



HAL
open science

Extraction d'un graphe de navigabilité à partir d'un nuage de points 3D enrichis.

Imeen Ben Salah

► **To cite this version:**

Imeen Ben Salah. Extraction d'un graphe de navigabilité à partir d'un nuage de points 3D enrichis.. Vision par ordinateur et reconnaissance de formes [cs.CV]. Normandie Université, 2019. Français. NNT : 2019NORMR070 . tel-02419559

HAL Id: tel-02419559

<https://theses.hal.science/tel-02419559v1>

Submitted on 19 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité : INFORMATIQUE

Préparée au sein de l'Université de Rouen Normandie

Extraction d'un graphe de navigabilité à partir d'un nuage de points 3D enrichis

Présentée et soutenue par

Imeen BEN SALAH

Thèse soutenue publiquement le 6 Décembre 2019 devant le jury composé de

Mme Véronique CHERFAOUI	Professeure des Universités, Université de Technologie de Compiègne	Rapporteuse
M. Thierry CHATEAU	Professeur des Universités, Institut Pascal	Rapporteur
Mme Michèle ROMBAUT	Professeure des Universités, Institut polytechnique de Grenoble	Examinatrice
M. Pascal VASSEUR	Professeur des Universités, Université de Rouen	Directeur de thèse
M. Cédric DEMONCEAUX	Professeur des Universités, Université de Bourgogne-Franche-Comté	Co-directeur de thèse
M. Sébastien KRAMM	Maître de conférences, Université de Rouen	Encadrant de thèse

Thèse dirigée par Pascal VASSEUR, laboratoire Litis

“La connaissance est une lumière qui nous guide et nous apporte la prééminence dans notre **sphère** d'évolution.”

Esther Jonhson

Écrivaine et poétesse

*Pour Mohamed Amine,
Pour ma famille,
Pour mon père, pour ma mère,
Pour Miled, pour Wided et pour Ikhlass*

REMERCIEMENTS

Je remercie avant tout DIEU pour son amour et sa miséricorde d'avoir guidé mes pas durant ses trois années d'études, de m'avoir donné le courage de me surpasser nonobstant toutes les difficultés rencontrées durant l'élaboration de ce travail et de m'avoir permis de le mener à terme. La réalisation de cette thèse a sollicité l'apport et l'aide d'un grand nombre de personnes à qui je voudrais témoigner toute ma gratitude.

Je tiens tout d'abord à remercier monsieur **Paquet THIERRY**, professeur à l'université de Rouen, d'avoir bien voulu diriger notre laboratoire LITIS et de m'avoir intégrée à la grande famille de cette université.

Je remercie en particulier les membres du jury d'avoir bien voulu examiner ce présent travail de recherche et de l'enrichir grâce à leurs recommandations. Je souhaite remercier tout particulièrement madame **Michèle ROMBAUT**, professeure à l'Institut Polytechnique de Grenoble d'être présidente du jury. Je voudrais remercier madame **Véronique CHERFAOUI** et monsieur **Thierry CHATEAU** d'avoir accepté de relire cette thèse et d'en être rapporteur.

J'adresse mes plus sincères remerciements à mon directeur de thèse, monsieur **Pascal Vasseur**, le responsable de l'équipe de Systèmes de Transports Intelligents (STI), d'avoir bien voulu m'orienter tout le long de l'élaboration et la rédaction de ce présent travail de recherche. Pascal, je voudrais vous remercier infiniment pour votre disponibilité, vos conseils avisés et précieux et pour toutes les connaissances scientifiques que vous m'avez transmises. Je suis très honorée de vous avoir eu pour directeur de thèse.

Que monsieur **Cédric DEMONCEAUX**, mon Co-directeur de thèse et professeur à l'Université de Bourgogne-Franche-Comté, trouve ici l'expression de mes vifs remerciements pour sa supervision éclairée, son temps et son aide inconditionnelle durant ces trois années. Cédric, je vous remercie d'avoir me diriger vers le bon chemin tout le long de cette thèse. Vos conseils et vos remarques m'ont énormément aidée à nourrir ma réflexion pour avancer dans mes recherches.

J'associe à ces remerciements monsieur **Sébastien KRAMM**, maître de conférences à l'université de Rouen, pour m'avoir offert un encadrement de qualité depuis mon stage de master jusqu'à aujourd'hui. Sébastien, merci pour votre aide, votre enthousiasme, votre sympathie et votre confiance. Votre patience et votre savoir-faire ont été prépondérants pour la réalisation de cette thèse. Vous m'avez beaucoup appris tout au long de ces années. J'ai appris de vous les règles de codage ainsi que les bonnes pratiques pédagogiques. Je vous remercie pour tout le temps que vous avez passé pour lire mes codes au point d'arracher les cheveux pour essayer de comprendre le sens souvent tortueux de mes lignes de code.

Je voudrais vous dire merci pour tous les moments que nous avons partagés ensemble lors des réunions et des conférences. Enfin, travailler avec vous a été un grand plaisir pour moi. J'adresse de sincères remerciements à tous les membres de l'équipe de recherche STI pour m'avoir accueillie chaleureusement au sein du laboratoire. Je remercie tous les thésards et les membres de l'équipe, en particulier, **Kamal, René Emmanuel, Danots, Lounis et Ahmad**. Je vous remercie pour la bonne ambiance de travail, les bons moments sympathiques que nous avons partagés et pour toutes nos discussions intéressantes lors des pauses-café.

Je suis également reconnaissante envers tous les partenaires dans le projet pLATINUM, en particulier, le thésard de l'IGN **Mohamed BOUSSAHA** d'avoir répondu à toutes mes questions. J'ai pris également un grand plaisir à travailler avec plusieurs partenaires et je leur remercie pour tout cela. Mes remerciements s'étendent également à madame **Brigitte DIARRA** et madame **Fabienne BOCQUET** pour leur dévouement à mener à bien les tâches administratives. Je remercie également monsieur **Jean-Francois Brulard** et madame **Elsa Planterose** qui ont contribué à créer une atmosphère chaleureuse et conviviale.

On n'oublie pas mes parents pour leur amour inconditionnel, leurs prières constantes et leur soutien moral indéfectible, en particulier pour m'avoir encouragée de compléter mes études jusqu'à ma thèse. Je remercie mes sœurs Wided et Ikhlass, et mon frère Miled, pour leurs encouragements. Je souhaite exprimer ma reconnaissance envers mon cher époux Mohamed-Amine qui a su me soutenir, me supporter, m'encourager durant toutes ces années. Amine, merci pour ton soutien moral et tes conseils inestimables. Cette thèse et moi te devons beaucoup.

Enfin, je voudrais remercier tous ceux qui ont contribué de près ou de loin à la réussite de ce travail de recherche et qui n'ont pas pu être mentionnés ici.

Merci à tous ...

Table des matières

Liste des figures	14
Liste des tableaux	14
1. Contexte de la thèse : le projet pLaTINUM	15
2. Objectifs	17
3. Contributions	17
4. Organisation du manuscrit	18
5. Liste des publications	18
I. État de l'art : solutions existantes de cartographie et de résumé des cartes	20
1. Introduction	22
2. Généralités sur la vision par ordinateur	22
A. Systèmes de vision	22
B. Imagerie panoramique	24
C. Extraction, texturation et sémantisation d'un maillage 3D	29
D. Construction d'une sphère virtuelle augmentée	33
E. Conclusion	40
3. État de l'art : solutions de cartographie existantes	41
A. Introduction	41
B. Méthodes de cartographie existantes	41
C. Organisation et combinaison des Cartes	46
D. Carte haute définition (HD)	49
E. Méthodes de résumé de cartes existantes	50
4. Conclusion	53

II.	Modélisation de l'environnement et extraction d'un graphe de navigabilité	55
1.	Introduction	57
2.	Sélection d'information : la saillance visuelle	57
A.	La saillance visuelle 2D	58
B.	La saillance visuelle 3D	60
C.	Sélection de point de vue optimal	63
D.	Solution choisie	65
E.	Conclusion	70
3.	Extraction d'un graphe de navigabilité	71
A.	Modélisation du problème	71
B.	Alpha Shape	72
C.	Critères de sélection des sphères du graphe	73
D.	Critères d'évaluation	76
E.	Algorithmes d'optimisation	77
4.	Conclusion	95
III.	Localisation en ligne et navigation dans un graphe de navigabilité	97
1.	Introduction	99
2.	Sélection de l'image de référence	99
3.	Recalage 3D : estimation de la position d'une image agent	100
A.	Introduction	101
B.	Go-ICP : a globally optimal solution to 3D ICP point-set registration	102
C.	Résultats	106
4.	Outils utilisés	130
5.	Optimisation et organisation du code	132
A.	Étapes de génération d'un graphe de navigabilité : organisation du code	133
B.	Optimisation du code	134
6.	Conclusion	137

IV.	Conclusion générale et perspectives	139
1.	Conclusion	140
2.	Perspectives	141
V.	Annexe A : détecteurs et descripteurs de points d'intérêt	142
1.	Harris	142
2.	Harris3D	142
3.	SUSAN	143
4.	FAST	143
5.	SIFT	145
6.	SIFT3D	147
7.	SURF	147
8.	MSER	147
9.	CenSurE	147
10.	AGAST	148
11.	PFH	148
12.	FPFH	150
VI.	Annexe B : organisation des données IGN	152
1.	Organisation des données IGN	152
A.	Organisation d'un fichier OBJ	152
B.	Organisation d'un fichier MTL	153
C.	Atlas de texture	153
VII.	Glossaire	155
VIII.	Bibliographie	156

Table des figures

0.1	Le projet pLaTINUM	15
1.1	Vision stéréoscopique	24
1.2	Procédé de photographie panoramique	25
1.3	Exemple d'un capteur Fisheye	26
1.4	Imagerie panoramique et stéréoscopique : système multi-caméras [2]	26
1.5	Image panoramique reconstruite à partir d'un système multi-caméras	27
1.6	Exemple d'une image panoramique construite à partir de 5 ou 6 images perspectives [2]	27
1.7	Exemple d'un capteur catadioptrique	28
1.8	(a) Système de caméras STEREOPOLIS utilisé par l'IGN [2] : un système panoramique en bleu et deux paires stéréoscopiques avant et arrière en vert. (b) les images utilisées pour construire les images panoramiques.	29
1.9	Véhicule STEREOPOLIS [2]	30
1.10	Zone d'acquisition STEREOPOLIS à Rouen [2]	30
1.11	Nuage de points laser obtenu lors de l'acquisition STEREOPOLIS à Rouen [2]	31
1.12	Extraction d'un maillage texturé et sémantisé [14]	31
1.13	Sémantisation des images sphériques [14]	32
1.14	Exemple de texturation et sémantisation d'un maillage sur une zone de 150 m à Rouen	32
1.15	Un maillage simple à 2 faces et 6 sommets	33
1.16	Coordonnées sphériques	34
1.17	Un exemple de lancer de rayon	35

1.18	Géométrie d'un triangle. Un triangle est défini par trois sommets qui définissent un plan. La normale est perpendiculaire à ce plan. Un rayon (en rose) coupe ce triangle.	35
1.19	Un rayon peut croiser plusieurs objets.	36
1.20	Intersection d'un rayon et d'un triangle. Le triangle est dans un plan. La valeur t est la distance entre l'origine du rayon et le point d'intersection.	36
1.21	Intersection rayon-triangle : le rayon en rose intersecte le triangle. Le rayon en marron rate le triangle et le rayon en vert est parallèle au triangle	38
1.22	Inside-Outside : soit un triangle (v_0, v_1, v_2) , A est l'arête définie par les deux sommets (v_0, v_1) , B est l'arête définie par les sommets (v_0, v_2) . B' est l'arête miroir de B . Si P est à gauche de A , le produit scalaire $N.C$ est positif. Si P est du côté droit ce produit est négatif.	39
1.23	Les coordonnées barycentriques peuvent être considérées comme l'aire des sous-triangles CAP (pour u), ABP (pour w) et BCP (pour v) dans le triangle ABC.	40
1.24	Exemple grille d'occupation.	43
1.25	Exemple d'une représentation géométrique [121]	43
1.26	Exemple d'une carte topologique [121]	45
1.27	Exemple d'utilisation d'une carte sémantique [78]	46
1.28	Technique de superposition	47
1.29	Organisation hiérarchique	48
1.30	Patchworks	49
2.1	Combinaison des approches descendantes et ascendantes	59
2.2	Détection de points saillants dans un nuage de points 3D à grande échelle [109]	63
2.3	Sélection de point de vue optimal. (a) peut être considérée comme un point de vue non pertinent et (b) comme un meilleur point de vue d'une scène contenant un cylindre et un cube.	64

2.4	Étapes de détection des zones saillantes [109]. La distinction des niveaux bas est d’abord calculée en identifiant les petites caractéristiques géométriques, telles que les dents et les pointes sur la tête et à l’arrière du dragon. Ensuite, l’association est appliquée, en regroupant les points saillants et en mettant l’accent sur les traits du visage du dragon. Ensuite, la procédure de distinction de haut niveau détecte des régions plus grandes, telles que la queue et la bouche. Enfin, les cartes sont intégrées pour produire la carte de saillance finale.	66
2.5	Un exemple d’extraction d’une carte de saillance [126]	67
2.6	La base de donnée de test (Réalité terrain)	68
2.7	Résultat de la détection des points saillants selon les méthodes d’extraction de la saillance 3D. -a- Méthode géométrique [109] - b- Méthode géométrique et photométrique [126] -c- Procédé photométrique [64] -d Harris3D- [46]	69
2.8	Évolution de F_β en fonction de θ	70
2.9	Exemple d’application de la méthode Alpha Shape sur un ensemble de points 2D	72
2.10	Exemple d’un problème de la galerie d’art : quatre caméras couvrent cette galerie.	73
2.11	Un exemple de polygone avec le Watchman route le plus court . . .	74
2.12	Évolution de l’Entropie E en se déplaçant dans une rue contenant un carrefour routier	75
2.13	Évolution de la localisabilité en se déplaçant le long de l’axe des x .	77
2.14	Optimisation séquentielle	79
2.15	Calcul de Watchman route dans un polygone.	81
2.16	Discrétisation et optimisation de l’Entropie sur un sous chemin du Watchman route	82
2.17	Graphe de navigabilité : chaque nœud est une image sphérique et chaque arc représente la distance entre les sphères.	85
2.18	Un nuage de points 3D de la ville de Rouen.(a) Nuage de points 3D sémantisé (b) Segmentation sémantique : 15 classes sémantiques . .	86
2.19	Image sphérique RGBD-L. La sphère RGB contient l’information photométrique. La sphère D contient l’information profondeur. La sphère L contient l’information sémantique	86

2.20	Extraction de polygones et positionnement de sphères	87
2.21	Un exemple qui illustre l'intersection de polygones de navigabilité et de visibilité	89
2.22	Un exemple de discrétisation des arêtes dans un graphe de barycentres	90
2.23	Maillage 3D à grande échelle représentant une grande zone à Rouen	91
2.24	Extraction de polygones de navigabilité et de visibilité : (a) et (c) : un nuage de points 3D, (b) : en rouge le graphe de navigabilité et en bleu le graphe de visibilité	91
2.25	Graphe optimal d'image sphérique RGB-D-L	92
2.26	Image sphérique RGB-D-L	92
2.27	Emplacement des nœuds du graphe de barycentres dans le deuxième maillage couvrant une trajectoire de 500 m	93
2.28	Les trajectoires de test à Rouen	94
2.29	Taux de compression en fonction de la résolution de la sphère	94
3.1	Entraînement d'un système de localisation basé vision en utilisant des informations auxiliaires [93]	100
3.2	Recalage 3D	101
3.3	ICP : Iterative Closest Point	102
3.4	Paramétrage de l'espace pour BnB. A gauche : l'espace de rotation. À droite : la translation est supposée être dans un cube 3D. Le cube jaune représente un sous-cube dans l'espace de recherche [125]. . . .	103
3.5	Zone d'incertitude [125]	104
3.6	Filtrage sémantique et recalage 3D à partir des images sphériques RGBD-L et un nuage de points 3D agent	106
3.7	Un exemple de recalage. Le nuage jaune représente l'image agent avant le recalage et le nuage vert le résultat de recalage	108
3.8	L'évolution de l'RMS et l'erreur en rotation en fonction du nombre de pixels N utilisés dans le recalage avec Go-ICP	110
3.9	Localisation dans un graphe	111
3.10	Localisation dans un graphe d'images panoramiques : en bleu la trajectoire réelle, en rouge l'estimation de la position de l'agent . .	111
3.11	Erreur de localisation et la distance entre l'agent et la sphère de référence en m	112

3.12	Erreur de localisation inférieure à 1m et la distance entre l'agent et la sphère de référence en m	113
3.13	Un exemple de localisation dans un graphe d'images panoramiques. En rouge : le véhicule navigue en marche avant. En vert : le véhicule effectue une marche arrière, des exemples d'images agent	114
3.14	Un exemple d'image panoramique acquise par l'IGN	115
3.15	Trajectoire d'essai à Rouen de longueur 150 m résumée avec 22 sphères RGBD-L	116
3.16	Scénario i : erreur de localisation obtenu avec/sans filtrage	117
3.17	Erreur de localisation avec et sans filtrage sémantique. 60 % des cas le filtrage sémantique améliore la précision de localisation	118
3.18	Erreur de localisation avec et sans filtrage sémantique en fonction de la distance entre l'agent et la sphère de référence : au-delà de 5m l'erreur de localisation augmente	119
3.19	Trajectoire d'essai : 150 m	119
3.20	Calibrage des caméras stéréoscopiques	121
3.21	Rectification des images stéréoscopiques	122
3.22	Extraction de la carte de disparité	122
3.23	(a) La rectification d'une image agent droite et une image gauche (b) La construction d'un nuage 3D agent à partir de deux images gauche-droite	124
3.24	Les séquences de la caméra ZED, vert : trajectoire du véhicule, bleu : trajectoire du piéton premier trottoir, rose : trajectoire du piéton deuxième trottoir	125
3.25	Résultats de recalage obtenus avec la séquence de l'agent sur la trajectoire piétonne : la sphère de référence (bleu), la zone piétonne (noire), le recalage réussi (vert), recalage échoué (rouge)	125
3.26	Résultats de recalage obtenus avec la séquence d'agent sur la trajectoire du véhicule : la sphère de référence (rouge), les résultats de localisation (noir), un exemple de sphère où le recalage a échoué	126
3.27	Recalage avec N sphères de référence	128
3.28	Recalage avec N sphères de référence : initialisation de la position de l'agent	128
3.29	Recalage d'image agent par rapport à N = 5 sphères de référence les plus proches	129

3.30	L'erreur de localisation avec $N=5$ et $N=3$	130
3.31	Diagramme de construction d'un graphe de navigabilité	133
3.32	Découpage d'une sphère selon l'angle θ en M secteurs horizontaux	134
3.33	Découpage d'un maillage selon l'angle θ	135
3.34	Découpage d'une sphère selon l'angle ϕ et θ en $N * M$ secteurs verticaux	136
3.35	Grille de recherche pour un rayon donné	137
5.1	Un exemple de détecteur de coins : SUSAN	144
5.2	Un exemple de détecteur de coins : FAST	145
5.3	Exemple de détection d'extremums dans l'espace des échelles	146
5.4	Construction du descripteur SIFT	146
5.5	Filtres utilisés par CenSurE (à gauche) et par STAR (à droite)	148
5.6	Voisinage d'un point requête p_q (en rouge) et ses voisins (en bleu) sont entièrement interconnectés dans un maillage	149
5.7	Repère (u, v, w) pour le calcul des histogrammes PFH	150
5.8	Voisinage d'un point requête p_q (en rouge) et ses voisins directs (cercle gris)	151
6.1	Organisation des données IGN	154
6.2	Exemple de texturation	154

Liste des tableaux

1	Résultats de détection des points saillants dans un nuage de points 3D	69
2	Résultats finaux	93
1	Classes sémantiques supprimées lors du filtrage sémantique	107
2	Résultats de recalage 3D avec Go-ICP en fonction de nombre de points 3D N utilisés dans le recalage	109
3	Score de recalage réussi avec et sans filtrage sémantique	117
4	Score de recalage réussi avec et sans filtrage sémantique, le taux de compression dans les scénarios	120
5	Recalage dans un graphe de sphère RGBD-L et une séquence d'images agent réelles	123
6	Résultats de recalage : Le score de localisation LS (%), l'erreur moyenne de localisation MLE (m). Le test est effectué avec filtrage f ou sans filtrage sémantique wf	124
7	Temps de calcul (s) de construction d'une sphère avant et après l'optimisation du code	137

Liste des algorithmes

- 1 Construction du graphe de sphères $X = \{X_i, i = 1..n\}$ 88
- 2 Go-ICP : a globally optimal solution to 3D ICP point-set registration 105

Résumé

Les caméras sont devenues de plus en plus communes dans les véhicules, les smartphones et les systèmes d'aide à la conduite ADAS (Advanced Driver Assistance Systèmes). Les domaines d'application de ces caméras dans le monde des systèmes intelligents de transport deviennent de plus en plus variés : la détection des piétons, les avertissements de franchissement de ligne, la navigation... La navigation basée sur la vision a atteint une certaine maturité durant ces dernières années grâce à l'utilisation de technologies avancées. Les systèmes de navigation basée sur la vision ont le considérable avantage de pouvoir utiliser directement les informations visuelles présentes dans l'environnement, sans devoir adapter le moindre élément de l'infrastructure. De plus, contrairement aux systèmes utilisant le GPS, ils peuvent être utilisés à l'extérieur ainsi qu'à l'intérieur des locaux et des bâtiments sans aucune perte de précision. C'est pour ces raisons que les systèmes basés sur la vision sont une bonne option car ils fournissent des informations très riches et précises sur l'environnement, qui peuvent être utilisées pour la navigation. Un axe important de recherche porte actuellement sur la cartographie qui représente une étape indispensable pour la navigation. Cette étape engendre une problématique de la gestion de la mémoire assez conséquente requise par ces systèmes en raison de la quantité d'informations importante collectées par chaque capteur. En effet, l'espace mémoire nécessaire pour accueillir la carte d'une petite ville se mesure en dizaines de *GO* voire des milliers lorsque l'on souhaite couvrir des espaces de grandes dimensions. Cela rend impossible son intégration dans un système mobile tel que les smartphones, les véhicules, les vélos ou les robots. Le défi serait donc de développer de nouveaux algorithmes permettant de diminuer au maximum la taille de la mémoire nécessaire pour faire fonctionner ce système de localisation par vision.

