



Grenoble INP – ENSIMAG

École Nationale Supérieure d’Informatique et de Mathématiques Appliquées

## Rapport de projet de fin d’études

### Find Your Way - Une solution de

Guillermain Nicolas  
3ème année – Option MMIS

12 février 2022 - 12 août 2022

**Aubay**  
13 rue Louis Pasteur,  
92100, Boulogne Billancourt  
FRANCE

**Responsable de stage**  
Éric Remilleret  
**Tuteur de l’école**  
Diane Larlus

## Table des matières

<b>1 Résumé</b>	<b>5</b>
<b>2 Problématique</b>	<b>6</b>
<b>3 L'entreprise</b>	<b>7</b>
<b>4 État de l'art</b>	<b>7</b>
4.1 SLAM . . . . .	7
4.2 ORB-SLAM2 : an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras [4] . . . . .	7
4.3 Towards Real-time Semantic RGB-D SLAM in Dynamic Environments [3]	8
4.4 Environments Wearable Travel Aid for Environment Perception and Navigation of Visually Impaired People[1] . . . . .	9
4.5 ORB-SLAM3 : An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM [2] . . . . .	9
4.6 DGS-SLAM A Fast and Robust RGBD SLAM in Dynamic Environments Combined by Geometric and Semantic Information [5] . . . . .	9
4.7 Conclusion d'état de l'art . . . . .	10
<b>5 Théorie</b>	<b>11</b>
5.1 Points d'intérêt . . . . .	11
5.1.1 Information . . . . .	11
5.1.2 Invariance . . . . .	12
5.2 Descripteur de point d'intérêt . . . . .	12
5.3 Appariement de descripteurs . . . . .	13
5.4 Orientated Brief and Rotated FAST : ORB . . . . .	14
<b>6 Méthodologie</b>	<b>15</b>
6.1 Intuition . . . . .	15
6.2 Environnements . . . . .	16
6.2.1 Windows . . . . .	16
6.2.2 WSL2 . . . . .	16
6.2.3 Machine virtuelle . . . . .	17
6.3 Installation librairies . . . . .	17
6.3.1 Eigen . . . . .	17
6.3.2 Pangolin . . . . .	17
6.3.3 OpenCV . . . . .	17
6.3.4 Système d'exploitation pour robot (Robot Operating System : ROS)	18

6.3.5	Kalibr	18
6.4	Prise en main	18
6.5	Architecture de la solution	18
6.5.1	Calibration	18
6.5.2	Recherche d'itinéraire : Pathfinding	20
6.5.2.1	Coordonnées discrétisées	20
6.5.2.2	Structures de données	21
6.5.2.3	Path	22
6.5.2.4	Instances dans la classe Map	23
6.5.2.5	A*	23
6.5.2.6	Retour à un lieu favori	24
6.5.2.7	FYWBack	24
6.5.3	Interaction avec l'utilisateur	25
6.6	Phase de projet	25
6.6.1	Nouvelle caméra	25
6.6.2	Environnement de l'application	26
6.6.3	Structure de l'application	27
6.6.4	Protocole de communication	28
6.6.5	Fonctionnement de l'outil	29
6.7	Plateforme de test	30
6.8	Améliorations	30
6.8.1	Précision ORB-SLAM3	30
6.8.2	Threads	31
6.8.3	Puissance de calcul embarquée	31
6.9	Analyse des résultats	32
<b>7</b>	<b>Bilan personnel</b>	<b>32</b>
<b>8</b>	<b>Impact environnemental</b>	<b>34</b>
<b>9</b>	<b>Conclusion</b>	<b>36</b>
<b>Glossaire</b>		<b>38</b>

## Remerciements

Je tiens à remercier toutes les personnes qui ont participé de près ou de loin à permettre le bon déroulement de ce stage.

Je remercie tout particulièrement mon maître de stage Éric Remilleret pour son attention au détail, pour ses conseils et son expérience précieux, pour son accueil et sa volonté de nous faire apprendre.

Je remercie Zachary Jessner, Florian Pisla, Nathan Cantat et Anne-France Galand, les responsables du projet Find Your Way, pour le temps qu'ils ont pris pour nous conseiller tout au long de ce stage, nous aiguiller et nous accompagner. Grâce à eux j'ai pu découvrir d'un peu plus près le domaine de la vision par ordinateur ainsi que de le traitement naturel du langage, et l'intelligence artificielle.

Je remercie mes co-stagiaires, N. Cisternas , V. Chaverot, J-N. Clink, J-B. Chanier, J. Menudier, M. Monneret, O. Phonchareun, M. Rasolofonora, R. Vidal, M. Wang, sans qui ce projet n'aurait pas pu être mené aussi loin. C'était un travail intense qui a vu une vraie émulation dans ce groupe de travail. Ce stage m'a permis de découvrir le travail en groupe, la prise de décision en commun et la volonté de mener un projet au bout.

Je remercie enfin l'entreprise Aubay et tous ses employés pour ce stage. Ils ont été accueillants, attentifs et pédagogues.

## 1 Résumé

*Find Your Way* est un projet qui a pour but de créer une solution de navigation en intérieur pour des personnes malvoyantes.

C'est un projet qui mêle des techniques d'apprentissage profond pour de la vision par ordinateur, du traitement naturel du langage et

**Mots clés :** *SLAM, ORB-SLAM, Extracteur ORB, SLAM dynamique, SLAM Multi-carte, Carte locale, Ssegmentation, Ttracking, Ttemps réel, Ssystème de navigation, Fframes d'intérêt, Ppoints d'intérêt, Vvision par ordinateur.*

## 2 Problématique

La cécité est une condition qui atteignait près de 35 millions de personnes dans le monde en 2015. C'est une condition qui peut être due à des maladies, des accidents ou bien qui peut être présente dès la naissance.

Les personnes étant atteintes de cécité doivent trouver des méthodes pour se débrouiller au jour le jour. La plupart utilisent une canne qui leur permet par un retour de force de savoir si des objets se trouvent sur leur chemin, d'autres utilisent des chiens d'aveugles qui les guident en utilisant leur propre vue. Ils développent généralement leurs autres sens. Celui qui progresse le plus est l'ouïe qui est utile pour savoir ce qui se rapproche.

Les personnes aux facultés visuelles diminuées (Visually Impaired Persons : VIP) se créent mentalement un monde pour éviter de se perdre. Ils ont souvent des itinéraires habituels, plus ou moins restreints. Cela leur laisse une certaine autonomie. En intérieur, chez eux ils doivent être très précautionneux et rangés pour retrouver toutes leurs affaires.

Les situations qui peuvent poser problème sont les phases de découvertes. En étant accompagnés, il leur est plus facile de se déplacer en intérieur comme en extérieur. Si par malchance ils se retrouvaient seuls, la découverte serait plus compliquée.

Ce projet s'inscrit dans cette optique. L'idée est de créer un prototype de système embarqué (lunettes, harnais, ...) qui permette d'analyser l'environnement en intérieur dans lequel se déplace le VIP et de lui fournir des directives de guidage, le prévenir des dangers potentiels et les chemins à ne pas prendre.

Il y a une première partie du projet qui s'applique à nommer les objets environnants la personne ainsi qu'à proposer un chemin "naïf" pour se rapprocher de portes en évitant les obstacles.

La deuxième partie adapte une méthode de Localisation et de cartographie simultanée (Simultaneous Localization and Mapping : **SLAM**) qui puisse permettre de proposer des itinéraires à l'utilisateur en se basant sur des chemins déjà empruntés, des points de vue déjà observés.

La troisième partie veut assurer la communication entre le système et l'utilisateur. C'est une partie qui utilise le traitement automatique du langage naturel (Natural Language Processing : NLP). Elle analyse les paroles prononcées par l'utilisateur, transforme le flux audio en un texte et communique avec les deux autres parties pour essayer de répondre au mieux aux questions posées.

L'objectif de ce stage est de proposer une adaptation du logiciel ORB-SLAM3 méthode de **SLAM** reconnue (considérée comme l'état de l'art par beaucoup), qui puisse proposer des trajets pour revenir à des points d'intérêts sauvegardés.

## 3 L'entreprise

## 4 État de l'art

L'étape d'état de l'art est une étape cruciale dans tout projet. Cette étape si elle est bien effectuée permet de se faire une bonne idée de l'avancée de la recherche sur une thématique. Elle a perduré pendant près de 4 semaines. Au cours de ces semaines de nombreuses publications scientifiques datant de ces dernières années ont été épluchées.

La recherche avait été commencée en se focalisant sur des méthodes de vision par ordinateur (Computer Vision : **CV**) et sur des systèmes embarqués qui aborderaient la problématique de déplacement en intérieur. C'est assez naturellement que les méthodes de **SLAM** sont apparues. Elles ont alors fait l'objet d'un état de l'art à part entière.

### 4.1 SLAM

Les méthodes de **SLAM** se basent sur la vision naturelle de l'homme. Si un homme est placé dans une rue qu'il connaît, il va la reconnaître presque instantanément. Comment parvient-il à reconnaître la rue immédiatement alors que les voitures ne sont plus les mêmes, la météo a changé depuis son dernier passage, il n'est pas forcément le même point de vue. L'homme a deux choses qui lui permettent d'effectuer le rapprochement. Tout d'abord il a une mémoire. Dans cette mémoire sont stockées les informations visuelles du point de vue, mais aussi la rue dans laquelle se situe ce point de vue. Ensuite il a un cerveau. Celui-ci lui permet de faire des rapprochements entre les bâtiments qu'il a vus et ceux qu'il voit en direct. Il peut aussi faire des associations entre des groupes de bâtiments qui apparaissent dans un même point de vue.

