



---

# Mission de fin d'études

## Find Your Way

---

AUBAY

*Maître de stage :*  
Nathan CANTAT

*Jury de stage :*  
Arnaud BANNIER

*Auteur :*  
**Nicolas CISTERNAS**

28 février - 31 août 2022

## Résumé

Le projet Find Your Way a pour but de guider des personnes mal-voyantes en milieu intérieur. Ce document relate le travail effectué autour de cette thématique dans le cadre de mon stage de fin d'études, en se focalisant sur la partie localisation et guidage de l'utilisateur. Un état de l'art concernant le sujet de la localisation et de la cartographie locale en temps réel est présenté. Ensuite une preuve de concept se basant sur une méthode en sources libres, contenant les améliorations que mon groupe et moi avons pu ajouter est expliquée dans ce document. Enfin ce document explique le déroulement de la phase projet au sein de la cellule Innov' de l'entreprise Aubay pour démontrer le résultat d'une collaboration entre 11 stagiaires. Le lecteur pourra y trouver des informations concernant ORB-SLAM3, FAST, les points d'intérêts, la création d'un protocole de communication, le développement de l'interface graphique et les problèmes que l'on peut rencontrer sur ce genre de projet multidisciplinaire.

## **Abstract**

Abstract in English

## Table des matières

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Remerciements</b>   | <b>5</b>  |
| <b>2</b> | <b>Introduction et contexte</b>  | <b>6</b>  |
| <b>3</b> | <b>État de l’art</b>   | <b>9</b>  |
| 3.1      | Recherches bibliographiques . . . . .  | 9         |
| 3.1.1    | Wearable Travel Aid for Environment Perception and Navigation of Visually Impaired People . . . . .  | 9         |
| 3.1.2    | CDSFusion : Dense Semantic SLAM for Indoor Environment Using CPU Computing . . . . .                 | 10        |
| 3.1.3    | Towards Real-time Semantic RGB-D SLAM in Dynamic Environments . . . . .                              | 13        |
| 3.1.4    | ORB-SLAM3 : An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM . . . . . | 14        |
| 3.2      | Synthèse . . . . .   | 16        |
| 3.3      | Conclusion . . . . .   | 18        |
| <b>4</b> | <b>Les dimensions techniques du projet (8 à 12 p)</b>  | <b>19</b> |
| 4.1      | Phase de preuve de concept . . . . .   | 19        |
| 4.1.1    | Installation . . . . .   | 19        |
| 4.1.2    | Calibration . . . . .  | 21        |
| 4.1.3    | Recherche de chemin . . . . .  | 21        |
| 4.1.4    | Test et résultats . . . . .  | 25        |
| 4.1.5    | Difficultés rencontrées . . . . .  | 26        |
| 4.1.6    | Conclusion . . . . .   | 27        |
| 4.2      | Phase projet . . . . .   | 27        |
| 4.2.1    | Protocole de communication . . . . .   | 27        |
| 4.2.2    | Développement de l’application . . . . .   | 29        |
| 4.2.3    | Conclusion . . . . .   | 30        |
| <b>5</b> | <b>Les dimensions humaines et managériales (3 à 5 p)</b>   | <b>31</b> |
| <b>6</b> | <b>Conclusion (2 à 3 p)</b>  | <b>32</b> |
| <b>7</b> | <b>Bibliographie</b>   | <b>33</b> |
| <b>8</b> | <b>Annexes</b>   | <b>34</b> |
| 8.1      | Point d’intérêt . . . . .  | 34        |
| 8.2      | FAST . . . . .   | 34        |

|     |                 |    |
|-----|-----------------|----|
| 8.3 | BRIEF . . . . . | 35 |
|-----|-----------------|----|

# 1 Remerciements

Je remercie Éric REMILLERET et Nathan CANTAT, mes maîtres de stage, ainsi que Wael NASR, pour m’avoir donné l’opportunité de rejoindre Aubay Innov’, qui s’est avérée être une expérience intéressante et instructive, ainsi que pour l’aide à la relecture de ce rapport.

Je remercie Anne-France GALLAND, qui a largement contribué à la phase de suivi et d’organisation du projet, par ses interventions lors des réunions hebdomadaires et pour le temps qu’elle a pris pour la relecture de ce rapport.

Je remercie Jean-Philippe CUNNIET, mon tuteur de stage ainsi qu’Arnaud BANNIER, le jury de cette mission de fin d’études, pour le temps qu’ils ont passé à relire et évaluer ce rapport de stage ainsi que la soutenance de stage.

Je tiens à remercier toute l’équipe avec laquelle j’ai travaillé durant mon stage : Nicolas GUILLERMAIN et Mathieu MONNERET avec qui j’ai travaillé lors de la phase d’état de l’art et de preuve de concept. Mélissa WANG et Jeffrey MENUJIER avec qui j’ai principalement travaillé lors de la phase projet. Jean-Baptiste CHANIER, Victor CHAVEROT, Jean-Noël CLINK, Ophélie PHONCHAREUN, Miora RASOLOFONERA et Rémi VIDAL avec qui j’ai échangé tout au long du stage concernant leurs parties du projet.

Je tiens également à remercier la direction générale d’Aubay pour leur accueil et leur écoute ainsi qu’Ophélie CHEVALIER, campus manager, pour tous les événements qu’elle a pu organiser et qui ont permis d’établir un climat chaleureux entre les différents groupes.

Enfin, je remercie les autres étudiants d’Aubay Innov’ avec lesquels j’ai eu l’occasion d’échanger aussi bien humainement que techniquement.

## 2 Introduction et contexte

Mon stage s'est déroulé dans une entreprise appelée Aubay, pour une durée de 6 mois (du 28 février 2022 au 31 août 2022). En détail, Aubay est une entreprise de services numériques (ESN) qui a été fondée par Christian Aubert en 1998 et dont le siège social est situé au 13 rue Louis Pasteur à Boulogne-Billancourt. L'entreprise est spécialisée dans les domaines liés à la finance, à l'assurance et à la banque et est également impliquée dans divers marchés, tels que les télécoms, les services, les réseaux, l'énergie et les transports. Aubay accompagne la transformation et la modernisation des systèmes d'information de ses clients. Ils opèrent sur des marchés à forte valeur ajoutée, en France comme en Europe. C'est un acteur référent de la transformation digitale. Son secteur d'activité est centré autour du conseil sur tout type de projet technologique. C'est une ESN cotée en Bourse (SBF 250) et 46% du capital est détenu par les managers. Les chiffres clés concernant Aubay sont présentés dans la Figure 1 ci-après. En 2022, l'entreprise emploie 7306 travailleurs, dont 2728 en France. Selon son site internet, Aubay est implantée dans 7 pays européens, et a réalisé un chiffre d'affaires de 470,6 M€ en 2021. D'ailleurs, ces dernières années, l'entreprise a connu une forte croissance de 10,4% en 2021, qui coïncide avec une augmentation constante des effectifs de l'entreprise, passant de 4600 employés en 2015 à plus de 7000 aujourd'hui.

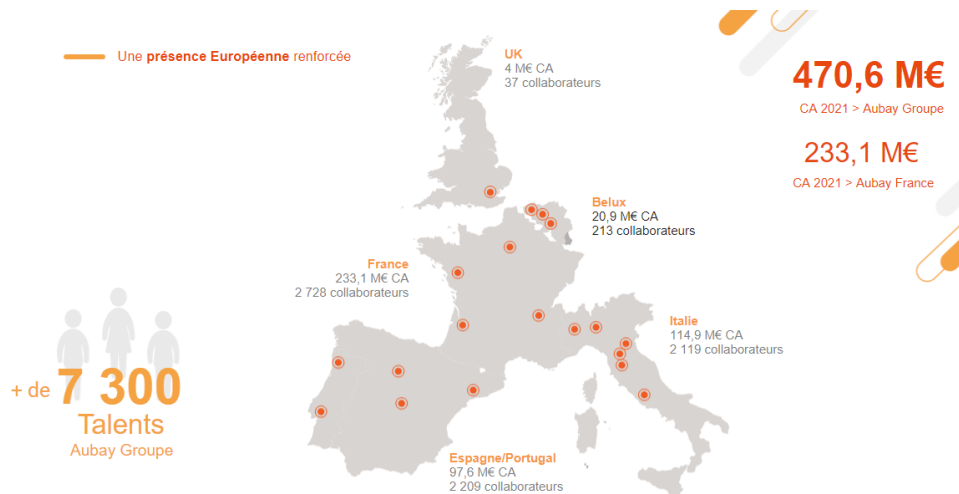


FIGURE 1 – Chiffres clés pour Aubay en 2021.