C'est dans ce contexte que se situe notre projet qui consiste à développer un nouveau système capable de résumer une carte 3D qui contient des informations visuelles collectées par plusieurs capteurs. Le résumé sera un ensemble des vues sphériques permettant de garder le même niveau de visibilité dans toutes les directions. Cela permettrait aussi de garantir, à moindre coût, un bon niveau de précision et de rapidité lors de la navigation. La carte résumant l'environnement sera constituée d'un ensemble d'informations géométriques, photométriques et sémantiques.

Mots clés : Navigabilité, vision, perception, carte, nuage 3D, résumé, maillage texturé, sphère, entropie, graphe, saillance, recalage, visibilité.

Abstract

Cameras have become increasingly common in vehicles, smart phones, and advanced driver assistance systems. The areas of application of these cameras in the world of intelligent transportation systems are becoming more and more varied : pedestrian detection, line crossing detection, navigation ... Vision-based navigation has reached a certain maturity in recent years through the use of advanced technologies. Vision-based navigation systems have the considerable advantage of being able to directly use the visual information already existing in the environment without having to adapt any element of the infrastructure. In addition, unlike systems using GPS, they can be used outdoors and indoors without any loss of precision. This guarantees the superiority of these systems based on computer vision. A major area of –research currently focuses on mapping, which represents an essential step for navigation. This step generates a problem of memory management quite substantial required by these systems because of the huge amount of information collected by each sensor. Indeed, the memory space required to accommodate the map of a small city is measured in tens of *GB* or even thousands when one wants to cover large spaces. This makes impossible to integrate this map into a mobile system such as smartphones , cameras embedded in vehicles or robots. The challenge would be to develop new algorithms to minimize the size of the memory needed to operate this navigation system using only computer vision. It’s in this context that our project consists in developing a new system able to summarize a 3D map resulting from the visual information collected by several sensors.

The summary will be a set of spherical views allow to keep the same level of visibility in all directions. It would also guarantee, at a lower cost, a good level of precision and speed during navigation. The summary map of the environment will contain geometric, photometric and semantic information.

Keywords : Navigability, graph, vision, perception, map, 3D cloud, summary, textured mesh, sphere, entropy, saliency, registration, visibility.

Introduction

1. Contexte de la thèse : le projet pLaTINUM

Ce travail de recherche a été mené pendant la période 2016-2019 à l'INSA de Rouen au sein du laboratoire d'Informatique, du Traitement de l'Information et des Systèmes (LITIS). Il s'inscrit dans le cadre d'un projet ANR (ANR-15-CE23-0010-01) intitulé pLaTINUM (Cartographie Long Terme pour la Navigation Urbaine). Ce projet est une collaboration entre le laboratoire LITIS, le laboratoire d'électronique, Informatique et Image (Le2I/VIBOT), l'équipe LAGADIC/CHORALE de l'INRIA : Asservissement visuel en robotique, vision et animation et le laboratoire MATIS : Méthodes d'Analyses pour le Traitement d'Images et la Stéréorestitution dans l'Institut national de l'information géographique et forestière (IGN).

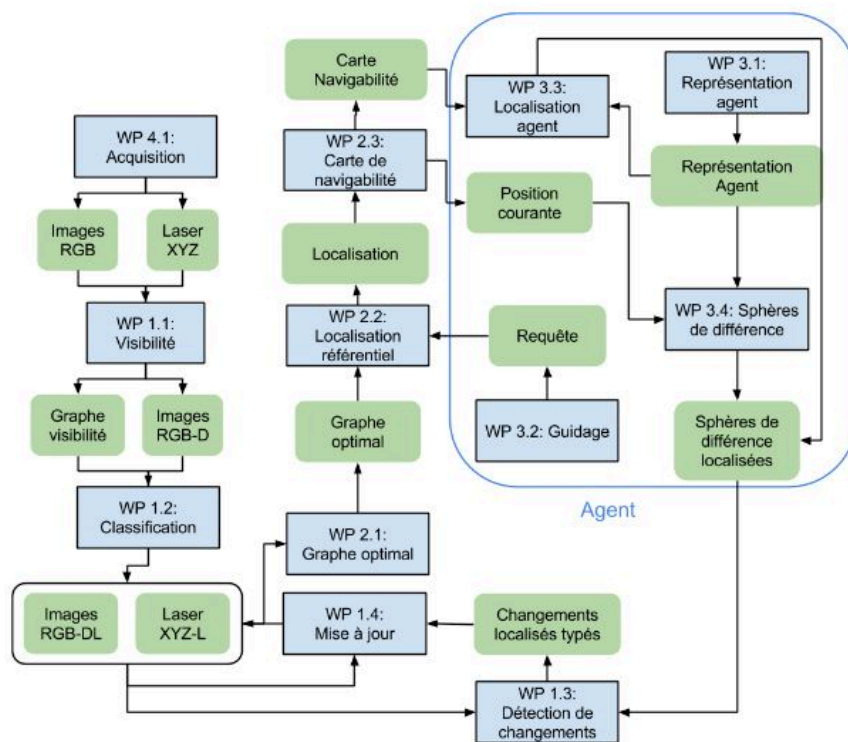


FIGURE 0.1 – Le projet pLaTINUM

Le projet pLaTINUM, porte sur le développement des méthodes et des algorithmes permettant de cartographier un environnement urbain, de l'enrichir et de le mettre à jour automatiquement à l'aide des capteurs visuels communicants embarqués par des utilisateurs. Un objectif majeur du projet pLaTINUM est l'exploitation

de données issues de systèmes de numérisation terrestres pour la localisation. Le projet se décompose en plusieurs parties comme il est indiqué sur la fig 0.1.

Cette figure permet de spécifier les entrées/sorties des briques du pipeline du projet pLaTINUM, représentées par les boîtes vertes dans la fig 0.1. La première étape *WP 4.1* est la numérisation de l’environnement. Cette étape est réalisée par l’IGN en utilisant un véhicule spécifique. Ce véhicule, qui parcourt l’espace public et le numérise spatialement est appelé Stéréopolis. Il est équipé de plusieurs capteurs tels que les caméras et le LIDAR. Par la suite, une étape de traitement de ces données acquises consiste à extraire un maillage texturé et sémantisé.

Ce maillage sera l’entrée de l’étape *WP 2.1*, qui représente notre brique. L’objectif de cette étape est de résumer le maillage de départ sous la forme d’un graphe de navigabilité. Chaque nœud du graphe sera associé à une sphère $RGB - D - L$ dont RGB représente l’information photométrique, D la profondeur et L l’information sémantique. Chaque arc est associé à des informations comme les commandes pour passer d’un nœud à un autre et la distance entre eux. Ce graphe sera embarqué sur un serveur ou un système de navigation à faibles ressources en calcul et en mémoire. Le calcul du graphe est effectué hors ligne. Ce graphe est considéré comme un graphe optimal en ce qui concerne la manière de placer ses nœuds. En fait, tous ses nœuds correspondent aux meilleurs points de vue de la scène. Ces points de vue permettent à la fois de garder une visibilité globale des informations pertinentes dans l’environnement tout en minimisant le nombre de nœuds dans ce graphe.

L’agent envoie une requête initiale pour se localiser ou pour obtenir le chemin le plus court à partir de sa position initiale vers une destination bien déterminée. Cette navigation est basée uniquement sur la vision. De ce fait, la requête envoyée par l’agent sera sous la forme d’image brute RGB accompagnée optionnellement de l’information de profondeur D et sémantique L . La localisation de l’agent se fait sur deux étapes. Au début, une localisation grossière est effectuée en calculant une position approximative de l’agent. Cette opération consiste à déterminer le nœud du graphe le plus proche de l’agent. Ensuite, une localisation fine et précise est réalisée en recalant l’image agent avec la sphère la plus proche. Pour guider l’agent, une carte de navigabilité est envoyée vers l’agent. Cette carte est un sous-graphe extrait du graphe optimal qui correspondra à une succession de vues associées à des commandes ou des indications de direction. La communication entre le serveur et l’agent se poursuit jusqu’à ce que l’agent atteigne sa destination finale. L’envoi des données de navigation du serveur vers l’agent se fait en temps réel. L’envoi des données se fait rapidement grâce au format sphérique des images. La dernière brique du projet consiste à mettre à jour la carte initiale. Pour ce faire, l’agent détecte les changements entre l’image de référence et son image acquise *WP 3.4*. Des sphères de différence seront envoyées vers le serveur pour mettre à jour la carte

de l’environnement. Cette étape est effectuée hors ligne d’une manière périodique chaque deux à trois semaines.

2. Objectifs

L’objectif de notre travail (*WP 2.1*) est de développer une méthode pour résumer une carte 3D initialement volumineuse représentant l’environnement. Cette carte est générée à partir de multiples capteurs permettant d’obtenir des modèles numériques de villes entières. L’utilisation de ce type de carte 3D sur des systèmes de navigation à faible ressource en mémoire ou en capacité de calcul, devient très compliquée. En effet la taille mémoire du modèle nécessite d’énormes capacités de stockage et de communication pour des applications du type temps réel. Cette méthode de résumé de carte doit garantir la diminution de la taille de la carte tout en gardant les informations nécessaires à la navigation.

3. Contributions

Les principales contributions de notre méthode se résument dans l’utilisation de l’information géométrique, photométrique et sémantique pour résumer efficacement un nuage de points 3D ou un maillage texturé et sémantisé. Les travaux effectués durant cette thèse ont donné lieu à plusieurs publications scientifiques :

- Dans BEN SALAH et al. [103], nous avons présenté le concept d’Entropie, qui est une mesure de la quantité d’informations pertinentes renvoyée par un point de vue en fonction des caractéristiques géométriques, photométriques et sémantiques. Cette mesure est utilisée pour sélectionner le meilleur point de vue.
- Dans BEN SALAH et al. [100], nous avons présenté un algorithme d’optimisation du critère d’Entropie. Cet algorithme permet de positionner les images sphériques d’une manière optimale. Cette méthode permet de résumer un nuage de points 3D sous la forme d’un graphe de navigabilité dont les nœuds sont des images sphériques augmentées avec l’information de profondeur et l’information sémantique.
- Dans BEN SALAH et al. [101], en se basant sur un problème connu dans la littérature sous le nom de la galerie d’art, nous avons proposé une nouvelle méthode d’optimisation du critère d’Entropie auquel nous avons ajouté un critère que nous avons appelé Visibilité. Nous avons proposé d’optimiser les deux critères d’une manière séquentielle. Cet algorithme prend en entrée un nuage de points 3D sémantisé.

- Dans BEN SALAH et al. [102], nous avons présenté une nouvelle méthode d’optimisation multi-objectifs. Cette méthode consiste à optimiser les deux critères Entropie et Visibilité simultanément. Cet algorithme prend en entrée un maillage texturé et sémantisé.

4. Organisation du manuscrit

Ce manuscrit est divisé en trois principaux chapitres :

- Dans le premier chapitre, nous présenterons quelques notions de la vision par ordinateur, de la reconstruction 3D et une étude bibliographique sur les méthodes de cartographie et de résumé des cartes 3D existantes.
- Dans le deuxième chapitre, nous présenterons la modélisation dense de l’environnement. Nous détaillerons la construction d’une sphère augmentée. Dans ce chapitre, nous aborderons la notion de saillance visuelle ainsi que différents détecteurs de points d’intérêt. Par la suite, nous détaillerons la modélisation du problème du résumé de carte. Nous présenterons finalement notre algorithme d’optimisation permettant l’extraction du graphe de navigabilité optimal.
- Dans le troisième chapitre, nous présenterons l’étape de la localisation dans un graphe de navigabilité. Cette étape est effectuée en temps réel tout en exploitant les avantages du graphe d’images sphériques augmentées et sémantisées. Nous montrerons l’avantage de l’utilisation de l’information sémantique pour obtenir une localisation précise et robuste en temps réel.

5. Liste des publications

- [103] Imeen Ben Salah, Sébastien Kramm, Cédric Demonceaux et Pascal Vasseur. Extraction d’un graphe de navigabilité à partir d’un nuage de points 3D enrichis. In : 16 èmes Journées francophones des jeunes chercheurs en vision par ordinateur (ORASIS 2017). 2017.
- [100] Imeen Ben Salah, Sébastien Kramm, Cédric Demonceaux and Pascal Vasseur.. Summarizing large scale 3d point cloud for navigation tasks. In : 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2017. p. 1-8.
- [101] Imeen Ben Salah, Sébastien Kramm, Cédric Demonceaux and Pascal Vasseur. Navigability Graph Extraction From Large-Scale 3D Point Cloud. In : 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018. p. 3030-3035.

- [102] Imeen Ben Salah, Sébastien Kramm, Cédric Demonceaux and Pascal Vasseur. Summarizing Large Scale 3D Mesh. In : 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018. p. 1-9.

Chapitre I.

État de l'art : solutions existantes
de cartographie et de résumé des
cartes

Carte Semantique

Carte Couleurs

Laser

Sphère RGB-D-L



1. Introduction

L'utilisation du GPS (Global Positioning System) est devenue de plus en plus commune dans les véhicules, les smart-phones, les smart-vélos... Cependant, l'erreur de localisation de ce système peut être de l'ordre de quelques dizaines de mètres, ce qui n'est pas compatible avec les contraintes des véhicules autonomes. En effet, guider un véhicule autonome sur des zones de navigabilité réduites demande une localisation beaucoup plus précise que le GPS. De plus, l'utilisation d'un GPS dans un environnement urbain, peut provoquer une perte de précision. En effet, la présence des bâtiments peut cacher des satellites ou refléter des signaux. Les tâches de localisation et de perception peuvent être résolues assez facilement en ajoutant de nouveaux capteurs tels que le LiDAR (Light Detection and Ranging) qui est capable de fournir une structure 3D de l'environnement étudié tout en gardant un haut niveau de précision et une haute vitesse d'acquisition. Plusieurs véhicules intelligents utilisent ce type de capteur comme le Velodyne. Cependant, les contraintes de l'industrie automobile ne sont pas toujours compatibles avec ce type de capteurs. En effet, le coût de ces capteurs est trop élevé ce qui empêche leurs commercialisations à grande échelle.

En revanche, les caméras sont devenues de plus en plus présentes dans les systèmes de navigation. Les caméras sont des capteurs bas coût pouvant fournir une présentation riche de l'environnement. De plus, contrairement aux systèmes utilisant le GPS, les caméras peuvent être utilisées à l'extérieur ainsi qu'à l'intérieur des locaux et des bâtiments sans aucune perte de précision. Voilà pourquoi la localisation basée sur la vision gagne de plus en plus de place et arrive à convaincre la communauté scientifique de travailler sur ce sujet avec plus d'intérêt. La première partie de ce chapitre présente quelques généralités sur la vision par ordinateur. La partie bibliographique détaille plusieurs méthodes de l'état de l'art pour la cartographie de l'environnement. Par la suite, nous détaillerons les méthodes existantes permettant de résumer une carte initialement volumineuse.

2. Généralités sur la vision par ordinateur

A. Systèmes de vision

a. Introduction

L'être humain sait parfaitement décrire et interpréter son environnement dans un repère tridimensionnel (3D). Sa perception des objets qui se trouvent dans cet environnement repose sur les informations de natures différentes fournies par

les différents sens dont la vue. L'image captée par l'œil humain est composée d'un ensemble de points (environ un million de pixels). Chaque pixel contient des informations sur la lumière (quantité et contenu spectral/couleur) reçue en ce point de la rétine. La vision humaine est extrêmement complexe d'où l'intérêt des nombreuses études menées par les spécialistes des neurosciences. La vision par ordinateur cherche à construire un modèle algorithmique qui, vu de l'extérieur, possède des propriétés semblables à la perception humaine.

b. Vision par ordinateur

La vision par ordinateur (aussi appelée vision artificielle ou vision numérique) est une branche de l'intelligence artificielle dont le principal but est de permettre à une machine d'analyser, traiter et comprendre une ou plusieurs images prises par un système d'acquisition (par exemple : caméras). Les outils fondamentaux de la vision par ordinateur sont la détection et la segmentation, l'extraction d'indices visuels, la localisation et la reconnaissance d'objets.

c. Vision monoculaire

Un système de vision monoculaire est un système de vision utilisant une seule caméra qui observe directement son environnement. Les algorithmes fondés sur l'utilisation de caméras classiques (perspectives ou orthographiques) sont plus aboutis que ceux fondés sur la vision omnidirectionnelle.

d. Vision stéréoscopique

La stéréoscopie est l'ensemble des méthodes pouvant reproduire une perception d'un objet à partir de deux images vues différemment par chaque caméra. Un système de vision est donc appelé stéréoscopique lorsqu'on dispose de deux caméras au moins (à des emplacements différents) qui regardent le même objet. La fig 1.9 illustre un système stéréoscopique. On cherche ici des informations sur la position des caméras et les coordonnées du point M de la scène en connaissant uniquement les coordonnées des points m_1 et m_2 images de (M) .

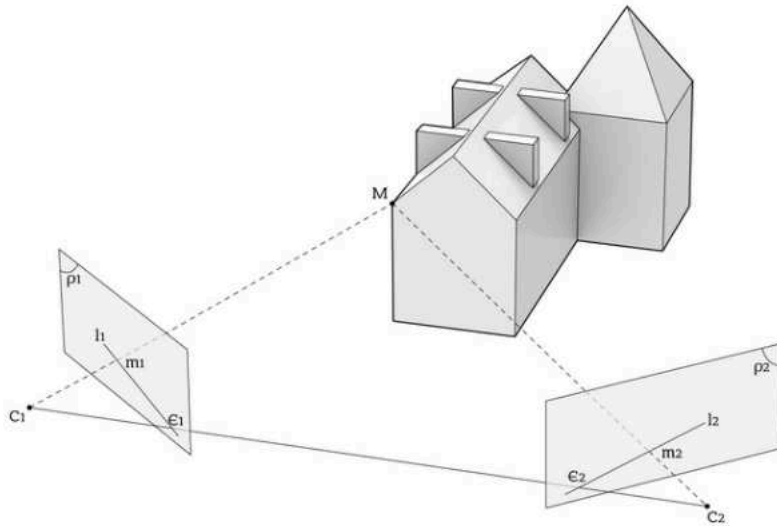


FIGURE 1.1 – Vision stéréoscopique

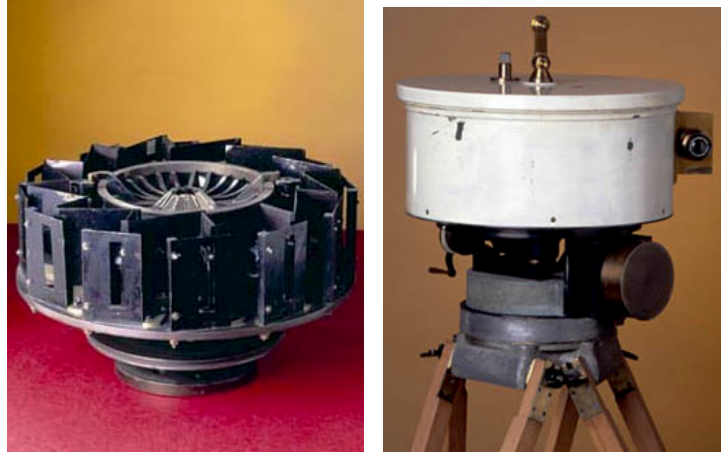
e. Vision omnidirectionnelle

Alors que la vision binoculaire humaine est limitée à 120° en moyenne, certains animaux comme les oiseaux, possèdent un grand angle de vue atteignant presque 360° . Une telle perception complète et large permet de mieux comprendre et analyser l'environnement. En effet, le principal inconvénient d'un système de vision classique est son champ de vue limité. Dans des applications telles que la robotique mobile et la télésurveillance, il est particulièrement important de pouvoir observer l'environnement dans toutes les directions et donc de supprimer le concept de direction du regard. C'est pour pallier cette limitation que la vision omnidirectionnelle a vu le jour ayant pour but d'augmenter le champ de vue d'un système de vision jusqu'à un angle de vue de 360° .

B. Imagerie panoramique

Ce concept de l'image panoramique est apparu en 1792 lorsque le peintre anglais R. Barker a créé la première peinture circulaire à 360° et invente le concept «Panorama» à partir des mots grecs «pan», qui signifie «tout» et «horama» qui signifie «vue». En 1900, le premier procédé de photographie panoramique permettant la reproduction complète de l'horizon, a été inventé par Louis Lumière. Le système Lumière est le premier à permettre à la fois la prise de vue et la projection

parfaite d'un tour d'horizon. Les prises de vues étaient réalisées avec le Périphote et la projection avec le Photorama comme le montre la fig 1.2.