L'idée du **SLAM** est de reproduire ce fonctionnement. En temps réel le système de **SLAM** doit être capable de chercher des points qui contiennent de l'information. Des points qui caractérisent un objet et qu'on ne pourrait pas confondre avec un autre objet. Si l'on trouve une manière d'obtenir ces points d'intérêt, il suffira de comparer les points de l'image qu'on voit en ce moment avec les points des images qu'on a déjà vues. Ainsi il faut créer une base de donnée qu'on étoffe au fur et à mesure qu'on utilise la méthode de **SLAM**.

### 4.2 ORB-SLAM2 : an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras [4]

Cette méthode de **SLAM** a fait l'unanimité dans la communauté à sa sortie et a pendant longtemps été considérée comme étant l'état de l'art.

Ce système propose une version de **SLAM** qui inclut la réutilisation des cartes observées, la jointure de boucles et la possibilité de se relocaliser.

ORB-SLAM2 apporte aussi un protocole de choix de frames d'intérêt. En effet effectuer un traitement sur tous les frames d'une vidéo, alourdit le temps de traitement et empêche d'avoir des performances en temps réel, mais cela implique aussi que le système ne prend pas en compte la grande similarité entre les frames consécutifs d'une vidéo et donc fait des calculs peu utiles.

C'est un système qui en plus d'être très précis n'est pas très demandeur en ressources et fonctionne très bien sur CPU.

Cette méthode détecte et décrit les points d'intérêt avec la méthode ORB (Oriented FAST and Rotated BRIEF), qui s'appuie sur les points d'intérêt FAST et sur les descripteurs BRIEF. Cette méthode est considérée comme étant aussi efficace que des méthodes telles que SIFT ou SURF.

Les auteurs du papier ont décidé de mettre en ligne ce logiciel de manière libre pour permettre à la communauté de s'inspirer de ce travail, mais aussi pour permettre à des chercheurs spécialisés dans d'autres sujets

### 4.3 Towards Real-time Semantic RGB-D SLAM in Dynamic Environments [3]

Nous avons ensuite découvert cet article qui propose une méthode innovante de **SLAM**, se basant sur ORB-SLAM2, utilisant des canaux de couleur d'une image ainsi qu'une carte de profondeur pour obtenir un logiciel fonctionnel.

Ce système propose un module géométrique chargé d'estimer à partir d'estimation d'erreur de reprojection si un objet point d'intérêt appartient à un objet dynamique ou non. Cela permet d'éviter de faire des calculs d'estimation de la position de la caméra sur des points appartenant à des objets dynamiques qui ne feraient que fausser l'estimation et donnerait de moins bons résultats à la méthode de **SLAM**.

Cette méthode utilise aussi une forme de tracking des objets d'une frame à la suivante pour prévenir la redondance qui apparaît dans les frames à l'utilisation du système.

Enfin il utilise un réseau de segmentation des objets (SegNet)

#### **4.4 Environments Wearable Travel Aid for Environment Perception and Navigation of Visually Impaired People[1]**

Cet article introduit une solution correspondant à la problématique de notre projet, puisqu'il propose un système embarqué pour des personnes ayant des problèmes de vision. Ce n'est pas qu'un article présentant une méthode de **SLAM** mais plutôt un article complet qui aborde aussi les problématique d'indication d'itinéraire et d'aide au déplacement.

Cet article utilise une méthode de **SLAM** pour assurer le déplacement en intérieur. Il propose aussi une méthode qui permet de choisir un point vers lequel se rendre et un logiciel qui permet d'estimer et d'assurer l'itinéraire.

#### **4.5 ORB-SLAM3 : An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM [2]**

ORB-SLAM3 est une évolution de la version 2 du logiciel. De même que son grand frère, il utilise la méthode d'extraction ORB, et a battu la plupart des autres systèmes du moment à sa sortie. Il est aussi proposé à la libre disposition des internautes pour permettre à tous d'utiliser cette solution et de pouvoir construire dessus.

Cette solution propose l'innovation d'utiliser la mémoire à court, moyen et long terme au moyen d'une base de donnée : l'Atlas. Cet atlas permet de garder en mémoire la carte courante active et les cartes passées temporairement passives.

Il utilise une représentation en sacs de mots dynamiques (Dynamic Bag of Words DBoW2), qui permet d'accélérer encore l'association entre le frame observé et les frames gardés en mémoire.

Il apporte aussi une représentation abstraite du modèle de la caméra qui ne nécessite que les paramètres intrinsèques de la caméra pour obtenir une méthode de **SLAM** fonctionnelle.

#### **4.6 DGS-SLAM A Fast and Robust RGBD SLAM in Dynamic Environments Combined by Geometric and Semantic Information [5]**

Cette dernière méthode construit sur ORB-SLAM3. Sa valeur ajoutée est l'ajout d'un module d'apprentissage profond qui a pour but de détecter les objets dynamiques. Ce module couplé aux informations géométriques de mouvement permet de segmenter les objets dynamiques. Cette segmentation permet alors de supprimer les points d'intérêt correspondant à des objets dynamiques dans l'estimation de déplacement pour ne se baser

que sur des objets statiques de l'image et ainsi améliorer l'estimation du déplacement de l'utilisateur.

C'est une méthode qui a l'air prometteuse et fonctionnelle. Cependant elle nécessite l'entraînement d'un réseau de neurone profond pour détecter des objets dynamiques mais aussi une grosse puissance de calcul pour pouvoir utiliser ensuite le réseau de neurone en temps réel. cet aspect sera à prendre en compte dans la phase de **POC** et la répartition du travail.

## 4.7 Conclusion d'état de l'art

Plusieurs articles sont intéressants à étudier pour mener à bien le projet Find Your Way. Tout d'abord le plus intéressant et abouti semble être ORB-SLAM3. En plus d'être considéré comme étant l'état de l'art à l'époque, il est aussi en license libre et propose une bonne base de travail.

En effet ORB-SLAM3 propose une solution efficace et fonctionnelle, qui permet de travailler en temps réel. Il permet aussi une implémentation facile de nouvelles fonctionnalités.

il pourrait être intéressant d'envisager la proposition de DGS-SLAM d'ajouter un module de segmentation d'objet dynamiques pour permettre une meilleure estimation du déplacement de l'utilisateur.

## 5 Théorie

### 5.1 Points d'intérêt

#### 5.1.1 Information

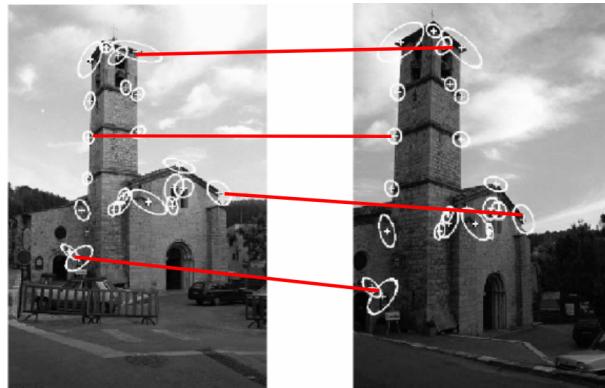
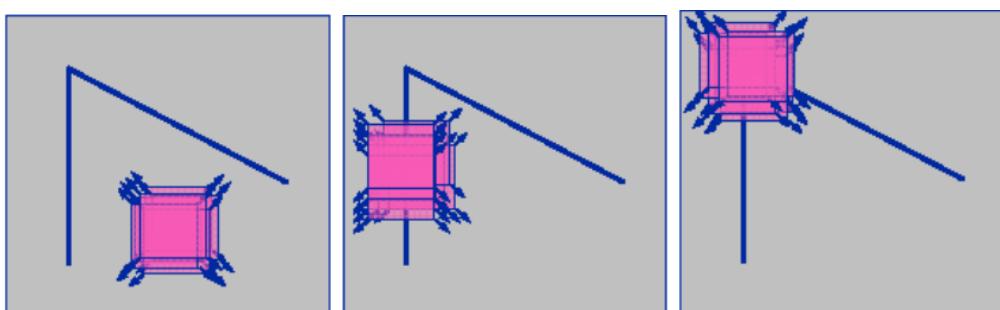


FIGURE 1 – Appariement de points

Prenons deux images d'un même bâtiment, prises avec deux points de vue légèrement différents. Comment est-ce qu'un algorithme pourrait déterminer si ces deux images correspondent effectivement à un même bâtiment ou bien si elles sont différentes ?

L'idée est d'être capable de trouver des points de l'image qui vont décrire précisément et localement des parties de l'image. En cherchant à faire des paires entre les nuages de point d'intérêt des deux images on sera capable de déterminer si les deux images sont du même bâtiment ou non. On appelle ces points des points d'intérêt (**POI**).

Un point d'intérêt est finalement un point qui va contenir beaucoup d'information et pour lequel la somme de l'intensité des pixels autour varie beaucoup avec un petit déplacement. Cela correspond en général à des parties angulaires, des coins,...



(a) **À-plat** : peu de changement d'intensité pour un petit déplacement (b) **Contour** : peu de changement d'intensité long du contour (c) **Coin** : grand changement d'intensité sur un petit déplacement

FIGURE 2 – Information contenue dans un point d'intérêt

### 5.1.2 Invariance

La détection d'un point d'intérêt doit suivre quelques règles pour être certain que sur deux images d'un même objet, les points d'intérêt détectés correspondent aux mêmes points sur les objets.