J'ai eu le privilège d'effectuer mon stage au sein de l'unité "Aubay Innov'" qui est la division d'Aubay France dédiée à la recherche et à l'innovation. Ses membres sont impliqués dans différents projets, chacun lié à la data science, l'analyse de données ou tout autre domaine liés aux nouvelles technologies numériques. L'objectif de l'unité est d'acquérir les connaissances et le savoir-faire pour construire des solutions durables et innovantes adaptées aux besoins futurs des clients. Les projets dirigés dans la cellule innovation d'Aubay ne sont donc soumis à aucun client pendant leur réalisation, ce qui laisse l'opportunité aux stagiaires d'expérimenter autant qu'ils le souhaitent durant leur stage. Chaque année, cette unité donne la chance à des dizaines de stagiaires d'améliorer ou de perfectionner leurs connaissances relatives à la science des données en leur laissant l'opportunité de découvrir et d'expérimenter les technologies les plus innovantes disponibles dans les domaines de la recherche. Par ailleurs, l'unité "Aubay Innov'" est largement considérée comme une source de recrutement pour l'entreprise, qui a souhaité cette année engager environ 800 nouveaux collaborateurs en France.

C'est dans ce cadre que ma mission de fin d'études a débuté. Le projet "Find Your Way" (FYW), qui représente l'expérience que je vais détailler dans ce document, a été lancé en février 2022 et a pour objectif de créer une application embarquée sur des lunettes, basée sur des algorithmes capables de reconnaître les éléments de l'environnement immédiat d'un utilisateur malvoyant pour le guider vers des lieux ou l'avertir d'obstacles présents sur son chemin tels que des chaises ou une personne par exemple. Un objectif majeur de ce projet est également de localiser l'utilisateur dans son environnement afin de lui permettre de retrouver son chemin jusqu'à un endroit précédemment enregistré et de le guider avec des indications de direction et d'orientation dans ses déplacements en intérieur en temps réel, le tout répondant à un système de commandes vocales.

Les systèmes d'aide automatisés au déplacement de personnes malvoyantes existent déjà sur le marché et nécessitent de fournir une carte du bâtiment avec les lieux importants préalablement renseignés afin de rendre le guidage possible. Ce projet vise à s'affranchir de cette contrainte afin de permettre une plus grande polyvalence pour ce genre d'outil. Le projet a été séparé en plusieurs parties délimitées par des dates clés qui sont présentées dans la Figure 2 avec les livrables associés à chaque fin de phase. La prise en main du projet a démarrée début février, s'en est suivi la partie concernant les états de l'art (EA sur la Figure 2) qui devait être réalisée entre mi-février et mi-mars en parallèle du Design Thinking qui nous a permis de cadrer



et définir le projet par rapport aux besoins de l'application et les tâches à accomplir. La phase de preuve de concept qui a suivi a duré jusqu'à mi-mai. Une fois ces étapes réalisées nous avons pu passer à la phase projet afin de réunir toutes les preuves de concept et concevoir l'application de démonstration nécessaire à la journée des stagiaires (JDS) du 7 juillet, moment phare pour le pôle innovation de chez Aubay où tous les stagiaires présentent leurs projets lors d'une démonstration auprès des directeurs généraux et directeurs commerciaux de l'entreprise.

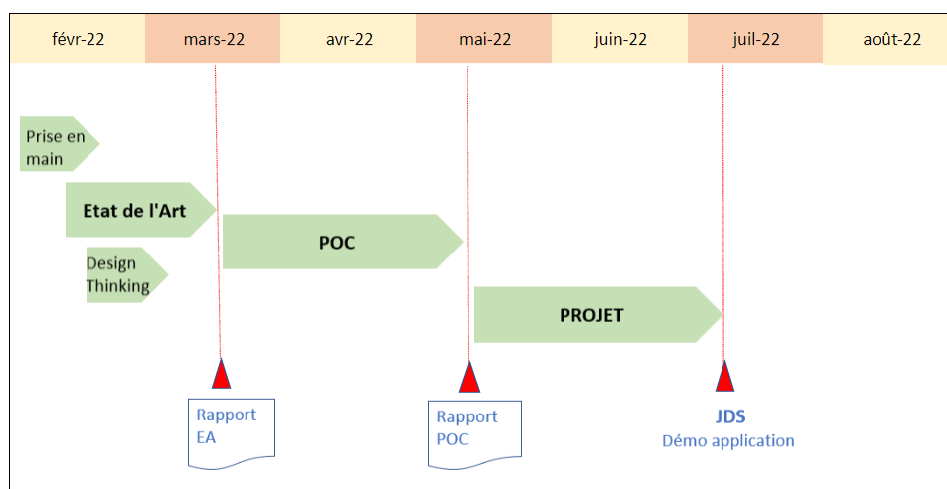


FIGURE 2 – Planning prévisionnel du projet FYW.

J'ai réalisé ce projet dans une équipe de 11 stagiaires, tous en stage de fin d'études. Étant donné les multiples parties concernant le projet, à savoir la détection d'objets, la localisation de l'utilisateur et la gestion des interactions vocales, nous avons dû nous séparer en 3 groupes afin de réaliser les états de l'art et les preuves de concept. J'ai personnellement travaillé sur la partie s'intéressant à la localisation et le guidage de l'utilisateur dans son environnement.

## 3 État de l'art

### 3.1 Recherches bibliographiques

Dans cette section je présente les recherches bibliographiques que j'ai pu effectuer avec mon groupe lors de la phase d'état de l'art. Cet état de l'art concerne la partie de localisation et de guidage de l'utilisateur. Nous avons retenu 8 publications, nous les avons confrontées et comparées afin de n'en sélectionner qu'une sur laquelle nous allons nous concentrer lors de la phase de preuve de concept. Je présente ici 4 des publications les plus pertinentes sur le sujet.

#### 3.1.1 Wearable Travel Aid for Environment Perception and Navigation of Visually Impaired People

L'appareil consiste en une caméra grand public rouge, vert, bleu avec de la profondeur (RGB-D pour red, green, blue, depth) et une unité de mesure inertielle (IMU pour Inertial Measurement Unit), c'est un capteur qui consiste généralement en des gyroscopes pour mesurer des vitesses angulaires et des accéléromètres pour mesurer la force [1]. Ces appareils sont montés sur une paire de lunettes et reliés à un téléphone. L'appareil proposé dans cette solution se sert de la continuité de la hauteur du sol entre les images adjacentes pour segmenter le sol avec précision et rapidité pour ensuite chercher la direction du mouvement en fonction du sol. Un réseau de neurones à convolution (CNN pour Convolutional Neural Network) léger est utilisé pour la reconnaissance d'objets (PeelNet avec l'ensemble de données "MS COCO" contenant des images de dimensions 640 x 640). Son schéma de fonctionnement est présenté dans la figure 3. Il permet de récupérer des informations concernant les endroits aux alentours et l'orientation des objets environnants. Le schéma de fonctionnement du système est présenté dans la Figure 4.

Le système de navigation contient un module de localisation intérieur qui va nous intéresser : un algorithme VSLAM (Visual Simultaneous Localization and Mapping) est utilisé. SLAM est un problème de computer vision visant à traquer les mouvements d'un module en se basant sur ce qu'il voit. Il faut traiter les objets dynamiques qui entrent et sortent du champ de vision pour ne pas les prendre en compte dans l'estimation de la position du module. L'estimation se fait en trouvant des points clés sur les images successives. Visual SLAM se sert des informations récupérées pour trianguler la position 3D du module.

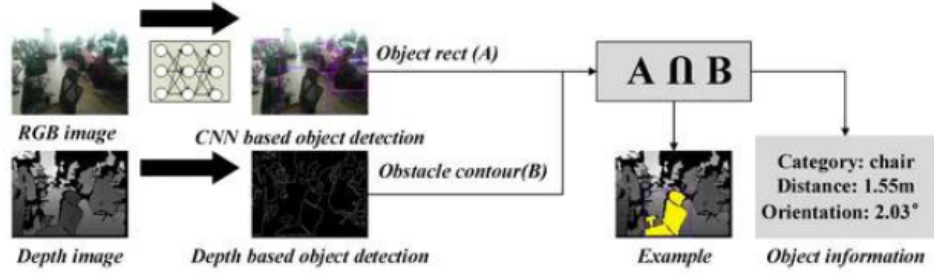


FIGURE 3 – Schéma présentant le fonctionnement du système de reconnaissance d’objets.

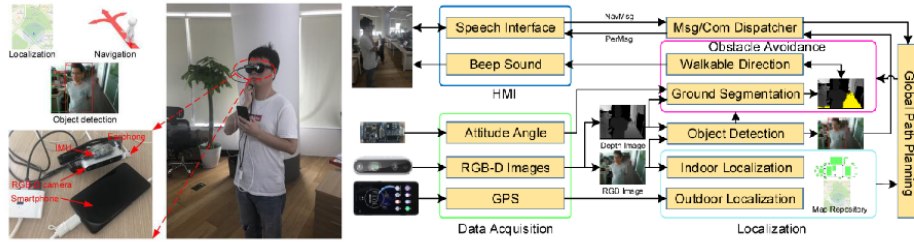


FIGURE 4 – Schéma de fonctionnement du système proposé.