(a) Le projecteur panoramique Photorama (b) L'appareil panoramique Périphote

FIGURE 1.2 – Procédé de photographie panoramique

Depuis cette invention, plusieurs systèmes permettant l'acquisition d'images panoramiques, ont été proposés. Nous pouvons classer ces systèmes en trois catégories : les caméras avec un champ de vision étendu, les caméras omnidirectionnelles catadioptriques et les systèmes multi-caméras.

a. Les caméras avec un champ de vision étendu : Fisheye

L'objectif fisheye permet de produire des photos avec un grand-angle de vue qui peut atteindre 180° dans la diagonale. Son nom est tiré de sa similarité avec l'œil de poisson. Cependant, ce type de capteur présente plusieurs défauts. En effet, une forte distorsion est présente sur les photos produites par ce capteur fisheye. Nous pouvons trouver aussi des aberrations chromatiques, ainsi que d'autres phénomènes d'optiques qui peuvent être gênant lorsque l'on désire obtenir une photo de haute qualité. Pour obtenir une vue panoramique complète, plusieurs techniques proposent d'utiliser plusieurs caméras fisheye (fig 1.3).



FIGURE 1.3 – Exemple d'un capteur Fisheye

b. Les systèmes multi-caméras

Ce type de système consiste à utiliser plusieurs caméras simultanément afin de produire une image panoramique [92]. La fig 1.4 montre un exemple de ce système utilisé par l'IGN.

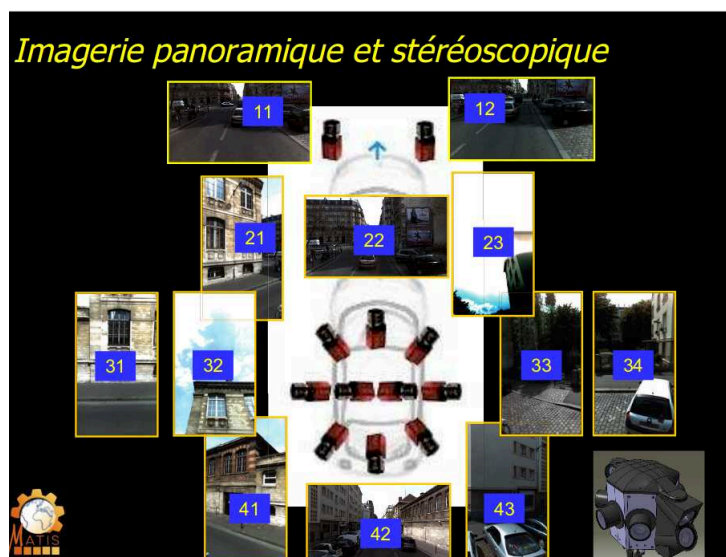


FIGURE 1.4 – Imagerie panoramique et stéréoscopique : système multi-caméras [2]

Une image panoramique complète sera construite à partir de plusieurs images perspectives. Pour ce faire, plusieurs algorithmes d'alignement d'images ont été proposés [7] ainsi que plusieurs techniques de mosaïquage [70, 89, 1]. Cette technique consiste à placer l'ensemble des informations vues dans les différentes images dans un repère commun. Les deux fig 1.6 et 1.5 montrent deux exemples de construction d'images panoramiques.

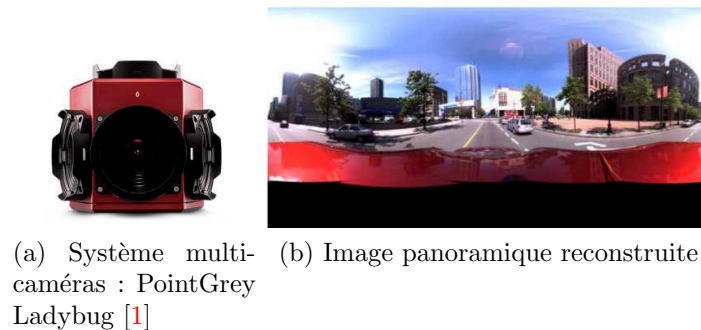


FIGURE 1.5 – Image panoramique reconstruite à partir d'un système multi-caméras

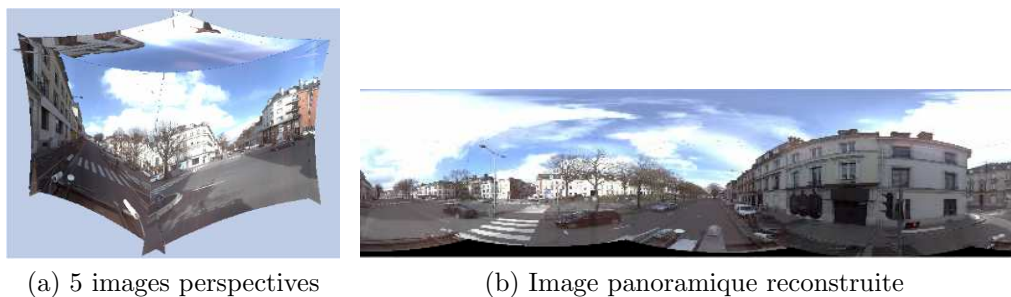


FIGURE 1.6 – Exemple d'une image panoramique construite à partir de 5 ou 6 images perspectives [2]

L'utilisation de plusieurs caméras permet d'obtenir des images panoramiques à haute résolution, mais la difficulté réside dans la calibration et la synchronisation entre les différentes caméras ainsi que dans l'assemblage des images perspectives.

c. Les caméras omnidirectionnelles catadioptriques

Ce type de caméra permet de couvrir un champ de vision horizontal de 360° produisant une image omnidirectionnelle. Ce système est composé d'une caméra perspective

et d'un miroir sous la forme hyperbolique ou parabolique [86]. Il est capable de fournir une image panoramique en une seule prise de vue et avec une seule caméra. Cependant la projection de 360° sur un seul capteur perspectif diminue la résolution de l'image. La fig 1.7 montre l'exemple d'un capteur catadioptrique.

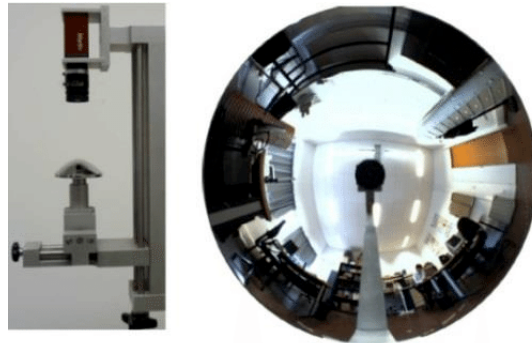


FIGURE 1.7 – Exemple d'un capteur catadioptrique

Pour compléter l'information, il faut ajouter la profondeur. Pour ce faire, deux solutions ont été utilisées dans la littérature. La première solution consiste à utiliser deux images omnidirectionnelles. La profondeur est extraite en appliquant des techniques de mise en correspondance entre les deux images [67, 60]. La deuxième solution consiste à ajouter un capteur actif comme le LiDAR qui permet d'obtenir la profondeur.

Tous ces systèmes permettent d'obtenir une image panoramique complète à 360°. Cependant, ils présentent une limitation lorsqu'on souhaite cartographier un environnement dynamique et assurer ses mises à jour. En effet, ils permettent d'acquérir une image panoramique dans un endroit fixe. Pour faire des mises à jour, il faut obligatoirement refaire l'acquisition d'une nouvelle image panoramique. Cela implique un coût supplémentaire et une perte de temps. Une autre alternative consiste à construire des images panoramiques virtuelles à partir d'un maillage ou d'un nuage 3D. La profondeur est ainsi obtenue simultanément lors de la construction de l'image panoramique. L'avantage de cette solution est sa capacité à créer des images panoramiques d'une façon dynamique sans refaire l'acquisition de nouvelles données. Dans notre projet, l'environnement de départ sera représenté par un nuage de points 3D ou un maillage texturé et sémantisé. Nous détaillerons par la suite la création d'un maillage texturé et sémantisé à partir des données brutes (caméras et LiDAR). Ensuite, nous présenterons la technique de création des images sphériques virtuelles.

C. Extraction, texturation et sémantisation d'un maillage 3D

Dans la littérature, rares sont les travaux qui proposent une présentation riche et dense de l'environnement. Dans le projet pLaTINUM, plusieurs capteurs (caméras, LIDAR, IMU...) ont été embarqués sur un système de numérisation mobile 3D hybride (laser-image) qui permet d'acquérir des infrastructures de données spatiales. La fig 1.9 montre le véhicule STEREOPOLIS de l'IGN utilisé pour cartographier l'environnement. Tous les 3 mètres, une acquisition de 5 images et un nuage de points produit par un LiDAR sont enregistrés. Ce processus permet d'obtenir le géopositionnement, les images ainsi que les nuages de points laser. Ce véhicule est équipé de plusieurs caméras : un système de caméras panoramiques et deux paires de caméras stéréoscopiques placées à l'avant et à l'arrière du véhicule. La fig 1.8 montre le système des caméras utilisé lors de l'acquisition STEREOPOLIS.



FIGURE 1.8 – (a) Système de caméras STEREOPOLIS utilisé par l'IGN [2] : un système panoramique en bleu et deux paires stéréoscopiques avant et arrière en vert. (b) les images utilisées pour construire les images panoramiques.

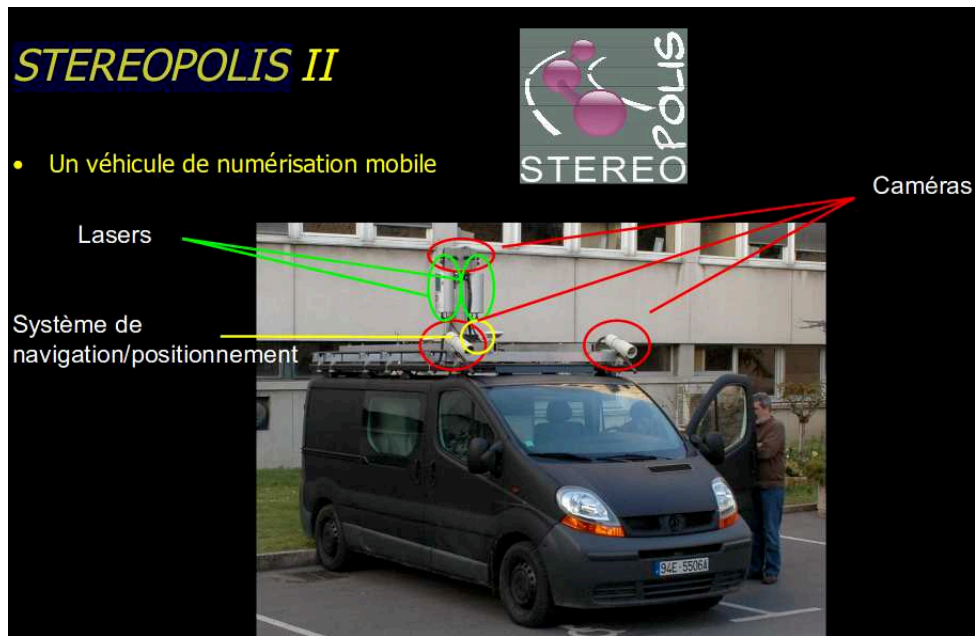


FIGURE 1.9 – Véhicule STEREO POLIS [2]

L'IGN a effectué une acquisition STEREO POLIS à Rouen (fig 1.10) afin de fournir les données nécessaires pour tester nos algorithmes. Une étape de prétraitement de données brutes acquises est appliquée afin d'extraire un maillage texturé et sémantisé. La fig 1.11 montre un exemple du nuage de points laser obtenu.



FIGURE 1.10 – Zone d'acquisition STEREO POLIS à Rouen [2]

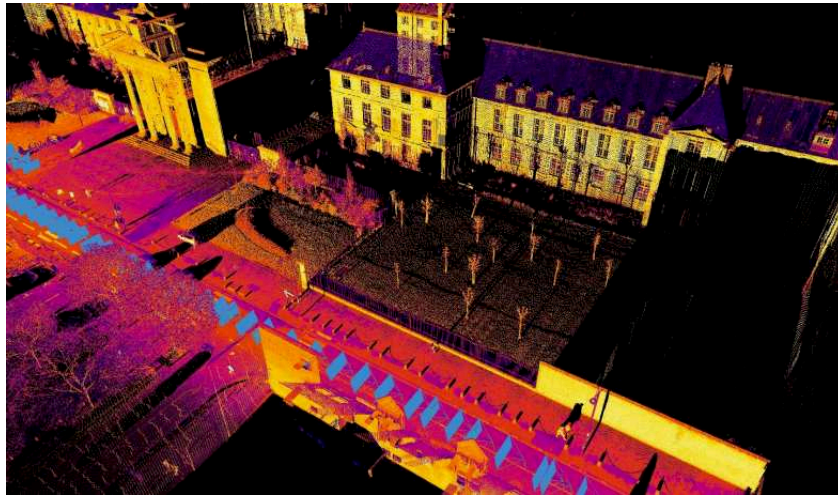


FIGURE 1.11 – Nuage de points laser obtenu lors de l’acquisition STEREOPOLIS à Rouen [2]

Pour produire un maillage texturé et labellisé, plusieurs étapes sont réalisées comme le montre la fig 1.12.

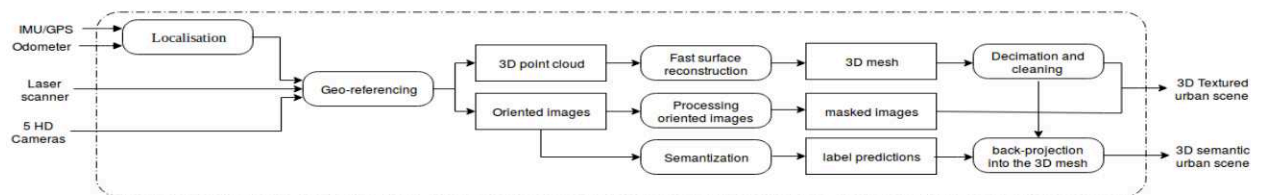


FIGURE 1.12 – Extraction d’un maillage texturé et sémantisé [14]

Le maillage est extrait en appliquant une triangulation sur les nuages de points 3D enregistrés par le LiDAR. Les images orientées servent à texturer et sémantiser les triangles du maillage 3D. Pour ce faire, la première étape consiste à construire des images panoramiques en utilisant la technique de mosaïquage [89] à partir d’un ensemble de 5 images perspectives.



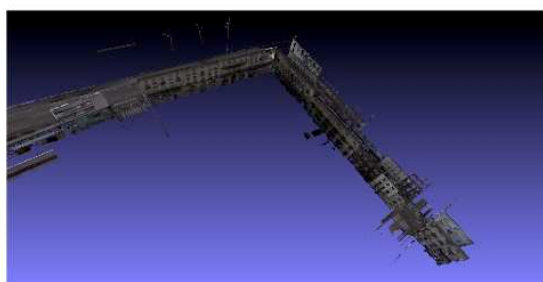
(a)



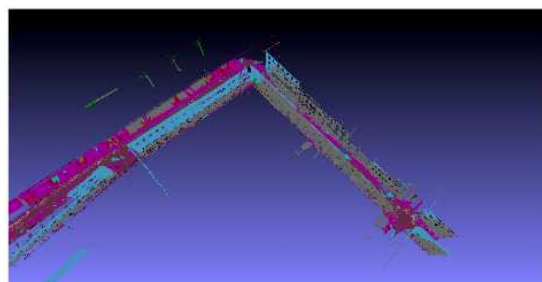
(b)

FIGURE 1.13 – Sémantisation des images sphériques [14]

En utilisant un CNN (Convolutional Neural Networks) une étape de sémantisation est réalisée sur les images panoramiques permettant de produire des images labellisées (fig 1.13). Les labels sémantiques et les couleurs sont ensuite reprojétés sur le maillage 3D. La fig 1.14 montre un exemple de texturation et de sémantisation d'un maillage couvrant une trajectoire de 150 m à Rouen.



(a) Exemple de texturation d'un maillage 3D



(b) Exemple de sémantisation d'un maillage 3D

FIGURE 1.14 – Exemple de texturation et sémantisation d'un maillage sur une zone de 150 m à Rouen

D. Construction d'une sphère virtuelle augmentée

a. Maillage 3D

Un maillage de polygones est constitué de sommets reliés les uns aux autres pour former des faces ou des triangles. Le maillage est défini par une liste de sommets ainsi qu'une liste de triangles.

Prenons l'exemple d'un maillage simple contenant 2 faces et 6 sommets comme l'illustre la fig 1.15. Dans cet exemple, pour présenter ce maillage, nous avons besoin d'une liste de sommets $\{v_0, v_1, v_2, v_3, v_4, v_5\}$. Nous avons également besoin d'une liste d'indices indiquant comment les sommets sont connectés les uns aux autres pour créer les faces. Par souci de simplicité et de généralité, il est généralement préférable de travailler avec un maillage triangulé. Dans ce cas, les polygones seront des triangles.

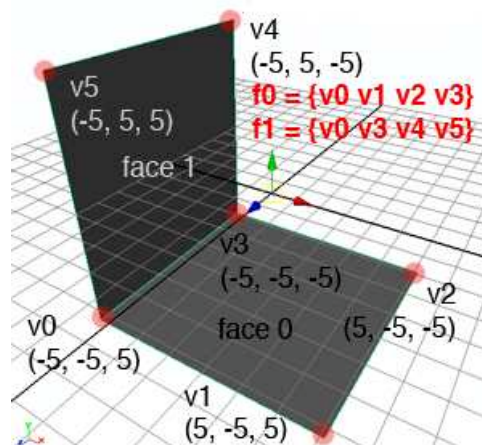


FIGURE 1.15 – Un maillage simple à 2 faces et 6 sommets

b. Coordonnées sphériques

Dans cette section, nous allons introduire les coordonnées sphériques. Les coordonnées sphériques [123], sont un système de coordonnées qui généralise les coordonnées polaires du plan (fig 1.16). Ce système permet de décrire un point Y sur une sphère par un rayon et deux angles $Y = (\rho, \theta, \phi)$. L'angle θ désigne la longitude et est compris entre 0 et 2π , et ϕ désigne la latitude et est compris entre 0 et π .

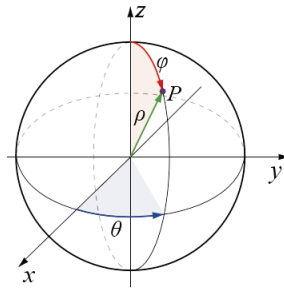


FIGURE 1.16 – Coordonnées sphériques

La conversion entre les coordonnées cartésiennes et les coordonnées sphériques se fait alors par les formules :

$$(x, y, z) = \begin{cases} \rho \cos(\phi) \cos(\theta), \\ \rho \cos(\phi) \sin(\theta) \\ \rho \cos(\phi) \end{cases} \quad (1.1)$$

c. Lancer de rayons (ray tracing)

Le « ray tracing », est une technique de calcul d'optique par ordinateur, utilisée pour le rendu d'image ou pour des études de systèmes optiques. Cette technique permet de calculer la visibilité entre les points. La visibilité entre deux points sera déterminée avec un lancement d'un rayon à partir d'un des deux points tout en interagissant avec des objets dans la scène.

Deux points de l'espace sont considérés visibles, si le segment de ligne qui les relie ne croise aucun obstacle. Une façon de produire une image à partir d'une scène 3D consiste à lancer des rayons à travers chaque pixel de l'image afin de déterminer la partie de la scène visible à partir de ce pixel. Le premier point d'impact du rayon sur un objet définit l'objet concerné par le pixel correspondant. La direction de ce rayon est indiquée par le vecteur allant de l'origine de la caméra au centre du pixel.

La fig 1.17 montre un exemple de lancement de rayon. Un rayon pouvant croiser un ou plusieurs objets dans la scène 3D. Dans d'autres cas, un rayon peut rater la totalité des objets de la scène.

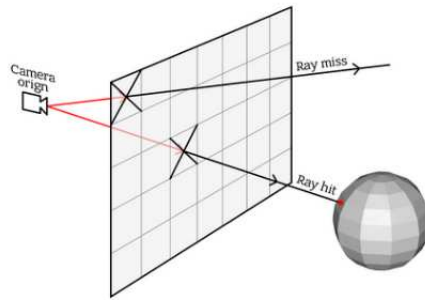


FIGURE 1.17 – Un exemple de lancer de rayon

Dans notre cas, la scène est représentée à l'aide d'un maillage triangulé. En effet, calculer l'intersection d'un rayon avec un triangle est une opération assez simple qui peut également être bien optimisée. La fig 1.18 montre la géométrie d'un triangle intersecté par un rayon.

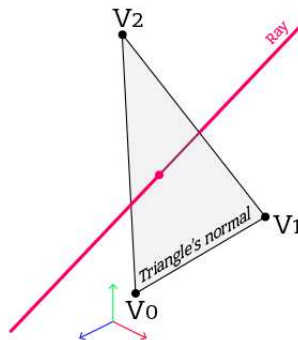


FIGURE 1.18 – Géométrie d'un triangle. Un triangle est défini par trois sommets qui définissent un plan. La normale est perpendiculaire à ce plan. Un rayon (en rose) coupe ce triangle.

À chaque lancer de rayon, on parcourt tous les triangles du maillage afin de trouver le triangle le plus proche intersecté par le rayon. Cependant, un rayon peut croiser plusieurs triangles (fig 1.19) dans un maillage, nous devons donc garder une trace de la distance d'intersection la plus proche lorsque nous parcourons les triangles.

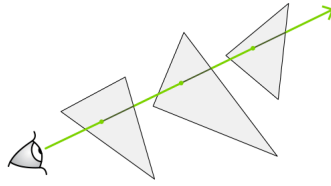


FIGURE 1.19 – Un rayon peut croiser plusieurs objets.