Ainsi un point d'une forme géométrique détecté comme un **POI** doit aussi être détecté comme un **POI** si la forme a subi une translation, une rotation, une variation affine de l'intensité,... Certaines méthodes permettent aussi d'assurer l'invariance au changement d'échelle.

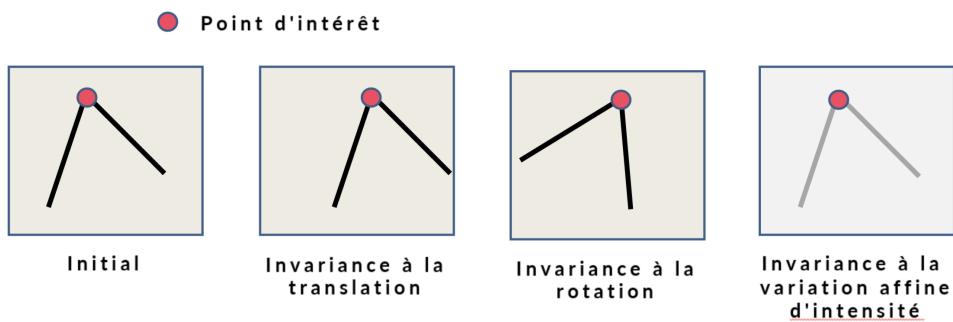


FIGURE 3 – Invariance de points d'intérêt

Il existe de nombreuses méthodes d'extraction de **POIs**. Les plus connues sont le détecteur de Harris, SIFT, ORB,...

### 5.2 Descripteur de point d'intérêt

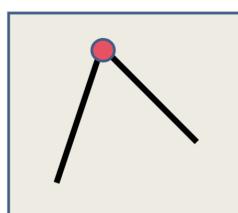


FIGURE 4 – Point d'intérêt extrait

Une fois qu'un point d'intérêt détecté, il faut être capable de décrire ce point de manière précise et légère pour permettre la comparaison entre les images. Le descripteur doit être local, il ne doit pas décrire un pixel mais doit décrire ce pixel dans son environnement. Ces descripteurs peuvent utiliser l'intensité des pixels autour du **POI**, ou encore la valeur des gradients autour du **POI**.

Un bon descripteur doit trouver le bon compromis entre être compact, discriminant et invariant.

De la même manière que pour l'extraction, il est primordial que la description soit invariante à beaucoup d'égards. Des normalisations de l'intensité de l'image ou encore des

gradients pour permettre de ne pas subir de petites variations.

On pourrait penser à un descripteur qui garderait toutes les valeurs des gradients et des pixels normalisés. Cela décrirait parfaitement le **POI** et serait une méthode très robuste. Cependant cette représentation serait trop lourde pour permettre de comparer efficacement deux images. Il faut ainsi trouver une forme compacte de représentation de cette information.

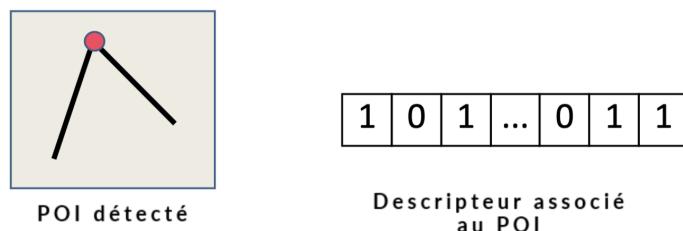


FIGURE 5 – Descripteur du point d’intérêt

En général, les descripteurs sont stockés dans des vecteurs de 128 bits. Cela permet une comparaison rapide entre les descripteurs des **POIs** mais permet aussi une description assez discriminante.

Les descripteurs les plus connus sont SIFT, HOG, SURF.

### 5.3 Appariement de descripteurs

L'appariement de **POIs** se fait en général par des calculs booléens sur les descripteurs.

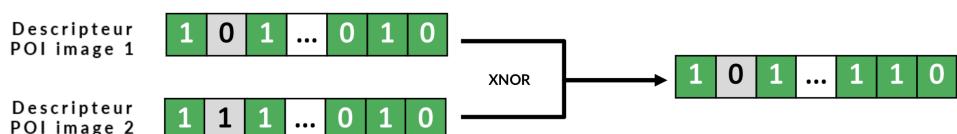


FIGURE 6 – Comparaison booléenne entre descripteurs

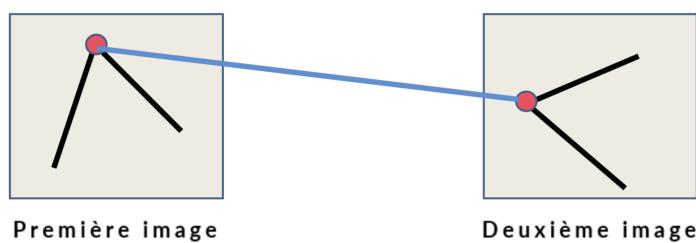


FIGURE 7 – Appariement de **POIs** entre deux images

## 5.4 Orientated Brief and Rotated FAST : ORB

ORB est une méthode d'extraction et de description de points d'intérêt. C'est une méthode qui est très efficace, qui permet de travailler sur des vidéos en temps réel et reste assez discriminante pour éviter de faire de mauvais appariements.

### Features from Accelerated and Segments Test : FAST

FAST est une méthode d'extraction de **POIs**. Cette méthode vérifie pour chaque pixel d'intensité  $I_p$  à l'intérieur de l'image si au moins 12 des 16 pixels du cercle autour du pixel ont une intensité supérieure ou inférieure à un pourcentage de  $I_p$  ( $I_j < 0.8 * I_p$ ,  $I_j > 1.2 * I_p$ ).

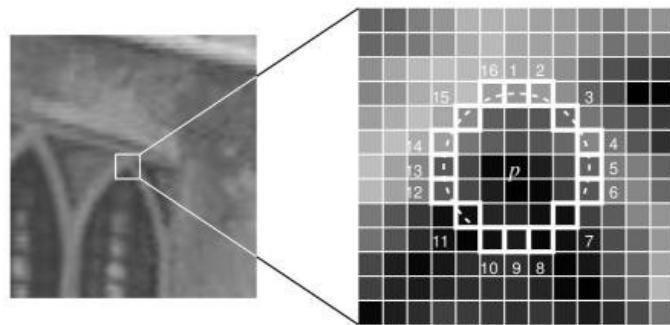
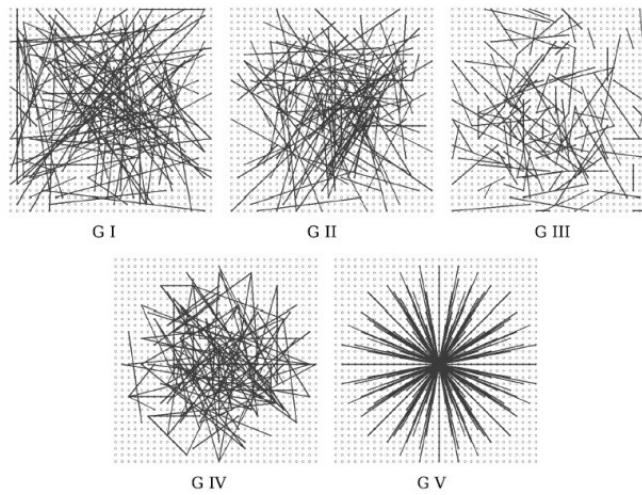


FIGURE 8 – Cercle de supériorité

### Binary robust independent elementary feature : BRIEF

Le vecteur descripteur est la concaténation de tests booléens de supériorité entre des valeurs choisies plus ou moins aléatoirement autour du **POI**.

Des paires de pixels sont générées aléatoirement. Sur ces paires vont être effectués les tests de supériorité.

FIGURE 9 – Paires de pixels autour du **POI** à tester

## 6 Méthodologie

Le stage s'est déroulé en deux phases. Une première phase de preuve de concept (Proof Of Concept : **POC**) puis une phase de projet.

La première phase vise à mettre en place et à vérifier, de manière disjointe, le bon fonctionnement de tous les modules du projet. Ainsi il y a eu trois **POCs** en parallèle. Le premier pour la partie de traitement automatique du langage naturel, **NLP**, le deuxième pour la partie de vision par ordinateur, **CV** et de détection de l'environnement et le troisième pour la partie de **SLAM**.

La seconde phase de projet a eu pour objectif de lier nos trois **POCs** en une application fonctionnelle, à laquelle on pourrait s'adresser avec un microphone pour se diriger vers des objets ou bien demander de se diriger vers un lieu favoris dans notre carte locale.

### 6.1 Intuition

Les méthodes de **SLAM** permettent de cartographier un environnement et de connaître la position de l'utilisateur dans celle-ci. Cette fonctionnalité est très intéressante mais il nous a semblé qu'avoir ces données brutes ne serait en définitive pas très utile. L'utilisation d'applications telles que Google Maps, Waze et autres a fait germer en nous l'envie de permettre à l'utilisateur malvoyant de découvrir un lieu, de le sauvegarder et que cette phase de découverte puisse être utile dans le cas où l'utilisateur

L'idée phare de notre phase de **POC** est tout d'abord de réussir l'installation du logiciel de **SLAM**, et enfin d'ajouter des fonctionnalités de sauvegarde de lieux favoris, de sauvegarde de la carte locale mais aussi de proposer la possibilité à l'utilisateur de retourner vers le lieu favori enregistré de son choix.