### 3.1.2 CDSFusion : Dense Semantic SLAM for Indoor Environment Using CPU Computing

La solution CDSFusion [4] se sert d’images RGB comme l’article précédent ainsi qu’un capteur IMU comme paramètres d’entrée et est composée de 3 modules imagés dans la Figure 5 :

**3.1.2.1 Le module VIO** Le module d’odométrie visuelle-inertielle (VIO pour Visual-Inertial Odometry) se sert des entrées pour estimer avec précision la position afin de proposer une trajectoire. Ce module VIO est basé sur VINS-Mono. VINS-Mono est un framework SLAM en temps réel pour les systèmes visuo-inertiels monoculaires. Il utilise une méthode de fenêtre glissante basée sur des optimisations pour fournir une odométrie visuelle-inertielle de haute précision. Les features FAST (Features from accelerated segment test) ont été adaptées pour accélérer le VIO à la place des feature Shi-Thomas (une manière de détecter les coins sur une image) et la profondeur a été introduite afin d’obtenir une échelle plus précise. Les résultats

expérimentaux montrent que les features FAST augmentent la rapidité du système d'une manière plus conséquente que les features Shi-Tomas et ORB avec la même précision et robustesse. Ce module est composé de 3 parties :

Visual-Inertial Frontend : prend en charge le traitement des données issues des capteurs. Les mesures effectuées par le capteur IMU sont pré-intégrées entre deux images consécutives, le frontend de vision détecte les FAST et les traque entre les images consécutives en utilisant l'algorithme KLT optical flow.

Back-End : est utilisé pour fusionner les mesures traitées afin d'obtenir l'estimation de la position. Une optimisation non linéaire est utilisée pour re-localiser et optimiser le calcul de la position en fonction de la boucle détectée, en utilisant le solveur Ceres.

Le module de détection de boucle : permet de relocaliser et optimiser le calcul de la position en fonction de la boucle détectée. En effet, une boucle est détectée lorsque l'on a réalisé une boucle dans le parcours d'un chemin, il devient alors inutile de recalculer la position tant que l'utilisateur se situe dans cette boucle, on peut alors facilement optimiser le calcul de la position en se basant sur des calculs précédemment effectués. Cela se base sur la bibliothèque Dynamic Bag of Words (DBoW2) qui est à l'état de l'art de la reconnaissance d'endroits par sac de mots (bag of words approach). De même, lorsqu'une boucle est détectée, une optimisation est possible pour le calcul de la position globale. Cette optimisation est similaire à la méthode VINS-Mono.

**3.1.2.2 Le module de segmentation sémantique** La segmentation sémantique résulte d'images RGB en entrée qui sont acquises en temps réel en utilisant le module de segmentation PSPNet. Le module de segmentation sémantique traite chaque image RGB et retourne des vecteurs indiquant la probabilité d'appartenance à une classe pour chaque pixel. Ils classifient et colorent chaque pixel en fonction de la plus haute probabilité d'appartenance à une classe. L'image segmentée finale est composée des pixels colorés et est transmise au module de reconstruction 3D.

**3.1.2.3 Le module de reconstruction 3D** Le nuage sémantique local est généré en utilisant une image sémantique (générée par le module précédent) et une carte de profondeur. Ce nuage local servira à produire un nuage global une fois qu'il sera combiné avec les estimations de position de

la caméra depuis le module VIO. Pour construire une carte 3D globale qui soit précise, un modèle basé sur des voxels est utilisé pour filtrer le bruit et extraire un mesh global. À chaque image importante (Keyframe), la carte de profondeur courante est transformée en nuage de points 3D, ensuite les options Voxelblox et FAST sont adaptées, le nuage local de points 3D est transformé en mesh local et ensuite ce mesh local est intégré dans le mesh global. L'ensemble de cette procédure est réalisée en temps réel sur CPU. Voxelblox est une bibliothèque de cartographie volumétrique basée principalement sur les TSDF (Truncated Signed Distance Fields).

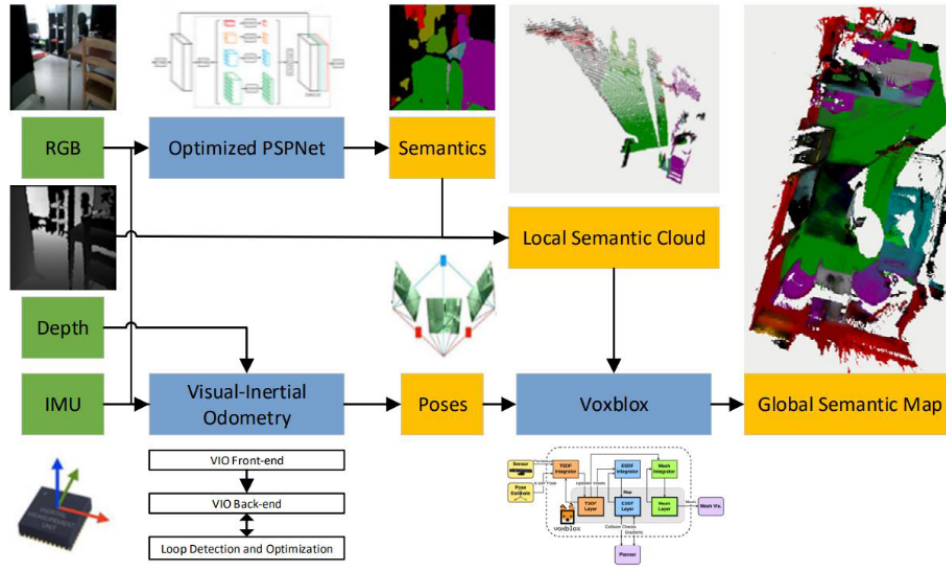


FIGURE 5 – Schéma présentant le fonctionnement du système proposé.

### 3.1.3 Towards Real-time Semantic RGB-D SLAM in Dynamic Environments

Cet article [3] présente une méthode de SLAM s’inspirant des features ORB (Oriented FAST and Rotated BRIEF). Le module de segmentation sémantique est une adaptation de SegNet, un réseau de neurones léger qui permet un traitement en temps réel. La segmentation se concentre sur des objets dynamiques dont les caractéristiques ne seront pas utilisées dans la création de la carte locale. La segmentation est faite uniquement sur les images clés les plus récentes, accélérant grandement le processus. La segmentation peut être très rapide, mais reconnaît principalement les objets qu’elle connaît déjà et est moins efficace dans des milieux inconnus avec des objets dynamiques nouveaux. Ce module essaye de résoudre ce problème. Pour chaque nouvelle image l’idée est d’utiliser l’algorithme du K-means pour segmenter l’image de profondeur en  $N$  clusters. Chaque cluster est considéré comme faisant partie du même objet. Pour chaque cluster, une erreur de reprojection est calculée. Si une des erreurs moyennes est relativement supérieure aux autres, le cluster est considéré comme un objet dynamique et les points caractéristiques de l’objet seront supprimés du traitement. Cette manière de traiter les images permet de réduire le taux de faux positifs. Les comparaisons sont effectuées d’une image clé à une autre puisqu’elles se ressemblent beaucoup, mais aussi entre la première estimation et les cartes locales obtenues.

### 3.1.4 ORB-SLAM3 : An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM

ORB-SLAM3 [2] est une amélioration d'ORB-SLAM2, présenté en Figure 6, qui consistait à travailler sur trois tâches simultanément : le suivi, la cartographie locale et la fermeture de boucle. Dans sa partie de suivi, ORB-SLAM2 fait correspondre les caractéristiques image par image et les compare avec une carte locale pour trouver l'emplacement exact de la caméra en temps réel. Il procède à un ajustement en fonction du mouvement afin de minimiser l'erreur de reprojection. Vient ensuite la partie cartographie locale dans laquelle ORB-SLAM2 crée des cartes locales et les optimise en utilisant des algorithmes tels que Iterative Closest Point (ICP) et effectue un ajustement local afin de calculer la position la plus probable de la caméra. Enfin, il utilise l'optimisation du graphe des positions pour corriger la dérive accumulée et effectuer une fermeture de boucle. Il est nécessaire d'effectuer un ajustement groupé après la fermeture de la boucle, afin que l'utilisateur se trouve à l'emplacement le plus probable dans la carte corrigée. Après l'ajout d'une image clé à la carte ou l'exécution d'une fermeture de boucle, ORB-SLAM2 peut démarrer un nouveau fil d'exécution qui effectue un ajustement de l'ensemble de la carte afin que l'emplacement de chaque image clé et des points dans celle-ci obtienne une valeur d'emplacement ajustée.

Pour traiter une image RGBD, on calcule d'abord les caractéristiques ORB sur l'image RGB, ensuite on estime les coordonnées de l'image de gauche depuis une paire d'images. Un point est associé à "proche" ou "éloigné" en fonction de sa profondeur. Chaque dénomination a des caractéristiques utiles, un point proche sera représentatif pour le changement d'échelle, la translation, la rotation alors qu'un point éloigné sera surtout utile pour la rotation et devra être supporté par un plus grand nombre d'images. À l'initialisation, le système nécessite seulement les premières images clés pour estimer la première carte locale en utilisant les informations de profondeur. L'insertion d'une nouvelle image clé est importante, car elle définit le nouvel environnement sur lequel se base l'estimation du mouvement de la caméra. Elle suit l'idée établie par la première version d'ORB-SLAM, d'insérer régulièrement de nouvelles images clés, même si cela implique de devoir les retirer si elles sont redondantes. L'ajout de points clés proches et éloignés permet un nouveau seuil d'apparaître pour déterminer si une nouvelle image clé est nécessaire. Si le nombre de points proches devient inférieur à un seuil, il sera nécessaire d'insérer une nouvelle image clé comportant au moins un certain nombre de points proches supérieur à un autre seuil.