Au cours des dernières décennies, plusieurs algorithmes ont été développés pour calculer l'intersection entre un rayon et un triangle. Parmi ces algorithmes d'intersection rayon-triangle les plus rapides, nous citons l'algorithme proposé par Möller-Trumbore [83] en 1997. Le problème d'intersection rayon-triangle est décomposé en deux étapes :

- Le rayon coupe-t-il le plan défini par le triangle ?
- Si tel est le cas, le point d'intersection est-il situé à l'intérieur du triangle ?

c.1 Solution géométrique

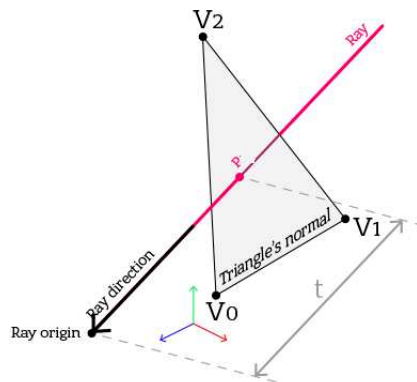


FIGURE 1.20 – Intersection d'un rayon et d'un triangle. Le triangle est dans un plan. La valeur t est la distance entre l'origine du rayon et le point d'intersection.

La fig 1.20 montre un triangle et le rayon qui l'intersecte. Calculer l'intersection du rayon et du triangle revient à trouver les coordonnées du point P . Nous savons

que P se trouve sur le rayon défini par son origine O et sa direction R . L'équation paramétrique du rayon est la suivante :

$$P = O + tR \quad (1.2)$$

Où t est la distance entre l'origine du rayon O et P . Pour trouver P , nous devons trouver t (fig 1.20). Pour ce faire, nous allons utiliser la normale du plan défini par cette équation :

$$Ax + By + Cz + D = 0. \quad (1.3)$$

Où A , B , C peuvent être vus comme les composantes (ou coordonnées) de la normale au plan ($N = (A, B, C)$) et D est la distance entre l'origine $(0, 0, 0)$ et le plan. Les variables x , y et z sont les coordonnées d'un point quelconque situé sur ce plan.

Pour calculer D , nous pouvons calculer le produit scalaire entre la normal N et un des sommets de triangle. Nous savons aussi que le point P qui est le point d'intersection du rayon et du plan se situe dans le plan. Par conséquent, nous pouvons substituer P (de l'équation 1.2) à (x, y, z) dans l'équation 1.3 et résoudre t (équation 1.4) :

$$t = -\frac{(N.O) + D}{(N.R)} \quad (1.4)$$

Après avoir calculé t , nous pouvons l'utiliser pour calculer la position de P (équation 1.2). Plusieurs situations peuvent se produire lors de calcul d'intersection rayon-triangle (Fig 1.21). Le rayon peut intersecter le triangle ou le rater. Si le rayon est parallèle au triangle, il n'y a pas d'intersection possible. Cette situation se produit lorsque la normale du triangle et la direction du rayon sont perpendiculaires (et que le produit scalaire de ces deux vecteurs est égal à 0).

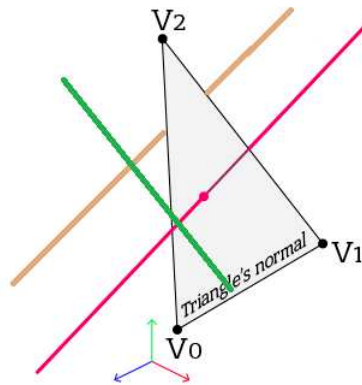


FIGURE 1.21 – Intersection rayon-triangle : le rayon en rose intersecte le triangle. Le rayon en marron rate le triangle et le rayon en vert est parallèle au triangle

Maintenant que le point P qui est le point d'intersection du rayon et du plan est trouvé, il reste à savoir si P est à l'intérieur ou à l'extérieur du triangle. La fig 1.21 illustre ces deux cas. La solution à ce problème s'appelle également le test d'Inside-Outside (fig 1.22). Dans cette technique, le même test est répété pour toutes les arêtes d'un triangle. Ce test consiste à calculer le produit vectoriel du vecteur défini par les sommets des deux arêtes et du vecteur défini par le sommet de la première arête et le point P . La deuxième étape consiste à calculer le produit scalaire du vecteur résultant et la normale N du triangle. Le signe du produit scalaire résultant détermine si le point P se trouve à droite ou à gauche de cette arête.

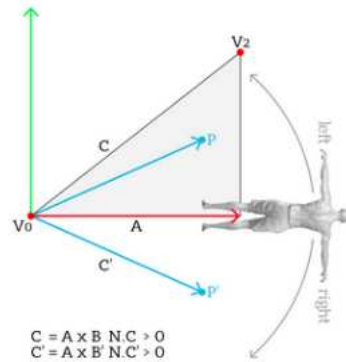


FIGURE 1.22 – Inside-Outside : soit un triangle (v_0, v_1, v_2) , A est l'arête définie par les deux sommets (v_0, v_1) , B est l'arête définie par les sommets (v_0, v_2) . B' est l'arête miroir de B . Si P est à gauche de A , le produit scalaire $N.C$ est positif. Si P est du côté droit ce produit est négatif.

c.2 Algorithme d'intersection rayon-triangle : algorithme de Möller-Trumbore

Avant de détailler l'algorithme de Möller-Trumbore, nous allons présenter les coordonnées barycentriques utilisées dans cet algorithme.

Coordonnées barycentriques

Les coordonnées barycentriques peuvent être utilisées pour exprimer la position d'un point situé sur le triangle avec trois scalaires (Fig 1.23). Ce point peut être à l'intérieur du triangle, sur une des trois arêtes du triangle ou un des sommets du triangle.

$$P = uA + vB + wC \tag{1.5}$$

Où A , B et C sont les sommets d'un triangle et (u, v, w) sont les coordonnées barycentriques tels que $u+v+w = 1$. On peut simplement utiliser deux coordonnées (u et v) pour exprimer la position d'un point P . L'équation 1.5 définit la position d'un point P sur le plan du triangle formé par les sommets A , B et C . Le point se trouve dans le triangle (A, B, C) si $0 \leq u, v, w \leq 1$. Si l'une des coordonnées est inférieure à zéro ou supérieure à un, le point est en dehors du triangle. Si l'un d'entre eux est zéro, P est sur l'une des lignes joignant les sommets du triangle.

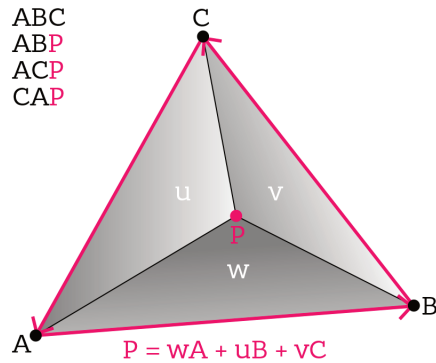


FIGURE 1.23 – Les coordonnées barycentriques peuvent être considérées comme l’aire des sous-triangles CAP (pour u), ABP (pour w) et BCP (pour v) dans le triangle ABC.

Algorithme de Möller-Trumbore

L’algorithme de Möller-Trumbore [83] est un algorithme d’intersection rapide rayon-triangle qui a été introduit en 1997 par Tomas Möller et Ben Trumbore. Il est toujours considéré comme un des algorithmes les plus rapides. Il est souvent utilisé comme une référence pour comparer les performances avec les autres méthodes. L’avantage de cet algorithme est l’utilisation des coordonnées barycentriques pour trouver le point d’intersection P . En remplaçant le point P par ses coordonnées barycentriques (équation 1.5) dans l’équation 1.2, nous obtiendrons une nouvelle équation 1.6.

$$O + tD = A + u(B - A) + v(C - A) \quad (1.6)$$

Dans l’équation 1.5, nous avons trois inconnues (t, u, v) et les termes connus A, B, C et D . Les termes sont réorganisés afin de trouver l’équation 1.7 suivante :

$$\begin{bmatrix} -D(B - A)(C - A) \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - A \quad (1.7)$$

Cet algorithme est plus rapide et simple à mettre en œuvre comparé à la solution géométrique.

E. Conclusion

Nous avons présenté quelques généralités sur la vision par ordinateur ainsi que l’avantage de la représentation panoramique. Les techniques de construction d’une

vue sphérique ont été présentées et en particulier, la technique de lancement de rayons. Dans la section suivante, nous détaillerons les solutions de cartographie existantes.

3. État de l’art : solutions de cartographie existantes

A. Introduction

La navigation autonome dans des environnements urbains complexes est l’un des sujets de recherche les plus importants dans la robotique et plus particulièrement le sujet de la localisation. En effet, la précision de la localisation avec un système GPS représente une problématique assez complexe. Cela est dû au phénomène de masquage causé par des multiples réflexions et réfractions que subit le signal GPS dans un milieu urbain plein de bâtiments et d’obstacles de tous genres. Les méthodes classiques de navigation sont basées sur l’odométrie classique qui consiste à utiliser différentes sortes de capteurs de vitesse et centrales inertielles. Ces méthodes se sont montrées inadaptées aux grandes distances de navigation en raison de leur tendance à la dérive et la complexité de mettre en œuvre ces différents capteurs dans le même système. Les améliorations récentes des performances du matériel informatique nous permettent désormais d’opter pour la vision par ordinateur en temps réel dans la navigation autonome. L’avantage de cette méthode réside dans la grande précision de ses résultats et son taux d’erreur inférieur à celui produit par des techniques plus coûteuses (IMU : Inertial Measurement Units) [48]) ce qui explique le recours au choix de l’odométrie visuelle pour la navigation autonome dans le cadre du projet pLaTINUM.

Dans la suite de ce manuscrit, nous allons dresser un état de l’art des méthodes de représentation et de cartographie de l’environnement en utilisant les informations visuelles. Ensuite, nous allons détailler les techniques existantes pour combiner différentes représentations de l’environnement afin d’obtenir un modèle plus riche. Ensuite, nous allons clôturer ce chapitre par un exposé sur les méthodes existantes pour résumer une carte 3D.

B. Méthodes de cartographie existantes

L’étape de la cartographie est l’une des étapes les plus importantes dans les applications utilisant la vision par ordinateur. Elle consiste à développer une représentation réaliste et peu volumineuse (en mémoire) de l’environnement étudié.

Dans la littérature, plusieurs méthodes ont été proposées pour représenter l'environnement. Nous dressons, par la suite, un état de l'art de ces différentes techniques de cartographie. Selon le type de relation spatiale liant les entités de base (mur, bâtiments, arbres...) qui constituent l'environnement étudié, nous distinguons deux différentes approches de représentation spatiale. La première approche est basée sur les coordonnées dans un repère absolu (représentations métriques) et la deuxième approche basée sur la définition d'un repère relatif reliant les entités de base observées entre elles (carte topologique). Dans la première méthode, les objets de l'environnement sont représentés par leurs coordonnées dans le repère absolu (carte métrique). Tandis que dans la deuxième approche, l'environnement est représenté par l'ensemble des relations entre les différents objets qui le constituent (graphe ou carte topologique).

a. Représentations Métriques : grille d'occupation

Cette méthode consiste à représenter l'environnement sous la forme d'une grille dans laquelle chaque objet (entité) de base sera représenté par une cellule ayant la même forme et la même taille. L'environnement étudié sera donc décomposé en plusieurs cellules ayant chacune un niveau d'occupation proprement attribué (par exemple 1 si elle est occupée même partiellement ; et 0 dans le cas contraire). Cette approche est utilisée généralement dans la localisation des robots grâce à sa simplicité d'application. Dans les premiers travaux de recherche dédiés à cette problématique, le niveau d'occupation prenait systématiquement la forme d'une valeur binaire (0/1). D'autres travaux plus récents commencent à le présenter sous la forme d'une probabilité ou une valeur de vraisemblance ($[0, 1]$) pour chaque cellule. Moravec et Elfes [32] ont été les premiers à proposer l'utilisation de représentations de la grille d'occupation pour la navigation de robots mobiles. Les grilles d'occupation sont maintenant utilisées dans de nombreux systèmes de cartographie [40, 44, 117]. Cette idée a également été adoptée pour modéliser l'espace 3D [84]. La fig 1.24 montre un exemple d'une grille d'occupation. Le principal inconvénient de ce type de représentation est le fait qu'elle ne soit pas adaptée aux environnements de dimensions importantes (villes entières) car elle nécessite un espace mémoire important et un temps de calcul assez long même pour des supercalculateurs très performants. L'uniformité de représentation est un autre inconvénient de cette méthode. En effet, toutes les zones de l'environnement sont représentées de la même façon indépendamment de leur degré d'importance (arbres, coins, fenêtres...). Ce qui représente un gaspillage considérable des ressources disponibles.

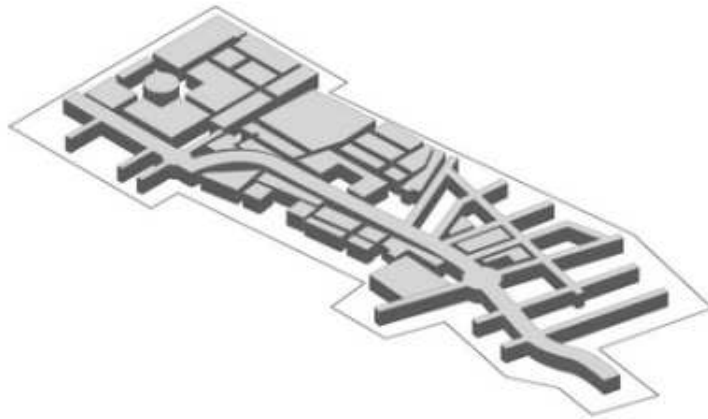


FIGURE 1.24 – Exemple grille d’occupation.

b. Représentations Métriques : carte géométrique

Dans cette représentation, les entités de base sont des formes géométriques (points, lignes, courbes, plans...). Cette méthode consiste à construire une représentation de l’espace à l’aide d’un ensemble d’objets géométriques situés dans le même système de coordonnées (repère absolu). La fig 1.25 montre un exemple simple de cette représentation.

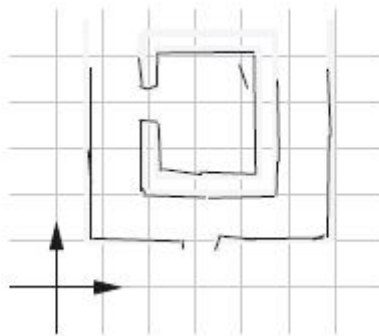


FIGURE 1.25 – Exemple d’une représentation géométrique [121]

Nous citons quelques exemples qui illustrent l’évolution de cette méthode : Chatila et Laumond [21] ont utilisé cette approche, pour décrire un environnement servant à la localisation d’un robot mobile. Crowley [25] a aussi utilisé cette approche, avec

les lignes comme des entités de base. Liu [69] a développé, en 2001, une approche différente basée sur l'extraction d'un modèle 3D compact à partir d'un ensemble donné de mesures dans l'espace 3D de l'environnement étudié.

c. Représentations Métriques : carte avec points-repères

Cette approche est basée sur l'extraction des points saillants appelés points de repère dans l'environnement étudié. Ces points sont choisis selon un ensemble de critères : leur présence en nombre suffisant dans l'environnement, la facilité de les distinguer des autres formes... Les positions de ces points sont spécifiées dans un système global de coordonnées (repère absolu). Voici quelques travaux qui illustrent l'utilisation de cette approche. Guivant et Nebot [41] ont utilisé les troncs d'arbre comme étant des points saillants pour créer une carte 2D d'un parc naturel. Neira et Tardos [87] ont utilisé les bords verticaux, les coins et les cadres des fenêtres comme étant des objets saillants pour cartographier l'environnement. Cette approche a également été utilisée avec succès pour les sous-marins avec des points de repères artificiels dont les coordonnées sont extraites à partir des données du sonar (profondeur, distance...).

Cette méthode s'adapte facilement aux environnements les plus volumineux grâce au nombre limité de points repères utilisés. Cela représente un gain considérable en espace mémoire et temps de calcul.

Cette représentation a un autre avantage en ce qui concerne la manipulation de l'incertitude, ce qui explique son utilisation dans les approches de SLAM (Simultaneous Localization And Mapping). Cependant, l'inconvénient majeur de cette méthode demeure dans la délimitation de l'environnement à cause de la discontinuité des points repères extraits. Ce qui inflige un handicap majeur à ce type de représentation dans l'application de planification d'itinéraires par exemple.

d. Représentations topologiques

Dans ce type de cartes, l'environnement étudié est transformé en un graphe illustrant les relations entre les différentes entités qui le constituent. Ces entités sont regroupées dans des ensembles appelés "les nœuds" suivant certains critères bien définis (co-visibilité, distance, type...). Ces nœuds, représentant des endroits distincts de l'environnement, sont reliés entre eux par des arcs permettant l'accessibilité entre ces différents endroits (fig 1.26). Dans ce contexte, Chapoulié [19, 20], a proposé en 2013 une segmentation topologique dont le choix des nœuds porte essentiellement sur la détection des changements dans les propriétés structurelles de l'environnement (textures, fréquence d'apparition, orientation des lignes et des

courbures, fréquence de répétition des motifs...). La faiblesse fondamentale de ce type de cartes réside dans le manque d'information métrique, diminuant ainsi leur précision ce qui réduit leur performance dans certaines tâches comme la localisation. Naviguer entre les nœuds en utilisant uniquement des informations de trajectoire pourrait fonctionner dans des environnements de structure statique (pas de changement dans la position des nœuds), contrairement à d'autres scénarios dynamiques plus complexes.

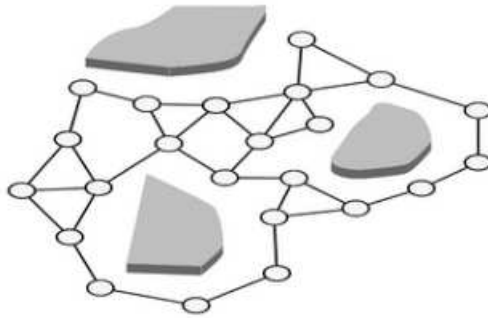


FIGURE 1.26 – Exemple d'une carte topologique [121]

e. Carte sémantique

L'incessante évolution dans le domaine de la cartographie a conduit à ce besoin accru d'enrichissement des informations des cartes avec une intervention humaine pour aider à mieux comprendre le contenu de ces cartes. Cela a induit le développement d'un nouveau type de cartographie appelé "la carte sémantique". Outre la construction des représentations simulant la structure 3D de l'environnement, ce type de carte nous permet aussi de modéliser le contenu sémantique de l'environnement en donnant un sens très proche de la perception humaine au contenu de la carte. L'attribution de labels sémantiques en lien avec la fonctionnalité de chaque objet de la carte (panneaux routiers, bâtiment, rues...) permet une meilleure interaction Homme-Robot ce qui rend la navigation plus efficace et plus simple. Un exemple illustrant cette technique a été réalisé par Drouilly [29] en 2014. Dans cet exemple, une nouvelle carte sémantique est ajoutée au modèle de base (carte topologique/métrique). Cette carte fournit un moyen puissant pour enrichir le modèle cognitif de l'environnement et représente ainsi un avantage pour la navigation. La fig 1.27 illustre la possibilité d'ajout d'une carte sémantique à d'autres types de représentation.

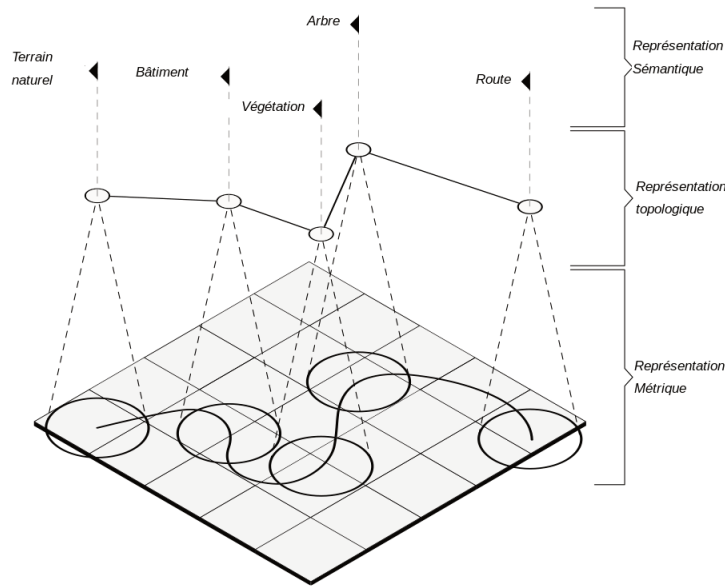


FIGURE 1.27 – Exemple d’utilisation d’une carte sémantique [78]

C. Organisation et combinaison des Cartes

Dans la section précédente, nous avons cité les différents types de représentation de base utilisés pour la modélisation de l’environnement. Par la suite, nous allons citer les méthodes existantes pour organiser ou combiner ces différents types de représentations afin d’obtenir un nouveau modèle plus riche tirant profit des avantages de chaque type combiné. Puis, nous allons examiner les approches existantes de combinaison de cartes en se concentrant sur les trois principales formes : la superposition, la représentation hiérarchique et l’approche Patchworks. Ces principales formes d’organisation/combinaison de représentations peuvent à leur tour être combinées entre elles pour former de nouvelles représentations encore plus complexes et plus riches d’informations.

a. Superposition de Cartes

Cette technique consiste à superposer plusieurs couches de représentations quelle que soit leur nature (métrique ou topologique), comme on peut le voir sur la fig 1.28. Chaque couche doit employer ses propres formalismes de représentation et couvrir l’intégralité de l’environnement. La liaison assurant cette superposition doit permettre la libre navigation entre les différentes couches de cartographie combinées. Un exemple illustrant cette technique a été réalisé par Thrun [117].