## 6.2 Environnements

### 6.2.1 Windows

Nous avons entrepris l'installation d'ORB-SLAM3 sur 3 distributions et machines différentes, sur Windows 10, **WSL2** et enfin sur une distribution Ubuntu 20.04 dans une machine virtuelle sur VirtualBox.

Nous avons utilisé le dépôt Git initial du projet pour démarrer l'installation. Celui-ci était prometteur. Bien qu'ayant à s'habituer au développement sur un environnement qui ne soit pas du tout Linux, nous avons travaillé pour faire avancer au maximum l'installation, mais des erreurs liées à ORB-SLAM3 développé pour Linux sont vite apparues. Nous avons trouvé une version d'ORB-SLAM3 qui serait spécifique à un environnement Windows, avec l'espoir de faire tourner cette méthode sur notre PC. Après une semaine, nous avons découvert qu'il y avait des librairies qui n'étaient pas disponibles sur Windows.

### 6.2.2 WSL2

Nous avons essayé l'installation sur le sous-système Windows pour Linux, (Windows subsystem for Linux : **WSL2**). Bien que soulevant des erreurs bien différentes de celles entrevues lors de l'installation sur Windows, telles que des conflits de version de certaines librairies nous avons réussi à avoir une version d'ORB-SLAM3 qui fonctionnait sur une vidéo prise par nos soins. Cependant l'objectif ultime de notre travail est de réussir à faire tourner notre méthode de **SLAM** en temps réel, avec un flux vidéo provenant de notre propre caméra. Le problème est que bien que **WSL2** ait accès aux périphériques USB de l'ordinateur, il n'a pas accès aux données contenues dans ces périphériques. Windows n'a pas prévu ce cas d'utilisation, ou en tous cas à prévu de ne pas le mettre en place. En effet le sous-système Linux pour Windows **WSL2**) est un système qui s'appuie sur le coeur de la machine. Si l'accès aux ports USB était disponible sans couche de sécurité, cela pourrait mener à de vraies failles de sécurité et il serait très facile de casser l'ordinateur. Ainsi une clé USB va pouvoir apparaître sur la liste des objets connectés, on ne pourra pas rentrer et avoir accès à ses documents. Bien que le fonctionnement d'une caméra USB soit un peu différent d'une clé USB, nous n'avions pas accès au flux en temps réel de la caméra et ne pouvions donc pas proposer une solution fonctionnelle pour répondre aux attentes du projet. Nous avons cependant eu confirmation, de nos propres yeux, que **SLAM** pouvait tourner. Cela a renouvelé notre motivation pour trouver une méthode pour avoir ORB-SLAM3 en temps réel.

### 6.2.3 Machine virtuelle

Nous avons finalement tourné notre recherche vers un environnement Ubuntu 20.04 sur une machine virtuelle VirtualBox de chez Oracle. Cette machine virtuelle nous a permis de travailler sur un environnement Linux complet, sans d'autres problèmes d'installation que le fait de trouver les bonnes versions des librairies, ou des problèmes plutôt liés à l'utilisation. C'est une technologie qui amène sa part de problèmes, que ce soient des problèmes de latence, les liens entre le clavier et la machine en bureau à distance, ou encore des systèmes corrompus et l'obligation de faire des instantanés régulièrement de notre système. C'est néanmoins la meilleure solution que nous ayons trouvé.

## 6.3 Installation librairies

### 6.3.1 Eigen

Eigen est une librairie de templates C++ pour faire de l'algèbre linéaire, du calcul matriciel, vectoriel, des solveurs numériques et autres algorithmes. elle est nécessaire pour avoir une méthode ORB-SLAM3 qui tourne bien.

Pour installer Eigen, nous avons du rechercher une bonne version de la librairie. Nous avons fait une recherche presque exhaustive avec nombre de versions qui ne correspondaient pas au projet et sommes finalement tombés sur la bonne version 3.3.1.

### 6.3.2 Pangolin

Pangolin est un ensemble de librairies portable et légère pour faire du prototypage 3D, numérique ou des programmes qui utilisent des vidéos. Elle est utilisée pour généraliser et ne pas être dépendant des plateformes de code et permettre de visualiser facilement les résultats.

Pour installer Pangolin nous sommes allés chercher sur le GitHub originel du projet, de même que pour Eigen, nous avons du trouver la version qui correspondait parfaitement à notre logiciel. Nous avons finalement trouvé la version 0.6 de github.

### 6.3.3 OpenCV

OpenCV est une librairie libre spécialisée dans le traitement d'image en temps réel. C'est une librairie qui a de nombreuses versions et qui, comme les deux librairies précédentes a nécessité un crible particulièrement fin pour trouver la version qui ne soulèverait pas de conflit avec le projet. La version qui a permis le bon lancement du projet est la 4.2.

### 6.3.4 Système d'exploitation pour robot (Robot Operating System : ROS)

ROS est un module utilisé principalement sur des robots pour assurer la communication entre le robot et le serveur qui le contrôle. C'est un module utilisé dans la méthode ORB-SLAM3 qui va permettre la bonne communication entre la caméra en temps réel et le module de traitement **SLAM**.

### 6.3.5 Kalibr

Ce module tiers est utilisé pour calibrer les caméras utilisées, il va estimer les paramètres intrinsèque de la caméra ainsi que la matrice de projection de la caméra pour avoir de meilleurs résultats avec ORB-SLAM3.

## 6.4 Prise en main

Après près d'un mois de tentatives d'installation ORB-SLAM3 est prêt à être utilisé.

ORB-SLAM3 est un module de plus de 33000 lignes de code sans compter les librairies utilisées. C'est donc un logiciel difficile à prendre en main notamment pour s'imbriquer dans le code existant et apporter des changements. La prise en main a été difficile au départ mais nous sommes arrivés au bout.

## 6.5 Architecture de la solution

### 6.5.1 Calibration



(a) Caméra de type pinhole

(b) Caméra de type fisheye

FIGURE 10 – Différents modèles de caméra

La première étape pour une bonne méthode de **SLAM** est d'être capable d'avoir une bonne calibration de la caméra utilisée.

La calibration d'une caméra consiste en l'estimation des paramètres intrinsèques et les paramètres de distortion d'une caméra. Ce sont les paramètres qui définissent comment un objet dans le monde réel va être projeté dans le plan caméra.

Les méthodes de **SLAM**, pour estimer le plus précisément possible les déplacements de l'utilisateur, utilisent les paramètres intrinsèques de la caméra. L'estimation du déplacement est un problème d'optimisation qui a pour objectif d'estimer la transformation géométrique entre deux nuages de points. Cette estimation de transformation se fait à partir des valeurs de distance dans le plan image. Si on a une bonne calibration, les calculs de distance vont être plus précis et on aura des résultats de meilleure qualité et une bonne estimation de la pose de l'utilisateur.

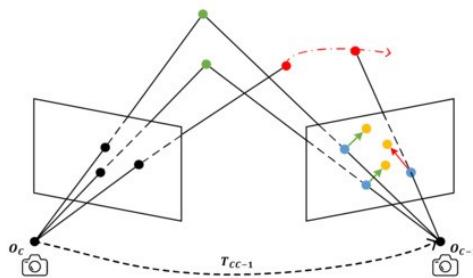


FIGURE 11 – Estimation de la transformation géométrique entre deux nuages de points

Ces paramètres intrinsèques de caméra varient entre les manufactures de caméra mais aussi entre les modèles dans les manufactures et entre les caméras du même type au sein d'une même marque. Il s'agit alors de les estimer précisément.

La calibration peut être effectuée à partir de deux application ROS et Kalibr. Cette première est une application facile à utiliser avec un retour graphique dynamique pour savoir si la calibration se déroule normalement et si elle est terminée. La dernière est une application plus difficile à utiliser dont les résultats apparaissent de manière très peu visuelle.

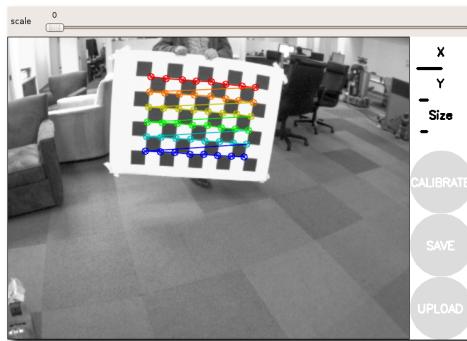


FIGURE 12 – Utilisation d'un modèle connu pour estimer la calibration d'une caméra

Le processus de calibration quelle que soit l'application utilisée se fait en utilisant un modèle géométrique connu (échiquier, apriltag, ...) qui contient des carreaux noirs et blancs de dimensions connues. L'idée est d'être capable d'estimer la déformation induite par la caméra sur un modèle connu, en utilisant différents points de vue de l'échiquier

(avec de la rotation, en variant les positions de l'échiquier dans le champ en changeant les distance entre le modèle et la caméra).

A partir de l'estimation de la transformation géométrique que subit un objet pour arriver dans le plan caméra, on peut déterminer la transformation inverse et appliquer celle-ci aux nouvelles images de la vidéo en temps réel et permettre la meilleure estimation de la pose de la caméra.

### 6.5.2 Recherche d'itinéraire : Pathfinding

Après avoir passé un certain temps pour comprendre les différentes parties dont est formé ORB-SLAM3, nous avons pu commencer à réfléchir à une manière de trouver son chemin entre notre position initiale et une position destination.