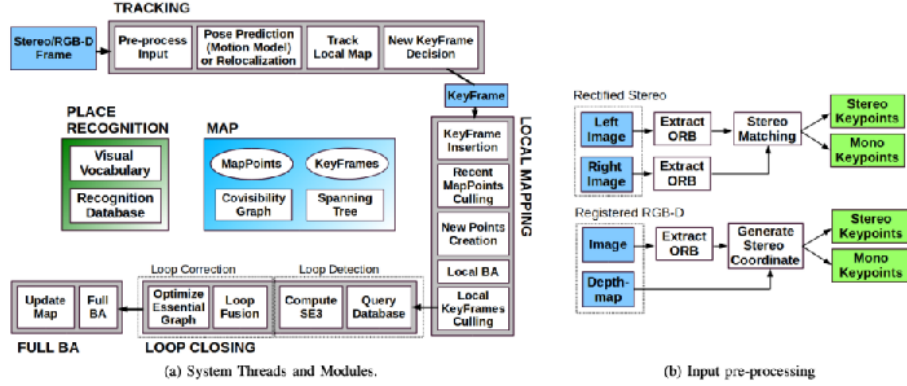


FIGURE 6 – Schéma présentant le fonctionnement d'ORB-SLAM2.

ORB-SLAM3 introduit l'utilisation de mémoire à court, moyen et long terme. En se servant des données que l'on a déjà vues, on peut retrouver facilement les boucles et ainsi réduire les erreurs de dérive. Une représentation en sac de mots dynamique (DBoW2) est utilisée. Ce système est présenté en Figure 7

Le thread de tracking traite chaque nouvelle image afin de vérifier si elle représente une nouvelle image clé afin de la localiser dans la carte active de l'atlas. Si le tracking est perdu, le système essaye de se relocaliser en cherchant des repères visuels qu'il a déjà rencontrés par le passé. Si cela ne fonctionne pas, il créera une nouvelle carte qui sera la nouvelle carte active dans l'atlas. On dispose donc d'un atlas de cartes locales, un ensemble de cartes déconnectées qui seront peut-être combinées avec le temps grâce au système de fermeture de boucles.



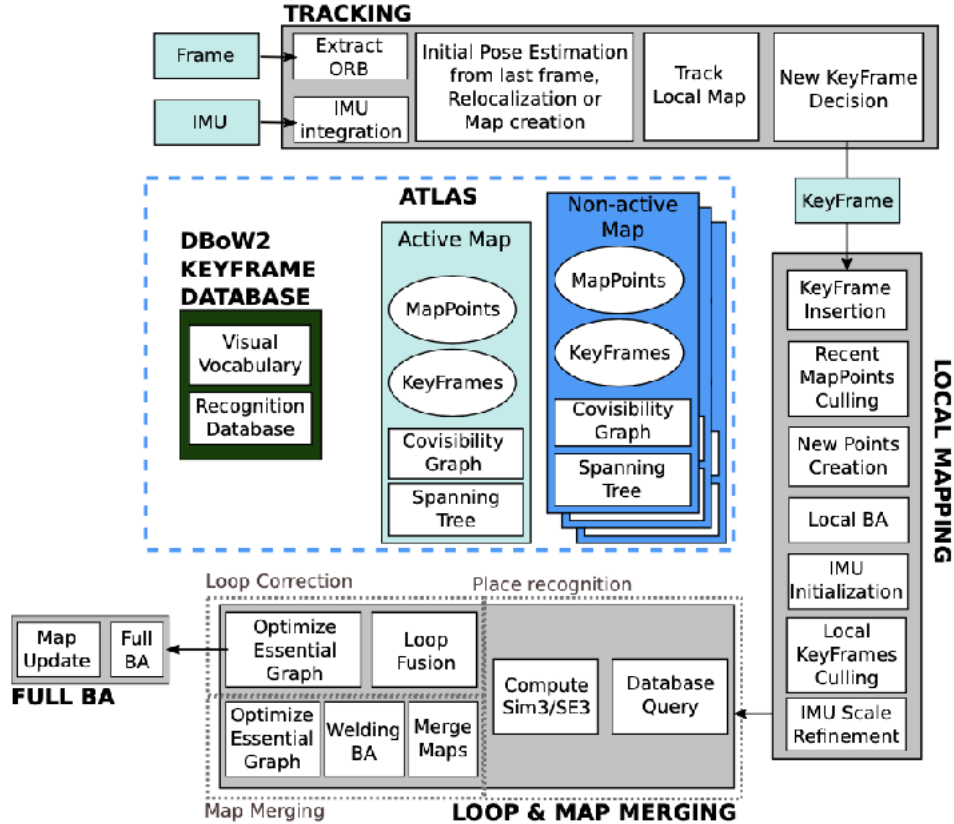


FIGURE 7 – Schéma présentant le fonctionnement d'ORB-SLAM3.

### 3.2 Synthèse

L'article "Wearable Travel Aid for Environment Perception and Navigation of Visually Impaired People" est très détaillé et demande beaucoup de temps pour saisir chaque subtilité. Au début de l'article, les auteurs listent les solutions qui existent déjà et expliquent leurs limitations respectives. Chaque partie de la solution est expliquée et peut être trouvée dans leurs travaux précédents. Cette solution prendrait beaucoup de temps à implémenter puisque les auteurs font surtout une description de leur travail sans mettre à disposition de code source. Il faudrait parcourir tous leurs travaux précédents pour comprendre l'intégralité de leur solution. De plus il faudrait adapter les travaux proposés afin de créer la carte locale en temps réel plutôt qu'avant, puisque c'est une des contraintes imposées par nos superviseurs.

L'article "CDSFusion : Dense Semantic SLAM for Indoor Environment Using CPU Computing" est très prometteur, mais n'est pas assez détaillé pour nous permettre de reproduire leur solution à partir de rien. De plus ils utilisent des capteurs IMU, que nous avons convenu de ne pas utiliser dans le cadre de ce projet.

L'article "Towards Real-time Semantic RGBD SLAM in Dynamic Environments" apporte une solution de tracking de la localisation d'un module en temps réel, ce qui correspond à ce que l'on recherche dans le cadre de ce projet. Cette solution aide à garder en mémoire les directions prises jusqu'à arriver à un point donné. Cependant les auteurs ne parlent que peu d'ORB-SLAM et se focalisent surtout sur leur valeur ajoutée permettant d'améliorer les résultats de ce dernier.

Enfin, l'article "ORB-SLAM3 : An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM" semble être le plus abouti que nous ayons pu lire, il est à l'état de l'art et obtient de meilleures performances que ses concurrents. L'article est assez bien expliqué et fournit un code source disponible sur Github. Leur solution semble réutilisable pour notre problème, cela apporterait un bon moyen de traquer une personne malvoyante dans une pièce. Certains aspects de la publication semblent compliqués à comprendre, mais la méthode en elle-même nous semble accessible. Il faudrait passer du temps sur leur code source afin de l'appréhender et se rendre compte de la faisabilité de sa mise en place.

### 3.3 Conclusion

Pour conclure sur cet état de l'art, nous avons découvert plusieurs méthodes permettant de traquer le mouvement d'une caméra dans un milieu intérieur, chacune avec son niveau de complexité et de faisabilité par rapport à nos contraintes qui étaient de n'utiliser aucun autre capteur qu'une caméra. Notre but était de fournir un module permettant de guider une personne d'un point A à un point B avec l'aide de notre module créant une carte locale de l'environnement en se basant sur des points et des lieux d'intérêt, s'emboîtant avec les modules de traitement du langage naturel, et le module de détection d'objets. L'article "ORB-SLAM3 : An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM" nous a semblé être le plus pertinent pour arriver à notre but. En effet nous pourrions nous baser sur leur solution de tracking et construire par-dessus pour répondre au besoin fonctionnel de ce projet. De plus le fait que les auteurs mettent à disposition leur code source donne une première vision de ce qui pourrait être réalisé en termes de performances avec notre propre environnement.

## 4 Les dimensions techniques du projet (8 à 12 p)

Description des objectifs/tâches qui vous ont été confiés, vous exposez les difficultés rencontrées et la manière dont vous les avez abordées. Vous mettez également en valeur l'originalité éventuelle de votre approche, les choix et décisions que vous aurez prises. Vous mettez vos travaux et réalisations en perspective par rapport à l'ensemble du projet et son historique.