Dans cet exemple, la représentation de base est une grille d'occupation classique qui couvre l'environnement étudié (représentation métrique). Une fois l'environnement cartographié, une représentation topologique en est dérivée avec des nœuds correspondant à des régions distinctes de l'environnement. Ainsi, nous obtenons une combinaison de plusieurs couches de représentations superposées couvrant à la fois l'intégralité de l'environnement : une représentation métrique par grille d'occupation et une représentation topologique dont chaque nœud couvre un ensemble de cellules de la grille d'occupation. Cette méthode de superposition de différents types de couches permet de choisir, selon le besoin, la mieux adaptée d'entre elles pour chaque opération à effectuer (localisation, planification d'itinéraire, détection d'obstacles...). Cela pourrait aussi engendrer un coût supplémentaire sur les ressources utilisées en termes de temps de calcul et d'espace mémoire pour assurer un niveau acceptable de cohérence et de compatibilité entre les différentes représentations. En effet, l'inconvénient de cette approche de superposition reste son inefficacité dans les environnements de dimensions importantes (cartographies complexes : ville entière...).

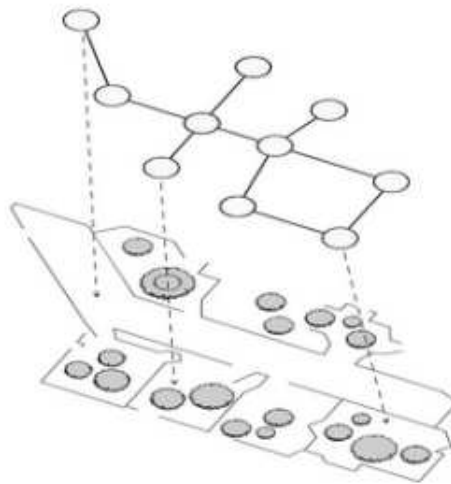


FIGURE 1.28 – Technique de superposition

b. Organisation Hiérarchique

L'organisation hiérarchique est très similaire à la superposition par le fait qu'elle soit composée de plusieurs couches couvrant l'intégralité de l'environnement. La particularité de l'organisation hiérarchique est que toutes les couches utilisent le même type de représentation (topologique/métrique) mais à différents niveaux de

granularité. La résolution en entités de base augmente en passant d'une couche supérieure à celle en dessous (fig 1.29). Le plus bas niveau fournit donc une carte assez détaillée de l'environnement, tandis que les niveaux supérieurs ont une résolution plus grossière et une carte plus abstraite. Dans la littérature, selon le type de représentation utilisé, nous pouvons trouver deux grandes catégories d'organisation hiérarchique : représentations basées sur les coordonnées (couches métriques) / représentations basées sur les relations (couches topologiques). Nous pouvons considérer le travail de Fernández et González [34] comme un exemple de cette technique avec la définition d'une organisation hiérarchique de graphe appelé le AH-Graph (Annotated Hierarchical Graph) composé de plusieurs couches, chacune sous la forme d'un graphe. Chaque nœud de chaque couche se décompose en un sous-graphe dans la couche inférieure avec une résolution plus importante en nœuds élémentaires. L'inconvénient de cette approche hiérarchique est l'effort supplémentaire (calcul/mémoire) nécessaire pour maintenir les liens entre les couches et maintenir les niveaux de représentation cohérents entre eux.

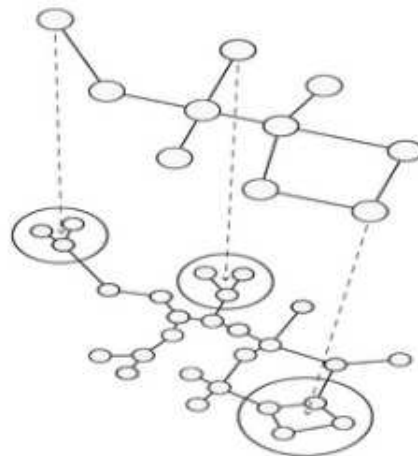


FIGURE 1.29 – Organisation hiérarchique

c. Organisation par Patchworks

Cette méthode consiste à décomposer l'environnement étudié en un ensemble de sous-environnements appelés plus communément "patches". Ces sous-régions de l'environnement sont représentées individuellement par des cartes locales à l'aide du même formalisme de représentation (métrique ou topologique). La fig 1.30 est un exemple de cette organisation. Les cartes locales sont alors liées à un niveau supérieur pour former une représentation globale de l'environnement. Dans

la littérature, les formes les plus communes d'organisations du type Patchworks utilisent des représentations (cartes) locales basées sur les coordonnées (type métrique) liées entre elles dans un graphe relationnel formant ainsi un niveau supérieur (type topologique) contenant la représentation globale (carte) de l'environnement. Une autre approche possible consiste à utiliser uniquement le type métrique à la fois pour les sous-cartes locales et aussi pour la carte globale de l'environnement.

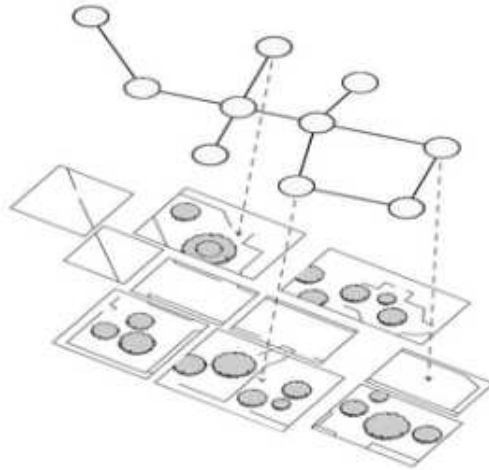


FIGURE 1.30 – Patchworks

Plusieurs cartes hybrides modernes ont adopté ou incorporé cette structure. Un exemple a été proposé dans [105, 82] où cette structure est utilisée pour produire une représentation détaillée de l'environnement.

D. Carte haute définition (HD)

Ces dernières années, une technologie clé pour la navigation des véhicules autonomes a été introduite. Cette technologie consiste à construire une carte haute définition (HD) [107]. L'utilisation de carte HD a fait une grande participation dans l'amélioration de la localisation basée sur la fusion de données (GPS, Lidar, caméra...).

Plusieurs fournisseurs de cartes numériques tels que TomTom, HERE, Bosch, Google et Apple fournissent aujourd'hui des cartes HD très riches et détaillées. Ces cartes permettent d'obtenir une navigation précise, en particulier dans les environnements urbains difficiles. La fusion des données des différentes sources (GPS, Lidar, caméra...) est réalisée pour produire ce type des cartes. Cependant, cette

fusion nécessite une haute puissance de calcul. De plus, le stockage des données collectées nécessite une capacité de plusieurs téraoctets. Le transfert de ces données de véhicule autonome vers un serveur de calcul n'est pas toujours compatible avec la contrainte de temps réel. D'où la nécessité de trouver une nouvelle méthode pour résumer ces cartes afin de réduire les ressources nécessaires (calcul / mémoire) pour faire fonctionner les systèmes de transport intelligents tout en préservant les informations de navigation essentielles (pixels saillants, nœuds importants, etc.).

E. Méthodes de résumé de cartes existantes

a. Pourquoi résumer une carte ?

La cartographie est une étape fondamentale qui précède la réalisation des différentes tâches de navigation (localisation, planification des chemins, évitement des obstacles...). La plupart des systèmes de navigation existants fonctionnent avec des ressources de calcul et de mémoire limitées d'où la difficulté rencontrée lors de la manipulation des cartes de taille importante et de l'intégration de ces cartes dans des systèmes embarqués. Dans certains domaines d'application, l'embarquement d'une carte complète de faible taille sur un dispositif mobile (automobile, robot..) ne pose aucune difficulté [6, 27, 50, 124, 66]. Cependant, le problème des ressources s'impose à d'autres applications où les cartes commencent à atteindre des tailles de plus en plus conséquentes malgré le recours à des solutions intermédiaires comme l'utilisation d'un serveur externe pour le stockage des cartes. Deux méthodes sont donc envisageables : le serveur externe envoie la partie demandée de la carte au dispositif mobile qui se charge d'effectuer les calculs nécessaires [119]. Ou bien, c'est le serveur qui se charge à la fois du stockage et du calcul [81, 120, 119]. Mais, dans les deux cas, plusieurs limitations menacent l'efficacité de ces solutions (coût d'utilisation du serveur, qualité de la connexion, temps supplémentaire de transfert d'informations, difficulté d'accès au réseau dans certains endroits...). D'où le besoin de trouver une nouvelle méthode permettant de résumer ces cartes afin de réduire les ressources nécessaires au fonctionnement du système (calcul/mémoire) tout en préservant les informations indispensables à la navigation (pixels saillants, nœuds importants...) sans pour autant diminuer la précision du système de navigation.

b. Méthodes de résumé de cartes

Pour réduire la taille d'une carte initialement volumineuse, deux approches sont possibles : la compression de carte ou la synthèse de carte. Cette synthèse consiste à utiliser uniquement des informations significatives. Au cours des ces dernières années, plusieurs approches de compression ont été proposées dans la littérature.

Certaines utilisent des structures de données spéciales telles que l'Octree [26] pour le codage progressif des nuages de points. Schnabel *et al.* [106] a proposé un schéma de prédiction pour réaliser la compression en utilisant l'approche octree. Une nouvelle méthode de compression a été proposée dans [56] pour coder uniquement les différences spatiales et temporelles au sein d'une structure octree. Jae-Kyun *et al.* [4] ont proposé un algorithme de compression géométrique permettant de compresser des nuages de points à grande échelle encodant les distances radiales en utilisant l'imagerie de distance. Les algorithmes de compression sont essentiels pour un stockage et une transmission efficaces, mais ils ne sont pas suffisants pour la navigation, car une étape de décompression est nécessaire pour traiter les données. De plus, ce type d'approche ne peut pas garantir l'efficacité des nuages compressés lors de la navigation. Plusieurs approches basées sur la sélection d'information caractéristique pour résumer une carte à des fins de localisation ont été présentées dans [30], [85], [115]. Ils proposent des fonctions permettant d'attribuer des scores pour classer les éléments de la carte en fonction de leur importance. Cette sélection permet de garantir une couverture maximale de la scène. Une approche de résumé de la carte a été proposée dans [116]. Son objectif est de sélectionner uniquement les lieux particulièrement adaptés à la localisation à l'aide d'une métrique appelée *location utility*.

Pour simplifier le processus de navigation basé sur l'apparence, un processus de sélection est appliqué afin de choisir les éléments clés dans l'environnement. Par exemple, dans les approches basées sur la mémoire visuelle, un ensemble d'images pertinentes et distinctives est acquis et utilisé pendant la navigation pour la comparaison avec la position actuelle. Dans le travail de Cobzas [24], une mémoire d'images panoramiques est créée en combinant des images acquises avec des informations de profondeur extraites d'un scanner laser. Dans cette base de données d'images, seules les informations saillantes seront conservées [10] sans dégrader les performances lors de la navigation. Afin de construire cette base de données d'images, certaines techniques ont été développées pour garantir une efficacité maximale dans le choix des informations utiles. D'autres méthodes basées sur le sac de mots (BoW) sont largement utilisées pour la localisation. Les méthodes BoW peuvent efficacement représenter une quantité de données importante en utilisant les occurrences de plusieurs vocabulaires visuels. En appliquant un dictionnaire hiérarchique au problème de navigation visuelle [36], les méthodes de BoW ont démontré une grande scalabilité et une grande précision dans les processus de localisation et de cartographie basés sur la vision. Une autre approche possible appelée "l'approche probabiliste" consiste à chercher le sous-ensemble de points 3D le plus compact possible qui pourrait représenter tout l'environnement en fournissant le nombre nécessaire de points utiles pour la résolution de l'algorithme Perspective-n-Points [65]. Dans cette approche la carte compacte finale

sera donc constituée par l'ensemble des points 3D ayant la plus grande probabilité d'apparition dans les images clés (un score attribué à chaque point d'intérêt). Dans certaines applications, l'utilisation de cette méthode heuristique a donné des résultats de compression impressionnants (jusqu'à 99%) tout en offrant une bonne qualité de localisation.

Une autre approche de localisation utilisant une technique qui s'appelle "la mémoire à base d'images" [79, 53, 96] est utilisée pour réaliser un résumé compact d'une carte. Dans cette approche de localisation, chaque estimation de position est calculée par rapport à une image de référence acquise au cours d'une phase d'apprentissage. Une mémoire de taille réduite à base d'images générées en combinant des informations de nature différente est utilisée dans la localisation à la place de la carte globale de l'environnement. Dans l'exemple de Cobzas [24], une mémoire d'images panoramiques est créée en combinant les images acquises par la caméra avec l'information de profondeur extraite à partir d'un scanner laser. Dans cette base d'images, seules les informations indispensables au processus de navigation sont retenues [10].

c. Comment construire une mémoire à base d'images ?

Pour construire cette base d'images, certaines techniques ont été développées afin de garantir le maximum d'efficacité dans le choix des informations utiles. La conversion des images brutes prises par le capteur en un nouveau type d'image (panoramique, sphérique...) engendre un certain taux d'erreur dû à la déformation de ces images lors de cette conversion. Le choix du bon type d'image permet donc de réduire ce taux d'erreur et de garantir une meilleure précision à la nouvelle base d'images ainsi construite.

Auparavant, les approches utilisées pour construire une mémoire à base d'images se basaient sur des images avec un champ de vue limité ne permettant pas de récupérer tous les détails de la zone ciblée de l'environnement. Le recours à un capteur ayant un large champ de vue devient alors nécessaire. Les caméras omni-directionnelles classiques ont un large champ de vue, mais leur inconvénient majeur est la mauvaise résolution dans la plupart des directions ce qui diminue fortement la précision de la localisation.

Une autre solution a été proposée par Maxime Meilland et al. [78, 77, 38] pour réaliser une représentation sphérique construite à l'aide d'un ensemble de caméras permettant de construire une image sphérique tout en gardant une haute résolution dans toutes les directions en combinant les images 2D venant des caméras avec l'information de profondeur venant d'un autre capteur laser.

Dans ce travail, le problème d'estimation de la position des sphères est formulé sous

la forme d'un problème d'optimisation. Cette position est liée à la mesure de luminosité et à la configuration géométrique 3D de l'environnement. L'étape de résumé de la carte et la construction des sphères est réalisée simultanément avec l'étape d'acquisition des images en utilisant un système à 6 caméras. Par conséquent, la carte sera composée d'images sphériques augmentées avec l'information de la profondeur. Cette information est extraite à partir du capteur laser ou aussi à travers la mise en correspondance entre les caméras stéréo. Pour sélectionner les sphères les plus représentatives de l'environnement au fur et à mesure de l'avancement du véhicule, un critère a été proposé. Ce critère est traduit sous la forme d'une mesure basée image : la valeur absolue des écarts à la médiane (MAD), qui représente l'échelle de la distribution de l'erreur. Une sphère est ajoutée au résumé si elle présente une différence significative par rapport à la sphère précédente et donc lorsque la valeur de la MAD dépasse un certain seuil.

Cette représentation sphérique augmentée est composée de plusieurs sphères dont chacune est définie par $S = \{I, P\}$. P est la position 3D de la sphère et I représente les intensités correspondantes. Le résultat final du processus de résumé aboutira donc à un ensemble de sphères augmentées stockées dans un système d'information géo-référencé qui sera utilisé par la suite au cours de la phase de navigation. Cette représentation sphérique présente les avantages suivants :

- La combinaison de différents types d'informations (luminosité, profondeur..)
- La localisation avec une grande précision dans toutes les directions.
- Une seule sphère suffit pour représenter différentes directions (comme les deux directions d'une rue).

4. Conclusion

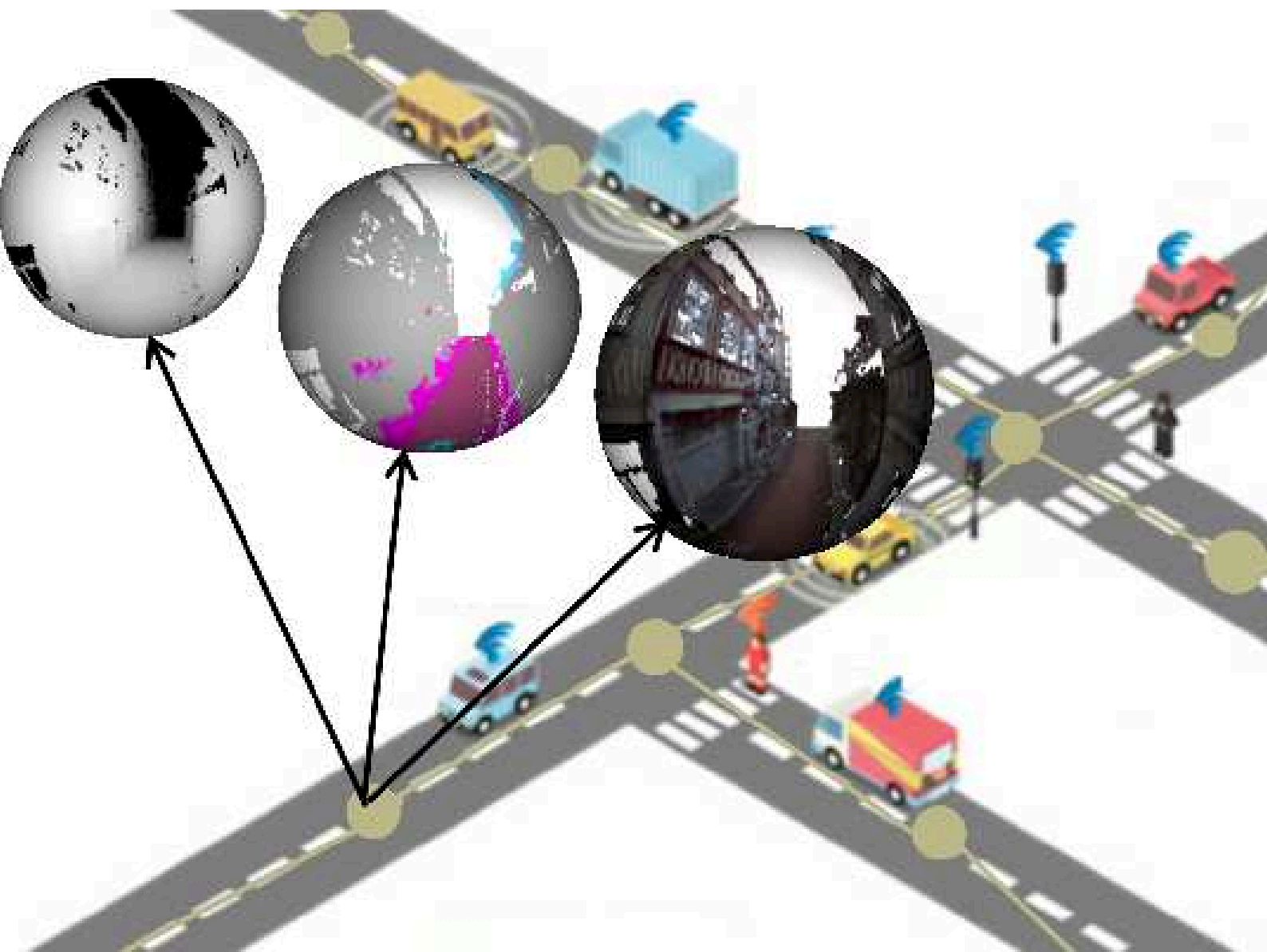
Dans ce chapitre bibliographique, nous avons dressé un état de l'art des systèmes permettant de représenter un environnement entier ainsi que les méthodes utilisées pour résumer une carte 3D. Nous avons fait un exposé détaillé des avantages et inconvénients des différents types de représentations (grilles d'occupation, cartes topologiques, cartes sémantiques...) avec la possibilité de combiner plusieurs types des cartes pour augmenter la robustesse du modèle des données. Nous avons présenté les méthodes existantes de résumé de carte. Toutes les méthodes existantes de résumé des cartes reposent principalement sur des caractéristiques géométriques

ou photométriques permettant de sélectionner les informations les plus pertinentes. Cependant, ces caractéristiques sont insuffisantes pour une bonne perception et compréhension de l'environnement. Une combinaison de caractéristiques géométriques, photométriques et sémantiques lors de la sélection des informations peuvent permettre d'avoir un résumé compact, précis et utile. Nous avons présenté quelques généralités sur la vision par ordinateur. En particulier, nous avons présenté l'imagerie panoramique. La représentation sphérique est l'une des meilleures solutions pour représenter un environnement 3D tout en gardant une visibilité globale dans toutes les directions.

Dans la suite de ce manuscrit, nous présenterons notre approche de résumé d'une carte 3D. Le résumé sera sous la forme d'un graphe de navigabilité dont chaque nœud est une image sphérique représentant une zone de la scène.

Chapitre II.

Modélisation de l'environnement et extraction d'un graphe de navigabilité



1. Introduction

Comme introduit précédemment, l'acquisition et la construction automatique de modèles 3D denses à grande échelle pose un problème de stockage et de transmission. Pour obtenir une carte réduite et utilisable pour la navigation, un processus de résumé de carte doit être appliqué. La représentation sphérique de l'environnement est plus appropriée pour représenter le plus fidèlement possible l'environnement. Ce genre de représentation a été utilisé dans plusieurs applications afin de fournir une couverture visuelle maximale. L'application "Google Earth" a utilisé cette représentation qui permet une immersion visuelle photoréaliste. Cependant, ses images sphériques ont été grossièrement positionnées dans l'espace ce qui réduit la précision de la localisation. Dans cette partie, une nouvelle approche alternative de cartographie est proposée. Pour cela, un graphe d'images sphériques est utilisé pour la navigation. Contrairement aux approches précédentes, cette approche permet de placer les images sphériques dans les meilleurs points de vue. Ce positionnement optimal permet de garantir une couverture maximale de l'environnement tout en gardant les informations pertinentes pour la localisation.