L'idée est d'être capable d'enregistrer un chemin emprunté par l'utilisateur quand on utilise le system, d'être capable de sauvegarder des lieux favoris et de trouver un trajet entre le point actuel et un des lieux favoris sauvegardés en utilisant l'algorithme A\* (prononcé A star).

Toutes ces fonctionnalités peuvent être mises en places en construisant à partir d'ORB-SLAM3.

#### 6.5.2.1 Coordonnées discrétisées

Les coordonnées données par ORB-SLAM3 sont de petits entiers qui simulent la continuité du déplacement de l'utilisateur. L'utilisation d'entiers est très intéressante pour avoir une idée précise du déplacement de l'utilisateur. Cependant pour donner des directives à une personne, il est plus aisné de travailler sur une version discrétisée des coordonnées. Nous avons dû trouver un moyen de les discrétiser pour être capable de trouver un chemin sur une grille de coordonnées et adapter la solution ORB-SLAM3.

L'idée de la projection est de supprimer l'excès de précision des coordonnées réelles.

Pour ce faire les coordonnées vont être multipliés par un facteur à déterminer. Les parties entières de ces nouvelles coordonnées sont stockées et divisées par l'inverse du facteur initial.

Pour les coordonnées  $X_0$  suivantes :

$$X = E(X_0 * \text{mult}) / \text{mult} \quad (1)$$

Par exemple :

$$\begin{pmatrix} 1.12546435987845 \\ 2.35496874325314 \end{pmatrix} \xrightarrow{\text{discret}} \begin{pmatrix} 1.1 \\ 2.3 \end{pmatrix}$$

Cette technique dont le coefficient multiplicateur peut évoluer pour permettre d'affiner ou d'élargir la précision a permis d'avoir une grille sur laquelle il est plus aisé de calculer des itinéraires mais aussi de donner des directions à suivre à l'utilisateur. L'idée étant d'ajuster le coefficient pour que les points sur la grille ne soient pas trop proches et qu'on perde l'intérêt d'avoir discréteriser les coordonnées et pas trop larges convenir à l'utilisation d'un humain qui se déplace à une certaine. A l'utilisation le coefficient a été choisi pour que deux points de la grille soient séparés en réalité d'une cinquantaine de centimètres. Une erreur dans la précision de ces coordonnées ne devrait pas avoir un grand impact quand l'ordre de grandeur est le centimètre.

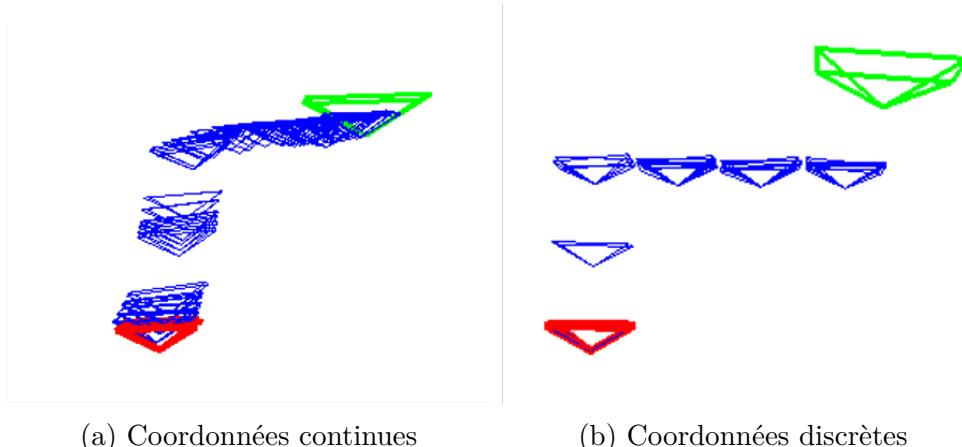


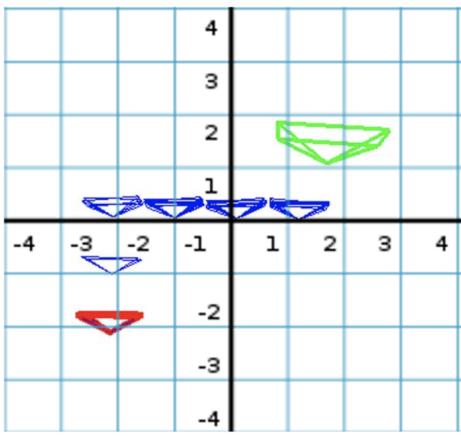
FIGURE 13 – Utilisation d'un modèle connu pour estimer la calibration d'une caméra

### 6.5.2.2 Structures de données

**Coordonnées** Une structure ***Coord*** en C++ a été créée. Elle définit les coordonnées entières utilisées par nos algorithmes pour la suite du traitement.

**Coordonnées absolues** Les coordonnées utilisées tout au long de notre programme peuvent prendre deux formes. Des coordonnées absolues et des coordonnées relatives. Les premières coordonnées correspondent aux coordonnées discréterisées renvoyées par ORB-SLAM3, elles peuvent être positives comme négatives (en considérant un utilisateur qui marcherait vers l'arrière par rapport à sa position initiale). Les dernières sont des coordonnées translatées assurant des coordonnées positives pour faciliter les calculs d'itinéraire à partir d'une matrice dont les coordonnées ne sont que positives.

La structure ***AbsoluteCoord*** a été définie. Elle contient 5 attributs différents :



(a) Coordonnées absolues discrétisées

$$\left\{ \begin{array}{l} (-2, -2) \\ (-2, -1) \\ (-2, 0) \\ (-1, 0) \\ (0, 0) \\ (1, 0) \end{array} \right\}$$

(b) Vecteur de coordonnées absolues

- *Int x\_min, x\_max, y\_min, y\_max* : Les dimensions de la carte.
- *Vector<Coord> coords* : Le trajet suivi par l'utilisateur sous la forme d'un vecteur de coordonnées 2D.

### Coordonnées relatives

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

#### 6.5.2.3 Path

La structure **Path** a été définie. Elle contient les attributs qui permettraient de définir un trajet :

*Coord start* : La position actuelle sur la carte quand le trajet est sauvegardé. Cette position va être donnée à l'algorithme de calcul d'itinéraire en tant que position de départ.

*vector<vector<int>> map2D* : La matrice représentant le trajet suivi par l'utilisateur sur laquelle va s'effectuer le calcul d'itinéraire. Elle a pour dimensions :

$$(x_{max} - x_{min} + 1, y_{max} - y_{min} + 1)$$

Cette carte 2D est créée à partir du vecteur de coordonnées absolues. L'idée est d'être capable de passer d'une représentation à l'autre, de passer d'une liste de position en coordonnées absolues à une matrice représentant la trajectoire dans des coordonnées relatives.

Les méthodes nécessaires à ce changement de représentation ont été mises en place. Le passage de la représentation en matrice vers une liste est fait de sorte que la liste de

coordonnées absolues ressorties soit en fait le trajet à suivre pour retourner au lieu favori.

#### 6.5.2.4 Instances dans la classe Map

Pour implanter du code dans le module ORB-SLAM3, il a fallu rechercher les endroits d'ouverture. Rechercher les emplacements les plus intéressants pour avoir de bonnes fonctionnalités. L'idée qui a été suivie est qu'un trajet en 2D devrait être sauvegardé en tant qu'attribut d'une instance de carte. En effet dans ORB-SLAM3, à tout moment le programme considère que l'utilisateur est dans une carte. Le trajet qu'il suit dans cette carte doit appartenir à cette carte.

**LocalPath** L'attribut *Path LocalPath* est une instance de la structure **Path** qui enregistre le trajet local suivi par l'utilisateur dans la carte actuelle et la position de départ pour l'algorithme de calcul d'itinéraire.

**Absolute2dCoordinates** L'attribut *AbsoluteCoord absolute2DCoordinates* est une instance de la structure **AbsoluteCoord** qui enregistre les coordonnées absolues visitées pas l'utilisateur.

**POIList** L'attribut **POIList** est une liste qui enregistre tous les lieux favoris sauvegardés dans une carte et atteignables tant que l'utilisateur se trouve dans la carte. Ces lieux favoris seront utilisés par l'algorithme de recherche d'itinéraire A\* en tant que destination.

Elle est sauvegardée en tant que *map<string, Coord>* qui associe à un nom de lieu favori une coordonnée.

Cette liste est étoffée au fur et à mesure à la demande de l'utilisateur. Il fait sa requête sur le viewer, saisit le nom voulu pour son lieu favori.

#### 6.5.2.5 A\*

L'algorithme A\* est un algorithme de recherche qui détermine le plus court chemin entre deux positions dans une matrice en passant par des positions disponibles. Il est utilisé dans de nombreuses applications notamment la recherche d'itinéraire dans des cartes.

En considérant une grille carrée contenant plusieurs obstacles disposés de manière aléatoire. En fournissant une position initiale A et une position finale B. L'objectif est d'arriver à la position finale en le temps le plus court.