### 4.1 Phase de preuve de concept

Le projet FYW vise à aider les personnes malvoyantes à se repérer en milieu intérieur. La preuve de concept (POC) sur laquelle j'ai travaillé avec mon équipe se concentrait sur la création d'une carte locale de l'environnement en nous basant sur des points d'intérêts (POI) entre les images vues successivement, c'est la méthode SLAM.

#### 4.1.1 Installation

Il a d'abord fallu se pencher sur l'installation de la solution que nous avons retenue, afin de construire notre module par-dessus. Nous avons été capables de produire une documentation d'installation de ORB-SLAM3 sur le Windows sub-system for Linux (WSL2) et sur une machine virtuelle (VM). Ces documentations d'installation appartiennent à Aubay et ne peuvent être divulguées dans ce document. Ensuite il a fallu calibrer la caméra que nous utilisions pour fonctionner avec ORB-SLAM3. La calibration de la caméra assure que l'estimation de la position de la caméra est la plus proche de la position réelle de la caméra. Enfin, l'idée était de développer un algorithme de recherche du plus court chemin entre deux points donnés, par exemple entre la position de l'utilisateur et une destination précédemment sauvegardée.

**4.1.1.1 General** Notre méthode se base sur ORB-SLAM3, une solution open source créée dans une université en Espagne. L'idée était simple, nous voulions l'installer et la faire fonctionner puis ajouter les fonctionnalités nécessaires à notre POC. Ce projet open source a été développé en C++ et se base sur de nombreuses bibliothèques de code telles que Eigen, Pangolin, Sophus, DBoW2, g2o, Kalibr et ROS. Avant d'installer ORB-SLAM3, nous avons dû installer Eigen et Pangolin. Sophus, DBoW2 et g2o étant déjà inclus dans leur dépôt Github.

Nous avons essayé d'installer ORB-SLAM3 sur Windows 10, Mac OSX, Ubuntu 20.04 sur VirtualBox et enfin WSL2. L'installation sous Windows 10

a rapidement été déclarée impossible puisqu'ORB-SLAM3 a été développé pour un système Linux en grande majorité. L'installation sur Mac OSX n'a été que temporaire, le temps que l'on obtienne l'autorisation de travailler avec WSL2. Nous avons donc réalisé l'installation d'ORB-SLAM3 sur WSL2 et réussi à lancer le SLAM sur la vidéo d'exemple fournie. Cependant notre POC devait fonctionner avec un flux vidéo en temps réel provenant d'une caméra, et non une vidéo préenregistrée. Le problème était que bien que wsl2 ai accès aux appareils USB (en prenant en compte que pour cela il était nécessaire de créer un nouveau noyau Linux contenant les bons pilotes) et que la caméra soit visible dans les pilotes USB, le sous-système n'était pas capable de créer la vidéo dans `/dev/video`, rendant le flux vidéo inaccessible à notre application. Nous avons choisi de basculer sur un système Ubuntu 20.04 sur VM afin de disposer des fonctionnalités de base nécessaires à notre avancée.

**4.1.1.2 Eigen** Eigen est une bibliothèque d'algèbre linéaire, elle traite les vecteurs, matrices, solveurs numériques et les algorithmes associés. Cette bibliothèque est nécessaire au bon fonctionnement d'ORB-SLAM3. Pour l'installer, nous avons dû chercher la version adaptée puisque la bibliothèque a évolué depuis la publication du dépôt Github d'ORB-SLAM3, rendant les versions les plus récentes de Eigen incompatible avec le projet. Nous avons essayé plusieurs versions qui n'ont pas fonctionné jusqu'à déterminer que la version 3.3.1 était la version utilisée au moment du développement d'ORB-SLAM3.

**4.1.1.3 Pangolin** Pangolin est un ensemble de bibliothèques utilitaires légères traitant de prototypage 3D et d'algorithmes numériques ou basés sur de la vidéo. Elles sont utilisées pour s'affranchir des contraintes d'environnement et rendre facile la visualisation des données. Pour l'installation, de même que pour Eigen, la version la plus récente ne correspondait pas alors il a fallu chercher dans l'historique des versions et tester jusqu'à trouver la version 0.6 sur leur Github qui nous a permis de lancer le projet.

**4.1.1.4 ORB-SLAM3** Pour installer ORB-SLAM3, il a d'abord fallu cloner le dépôt Github de la version 1.0. Il y avait des prérequis qui ne correspondaient pas à notre environnement tels que la version d'OpenCV qu'il a fallu adapter, nous avons déterminé que la version 4.2 était la bonne. Pour chaque bibliothèque nécessaire, nous avons dû trouver la version adaptée à notre projet. Ensuite nous avons pu installer les bibliothèques tierces déjà

contenues dans le dossier du projet.

**4.1.1.5 ROS** Robot Operation System (ROS) est utile au moment de l'accès à la caméra par ORB-SLAM3 pour traiter le flux vidéo. Ce package censé être rapide à installer nous a posés beaucoup de soucis de packages cassés dans les dépôts en ligne. Nous avons essayé diverses solutions sans succès sur la VM, sans succès, pour finalement réussir à faire fonctionner le tout sur WSL2.

#### 4.1.2 Calibration

Comme dit précédemment, la calibration de la caméra était une phase importante afin d'obtenir une bonne estimation de la position de la caméra utilisée. En effet, les paramètres intrinsèques et les distorsions de la caméra sont ce qui définit la manière dont un objet dans le monde réel va être projeté dans l'espace image. C'est la première étape nécessaire à obtenir une méthode de SLAM performante. Ces paramètres varient d'une marque à une autre, mais aussi entre les différents modèles d'une même marque.

Ros et Kalibr sont deux packages qui permettent d'effectuer une calibration. Le premier est pratique à utiliser puisqu'il donne un retour instantané sur la calibration et indique quand celle-ci est finalisée. Le second est moins visuel, mais c'est le module conseillé par les développeurs d'ORB-SLAM3.

Le processus de calibration est le même pour chaque package. Le principe est d'utiliser un modèle de dimension connue comme un plateau d'échec contenant des cases blanches et noires de taille fixe. L'idée est donc d'estimer la déformation de la caméra sur ces dimensions connues en montrant plusieurs plans de vue du plateau à la caméra.

#### 4.1.3 Recherche de chemin

Après avoir passé un certain temps sur le code source d'ORB-SLAM3 pour comprendre comment chaque partie fonctionnait, nous avons été en capacité de réfléchir à comment nous implanter dedans pour ajouter notre contribution. L'idée était d'être capable de garder en mémoire les lieux importants pour l'utilisateur (POI) et de lui permettre d'y retourner en trouvant le chemin le plus court entre la position courante et la destination voulue grâce à l'algorithme A\*.

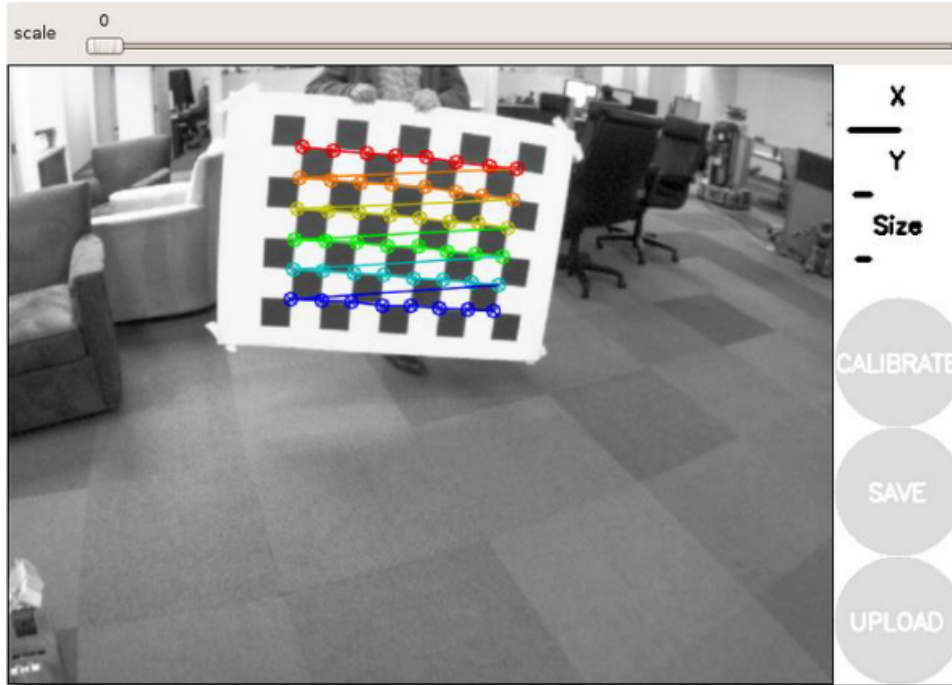


FIGURE 8 – Exemple de calibration.

**4.1.3.1 Matrice de coordonnées** Les coordonnées résultantes d'ORB-SLAM3 sont des nombres flottants généralement compris dans l'intervalle  $[0, 1]$  pour les vidéos courtes. Nous avons besoin d'un moyen de projeter ces coordonnées sur une grille afin de déterminer le chemin à parcourir et de donner des indications de direction alors il a fallu adapter le résultat fourni par ORB-SLAM3.