Dans ce chapitre, nous allons présenter la notion de la saillance visuelle. Nous présenterons notre contribution où nous avons adapté la saillance visuelle pour proposer une nouvelle méthode permettant de résumer efficacement une carte volumineuse. Nous détaillerons la modélisation de notre méthode ainsi que tous les algorithmes utilisés pour extraire le graphe de navigabilité. Nous allons définir deux stratégies d'optimisation ainsi que les résultats obtenus par chaque méthode.

2. Sélection d'information : la saillance visuelle

La saillance visuelle est la capacité de notre système de perception à distinguer certains objets de leur voisinage en fonction de leurs caractéristiques et d'attirer immédiatement notre attention. Il est à noter que plusieurs applications ont donné une grande attention à la saillance visuelle grâce aux avancées récentes dans la théorie de l'attention humaine, la psychologie cognitive, et la neurobiologie. Bien que de nombreux articles scientifiques sur la saillance visuelle ont été publiés au cours des dernières décennies, le domaine de la saillance continue à se développer très rapidement. Dans le domaine de la vision par ordinateur, la saillance visuelle a été activement abordée ces dernières années grâce à l'utilisation massive des données 3D. Ces données sont de plus en plus accessibles grâce à la grande diversité des types de capteurs utilisés (LIDAR, LASER, GPS...). La saillance visuelle représente un problème clé dans le domaine de la cartographie et la navigation. En effet lors du traitement d'un nuage de points 3D ou d'un maillage texturé, la

sélection des points qui formeront la carte compacte de l’environnement repose sur le choix des points les plus représentatifs de l’environnement. Ces points sont généralement appelés points clés ou points d’intérêt. Dans la littérature, il existe de nombreux algorithmes utilisés pour la détection de la saillance visuelle 3D. Les modèles de détection de saillance en 3D peuvent être regroupés selon leurs types de données d’entrée. Un premier modèle représente l’environnement sous la forme d’un ensemble d’images RGB augmentées avec leurs cartes de profondeur D . Dans ce cas, plusieurs méthodes ont été proposées pour détecter la saillance 3D sur des images $RGB - D$ [55], [91], [122]. Un deuxième modèle utilise un maillage 3D de l’environnement pour extraire les zones saillantes [62], [63], [114]. Le troisième modèle consiste à utiliser un nuage de points 3D comme entrée pour produire une carte de saillance [64], [109], [126]. Dans un premier temps, nous présenterons brièvement les différents modèles d’attention visuelle utilisés avec des données 2D (images). Ensuite, nous décrirons les modèles de saillance visuelle dans les environnements 3D ainsi que les différentes méthodes utilisées pour sélectionner le meilleur point de vue dans une scène.

A. La saillance visuelle 2D

La saillance visuelle est un processus qui permet rapidement au cerveau de réaliser des processus plus complexes sur certaines zones de la scène. L’être humain dispose d’un accès à une carte de saillance calculée par le cerveau afin de détecter les objets les plus importants d’une scène. La définition de la saillance dépend de différents facteurs (culture, intérêt, tâche à effectuer ...). Des nombreuses méthodes ont été proposées pour détecter les points d’intérêt dans une image afin de faciliter des tâches plus complexes (reconnaissance d’objet, reconnaissance de lieu, résumé vidéo ...).

Les modèles de détection de la saillance sont inspirés de deux lois : l’être humain est souvent attiré par le centre (objet, mot, image, etc.) [95], et l’attention de l’être humain est davantage attirée par les objets que par l’arrière-plan [17]. Classiquement, les algorithmes de détection de saillance se concentrent sur l’identification de points de fixation sur lesquels un observateur humain se concentrerait en un coup d’œil [51, 45, 16]. D’autres travaux se sont focalisés sur la détection d’un objet dominant dans une image [47, 42, 68].

Dans la littérature, il existe deux approches pour définir et extraire la saillance. L’approche la plus utilisée s’appelle l’approche «bottom-up» ou approche ascendante, elle repose uniquement sur les caractéristiques visuelles d’une image : couleurs, intensité, orientation, statistiques (covariance), caractéristiques temporelles (flux optique)...

La deuxième approche, appelée « top-down » ou approche descendante, est basée sur l'influence de l'attention humaine pour définir la saillance. Citant, par exemple, les approches basées sur les tâches qui consistent à donner la tâche aux observateurs avant d'extraire les zones saillantes dans une séquence d'images. Dans ce contexte une expérience d'Ali Borji [13] a été réalisée afin d'étudier le jugement de la saillance et les comportements de l'être humain. Il a montré que la saillance visuelle est influençable par la tâche demandée au départ. Ces deux approches présentent certaines limites, l'approche ascendante étant souvent insuffisante pour détecter les zones saillantes d'une scène. Avec cette approche, le contexte ou la tâche à exécuter n'est pas pris en compte.

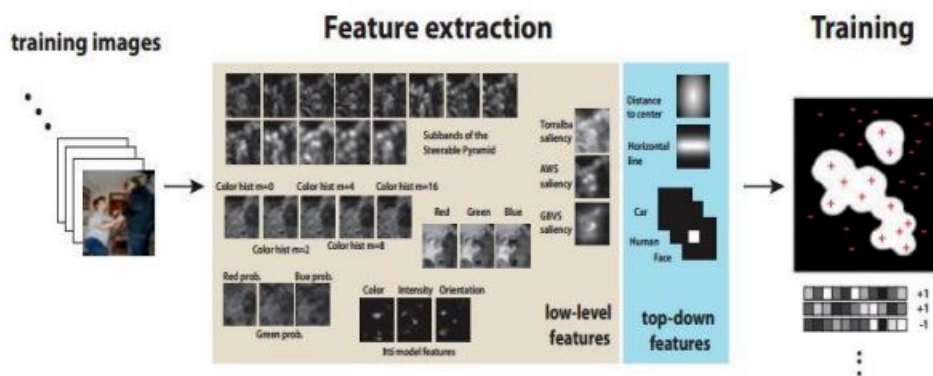


FIGURE 2.1 – Combinaison des approches descendantes et ascendantes

En revanche, les approches descendantes sont difficiles à mettre en œuvre, car le contexte est toujours difficile à modéliser. D'autres travaux ont proposé la combinaison de méthodes ascendantes et descendantes en utilisant des caractéristiques visuelles de bas niveau et des facteurs de haut niveau. La fig 2.1 montre la combinaison des approches descendantes et ascendantes. Dans [12], une classification bayésienne naïve est proposée pour estimer la saillance. Soit s une variable binaire indiquant la saillance d'un pixel d'une image à l'emplacement $X = (x, y)$. s est égale à 1 si le pixel est saillant. Soit f un vecteur de caractéristiques de pixel X (couleurs, intensité...). La probabilité que le pixel X soit saillant est écrite comme suit :

$$p(s|f, X) = p(s|f)p(s|X) \quad (2.1)$$

Le premier terme mesure la saillance en fonction des caractéristiques d'un pixel dans une image (estimée à l'aide d'une méthode de classification). Le second terme mesure la saillance uniquement en se basant sur l'emplacement d'un pixel. $P(s|X) = 1 - d(X, x_o)$ ou $d(X, x_o)$ est la distance séparant le pixel x du pixel

au centre x_o . Les performances de cette méthode dépassent celles de la majorité des modèles existants. Elle présente un avantage majeur : il s’agit d’une méthode générique, car il est possible d’ajouter plusieurs autres types de caractéristiques (ascendante/descendante). Parmi les approches ascendantes, nous citons la méthode d’Itti [52] qui présente une référence pour toutes les œuvres existantes. Cette méthode multi-échelle utilise trois caractéristiques de bas niveau : couleur, intensité et orientation pour calculer une carte de saillance d’une image. La saillance est définie comme le degré de divergence entre les régions voisines. Dans ce modèle, l’image d’entrée est d’abord décomposée en plusieurs cartes de caractéristiques (couleurs, intensité, orientation, etc.).

Dans chaque carte, un processus de sélection entre les différents emplacements spatiaux est ensuite mis en œuvre afin de sélectionner des emplacements différents de leur environnement. Après normalisation de toutes les cartes, une combinaison est faite pour former une seule carte globale de saillance. Il existe d’autres modèles basés sur la notion de graphe. Nous citons les travaux de Harel et al. [45], ils ont proposé un modèle fondé sur les graphes (GBVS). À partir d’une image, ils ont extrait des cartes caractéristiques (intensité, couleur et orientation) puis, pour chaque carte, un graphe connecté est construit. Les poids entre deux nœuds sont attribués proportionnellement au déséquilibre des valeurs de leurs caractéristiques et de leur distance spatiale. Le graphe obtenu est traité comme une chaîne de Markov. Les nœuds très différents des nœuds voisins recevront des valeurs de saillance élevées.

Parmi les approches descendantes, nous trouvons plusieurs modèles de classification qui ont été utilisés pour modéliser l’attention visuelle. Ces modèles utilisent des facteurs de haut niveau qui influencent l’attention visuelle [59]. Une évaluation de tous ces modèles a été réalisée dans une plate-forme de saillance (<http://saliency.mit.edu/>). Cette plate-forme contient deux bases de données : MIT300 (300 images) et CAT2000 (4000 images). Chaque image est accompagnée de sa carte de saillance (vérité terrain). Pour évaluer la performance de tous les modèles, huit critères d’évaluation ont été proposés. Parmi ces critères : AUC-Judd qui traduit l’aire sous la courbe ROC (Area Under the Curve), SIM qui présente la similitude entre deux cartes de saillance. Aujourd’hui, les modèles de classification et d’apprentissage sont les mieux adaptés pour prédire la saillance d’une image [61, 49].

B. La saillance visuelle 3D

.1 Images RGB-D Dans cette approche, l’environnement est représenté avec des images augmentées, la détection de la saillance visuelle est effectuée sur les deux types de données séparément. Une première étape consiste à calculer une

carte de saillance sur les images RGB. Pour ce faire, plusieurs modèles d'attention visuelle ascendantes et descendantes ont été proposés. D'autres modèles ont été proposés récemment utilisant l'oculométrie pour prédire les zones saillantes dans une image. La deuxième étape consiste à calculer une carte de saillance à partir de la carte de profondeur. Une combinaison de deux cartes produites séparément permet d'obtenir une carte de saillance globale.

.2 Maillage 3D Le système visuel humain peut percevoir facilement les surfaces et les formes géométriques. Plusieurs travaux ont été proposés pour construire un maillage 3D à partir d'un nuage 3D afin de produire une représentation compacte de l'environnement. Les premiers travaux [43] sur la détection de saillance pour les modèles 3D ont été inspirés par le calcul de la saillance 2D dans une image en se basant sur la détection des bords et en produisant des surfaces lisses à partir des données. D'autres travaux [90] ont été proposés pour calculer la saillance dans un maillage en utilisant les variations sur la surface. Dans [62] la saillance dans un maillage est calculée en utilisant le principe de dissimilarité d'une région saillante centrale avec ses régions voisines. Ils ont proposé d'utiliser la courbure comme étant un critère principal de calcul de saillance et de sélection de points de vue optimale. Dans [37], une méthode de mise en correspondance entre les différentes zones de maillage a été proposée. Chaque partie de maillage est représentée géométriquement à l'aide d'un descripteur basé sur sa courbure et la variation de cette courbure par rapport à son environnement. Plusieurs travaux ont été effectués pour trouver le point de vue optimal dans une scène. Parmi ces travaux, on cite le travail de [33] dont ils proposent de calculer la saillance d'un maillage et de sélection des points de vue en se basant sur l'information mutuelle entre les polygones. Ils définissent cette information mutuelle comme pour sélectionner les vues les plus représentatives d'un objet. Dans [108], les auteurs ont proposé une méthode pour calculer la saillance sur une surface 3D en détectant les régions saillantes. Ils comparent les différents patchs d'une surface 3D avec une base de données construite par plusieurs patchs saillants classés selon leurs appartenances à un même objet. Dans [22] une méthode de prédiction de saillance dans un maillage a été proposée en utilisant les "Points Schelling" sur la surface 3D. Une base de ces points est construite à partir de données recueillies dans le cadre d'un jeu de coordination où les utilisateurs choisissent les points qu'ils pensent être sélectionnés par d'autres utilisateurs. D'autres travaux [63, 109] proposent d'utiliser la distinction de région pour extraire les zones saillantes en s'appuyant sur le fait que les êtres humains sont attirés par les différences. Contrairement à la plupart des approches antérieures, qui ne considèrent que la distinction locale, ce travail prend en compte la distinction globale. De plus, il considère le fait que les formes visuelles peuvent posséder un ou plusieurs centres de gravité sur lesquels

la forme est organisée, cela peut influencer le critère de recherche de la saillance. D'une façon générale, les extrémités sont souvent considérées comme saillantes par les humains, c'est pour cette raison, ils ont choisi de calculer la saillance sur des sommets extrêmes d'un maillage. Les sommets saillants sont les sommets dont la géométrie est unique. Ceci est fait en calculant, pour chaque sommet, un descripteur qui caractérise sa forme. Un sommet est distinct si son descripteur est dissemblable à tous les autres descripteurs de sommets du maillage. Dans [113], un nouveau détecteur de point d'intérêt dans un maillage a été proposé. Cette méthode est basée sur l'analyse spectrale de données. La saillance est définie à partir d'irrégularité du spectre laplacien d'un maillage. De plus, ils utilisent la courbure gaussienne pour mesurer la géométrie locale pour aider à améliorer la localisation des points d'intérêt. Plusieurs modèles ont été proposés dans la littérature pour calculer la saillance globale et locale.

.3 Nuage 3D Une troisième famille consiste à traiter la saillance visuelle dans un nuage de points 3D. De la même façon que les algorithmes précédents de détection de saillance, qui fonctionnent sur d'autres types de données, ces algorithmes se basent aussi sur la distinction entre les régions pour prédire la saillance dans un nuage. Contrairement aux autres types des données traitées, les travaux qui traitent la saillance dans un nuage de points sont très rares en raison du manque de connectivité entre les points 3D. Dans [110] un algorithme de détection de saillance dans un nuage de points 3D à grande échelle a été proposé. Cette méthode consiste à utiliser un descripteur de point 3D appelé FPFH (Fast Point Feature Histogram) [98] pour caractériser la géométrie du voisinage d'un point. Un point est considéré distinct si son descripteur est dissemblable à tous les autres descripteurs de points du nuage. Cette opération est effectuée sur plusieurs échelles avec différentes tailles de voisinage. Ensuite, une somme pondérée des cartes de saillance de plusieurs niveaux est calculée pour obtenir une carte de saillance globale. Cependant, cette méthode détecte la saillance en se basant uniquement sur la géométrie de la scène sans l'utilisation d'autres types d'information comme la couleur.

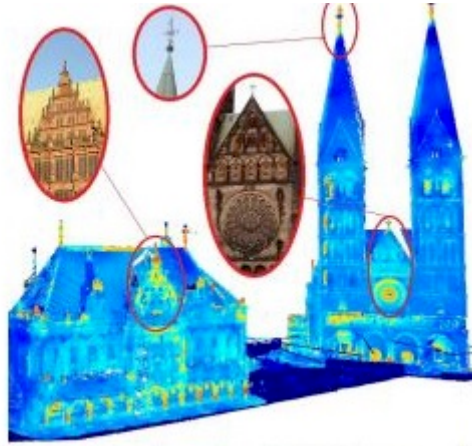


FIGURE 2.2 – Détection de points saillants dans un nuage de points 3D à grande échelle [109]

Un nouvel algorithme de détection de saillance pour les nuages a été proposé dans [127]. Cet algorithme consiste à exploiter les caractéristiques géométriques et les caractéristiques de couleur ensemble pour estimer la saillance dans un nuage de points coloré. Tout d’abord, tous les points 3D sont classés en plusieurs supervoxels. Ensuite une mesure de saillance pour chaque supervoxel est calculée à l’aide des caractéristiques géométriques et photométriques de ses voisins. Ce processus est appliqué sur plusieurs niveaux. En faisant la moyenne de cartes multi-échelles, une carte de saillance globale est obtenue. Les résultats expérimentaux ont démontré que l’algorithme proposé détecte les objets d’avant-plan globaux ainsi que les objets les plus fins d’une manière très efficace.

C. Sélection de point de vue optimal

La sélection de point de vue optimal a des applications dans plusieurs domaines : l’asservissement visuel, le mouvement de robot, etc. De plus, il devient un problème clé en vision par ordinateur. Il est difficile de définir avec précision le terme «point de vue optimal». Il semble intuitif de considérer qu’une vue est bonne si elle fournit une grande quantité d’informations pour une scène. Par exemple, la fig 2.3 (b) semble être meilleure que (a), car elle nous donne plus d’informations sur l’objet vu. Si la géométrie de la scène est connue, l’ensemble des faces de tous les objets peut être considéré comme l’information avec laquelle nous pouvons sélectionner le meilleur point de vue.

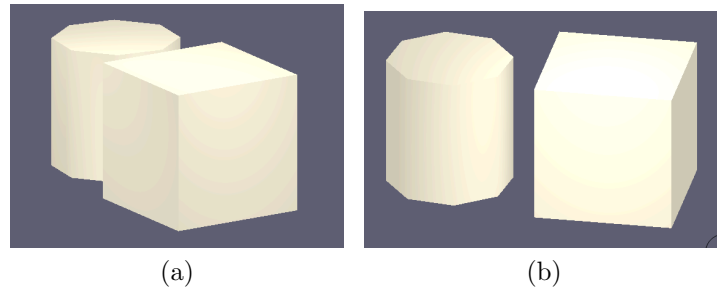


FIGURE 2.3 – Sélection de point de vue optimal. (a) peut être considérée comme un point de vue non pertinent et (b) comme un meilleur point de vue d’une scène contenant un cylindre et un cube.

Le calcul des meilleures vues dans un nuage de points ou dans une séquence d’images devient une question clé dans la vision par ordinateur. Il est devenu important d’élaborer des algorithmes permettant de sélectionner automatiquement les meilleures vues qui permettent d’élucider les caractéristiques les plus importantes d’un objet ou d’une scène. Ce type d’information peut être utile pour plusieurs domaines : la navigation urbaine, le dessin du graphique, la cartographie... Un point de vue est dit optimal si la quantité d’informations qu’il donne sur une scène est maximale. Et encore une fois, le terme ”information” est imprécis. Dans la littérature, plusieurs méthodes ont été proposées pour évaluer la qualité d’un point de vue dans une scène. On peut classer ces méthodes selon la nature des informations d’entrée. La première famille contient les méthodes qui utilisent uniquement la quantité des surfaces visibles à partir d’un point de vue pour l’évaluer. Une méthode de cette famille a été proposée par Plemenos et Benayada [94] qui considère un point de vue comme étant optimal s’il minimise l’écart d’angle entre une direction de vue et les normales aux faces. Dans [118] un algorithme de sélection de meilleur point de vue a été introduit. Cette méthode, fondée sur la théorie de l’information, propose de maximiser une fonction appelée ”Entropie” pour sélectionner le point de vue optimal. Un deuxième groupe ne prend pas en compte uniquement la quantité des surfaces visibles, mais aussi la géométrie de ces surfaces. Dans [118] Sokolov et Plemenos ont proposé de prendre en compte la courbure totale des surfaces visibles pour la sélection du point de vue optimal. Dans [112] une fonction pour mesurer la qualité d’observation pour un objet a été introduite. Cette fonction est utilisée pour définir le meilleur point de vue dans une scène initialement segmentée en plusieurs objets. Une autre méthode proposée dans [62], consiste à exploiter la saillance calculée sur un maillage de points. Le point de vue optimal maximise la somme de saillances des régions visibles depuis ce point de vue. La saillance d’une région est calculée à partir de sa courbure.

D. Solution choisie

Parmi les méthodes de détection des zones saillantes dans un nuage de points, nous détaillons trois méthodes basées sur l'information géométrique et photométrique. La première méthode (Méthode géométrique) a été proposée dans [109]. Cette méthode permet de calculer efficacement la carte de saillance à partir d'un nuage de points. Puisque l'attention humaine est attirée par les différences, les points saillants sont les points dont le voisinage est géométriquement unique par rapport aux autres voisins. La fig 2.2 montre un exemple des zones saillantes extraites à partir d'un nuage de points 3D. Pour mesurer la distinction, cette méthode utilise un descripteur de point 3D appelé FPFH (Fast Point Feature Histogram) [98] afin de caractériser la géométrie du voisinage d'un point 3D (Annexe A). Un point est considéré comme distinct si son descripteur est différent de tous les autres descripteurs dans son entourage. L'algorithme de détection de la saillance s'effectue de manière hiérarchique. La distinction est calculée sur plusieurs niveaux. Cette méthode recherche des régions saillantes plutôt que des points isolés. Cette considération découle de la tendance humaine à regrouper des éléments proches. Cette opération est réalisée sur deux niveaux avec des tailles de voisinage différentes. Tout d'abord, une distinction de bas niveau est calculée pour détecter ces petites caractéristiques D_{low} . Par la suite, une valeur d'association est calculée pour détecter les points saillants au voisinage des points les plus distincts A_{low} . Ensuite, une distinction de haut niveau est calculée pour sélectionner les grandes caractéristiques D_{high} . Enfin, les trois composantes ci-dessus sont intégrées à la carte de saillance finale S définie pour un point p_i comme suit :

$$S(p_i) = \frac{1}{2} * (D_{low}(p_i) + A_{low}(p_i) + D_{high}(p_i)) \quad (2.2)$$

La fig 2.4 illustre un exemple de détection des zones saillantes dans un nuage de points 3D.