Cet algorithme prend en compte 3 paramètres :

- $C_{I,c}$  : le coût pour passer de la cellule initiale à la cellule courante. C'est la somme des coûts de toutes les cellules par lesquelles l'algorithme est passé pour arriver à

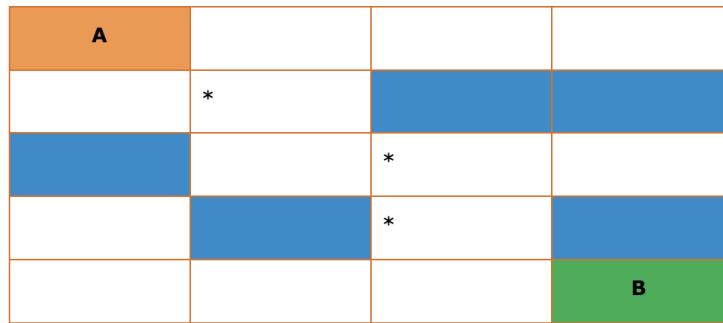


FIGURE 15 – Estimation de la transformation géométrique entre deux nuages de points

la cellule courante.

- $C_{c,F}$  : le coût estimé pour passer de la cellule courante à la cellule finale. Le coût réel ne peut être calculé qu'une fois que la cellule finale a été atteinte. Ainsi  $H$  est le coût estimé. il faut faire attention à ne jamais sur-estimer ce coût.
- $C$  : la somme des deux coûts précédents.

$$C = C_{I,c} + C_{c,F}$$

L'algorithme prend ses décisions en fonction de la valeur de  $C$ . L'algorithme choisit la cellule avec la valeur  $C$  minimale et se déplace vers cette cellule. Le processus continue jusqu'à ce que l'algorithme atteigne la cellule cible.

#### 6.5.2.6 Retour à un lieu favori

Pour retourner à un lieu favori, il va être demandé à l'utilisateur de saisir le nom du lieu favori vers lequel il veut se diriger. Si ce nom correspond véritablement à un lieu favori existant dans la carte courante, les coordonnées correspondantes vont être données à l'algorithme A\* pour trouver un trajet utilisable de la position courante vers le lieu favori choisi. Ce trajet va être donné à la fonction FYWBack pour assurer la transmission des consignes à l'utilisateur.

#### 6.5.2.7 FYWBack

Cette fonction FYWBACK utilise la sortie donnée par l'algorithme A\* pour guider l'utilisateur vers un lieu favori choisi. La sortie d'A\* est donnée sous la forme d'une liste de coordonnées triée dans l'ordre du trajet à effectuer depuis la position courante à la position destination.

À partir de ces coordonnées on peut découper le trajet à suivre en sections. Ces sections permettent d'estimer la direction à suivre par l'utilisateur. Or **SLAM** estime, entre autres choses, la direction suivie par l'utilisateur, cette direction correspond en langage mathématique au vecteur normal au plan de la caméra. En calculant l'angle entre la direction

objectif et la direction véritable de l'utilisateur l'algorithme est capable de maintenir un cap et de le redresser si nécessaire.

Si l'utilisateur se trouve "proche" de la fin d'une section la solution va passer automatiquement à la section suivante et permettre d'assurer la continuité du guidage.

### 6.5.3 Interaction avec l'utilisateur

ORB-SLAM3 propose en option un retour visuel de ce qui se passe avec le module. Cette interface homme-machine (IHM), affiche le flux caméra sur lequel sont ajoutés les points d'intérêt déterminés de type ORB, la carte 3D construite dynamiquement au fur et à mesure de la vidéo ainsi qu'un certain nombre de boutons permettant des interruptions du programme et de lancer différents modes de traitement de la vidéo.

La majorité des fonctionnalités mises en place dans notre solution va utiliser le principe d'interruptions, que celles-ci soient faites à partir d'un bouton ou à partir de la voix (avec le module de **NLP**).

Nous sommes enfin arrivés à la partie centrale de notre installation, ORB-SLAM3. Le gâteau qui va bientôt recevoir sa cerise... Notre apport au projet.

## 6.6 Phase de projet

L'aboutissement de ce stage se trouve dans la journée des stagiaires (**JDS**). C'est une journée de présentation des résultats de tous les stages de la cellule INNOV, mais aussi l'aboutissement des projets de d'alternance. C'est une journée qui correspond pour nos projets à des présentations clients pour des consultants.

Durant celle-ci chaque équipe présente, en dix minutes, sa solution devant la direction générale d'Aubay ainsi que certains directeurs de business units. Cette présentation est suivie par une période de démonstration des résultats du projet en temps réel.

Cette partie décrit les réalisations durant la phase projet. L'objectif est de développer une application qui permet de guider un malvoyant en temps réel, en intérieur. Elle est présentée à la **JDS** devant la direction générale et les autres stagiaires.

### 6.6.1 Nouvelle caméra

Avant de commencer à développer l'application, nous avons pu commander une nouvelle caméra. Le choix a été complexe, car nous voulons une caméra qui fonctionne en plug-and-play avec un assez bon champ de profondeur. Une caméra plug-and-play est une caméra qui fonctionne comme une webcam, il y a juste besoin de la brancher en USB pour qu'elle fonctionne. Nous aurions pu prendre des caméras de bonnes qualités comme le font les

streamers mais elles ne fonctionnent pas en plug-and-play. Il aurait fallu acheter une carte d'acquisition en plus et nous n'avons pas le budget pour. Nous avons donc opté pour la caméra Svpro USB Web Camera 1280x720, trouvée sur Amazon. Nous avons pris une caméra de basse résolution, pour éviter un nombre de calculs trop conséquent et donc une perte de temps, la résolution du flux vidéo en entrée est de 640x480. Sur la figure suivante, nous pouvons voir une comparaison des deux caméras avec à gauche l'ancienne (avec une résolution de 1080x720) et à droite la nouvelle (avec une résolution de 640x480).

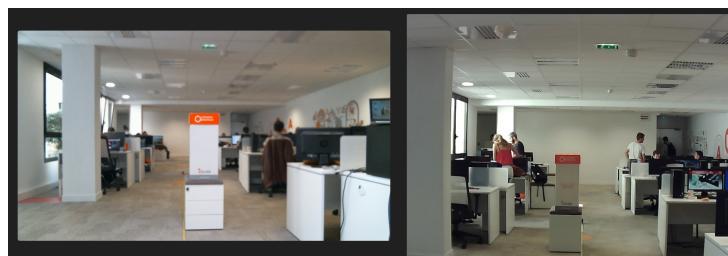


FIGURE 16 – Comparaison de la qualité des deux caméras

Comme nous avons une nouvelle caméra, il a fallu estimer ses paramètres intrinsèques pour obtenir une calibration fonctionnelle et pouvoir utiliser le module de **SLAM** sans problème.

Ceci a été fait avec le module Kalibr sans trop de problème et a permis de se lancer pleinement dans la réalisation de l'application.

### 6.6.2 Environnement de l'application

Cette application est réalisée en rassemblant les différents **POCs** précédemment décrits : **CV**, **NLP** et **SLAM**. Il a fallu trouver un environnement adéquat pour pouvoir développer l'application. En effet, **SLAM** est codé en C++ et fonctionne exclusivement sur un système Linux, tandis que les deux autres **POCs** codés en Python fonctionnent sous Windows.

Travaillant sur des ordinateurs sous Windows, la partie **SLAM** a été développée sur une machine virtuelle Linux ainsi que sur **WSL2**, cependant pour une application fonctionnelle en temps réel, **WSL2** ne permettait pas d'avoir accès au flux vidéo de la caméra. Nous avons donc tout d'abord essayé de travailler dans une machine virtuelle. Cependant, nous n'avons pas pu continuer longtemps, car les parties **CV** et **NLP** ont besoin d'un **GPU** pour fonctionner en temps réel et avec une machine virtuelle, nous n'y avions pas accès.

Nous nous sommes donc tournés vers Windows Subsystem for Linux **WSL2**, avec en tête de trouver une solution pour obtenir un flux d'image de la caméra et pouvoir transmettre ce flux au module **SLAM**. Le fait de travailler sur un même matériel a fait surgir l'idée de serveurs locaux sur Windows et sur **WSL2**. Le premier serait serveur et distribuerait le flux à son client sur **WSL2**. Cette technique semble un petit peu barbare mais est la seule

solution qui a été trouvée pour pallier le problème de caméra sur **WSL2**. Cette solution a nécessité une grande adaptation du code d'ORB-SLAM3 pour permettre de prendre en entrée un flux provenant d'un serveur, il a fallu ouvrir plusieurs ports locaux sur **WSL2** à partir du pare-feu Windows pour permettre de transférer les informations souhaitées.

Le serveur utilisé est un serveur Flask, c'est un serveur facile à mettre en place ce qui a permis d'avoir des résultats rapidement.

### 6.6.3 Structure de l'application

Après avoir déterminé l'environnement, nous pouvons développer l'application. Elle se divise en deux parties :

- le back-end, c'est le fond du programme, ce qui constitue l'application en elle-même. Elle prend en entrée le flux vidéo et le flux audio.
- le front-end, c'est l'interface avec l'utilisateur. À des fins de présentation il a fallu être capable de présenter les résultats de manière visuelle, bien que le système soit destiné à des personnes malvoyantes. Sont affichés les flux de caméra, les boutons permettant de lancer les traitements du back-end, ainsi que les résultats des **POCs CV** et **SLAM**.

Sur la figure est représentée un schéma de la structure de l'application. Un diagramme des composants, plus précis, est en annexe.