La projection est assez simple, nous multiplions la coordonnée flottante de la caméra par 10, on arrondit au supérieur afin d'obtenir un nombre entier puis on divise le résultat par 10. Cela nous a permis d'obtenir une grille de coordonnées discrétisées qui ne soit pas trop petite (une grille trop petite correspondrait à des espacements plus petits qu'un pas dans le monde réel), ce qui nous permet de ne pas guider l'utilisateur trop souvent. Un exemple de cette méthode de discrétisation est présenté dans la Figure 9.

**4.1.3.2 Structures de données définies**

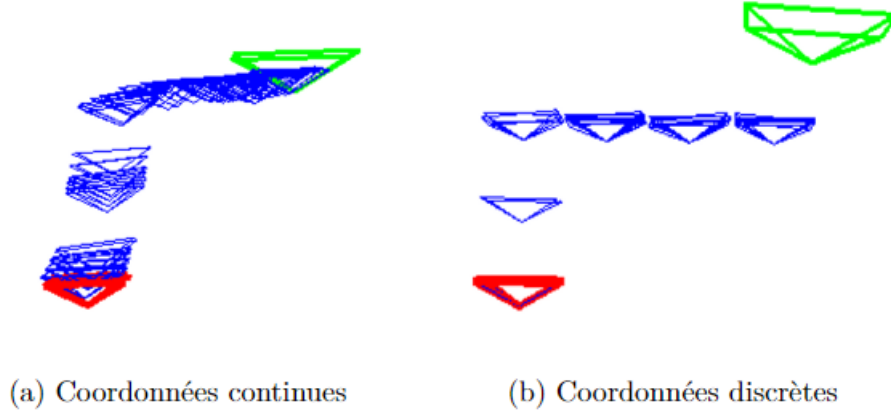


FIGURE 9 – Exemple de discrétisation.

**4.1.3.2.1 Coord** Nous avons créé une structure *Coord* en C++, équivalente à un vecteur 2D. Cette structure est utilisée par tous nos algorithmes.

**4.1.3.2.2 AbsoluteCoord** Les coordonnées absolues sont les coordonnées de la caméra qui ont été projetées sur une grille discrète. Nous avons créé une structure *AbsoluteCoord* contenant 5 champs :

- *int*  $x_{min}, x_{max}, y_{min}, y_{max}$  : les dimensions de la grille.
- *vector*<*Coord*> *coords* : Le chemin parcouru par la caméra.

**4.1.3.2.3 Path** Nous avons créé une structure *Path* contenant 2 champs :

- *Coord start* : la position courante sur la carte quand le chemin est sauvegardé, celle qui sera fournie à l'algorithme A\* pour la recherche de chemin.
- *vector*<*vector* < *int* >> *Map2D* : La matrice représentant les chemins explorés par l'utilisateur. Cette carte 2D est créée depuis les coordonnées absolues.

**4.1.3.2.4 PoiList** Nous enregistrons la liste des POI sauvegardés sous la forme d'un dictionnaire *map*<*string*, *Coord*> *poiList* qui associe un nom à une position. Ce dictionnaire est mis à jour lorsque l'utilisateur demande la sauvegarde d'un endroit.



**4.1.3.3 L’algorithme A\*** L’algorithme A\* est un algorithme de recherche dont le but est de trouver le plus court chemin entre 2 positions dans une matrice. Imaginons une grille carrée contenant des obstacles répartis aléatoirement, on fournit la position de départ *A* et la destination *B*. Le but est de rejoindre la destination le plus rapidement possible. L’algorithme A\* prend 3 paramètres :

- *G* : Le coût de déplacement de la cellule initiale jusqu’à la cellule courante. Cela correspond à la somme de toutes les cellules qui ont été visitées depuis le départ de la première cellule.
- *H* : La valeur heuristique, c’est une estimation du coût de déplacement depuis la cellule courante jusqu’à la destination. Il faut faire attention à ne pas surestimer cette valeur.
- $F = G + H$

A\* base sa décision sur la valeur de *F*, il cherche toujours à minimiser cette valeur jusqu’à atteindre sa destination, comme le montre la Figure 10.

|          |   |   |          |
|----------|---|---|----------|
| <b>A</b> |   |   |          |
|          | * |   |          |
|          |   | * |          |
|          |   | * |          |
|          |   |   | <b>B</b> |

FIGURE 10 – Exemple de A\*.

**4.1.3.4 Retourner à un point d’intérêt** Dans cette partie nous avons ajouté un bouton à l’interface graphique fournie avec ORB-SLAM3 qui affiche la liste des POI et demande à l’utilisateur de choisir sa destination. La coordonnée associée à la destination choisie est donnée à A\* pour trouver le plus court chemin praticable. Le chemin trouvé est ensuite donné à la fonction *FYWBack()*.

**4.1.3.5 FYWBack** Après avoir récupéré la sortie de l’algorithme A\* sous la forme d’une liste de coordonnées, *FYWBack()* fourni des indications de directions en se basant sur la normale sur le plan de la caméra et calcule

l'angle avec la section de chemin le plus proche afin guider l'utilisateur. Le chemin à parcourir est régénéré section par section.

#### 4.1.4 Test et résultats

Nous avons testé notre travail sur plusieurs vidéos prises dans l'openSpace. Nous avons d'abord essayé de créer une grille 2D de la carte locale comme présentée dans la Figure 11. Ensuite, nous avons calculé les normales des images clés en temps réel comme présenté dans la Figure 12. Nous avons ensuite réussi à sauvegarder une position, appelée POI, et la nommer comme présenté en Figure 13. Enfin nous avons créé une fonction de retour à un POI depuis la position courante, montrée dans la Figure 14.

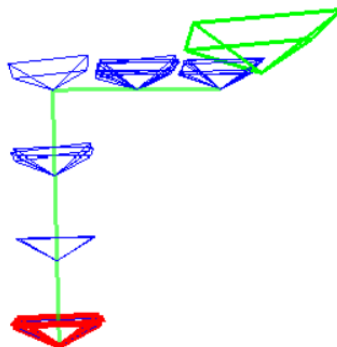


FIGURE 11 – Grille 2D de la carte locale.

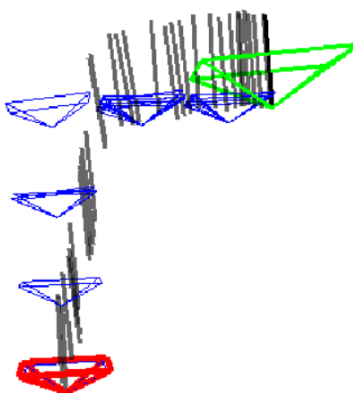


FIGURE 12 – Normales des images clés en temps réel.

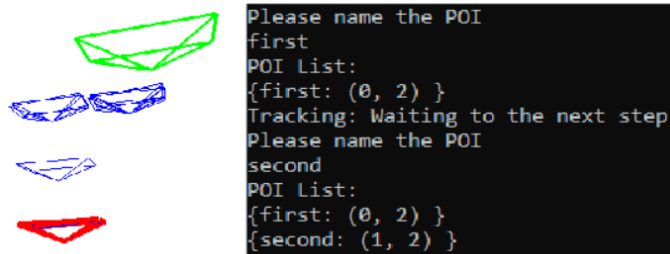


FIGURE 13 – Sauvegarde d'un point d'intérêt.

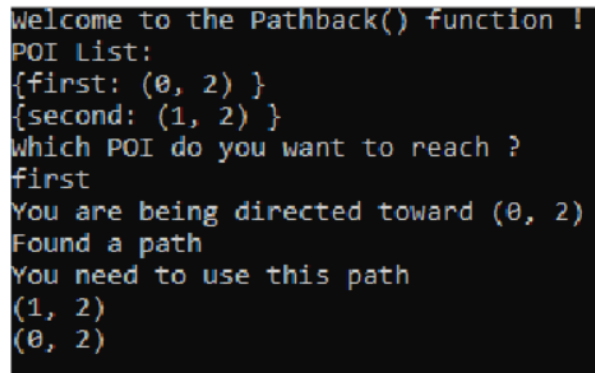


FIGURE 14 – Retour à un point d'intérêt.

#### 4.1.5 Difficultés rencontrées

Nous avons passé la majeure partie du temps qui nous était imparti pour le POC à essayer d'installer ORB-SLAM3. La recherche des bonnes versions pour chaque bibliothèque nous a beaucoup retardés. Certaines parties du code source ont dû être éditées, notamment au niveau de CMakeLists.txt pour faire fonctionner le tout. De nombreux problèmes sont survenus concernant quasiment chaque package qu'il fallait installer, la plupart du temps à cause de problèmes de compatibilité ou de documentation manquante. La seule solution que nous ayons trouvée a été de chercher sans relâche jusqu'à trouver une solution. Lorsque nous travaillions sur la VM, des latences et des crashes inattendus nous ont également fortement ralenti.