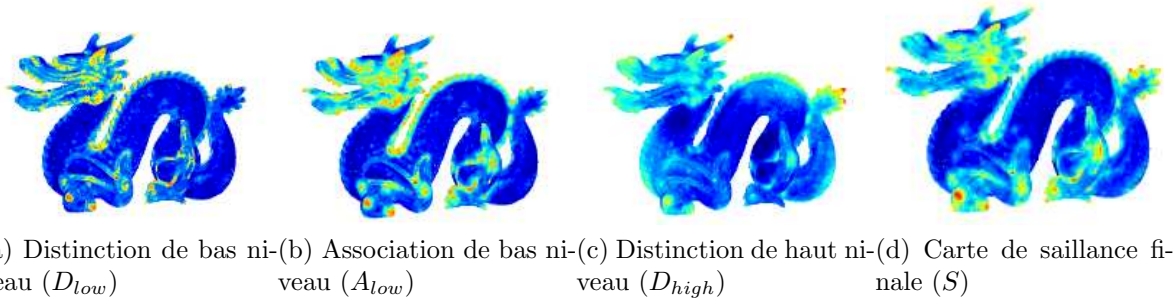


FIGURE 2.4 – Étapes de détection des zones saillantes [109]. La distinction des niveaux bas est d’abord calculée en identifiant les petites caractéristiques géométriques, telles que les dents et les pointes sur la tête et à l’arrière du dragon. Ensuite, l’association est appliquée, en regroupant les points saillants et en mettant l’accent sur les traits du visage du dragon. Ensuite, la procédure de distinction de haut niveau détecte des régions plus grandes, telles que la queue et la bouche. Enfin, les cartes sont intégrées pour produire la carte de saillance finale.

Cependant, cette méthode utilise uniquement la géométrie de la scène sans aucun autre type d’information, telles que les couleurs. Un nouvel algorithme de détection de saillance dans un nuage de points 3D a été présenté dans un travail récent [126]. Cette deuxième méthode (Méthode géométrique et photométrique) consiste à exploiter conjointement les caractéristiques géométriques et les caractéristiques de couleur pour estimer la saillance dans un nuage de points colorés. Cette méthode permet de regrouper les points 3D dans un nuage en supervoxels. Ensuite une valeur de saillance est calculée pour chaque supervoxel. La saillance d’un supervoxel est déterminée à l’aide de ses caractéristiques géométriques et photométriques et celles de ses voisins. Pour ce faire, une mesure de distinction est calculée pour chaque cluster en mesurant son contraste photométrique et géométrique par rapport à celle de chaque cluster adjacent. Le contraste d’un cluster C est calculé comme suit :

$$\rho(C) = \theta \rho_{geo}(C) + (1 - \theta) \rho_{color}(C) \quad (2.3)$$

Où ρ_{geo} et ρ_{color} sont les contrastes de caractéristiques géométriques et de couleurs normalisés pour un cluster C et θ est un paramètre de pondération qui est fixé empiriquement à 0.5 dans [126]. Les résultats expérimentaux démontrent que l’algorithme proposé extrait les régions globalement et localement saillantes des nuages de points 3D colorés à grande échelle en utilisant conjointement les caractéristiques géométriques et photométriques. La fig 2.5 montre un exemple de détection des zones saillantes dans un nuage de points 3D. Pour mieux étudier l’influence des

informations géométriques et photométriques, nous avons testé différentes valeurs de θ .

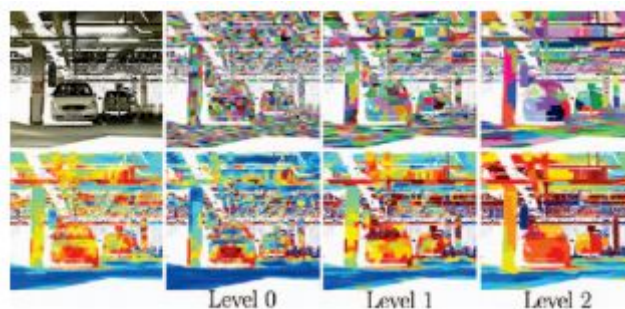


FIGURE 2.5 – Un exemple d’extraction d’une carte de saillance [126]

Une troisième méthode (Méthode photométrique) a été proposée dans les travaux de Leroy [64] basée uniquement sur la rareté de supervoxels. Pour chaque supervoxel v , une mesure de la rareté S_i est calculée en utilisant uniquement les caractéristiques photométriques. La rareté est obtenue sur plusieurs représentations d’espace colorimétrique (HSV, HLS, YUV, RGB, Lab, Luv).

$$S_i(v) = -\log\left(\frac{P_i}{N}\right) \quad (2.4)$$

Le mécanisme de rareté consiste, pour chaque supervoxel v , à calculer la probabilité d’occurrence croisée de chacun des N supervoxels. Pour chaque composante de couleur i , la probabilité de présence du supervoxel v est obtenue et N est le nombre de supervoxels. Ensuite, un score d’attention est attribué à chaque supervoxel. Ce mécanisme fournit des scores plus élevés pour les régions rares. La valeur de rareté est comprise entre 0 si tous les supervoxels sont identiques et 1 si un supervoxel est différent de tous les autres. Nous avons choisi d’implémenter ces trois méthodes (géométrie, photométrie et la combinaison des deux).

Dans la littérature, plusieurs détecteurs et descripteurs de points d’intérêts ont été proposés. Nous citons parmi ces algorithmes, le détecteur SUSAN [111], le détecteur FAST [97], le détecteur SIFT [71, 39], le détecteur SURF [8], le détecteur MSER [76], le détecteur CenSurE [3], le détecteur AGAST [74] et le détecteur Harris3D [46] qui est reconnu pour sa simplicité et son efficacité dans les applications de vision par ordinateur. Certains détecteurs ont été adaptés pour traiter des données 2D. Ces algorithmes sont décrits dans l’annexe A. Nous avons évalué certains détecteurs sur un nuage de points 3D afin de choisir les algorithmes les plus appropriés pour notre cas.



FIGURE 2.6 – La base de donnée de test (Réalité terrain)

Pour évaluer le niveau de la saillance capturée par ces méthodes, nous avons proposé de calculer un critère appelé dans la littérature : F_β mesure [75]. Ce critère nous permet d’avoir une idée sur la pertinence de l’information renvoyée par chaque méthode. F_β atteint sa meilleure valeur à 1 et la plus mauvaise à 0.

Cette mesure est calculée comme suit :

$$F_\beta = \frac{(1 + \beta^2) * (Precision * Rappel)}{\beta^2 * Precision + Rappel} \quad (2.5)$$

$$Rappel = \frac{VP}{(VP + FN)} \quad (2.6)$$

$$Precision = \frac{VP}{(VP + FP)} \quad (2.7)$$

Où β est un paramètre qui contrôle la préférence entre la détection complète et la sur-détection. Le rappel et la précision sont tout aussi importants si β est égale à 1. Si la valeur de β est inférieure à 1 on donne plus de poids à la précision, tandis qu’une valeur supérieure à 1 permet de favoriser le rappel. Nous avons décidé de donner beaucoup plus de priorité à la précision qu’au rappel pour éviter un rappel maximal de points (aucune discrimination). Pour ce faire nous avons choisi une valeur de β égale à 0.5 qui est une des valeurs les plus couramment attribuées à β . Dans ce cas, le rappel est deux fois moins important que la précision.

- Vrai Positif (VP) : nombre de points correctement classés comme pertinents pour la localisation
- Faux positifs (FP) : nombre de points classés à tort comme pertinents pour la localisation

Méthodes	F_β
Méthode géométrique et photométrique ($\theta = 0.7$) [126]	0.7401
Méthode géométrique [109]	0.7234
Méthode géométrique et photométrique ($\theta = 0.5$) [126]	0.6696
Méthode photométrique [64]	0.5363
Harris3D [46]	0.4536

TABLE 1 – Résultats de détection des points saillants dans un nuage de points 3D

- Faux négatif (FN) : nombre de points classés à tort comme non-pertinents pour la localisation

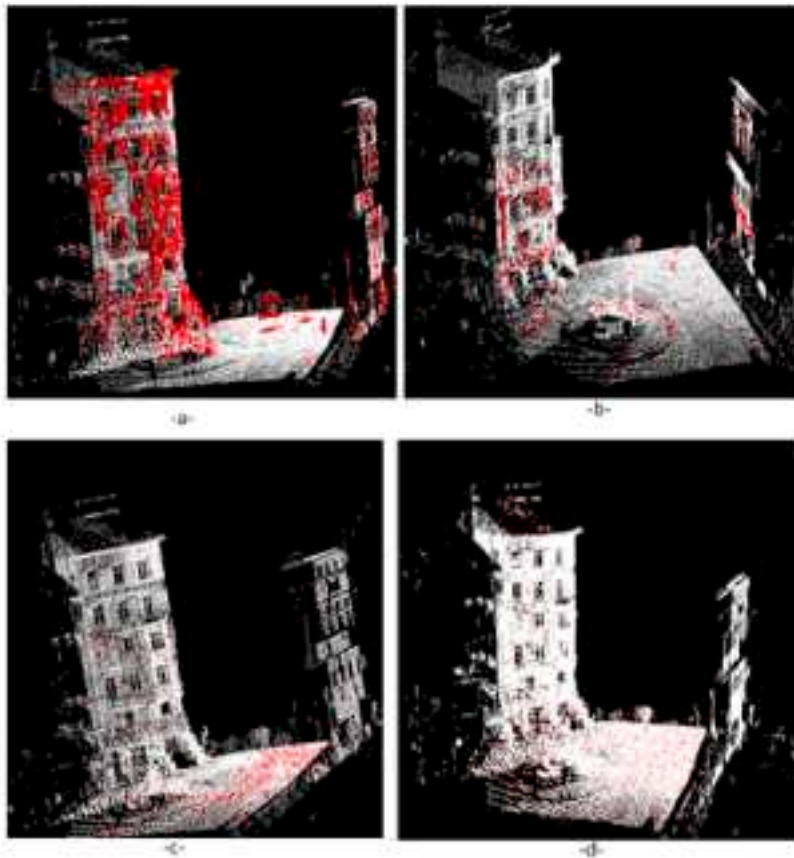


FIGURE 2.7 – Résultat de la détection des points saillants selon les méthodes d'extraction de la saillance 3D. -a- Méthode géométrique [109] -b- Méthode géométrique et photométrique [126] -c- Procédé photométrique [64] -d Harris3D- [46]

Nous avons testé ces méthodes sur une première base de données contenant 60 000 points 3D [128]. Ce nuage de points, développé à l'origine sans information sémantique, représente un environnement urbain et couvre environ $400 m^2$. Nous avons utilisé ce nuage de points 3D pour comparer les méthodes d'extraction de la saillance 3D. Pour construire une réalité terrain, nous avons procédé à une segmentation manuelle du nuage de points. Pour ce faire, nous avons choisi les points les plus saillants et utiles pour la localisation. Parmi ces points, nous avons choisi ceux qui appartiennent à la façade du bâtiment, aux panneaux de signalisation et au marquage au sol. Ainsi, tous les points appartenant à des objets mobiles comme les voitures et les piétons sont considérés comme non-pertinents. Cette vérité terrain a permis d'évaluer les résultats obtenus. La fig 2.6 montre le nuage de points utilisé dans l'évaluation des méthodes d'extraction de la saillance. Le tableau 1 montre les valeurs obtenues par les quatre méthodes. La fig 2.7 résume les résultats obtenus avec les différentes méthodes d'extraction de saillance. Parmi les détecteurs de points d'intérêts, nous avons gardé le détecteur Harris 3D qui a donné des meilleurs résultats. Les deux algorithmes utilisant la géométrie pour calculer la carte de saillance ont donné de meilleurs résultats que les méthodes utilisant uniquement la photométrie. En effet, les points d'intérêt pour la navigation sont généralement plus saillants géométriquement que photométriquement. La méthode qui utilise les caractéristiques géométriques et photométriques permet d'obtenir le meilleur score F_β . Ces résultats sont atteints lorsque le paramètre θ est égal à 0.7. Cette valeur permet de favoriser la distinction géométrique sans négliger la distinction photométrique.

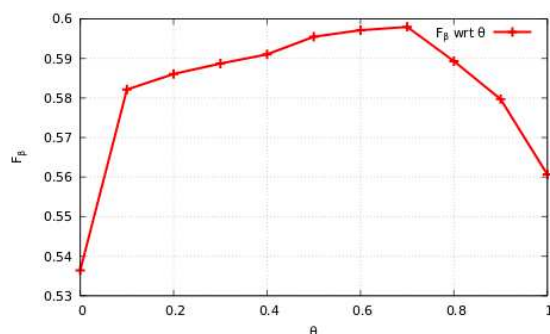


FIGURE 2.8 – Évolution de F_β en fonction de θ

E. Conclusion

Dans cette section, nous avons discuté des progrès récents dans la modélisation de l'attention visuelle dans des environnements 3D avec un accent sur les modèles

de saillance en 2D. Plusieurs recherches antérieures ont montré que la saillance visuelle dépend fortement des facteurs de bas niveau (intensité, couleurs, orientations...) et des facteurs de haut niveau (nature de cible, tâche à accomplir...). Dans les environnements 3D plusieurs travaux ont montré l'efficacité des caractéristiques géométriques dans une scène pour prédire sa saillance. Cependant, il reste plusieurs autres facteurs à découvrir et à étudier. L'intégration de ces facteurs supplémentaires peut aider à combler l'écart entre une carte de saillance produite par les êtres humains et une carte calculée à l'aide d'une des méthodes existantes pour prédire la saillance. La plupart des recherches précédentes sur la modélisation ont été axées sur la composante ascendante de l'attention visuelle, alors que la composante descendante est prometteuse. En outre, il n'existe pas encore une compréhension unique de principe de la saillance visuelle qui devrait être clarifiée à l'avenir.

3. Extraction d'un graphe de navigabilité

A. Modélisation du problème

Notre objectif est de résumer un maillage ou un nuage de points 3D en un ensemble compact d'images sphériques. Ce processus de résumé consiste à réduire le nombre de sphères tout en maintenant un niveau élevé d'informations significatives pour la navigation et la localisation. Pour modéliser ce problème, nous introduisons les notations suivantes. Soit une carte 3D sous la forme d'un nuage de points composée de N points 3D avec des informations couleur et sémantiques. $\xi = \{P_i(X_i, Y_i, Z_i) \rightarrow \{R_i, G_i, B_i, L_i\}\}, i = 1..N$. Dans ce modèle 3D, nous définissons les zones de navigabilité Γ comme un polygone 2D dont les arêtes représentent les frontières des zones de navigabilité. Nous définissons les zones visibles Σ comme un polygone 2D dont les arêtes représentent les frontières des zones visibles. Le résumé de la carte sera sous la forme d'un graphe de navigabilité. Les n nœuds de ce graphe sont des images sphériques augmentées $g_n = \{g_j, j = 1..n\}$. L'ensemble de sphères doit garantir un compromis entre les deux objectifs que nous détaillerons par la suite :

- Maximiser la Visibilité
- Maximiser l'Entropie

Pour extraire les polygones 2D, nous avons utilisé une méthode appelée Alpha Shape. Nous détaillerons par la suite le principe de cette méthode. Ensuite, nous présenterons les critères à optimiser ainsi que les différentes stratégies d'optimisation.

B. Alpha Shape

Supposons que nous avons un ensemble de points en 2D ou 3D et que nous aimerions savoir la forme construite par ces points. Ce problème est une notion assez connue dans la littérature et il existe de nombreuses interprétations possibles. Alpha Shape, est une méthode utilisée pour la reconstruction de la forme d'un ensemble dense et non organisé de points 2D/3D. En effet, une forme est délimitée par une frontière, qui est une approximation linéaire de la forme originale. Le concept Alpha Shape a été défini pour la première fois par Edelsbrunner, Kirkpatrick et Seidel en 1983 [31]. Cette méthode est une généralisation du concept de "convex hull". Alpha Shape dépend d'un paramètre α d'après lequel elle est nommée.

Pour un ensemble de points, l'algorithme dessine et déplace les cercles d'un rayon α vers l'ensemble de points, et assigne deux points comme points limites si les deux points touchent le même bord d'un cercle en même temps et aucun autre point ne se trouve dans le cercle comme illustré à la fig 2.9. Lorsque le rayon α est proche de 0, chaque point serait une limite. Par contre, quand α est proche de l'infini, la frontière sera la "convex hull" de l'ensemble de points.

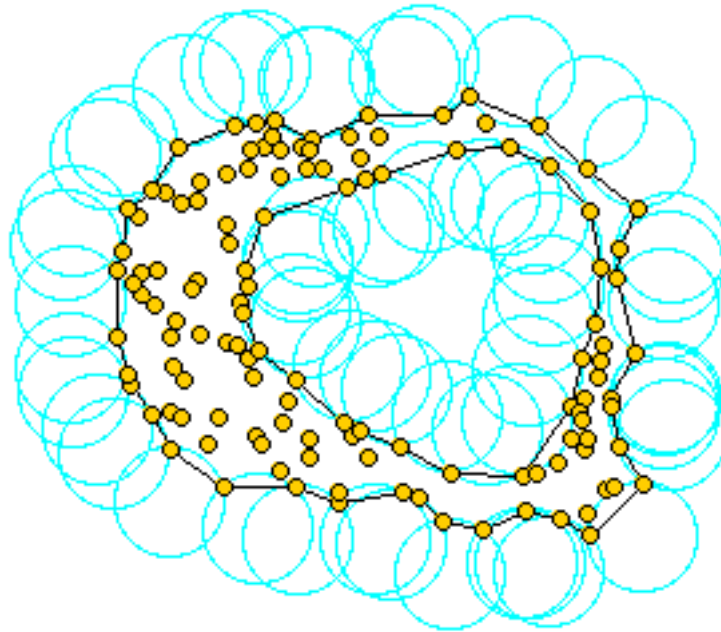


FIGURE 2.9 – Exemple d'application de la méthode Alpha Shape sur un ensemble de points 2D

C. Critères de sélection des sphères du graphe

a. Visibilité

Le premier objectif est de maximiser la visibilité V de l'environnement étudié. Nous définissons la visibilité d'un point 3D c dans les zones de navigation Γ comme l'ensemble des sommets de Σ visibles à partir de ce point. Géométriquement, le point p est dit visible à partir de c si le segment pc ne croise aucun obstacle ni aucune arête de Σ . Nous définissons la visibilité V comme suit :

$$V(c) = \{p \in \text{sommets de } \Sigma \text{ telque } p \text{ est visible depuis } c\} \quad (2.8)$$

Nous définissons $\#V$ par la cardinalité de l'ensemble V .

Dans la littérature, la visibilité dans un polygone a été traitée par un problème connu par le nom de la galerie d'art. Ce problème a été présenté pour la première fois sous la forme d'une question d'un peintre allemand Paul Klee. La question est : combien de gardes avons-nous besoin pour surveiller une galerie d'art ?

La galerie d'art sera représentée à l'aide d'un polygone 2D. Ce problème consiste à trouver dans ce polygone le nombre optimal et les positions de gardes nécessaires pour visualiser la totalité du polygone. La fig 2.10 montre un exemple de résolution de ce problème de visibilité. On peut voir dans cet exemple qu'il faut quatre caméras pour avoir une visibilité totale de la galerie d'art.

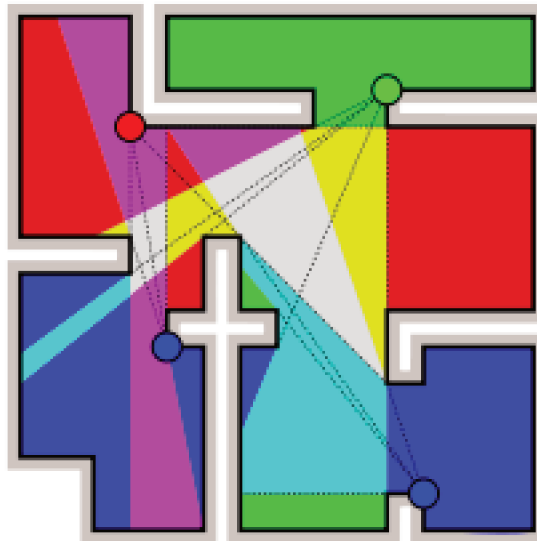


FIGURE 2.10 – Exemple d'un problème de la galerie d'art : quatre caméras couvrent cette galerie.

Une variété de ce problème est connue sous le nom de "Watchman route". Ce problème consiste à trouver un chemin à l'intérieur d'un polygone Γ afin que chaque point de Γ soit visible à partir d'un certain point de ce chemin. Plusieurs algorithmes ont été proposés pour calculer la Watchman route en prenant comme entrée un ensemble de points cibles à visiter [88, 18, 28]. Une autre méthode connue sous le nom de problème de Zookeeper a été proposée dans [54]. Étant donné qu'un simple polygone (le zoo) contient un ensemble de k polygones disjoints (les cages) convexes, cette solution consiste à trouver le chemin le plus court à l'intérieur du zoo qui touche toutes les cages sans y pénétrer. La majorité de tous ces algorithmes recherche le chemin optimal qui minimise la distance totale parcourue tout restant à l'intérieur d'un polygone. La fig 2.11 montre un exemple de Watchman route le plus court dans un polygone.