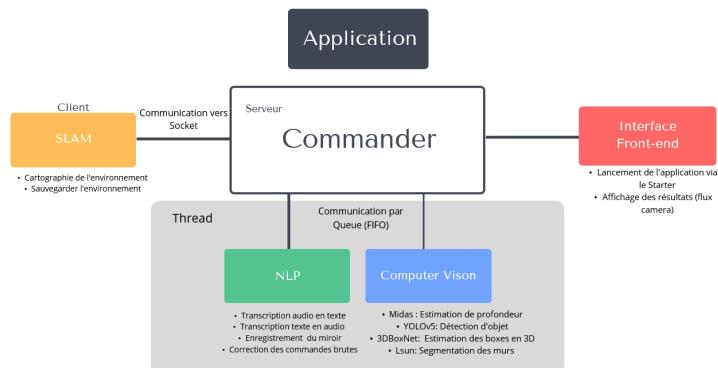


FIGURE 17 – Schéma de la structure de l'application

C'est un bouton sur l'interface qui va lancer les fonctionnalités via le Starter. Le Starter s'occupe de récupérer le flux vidéo, de lancer le serveur Flask afin de lancer **SLAM** et de lancer le Commander. **SLAM** ne fonctionnant que sous Linux, il a fallu écrire des scripts shell pour le lancer. Le Commander a pour rôle de recevoir les messages des différents modules et de les envoyer au bon endroit. Les modules **CV** et **NLP** communiquent au Commander à l'aide de threads alors que le module **SLAM**, n'étant pas dans le même environnement, communique à l'aide de socket en tant que client. Les données à envoyer à l'interface sont stockées dans un objet python.

L'interface de l'application a été faite sous PyQt, un outil permettant de lier le Python à la bibliothèque Qt. Des images de l'application sont consultables en annexes. L'interface est composée de 4 pages, une page d'accueil, une page d'information, une page récapitulant les commandes vocales et la page principale.

#### 6.6.4 Protocole de communication

Le fait de travailler sur un environnement séparé du module principal de communication, le Commander, le module **SLAM** doit trouver un moyen de communiquer avec celui-ci pour recevoir des commandes mais aussi renvoyer des informations, telles que des instructions de direction à suivre mais aussi des images de la carte locale créée, de l'image avec les points d'intérêts extraits.

Le moyen de faire cette communication a été la socket. Une socket est une interface logicielle entre deux processus. Cette interface permet le transfert de donnée plus ou moins lourdes et une communication facile entre les éléments. Un des aspects les plus intéressants est le fait que les sockets sont accessibles depuis la plupart des langages de programmation. Ainsi C++ et Python ont tous deux accès aux sockets. **SLAM** pourrait donc être connecté au commander. Pour ce faire, de nombreux tests ont été effectués des tests d'autorisations au niveau du proxy, du pare-feu, de l'installation de **WSL2**, du noyau Linux utilisé,...

Il se trouve que le plus grand problème venait du pare-feu qui ne laissait pas remonter des données depuis /glsWSL2 vers Windows. En ouvrant les ports nécessaires le lien a pu être fait.

Il a alors fallu mettre en place la possibilité de faire des interruptions depuis le commander, c'est-à-dire de l'application, sur le module **SLAM**.

Ces interruptions ont été gérées en utilisant des threads, c'est-à-dire des processus d'exécution qui tournent en parallèle et peuvent être à l'écoute d'interruptions et de commandes provenant d'autres parties du code.

Ces threads sont aussi présents dans la plupart des langages de programmation, notamment C++. L'utilisation de ceux-ci a permis tout d'abord d'interrompre ORB-SLAM3 en utilisant les touches du clavier, puis ensuite en utilisant deux serveurs sur **WSL2** et enfin en utilisant un serveur sur Windows et un serveur sur **WSL2**.

Ces threads peuvent être d'écoute ou d'envoi au sein de l'application **FYW** :

- Les threads d'écoute vont effectuer une action lorsqu'on va les appeler. En l'occurrence lorsqu'une variable va changer de valeur.
- Les threads d'envoi vont être appelés dès qu'une donnée est prête à être envoyée par le programme sur la socket vers le Commander.

Ce protocole de communication a été long à mettre en place, il a nécessité beaucoup d'apprentissage de la part de l'équipe pour appréhender des technologies peu connues pour des étudiants en fin de cursus scolaire. Cela a été cependant très positif et a éveillé une grande curiosité.

### 6.6.5 Fonctionnement de l'outil

Notre outil est composé de l'application fonctionnant sur un ordinateur avec une carte graphique, d'une caméra, d'un micro et d'un haut-parleur. Une personne doit être devant l'ordinateur pour activer le microphone et prononcer les commandes vocales. L'utilisateur, lui, tient la caméra qui est branchée à l'ordinateur.

Une fois l'application lancée et étant sur la page principale, nous pouvons démarrer la caméra. Un bouton est présent pour activer le microphone et lancer l'enregistrement de la commande vocale.

En fonctionnement, 3 fenêtres sont affichées. La fenêtre principale affiche la vue frontale de la caméra, avec les objets détectés et leur distance à l'utilisateur. Lorsqu'un itinéraire est demandé, la carte en deux dimensions s'affiche avec le chemin jusqu'à la cible, en haut à droite. En bas à droite sont affichés les points d'intérêt de l'environnement 3D construit par l'algorithme **SLAM**. L'application en fonctionnement est visible sur la figure suivante.

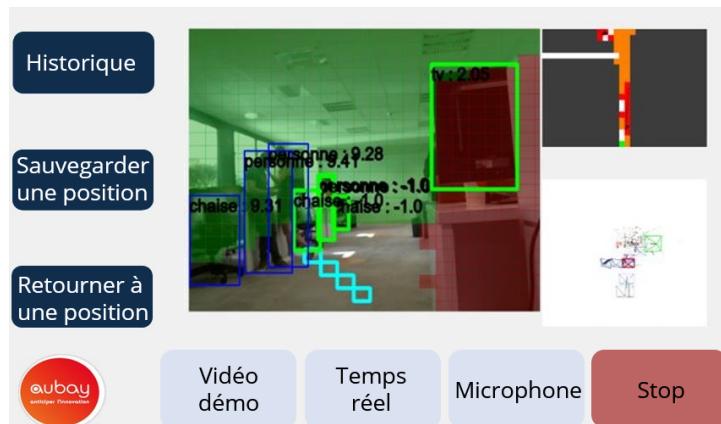


FIGURE 18 – Application en fonctionnement

Notre application dispose de plusieurs fonctionnalités pouvant être déclenchées par commandes vocales. L'utilisateur peut demander :

- à ce qu'on lui décrive l'environnement. L'application lui listera les 5 objets détectés les plus proches.
- à être guidé vers un objet présent dans l'image. S'il y a plusieurs itérations de l'objet demandé, il sera guidé vers le dernier détecté.
- à sauvegarder sa position actuelle, il devra la nommer.
- à lister les positions sauvegardées.

- à retourner à une position précédemment sauvegardée.

En cas de nécessité, il est également possible de faire une demande de sauvegarde de position ou de choisir un retour vers une position directement via l'interface graphique sans passer par le microphone.

Par défaut, si un objet est trop proche, moins de 2 mètres, l'utilisateur en sera averti.

## 6.7 Plateforme de test

Tout travail informatique doit être testé pour être certain que la solution ne se comportera pas de manière aberrante. Ces tests peuvent être plus ou moins quantitatifs. Certains aspects développés requièrent des tests de précision, d'autres des tests fonctionnels pour être sûr que dans tous les cas d'utilisation la solution réponde normalement.

La plupart des fonctionnalités mises en place sont difficiles à évaluer avec une plateforme de tests unitaires. Les tests effectués ont été plutôt qualitatifs. En travaillant sur des vidéos prises dans l'open-space, les fonctionnalités ont pu être testées pour vérifier le comportement normal de tout ce qui a été mis en place.

Ainsi pour tous les tests d'utilisation de notre solution

Il se trouve que les fonctionnalités mises en place dans le projet Find Your Way sont pour beaucoup des fonctionnalités comportementales.

## 6.8 Améliorations

La solution **FYW** pourrait être améliorée sur certains aspects. Ces améliorations sont en majeure partie dues à des restrictions matérielles, des limitations de l'état de l'art, mais aussi des choix d'architecture de la part de l'équipe.

### 6.8.1 Précision ORB-SLAM3

ORB-SLAM3 est considéré comme l'état de l'art, c'est un papier reconnu dans le domaine qui fonctionne très bien. Cependant quand on s'intéresse d'un peu plus près aux vidéos de présentation d'ORB-SLAM3 en fonctionnement on se rend compte que la vidéo a été accélérée.

Cette découverte explique en grande partie la difficulté éprouvée par cette méthode lorsque la caméra effectue des rotations à ne pas se perdre. En effet une grande rotation implique que beaucoup de points d'intérêt détectés ne vont pas avoir d'appariement, la majeure partie des **POIs** étant nouveaux.

Ce point est gênant mais peut être contourné en travaillant dans les mêmes conditions que dans les vidéos de l'article et en faisant attention de ne pas bouger trop vite en rotation.

En revanche il n'y a aucun problème pour les mouvements de translation. Il faudrait donc trouver un moyen de renforcer les méthodes de **SLAM** pour les mouvements de rotation. Une manière de faire cela pourrait-être d'augmenter le nombre de points d'intérêt extraits dans chaque image. Si la puissance de calcul est assez bonne cela pourrait permettre de garder la possibilité de travailler en temps réel et d'assurer le lien entre toutes les images de la vidéo ce qui permettrait de ne pas se perdre.

### 6.8.2 Threads

Le choix a été posé de travailler avec des threads pour assurer les interruptions du programme. Un thread (ou fil d'exécution en français) est un processus qui permettent d'exécuter des commandes sur la machine. Ils ont l'avantage de pouvoir effectuer des commandes asynchrones et faire tourner des parties du programme en parallèle.