#### 4.1.6 Conclusion

Pour conclure sur la phase de POC, nous avons réussi à faire fonctionner un système SLAM pour la première fois chez Aubay, d'autres équipe ayant essayé sans succès par le passé. En plus de réussir l'installation, nous avons ajouté les fonctionnalités nécessaires pour atteindre notre but et avoir un module prêt à intégrer la phase projet pour venir se lier aux autres POC de notre projet.

### 4.2 Phase projet

Une fois la phase de POC terminée, nous devons passer à la phase projet nécessitant d'associer nos POCs afin de créer une application de démonstration pour la JDS du 7 juillet 2022. Cette journée est un jalon très important pour tout stagiaire au sein de la cellule Innov' de chez Aubay. Elle sert à présenter le projet sur lequel chaque équipe a travaillé afin de montrer à la direction générale ainsi qu'aux directeurs commerciaux ce dont les stagiaires sont capables et ce qu'Aubay pourrait proposer à ses clients dans le futur. Nous avons donc réfléchi à la création d'une application et d'une interface graphique nous permettant de démontrer le fonctionnement de notre application, bien que l'application finale soit destinée à des personnes malvoyantes, ne nécessitant qu'une interface vocale.

#### 4.2.1 Protocole de communication

Il est important de noter que nos 3 POCs ne fonctionnent pas sur les mêmes systèmes, en effet, le POC SLAM développé en C++ fonctionne uniquement sur WSL2, tandis que les POC Computer Vision (CV) et Natural Language Processing (NLP) sont développés en Python sur Windows. Le problème étant que WSL2 ne nous permettait pas d'accéder au flux vidéo de la caméra nécessaire aux POCs SLAM et CV, qu'une VM nous le permettait, mais ne donnait pas accès aux unités de traitement graphiques (GPU) nécessaires aux POCs CV et NLP. Nous avons opté pour trouver une solution concernant le problème de flux vidéo sur WSL2. Étant donné que les 3 POCs devaient fonctionner sur le même réseau local, nous avons eu m'idée de mettre en place un serveur afin d'établir une communication entre Windows et WSL2 pour transmettre le flux vidéo de Windows à WSL2. Cette solution a nécessité d'adapter la manière dont ORB-SLAM3 récupérait le flux vidéo avant de l'incorporer à son fonctionnement normal.

L'application est composée d'un backend se chargeant des fonctionna-

lités tandis que le frontend s'occupe du rendu graphique afin de présenter les résultats à l'utilisateur. La structure de l'application est simple : Les 3 POCs communiquent à un chef d'orchestre pour envoyer leur requête ou récupérer le résultat d'une requête précédemment effectuée et le frontend récupère les résultats à afficher après avoir sélectionné le mode de démonstration adapté. Ce fonctionnement est illustré dans la Figure 15.

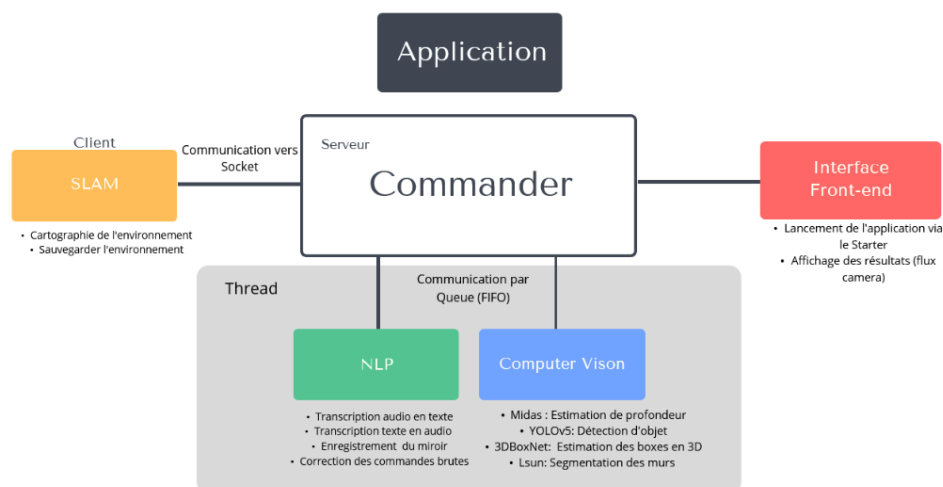


FIGURE 15 – Schéma présentant l'architecture de l'application.

Le POC SLAM se situant sur un système différent et utilisant un langage de programmation différent des POCs CV et NLP, il a fallu recourir aux sockets qui sont disponibles dans beaucoup de langages de programmation dont C++ et Python. Après avoir autorisé les communications au niveau du proxy et du par-feu en ouvrant les bons ports, cette communication par socket a pu être réalisée. Une fois la communication rendue possible, il a fallu ajouter des threads au code d'ORB-SLAM3 permettant des interruptions pour envoyer les données nécessaires après leur traitement ou recevoir les instructions du chef d'orchestre afin d'exécuter les fonctions utiles au bon moment. Un schéma global des communications établies au sein du projet est disponible en annexe.

#### 4.2.2 Développement de l'application

L'application fonctionne avec une caméra branchée pour transmettre le flux vidéo aux différents modules s'en servant ainsi qu'avec un micro et un haut-parleur. L'interface graphique donne la possibilité de lancer une démonstration et de démarrer les parties SLAM, CV et NLP via le serveur Flask et des sockets. Cette interface a été développée avec Python grâce à la bibliothèque PyQt5. La page principale de l'application est montrée en Figure 16, on y voit les résultats des POCs CV et SLAM ainsi que les différents boutons permettant à l'utilisateur d'interagir pour tester l'application. Ces boutons servent à :

- Lancer une vidéo de démonstration.
- Lancer l'application en temps réel.
- Utiliser le microphone pour exprimer une commande vocale lorsque l'application s'exécute en temps réel.
- Sauvegarder la position courante comme un lieu favori afin de pouvoir s'y faire guider plus tard.
- Demander à se faire guider à un lieu précédemment sauvegardé.
- Afficher l'historique des commandes utilisées et réponses reçues.

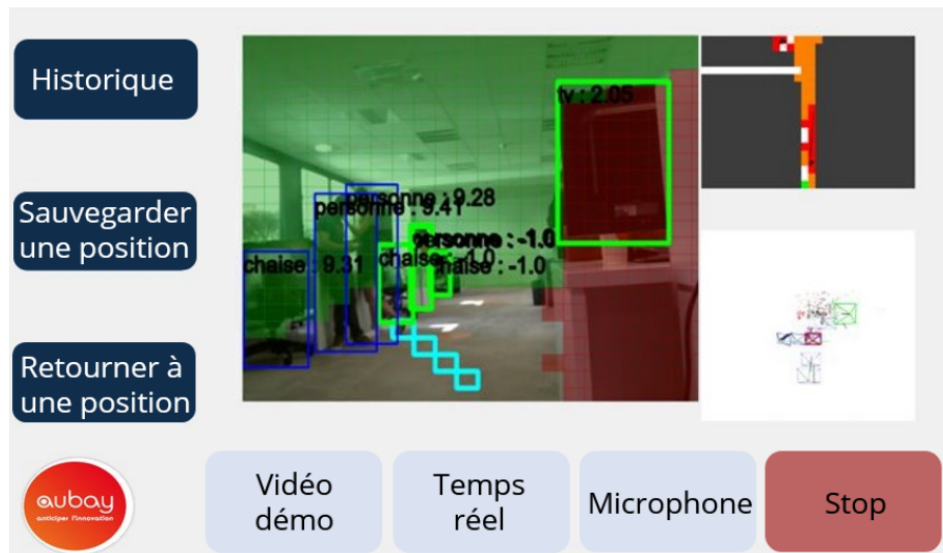


FIGURE 16 – Page principale de l'application.

### **4.2.3 Conclusion**

Pour conclure sur la phase projet, nous avons travaillé ensemble sur la construction d'une application complète liant nos POCs afin de fournir une démonstration le jour de la JDS. Cette journée s'est parfaitement déroulée et nous avons pu échanger avec différentes personnes de la direction d'Aubay ainsi que des directeurs commerciaux pour leur expliquer le fonctionnement de notre application. Nous avons appris à faire communiquer nos applications grâce à Flask et les sockets en établissant un protocole de communication.

## **5 Les dimensions humaines et managériales (3 à 5 p)**

Les dimensions humaines et managériales internes à l'organisme d'accueil. Cette partie consiste en une présentation analytique des processus d'entreprise. Selon les cas, elle portera sur la conduite de projet, les aspects organisationnels, la gestion du changement, le travail en groupe, l'énoncé des objectifs individuels et de l'équipe, la contribution à l'atteinte des objectifs, les difficultés rencontrées, les aides reçues, etc. Les difficultés propres à l'entreprise ou au service dans lequel la mission a été effectuée doivent être abordées de façon professionnelle pour que le jury ait une appréciation réaliste des conditions du travail réalisé.

