annexe C

Comparaison entre les différents types de détecteurs de points caractéristiques

Le tableau **??**uivant présente une comparaison entre les différents types de descripteurs de l'état de l'art .

	Corner detector	Blob detector	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	x		x			+++	+++	++	++
Shi-Tomasi	x		x			+++	+++	++	++
FAST	x		x	x		++	++	++	++++
SIFT		x	x	x	x	+++	++	+++	+
SURF		x	x	x	x	+++	++	++	++
CenSurE		x	x	x	x	+++	++	+++	+++

FIGURE *C.1 – Comparaison entre les différents détecteurs de points saillant selon les propriétés et les performances [99].*

Annexe D

Outil de visualisation de l'architecture sur ROS

L'outil de visualisation **rqt_graph** est disponible sous le paquet d'outils **rqt_common_plugins**[1] de ROS. Il permet de surveiller l'exécution du système en temps-réel.

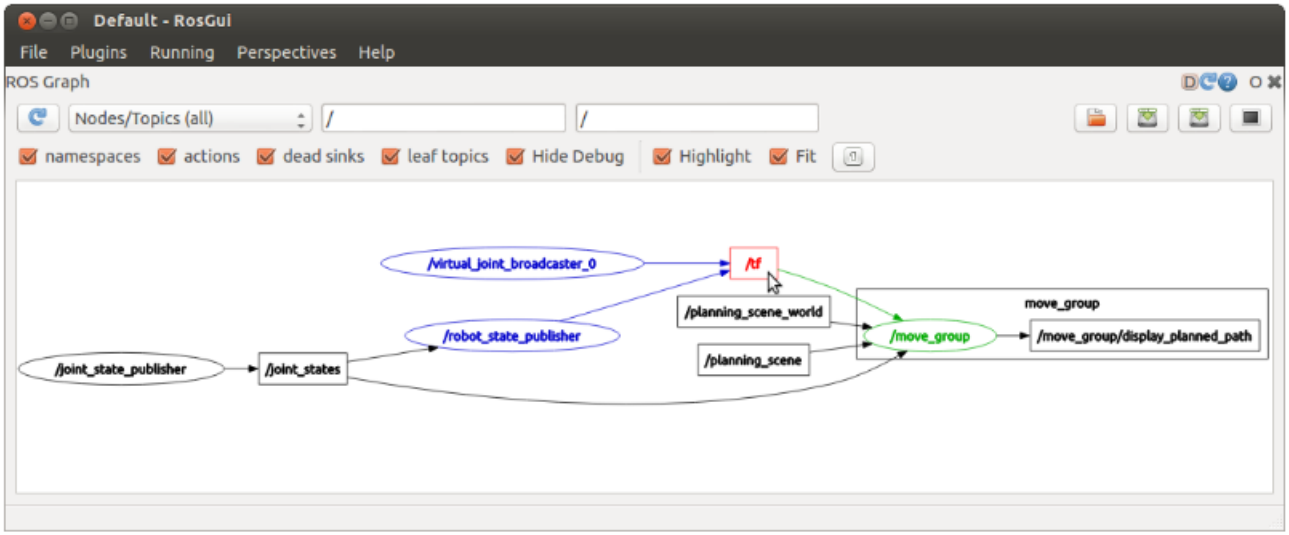


FIGURE *D.1 – Exemple d'utilisation de rqt_graph.*

La figure D.1 montre un exemple d'utilisation de l'outil **rqt_graph**. Les *nodes* (processus) ROS qui exécutent les algoithmes sont représentés par des formes elliptiques, tandis que les *topics* (queues de messages) qui permettent la communication entre les différents processus sont représentés par des formes rectangulaires. Les flèches représentent une souscription ou bien une publication dans une queue de messages à partir d'un processus, ils modélisent donc une relation *node-topic*.

1. http://wiki.ros.org/rqt_graph.