Il se trouve que dans la solution finale proposée, il y a certains threads qui sont en attente. C'est-à-dire qu'ils attendent une intervention d'une autre partie du programme qui peut potentiellement arriver dans longtemps et prennent ainsi une partie de la mémoire de la machine en réduisant les performances du programme.

Une solution envisagée est d'utiliser des processus, plus légers

### 6.8.3 Puissance de calcul embarquée

Ce projet avait pour volonté, au départ, de créer un système embarqué, sous forme de lunettes connectés ou d'un harnais que porterait l'utilisateur. En effet, des solutions comme cela sont très pratiques, facilement utilisables. Cependant elles viennent avec leur lot de problèmes. Principalement la faible puissance de calcul disponible. En effet, le projet est à dominante d'intelligence artificielle. Ces modèles parfois lourds ont des besoins matériels importants, notamment des **GPU** qui permettent de stocker une grande puissance de calcul et permettent de faire tourner notre application en temps réel.

La solution serait donc, soit d'avoir des **GPUs** embarqués dans notre paire de lunette ou notre harnais, soit de travailler sur une version cloud de notre application qui recevrait notre flux vidéo en direct, serait capable d'effectuer les calculs nécessaires en temps réel et de renvoyer les résultats au système. Ainsi il faudrait aussi que notre système soit équipé d'une connexion wi-fi pour être toujours connecté au cloud.

Pour le moment, l'application tourne sur un ordinateur fixe, avec une connexion USB pour la caméra et le micro.

## 6.9 Analyse des résultats

La solution finale est une solution fonctionnelle qui permet de faire tout ce que nous souhaitions mettre en place. En effet, un utilisateur peut se saisir de la caméra et du micro, il peut lancer l'application, saisir des commandes vocales, se déplacer en intérieur, être prévenu des objets qui sont proches de lui, sauvegarder des lieux favoris, y retourner, demander la description de l'environnement.

En ce qui concerne la partie **SLAM**, les résultats sont concluants mais pourraient être améliorés si ces méthodes étaient performantes pour un mouvement de rotation.

Les tests effectués pour des mouvements de translation dans l'open-space montrent que l'utilisateur peut se balader en intérieur, sauvegarder les lieux favoris qu'il souhaite (Bureau de Lucie, Salle de réunion,...) il peut se déplacer dans le milieu et demander à retourner au lieu de son choix et l'algorithme fera en sorte qu'il arrive à destination.

## 7 Bilan personnel

Ce stage s'est effectué dans une équipe de onze stagiaires avec deux responsables présents aux réunions hebdomadaires. De part les effectifs et le nombre de projets de chaque responsable, Find Your Way a été développé en majeure partie par les stagiaires, c'est nous qui avons fait des choix dans l'implémentation de notre solution, dans les fonctionnalités que nous voulions ajouter et dans les méthodes de travail. **FYW** est le fruit d'une vraie émulation entre tous les stagiaires.

Cependant cette liberté a un pendant, chaque responsable ne peut pas être chargé de la bonne montée en compétence de chacun, et j'ai parfois ressenti un manque de structure autour de notre projet.

De plus nous avons vu passer 3 responsables différents en tant que responsable technique (L'un a quitté l'entreprise, un autre a été appelé par un projet en tant que consultant). Cet instabilité a renforcé le fait que nous devions travailler de notre côté pour faire avancer le projet comme bon nous semblait.

La volonté d'Aubay pour ses stages est de proposer des sujets innovants, sur des thématiques intéressantes pour des étudiants en fin de parcours mais aussi de les faire se creuser la tête sur certaines problématiques sans que la solution ne soit jamais déployée. C'est une manière de ne pas nous considérer comme de faux employés payés moins que des salariés pour faire le même travail et cela met en place un vrai tremplin entre l'école et le monde du travail sans autre pression que celle de la **JDS** (une journée de présentation de l'aboutissement de tous les projets face à la direction générale d'Aubay ainsi que les directeurs de business units (**BU**) et les commerciaux).

Une réflexion qui m'est venue assez rapidement au cours de ce stage est que le travail effectué dans ce stage ne m'a pas réellement permis d'imaginer le métier d'un consultant en informatique. Le lien a mis du temps à se mettre en place. Il a pris la forme de présentation de parcours d'anciens stagiaires ainsi que les présentations des BU. Mais notre travail au jour le jour ne reflétait pas du tout ce qu'allait pouvoir vivre un consultant dans une journée type.

## 8 Impact environnemental

Les stages dans la cellule INNOV chez Aubay sont des pistes de travail qui permettent de tester les nouvelles technologies, de les adapter à des problématique précises et d'en évaluer les résultats.

Or depuis une dizaine d'années, les méthodes d'apprentissage profond sont de plus en plus en vogue, elles sont porteuses de beaucoup d'innovation et sont en général étudiées pour des projets tels Find You Way. Cependant, souvent les méthodes d'apprentissage profonds sont utilisées au détriment de certaines méthodes plus classiques, se basant simplement sur une analyse mathématique d'un problème donné.

La découverte des méthodes de localisation et de cartographie en simultané a permis de s'éloigner de ce travers et de s'attaquer à une méthode bien moins gourmande en ressources pour un ordinateur, qui consomme bien moins d'énergie en tournant sur un simple processeur (CPU : Central Processing Unit) et non pas sur une carte graphique (**GPU**). Ce qui est intéressant c'est de voir qu'avec des méthodes "plus simples" (bien que nécessitant une analyse et une modélisation mathématique complexe) il peut arriver à des résultats plus que satisfaisants.

L'utilisation d'une telle méthode permet aussi d'envisager le produit fini comme un objet embarqué, une paire de lunettes ou encore un harnais, sur lesquels les ressources computationnelles sont plus limitées que sur un poste fixe.

Ce projet utilise aussi des réseaux de neurones qui sont pré-entraînés par des grands groupes telle que Google ou encore Facebook, ceux-ci sont très performants et ne nécessitent qu'un petit ré-apprentissage pour s'ajuster aux besoins de différents projets. Cela permet aussi d'éviter des périodes d'apprentissage sur-proportionnées de nos algorithmes pour arriver à des résultats satisfaisants.

La société Aubay a une politique RSE de plus en plus active, ses salariés sont poussés au télé-travail deux à trois jours par semaine ce qui permet de réduire l'impact environnemental lié au déplacement.

Aubay mène aussi une grande politique de tri des déchets via des partenariats avec des entreprises adaptés, des recyclages des téléphones et des collectes solidaires de lunettes vêtements et stylos. Aubay veut supprimer les gobelets à usage unique et participer au recyclage des mégots de cigarette pour qu'ils ne soient pas jetés par terre.

Aubay vise la compensation de son impact carbone et agit en permanence pour optimiser la consommation d'électricité de ses sites.

Ce projet a pour objectif premier d'aider les personnes malvoyantes, son impact sociétal doit être positif pour continuer de travailler pour une meilleure vie pour les personnes

en situation de handicap. C'est un projet qui n'aura pas d'impact sur la vie privée des personnes, les données n'étant pas traitées sur des serveurs à distance mais bien dans la solution développée, et devrait améliorer la vie au travail pour les personnes atteintes de cécité ou de problèmes de vue.

## 9 Conclusion

## Références

- [1] Jinqiang BAI et al. « Wearable Travel Aid for Environment Perception and Navigation of Visually Impaired People ». In : *Electronics* 8.6 (2019). ISSN : 2079-9292. DOI : [10.3390/electronics8060697](https://doi.org/10.3390/electronics8060697). URL : <https://www.mdpi.com/2079-9292/8/6/697>.
- [2] Carlos CAMPOS et al. « ORB-SLAM3 : An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM ». In : *CoRR* abs/2007.11898 (2020). arXiv : [2007.11898](https://arxiv.org/abs/2007.11898). URL : <https://arxiv.org/abs/2007.11898>.
- [3] Tete JI, Chen WANG et Lihua XIE. « Towards Real-time Semantic RGB-D SLAM in Dynamic Environments ». In : (mai 2021). DOI : [10.1109/icra48506.2021.9561743](https://doi.org/10.1109/icra48506.2021.9561743). URL : <https://doi.org/10.1109%2Ficra48506.2021.9561743>.
- [4] Raul MUR-ARTAL et Juan D. TARDOS. « 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. DOI : [10.1109/tro.2017.2705103](https://doi.org/10.1109/tro.2017.2705103). URL : <https://doi.org/10.1109%2Ftro.2017.2705103>.
- [5] Li YAN et al. « DGS-SLAM A Fast and Robust RGBD SLAM in Dynamic Environments Combined by Geometric and Semantic Information ». In : (fév. 2022). DOI : [10.3390/rs14030795](https://doi.org/10.3390/rs14030795).

## Glossaire

**BU** Business Unit 28, 29

**CV** Computer Vision 5, 13, 24, 25

**FYW** Find Your Way 27, 28

**GPU** Graphics Processing Unit 24, 30

**JDS** Journée des stagiaires 23, 28

**NLP** Natural Language Processing 4, 13, 23–25

**POC** Proof of Concept 7, 13

**POCs** Proofs of Concept 13, 24

**POI** Point of interest 9–12

**POIs** Points of interest 10–12, 27

**SLAM** Simulatneous Localization and Mapping 4–7, 13, 14, 16, 17, 22, 24–27

**WSL2** Windows Subsystem for Linux 14, 24