Méthode agile Daily Comité le mardi à 14h45 Jira pour suivre le projet et ses avancées, sprint de 2 semaines Scrum Master élu chaque semaine 3 groupes de 3, 3 et 4 indépendants puis mise en commun lors de la phase projet



## 6 Conclusion (2 à 3 p)

La conclusion générale, de quelques pages, porte sur l'ensemble de votre expérience technique et humaine. Une première partie correspond au bilan et une deuxième partie expose les possibilités d'évolution du projet, du produit. Enfin, vous présenterez vos perspectives d'évolution par rapport à votre projet professionnel initial. Vous préciserez les compétences que vous avez développées en école d'ingénieurs et durant votre mission de fin d'études et préciserez vos axes d'amélioration.

Bilan technique sur le projet

Possibles évolutions

## 7 Bibliographie

1. Jinqiang BAI et al. « Wearable Travel Aid for Environment Perception and Navigation of Visually Impaired People ». In : *Electronics* 8.6 (juin 2019), p. 697. DOI : [10.3390/electronics8060697](https://doi.org/10.3390/electronics8060697).
2. Carlos CAMPOS et al. « ORB-SLAM3 : An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM ». In : *IEEE Transactions on Robotics* 37.6 (déc. 2021), p. 1874-1890. DOI : [10.1109/TR0.2021.3075644](https://doi.org/10.1109/TR0.2021.3075644).
3. Tete JI, Chen WANG et Lihua XIE. « Towards Real-time Semantic RGB-D SLAM in Dynamic Environments ». In : *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Mai 2021, p. 11175-11181. DOI : [10.1109/ICRA48506.2021.9561743](https://doi.org/10.1109/ICRA48506.2021.9561743). arXiv : [2104.01316 \[cs\]](https://arxiv.org/abs/2104.01316).
4. Sheng WANG et al. « CDSFusion : Dense Semantic SLAM for Indoor Environment Using CPU Computing ». In : *Remote Sensing* 14.4 (jan. 2022), p. 979. DOI : [10.3390/rs14040979](https://doi.org/10.3390/rs14040979).

## 8 Annexes

### 8.1 Point d'intérêt

Le but d'un point d'intérêt est de permettre à un algorithme de déterminer que deux objets sont bien les mêmes sur deux images différentes en ayant appliqué des modifications telles qu'un changement d'échelle, une rotation, une translation ou même une variation de l'intensité lumineuse. L'idée est donc de trouver des points caractéristiques sur les images récupérées de manière à ce que ces points décrivent au mieux l'objet en cours de traitement. En regroupant ces points caractéristiques, appelés "points d'intérêts", on est capable de déterminer si deux objets avec un point de vue différent sont les mêmes ou non. Les méthodes d'extractions de POI les plus populaires sont les méthodes SIFT et ORB par exemple. Une fois un point d'intérêt détecté, on veut être capable de résumer son information d'une manière simple afin de pouvoir le comparer à d'autres points d'intérêts. On peut utiliser l'intensité lumineuse ou le gradient des pixels environnant le point d'intérêt pour former un descripteur correct, tout en respectant les règles de discrimination et d'invariance. Les descripteurs sont en général stockés dans des vecteurs de 128 bits, ce qui permet une comparaison rapide. Un exemple d'appariement des POI est présenté dans la Figure 20.

La comparaison entre deux descripteurs se fait généralement par des calculs booléens tels que présenté dans la Figure 17.



FIGURE 17 – Association de descripteur avec une opération booléenne.

### 8.2 FAST

Cette méthode d'extraction de POI vérifie pour chaque pixel de l'image d'intensité  $I_p$  si au moins 12 des pixels du cercle autour du pixel ont une intensité supérieure ou inférieure à une proportion de  $I_p$ . Autrement dit on regarde s'il y a une grosse démarcation d'intensité entre le pixel et ses voisins comme le montre la Figure 18.

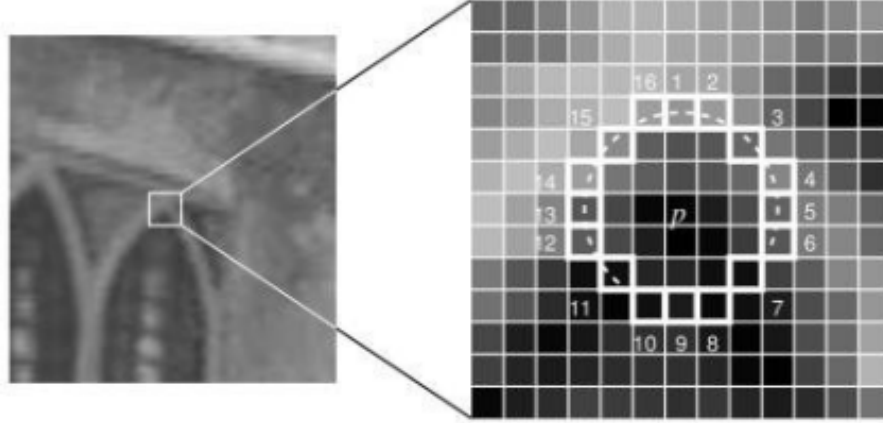


FIGURE 18 – Cercle de supériorité associé à un pixel.

### 8.3 BRIEF

Des paires de pixels autour du POI sont générées aléatoirement et des tests de supériorités sont effectués dessus. Un vecteur descripteur issu de la concaténation des résultats de ces tests de supériorité est créé. Un exemple est présenté dans la Figure 19.

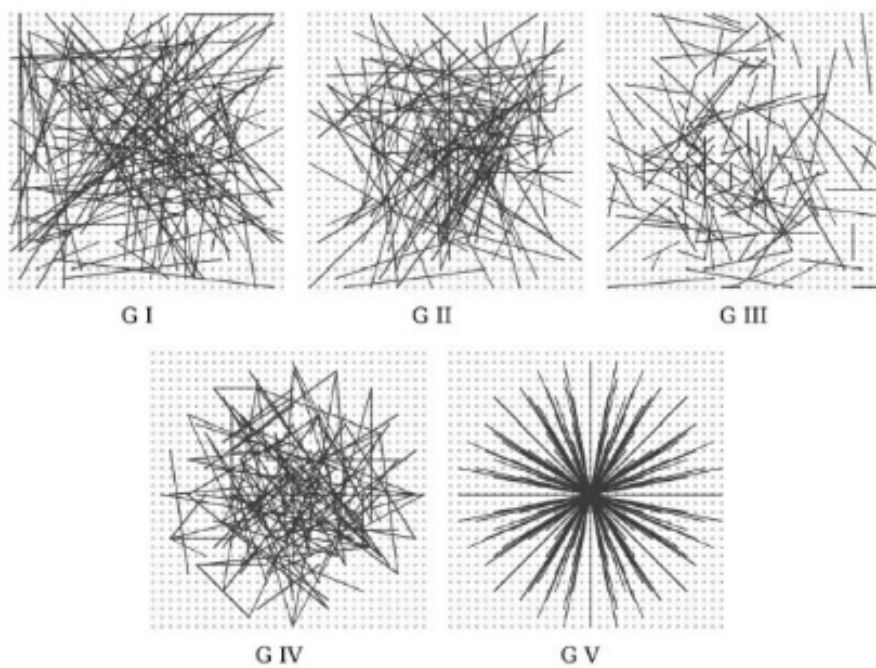


FIGURE 19 – Exemple de paires de pixels à tester.

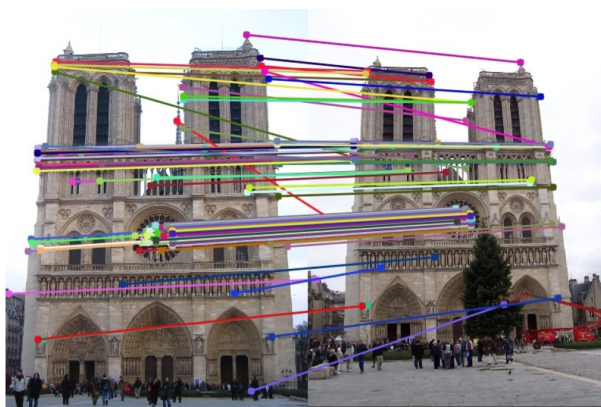


FIGURE 20 – Exemple d'association de POI.

## Acronymes

**BRIEF** Binary Robust Independent Elementary Features.

**CNN** Convolutional Neural Network.

**CPU** Central Processing Unit.

**CV** Computer Vision.

**DBoW2** Dynamic Bag of Words.

**EA** État de l’art.

**ESN** Entreprise de Services Numériques.

**FAST** Features from accelerated segment test.

**FYW** Find Your Way.

**ICP** Iterative Closest Point.

**IMU** Inertial Measurement Unit.

**NLP** Natural Language Processing.

**ORB** Oriented FAST and Rotated BRIEF.

**POC** Proof of Concept.

**POI** Point of Interest.

**RGB-D** Red, Green, Blue and Depth.

**ROS** Robot Operating System.

**SLAM** Simultaneous Localization and Mapping.

**TSDF** Truncated Signed Distance Fields.

**VIO** Visual-Inertial Odometry.

**VM** Virtual Machine.

**WSL2** Windows Subsystem for Linux 2.