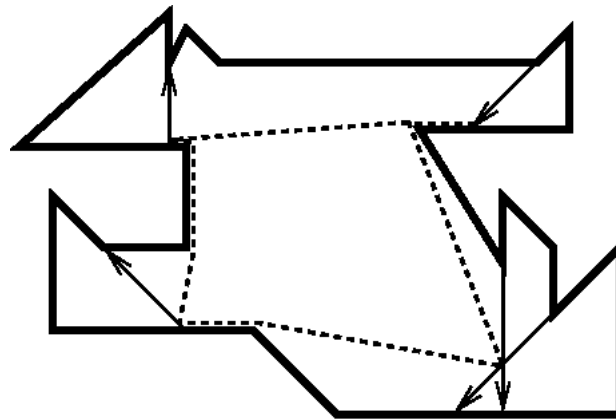


FIGURE 2.11 – Un exemple de polygone avec le Watchman route le plus court

b. Entropie

Le deuxième critère utilisé pour sélectionner l'emplacement des sphères dans le graphe est l'entropie E . Ce critère exprime la quantité d'informations saillantes visibles d'un point de vue. Nous définissons l'entropie d'une sphère comme la quantité d'informations significatives projetées sur celle-ci. En utilisant cette entropie, nous sommes en mesure de sélectionner le point de vue optimal qui représente le mieux possible une carte 3D. L'idée est de calculer le nombre de points saillants de l'image sphérique. Pour décider de la saillance du point 3D sur la sphère, nous proposons de définir deux niveaux de saillance. La saillance de bas niveau basée sur les caractéristiques de bas niveau (photométrique et géométrique) d'un point 3D. La saillance de haut niveau est basée sur les informations sémantiques de chaque point 3D. Dans notre processus de synthèse, les points étiquetés "bâtiments" sont considérés comme les points les plus saillants de la localisation. En utilisant ces

deux types de saillance, nous calculons le nombre de points d'intérêt sur la sphère en fonction de leur pertinence. Par conséquent, nous avons quatre combinaisons possibles comme suit :

- n_{00} : nombre de points non-pertinents sémantiquement, photométriquement et géométriquement.
- n_{10} : nombre de points pertinents uniquement par photométrie et géométrie.
- n_{01} : nombre de points pertinents uniquement sémantiquement.
- n_{11} : nombre de points pertinents sémantiquement, photométriquement et géométriquement.

L'entropie d'une sphère sera caractérisée par l'entropie de son centre c . L'entropie est donnée par l'équation :

$$E(c) = -\alpha_i \sum_{i,j=0,1} \frac{n_{ij}}{h} \log \frac{n_{ij}}{h} \quad (2.9)$$

Les α_i sont des poids attribués à chacune des quatre combinaisons. Dans la suite, nous avons utilisé des poids unitaires. Nous avons testé l'évolution de la valeur d'entropie dans une rue qui comporte un carrefour routier. Nous avons calculé la valeur d'entropie dans le nuage de points 3D représentant la rue. La fig 2.12 montre les résultats obtenus. Nous constatons que l'entropie atteint sa valeur maximale au centre du rond point. Cela est dû au fait qu'au centre du carrefour la visibilité ainsi que le nombre des pixels pertinents augmentent.

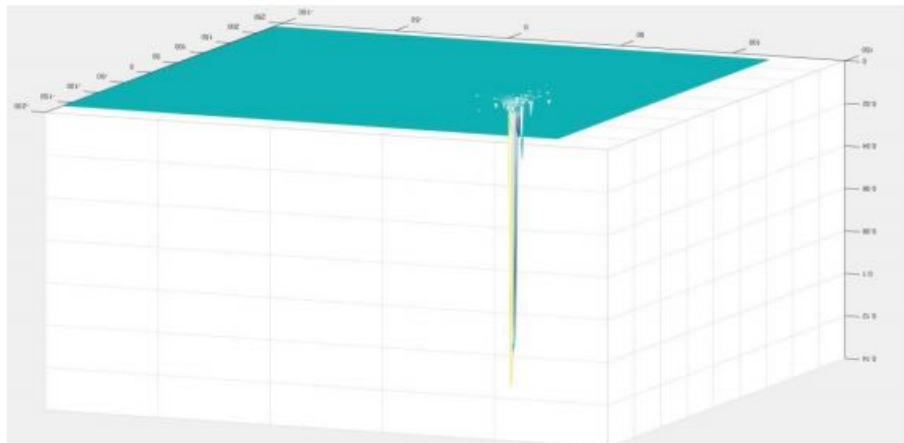


FIGURE 2.12 – Évolution de l'Entropie E en se déplaçant dans une rue contenant un carrefour routier

D. Critères d'évaluation

.1 Localisabilité Pour évaluer la qualité d'une carte et estimer ses performances lors de la localisation, plusieurs solutions ont été proposées. Dans [80], les auteurs proposent de prédire les performances de localisation dans le contexte de la cartographie visuelle grâce à un score estimé pour chaque position v en fonction de la structure de la carte. Ce score est calculé en comptant le nombre de points d'intérêt visibles de v et de ses voisins les plus proches. Les auteurs ont montré que le score fourni par leur méthode est fortement corrélé à la performance réelle de la localisation. La solution proposée par Zhen et al. [131] consiste à calculer un critère appelé localisabilité qui traduit la possibilité de localisation. Ce critère est calculé à partir des caractéristiques géométriques dans chaque direction pour chaque point de la carte. Pour estimer la localisabilité d'une position donnée, ils déterminent l'ensemble des points visibles à partir de cette position. Ensuite, ils estiment les normales à la surface pour chaque point visible. Après cela, ils accumulent les normales dans une matrice N qui décrit l'ensemble des contraintes observables de la position donnée.

$$N = \begin{bmatrix} n_{1x} & n_{1y} & n_{1z} \\ n_{2x} & n_{2y} & n_{2z} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ n_{kx} & n_{ky} & n_{kz} \end{bmatrix}$$

Dans [131], ils ont montré que, pour localiser précisément un robot, nous devons être en mesure de contraindre correctement sa pose dans les trois dimensions. Ils ont proposé d'examiner les valeurs singulières de N . Théoriquement, l'agent devrait pouvoir se localiser tant que les trois valeurs singulières sont non nulles. En pratique, ils ont calculé la possibilité de localisation L comme valeur singulière minimale de N :

$$L = \min(\text{diag}(\Sigma)) \quad (2.10)$$

Pour tester ce critère, nous avons sélectionné plusieurs positions sur une trajectoire, puis nous avons calculé la localisabilité pour chaque sphère dans chaque position. Toutes les sphères sont dans la même référence (x, y, z) et ont la même hauteur $z = 1,5m$ correspondant à la hauteur moyenne de l'agent. La fig 2.13 montre l'évolution de la possibilité de localisation en se déplaçant le long de l'axe des x . Nous constatons que la localisation est meilleure lorsque l'image sphérique contient moins de zones vides. Ces zones vides sont les zones invisibles de ce point de vue.

Ce critère reflète donc la quantité de contraintes géométriques dans une position donnée. Si ce point de vue permet une visualisation suffisante des contraintes géométriques, il sera facilement localisable. Dans notre cas, il faut que chaque sphère du graphe optimal X ait une bonne valeur de localisabilité.

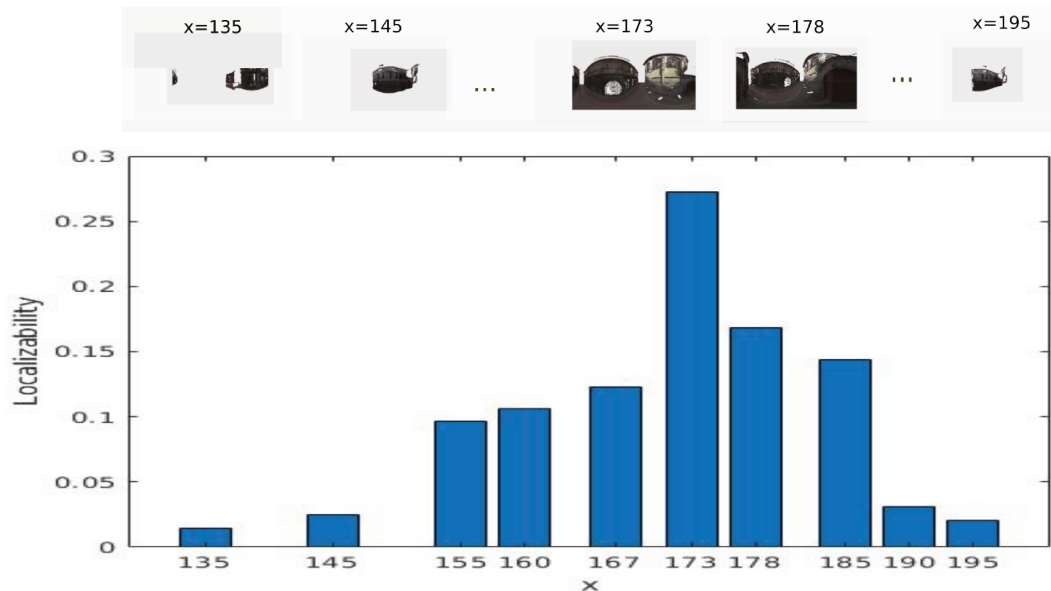


FIGURE 2.13 – Évolution de la localisabilité en se déplaçant le long de l’axe des x

.2 Taux de compression Pour évaluer le gain en mémoire, nous avons proposé de calculer le taux de compression (TC) exprimé en pourcentage. Il est défini comme suit :

$$TC = \left(1 - \frac{T_i}{T_f}\right) * 100 \quad (2.11)$$

T_i est la taille de la carte 3D initiale. T_f est la taille du graphe de navigabilité optimal. Ce graphe est le résultat du processus de résumé de la carte initiale.

E. Algorithmes d’optimisation

Nous formulons ce processus de résumé d’un modèle 3D sous la forme d’un problème d’optimisation à deux objectifs. Nous avons proposé deux stratégies d’optimisation de ces deux critères. La première solution consiste à optimiser les deux objectifs simultanément alors que la deuxième solution consiste à les optimiser d’une façon séquentielle. Nous détaillerons par la suite les deux méthodes d’optimisation.

a. Optimisation séquentielle

Dans cette partie, nous détaillerons l'optimisation séquentielle de deux critères Entropie et Visibilité afin de résumer le nuage de points d'entrée ξ .

Le résultat de ce processus sera un ensemble minimal de sphères $g_n = \{g_j, j = 1..n\}$ permettant de résumer la carte 3D initialement volumineuse. Dans la littérature, plusieurs méthodes d'optimisation ont été proposées. Une des approches proposées pour résoudre des problèmes multicritères est l'approche lexicographique [35]. Cette méthode consiste à optimiser les objectifs séparément et séquentiellement et génère une solution optimale à chaque étape d'optimisation séquentielle. Nous avons appliqué cette approche pour optimiser les critères suivants :

- Maximiser la Visibilité
- Maximiser l'Entropie
- Minimiser le nombre de sphères

Chaque critère est présenté par une fonction objective. Les fonctions objectives sont initialement classées par ordre d'importance. Premièrement, nous devrions garantir que la scène entière a été visitée lors du choix des meilleurs points de vue en maximisant la visibilité. Ensuite, seules les sphères ayant une valeur d'entropie supérieure à un certain seuil seront sélectionnées. Enfin, nous minimisons le nombre de sphères afin d'éliminer la redondance d'information.

La fig 2.14 montre les différentes étapes du processus d'optimisation.

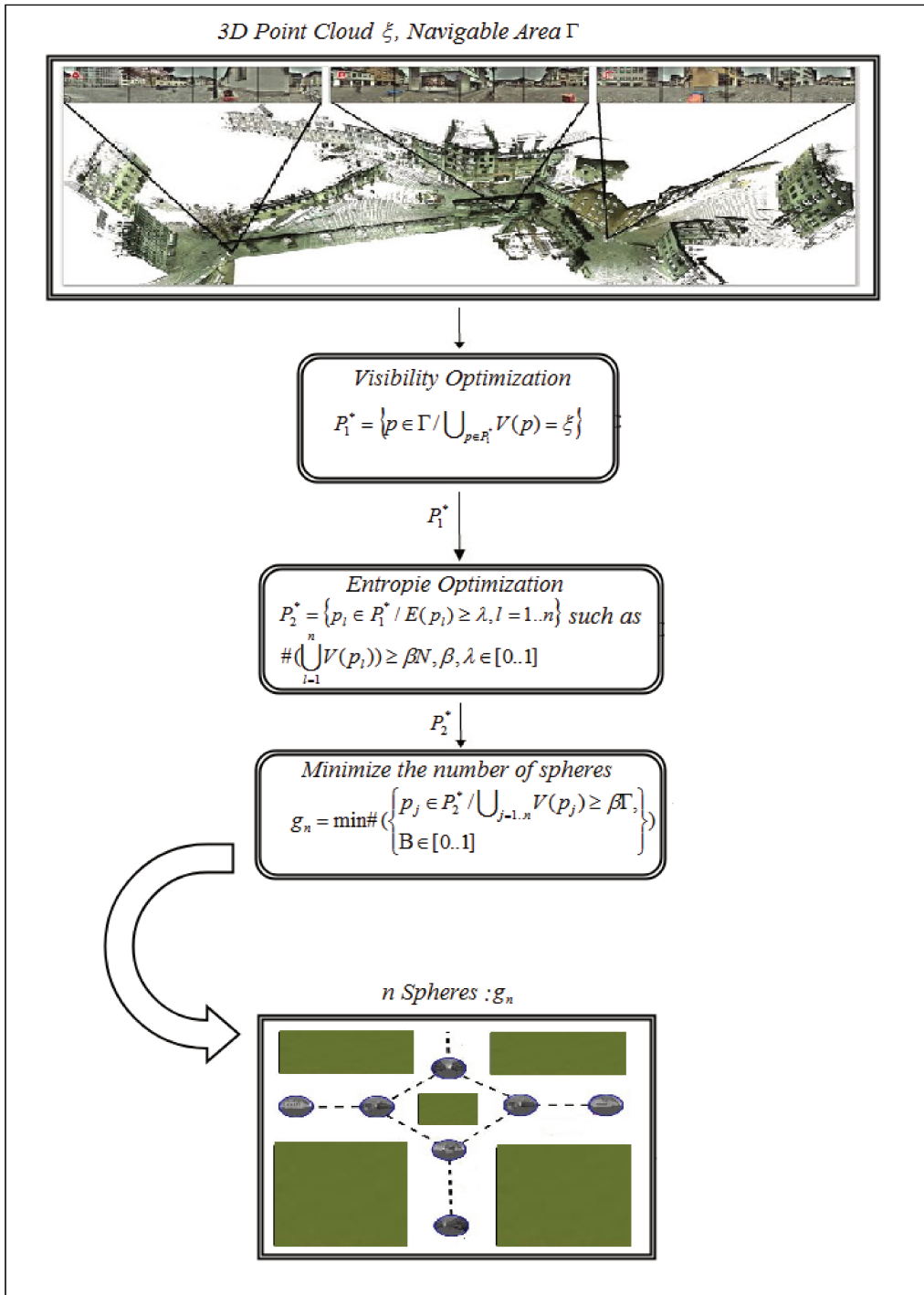


FIGURE 2.14 – Optimisation séquentielle

a.1 Optimisation de la visibilité

La première étape du processus d’optimisation consiste à maximiser la visibilité. Pour ce faire, nous devons garantir que chaque point du nuage est visible depuis au moins un point de vue. Notre espace de recherche des meilleurs points de vue est réduit à la zone navigable. Comme nous avons expliqué auparavant, la zone de visibilité est représentée par un polygone 2D Γ . À chaque itération, un point de vue P appartenant au polygone Γ est sélectionné. La visibilité à partir de ce point de vue est alors calculée. Ce processus de sélection des meilleurs points de vue (\mathcal{P}_1^*) sera répété jusqu’à l’obtention d’une visibilité globale de la scène. L’équation 2.12 montre la première fonction objective f_1 qui permet l’optimisation de la Visibilité.

$$f_1 : \begin{cases} \mathcal{P}_1^* = \{P \in \Gamma\}, \\ \text{tel que} \\ \cup_{P \in \mathcal{P}_1^*} V(P) = \xi \end{cases} \quad (2.12)$$

Le résultat de cette première étape d’optimisation sera le chemin \mathcal{P}_1^* construit à partir d’un ensemble de points garantissant une visibilité maximale de l’environnement étudié. En effet, chaque point de ξ est visible par au moins un point de \mathcal{P}_1^* . Pour trouver les points constituant le chemin \mathcal{P}_1^* , nous proposons d’adapter le problème de la galerie d’art expliqué précédemment. Nous pouvons considérer ce chemin comme un Watchman route. Dans un premier lieu, nous devons assurer la visibilité des arêtes de polygone de navigabilité. Pour ce faire, nous proposons de décomposer le polygone de navigabilité en k polygones convexes. Pour simplifier le problème, nous considérons que la visibilité à travers les arêtes extérieures des polygones est possible et nous considérons que les arêtes intérieures des polygones sont des obstacles.

Pour visualiser la totalité d’une zone convexe, un seul point de vue (le centre) suffit. En se basant sur cette règle, le chemin sera construit à partir des points de vue P_k sélectionnés dans chaque zone convexe. Plusieurs algorithmes ont été proposés pour décomposer un polygone en plusieurs zones convexes. L’algorithme de Keil [58, 57] consiste à décomposer un polygone en plusieurs sous-polygones convexes en éliminant tous les sommets réflexes. Dans un polygone, un sommet est appelé ”convexe” si l’angle formé par les deux arêtes au sommet, est inférieur à π radians. Dans le cas contraire, cela s’appelle ”réflexe”.

L’algorithme de Keil fonctionne en examinant tous les sommets pour supprimer ceux qui sont réflexes. Nous suggérons l’utilisation de l’algorithme de Mark Bayazit [9], qui est une version optimisée de l’algorithme Keil. Les meilleurs points de vue sélectionnés sont les centres des sous-polygones convexes. En reliant chaque centre de sous-polygone au centre le plus proche, nous obtiendrons le chemin optimal

\mathcal{P}_1^* . Ce chemin permet de garantir la couverture totale de l'environnement étudié. La fig 2.15 montre un exemple de décomposition de polygone en plusieurs zones convexes et le Watchman route obtenu.

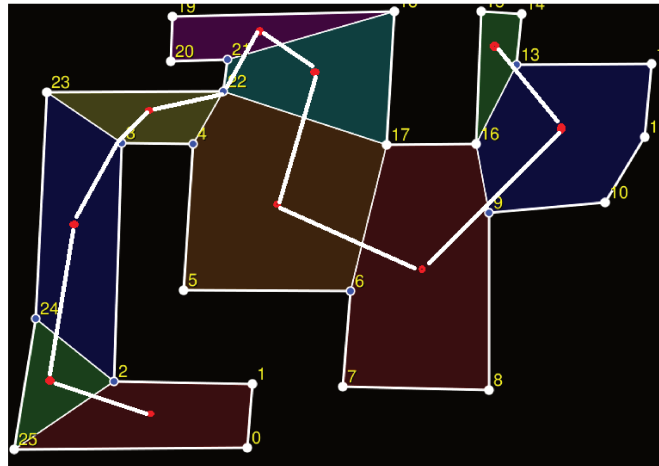


FIGURE 2.15 – Calcul de Watchman route dans un polygone.

Ce Watchman route \mathcal{P}_1^* devient une nouvelle contrainte pour l'optimisation de l'Entropie E .

a.2 Optimisation de l'Entropie

Le Watchman route \mathcal{P}_1^* calculé dans l'étape précédente, sera considéré comme l'espace de recherche dans la nouvelle étape d'optimisation. L'optimisation d'entropie consiste à discrétiser \mathcal{P}_1^* pour sélectionner les points qui maximisent l'entropie E . Cette discrétisation est effectuée tout en veillant à ce que la majorité des zones de la scène soient toujours visibles. L'optimisation de l'entropie E revient à résoudre l'équation 2.13 suivante :

$$f_2 : \begin{cases} \mathcal{P}_2^* = \{P_l \in \mathcal{P}_1^* , l = 1..n / E(P_l) \geq \lambda \quad \lambda \in [0..1]\} & \text{tel que} \\ \#(\cup_{l=1}^n V(P_l)) \geq \beta.N, \beta \in [0..1] \end{cases} \quad (2.13)$$

\mathcal{P}_2^* est un ensemble de points (centres de sphères) permettant de trouver un compromis entre les deux objectifs Visibilité et Entropie. Les deux paramètres λ et β sont utilisés pour la régulation de deux critères Entropie et Visibilité. Afin de simplifier le traitement des nuages des points à grande échelle, nous proposons de décomposer le nuage de départ en plusieurs sous-nuages. Pour chaque point dans \mathcal{P}_2^* , nous définissons une fenêtre de recherche cylindrique de rayon r . Chaque fenêtre est un sous-nuage de ξ contenant une partie \mathcal{P}_1^* du Watchman route. L'optimisation de l'entropie est effectuée dans chaque sous-nuage sur \mathcal{P}_1^* . Dans chaque

sous-nuage, nous cherchons à trouver sur π l'ensemble des sphères qui maximisent E tout en gardant une visibilité maximale de la scène. Cette visibilité globale est l'union de toutes les visibilités obtenues dans chaque sous-nuage. Pour discrétiser le sous chemin de \mathcal{P}_1^* , nous avons évalué l'entropie de ses extrémités ainsi que son milieu.

Nous gardons la moitié ayant une entropie supérieure. Cette étape sera répétée jusqu'à obtenir un point ayant une valeur d'entropie supérieure au seuil λ . La fig 2.16 montre un exemple de ce processus de discrétisation. Dans un premier temps, l'entropie des sommets colorés en noir et de milieu de chaque paire de sommets consécutifs est calculée. Sur deux sommets consécutifs et leurs centres colorés en bleu, nous ne conservons que deux points ayant la plus grande entropie. Pour chaque sous-nuage, nous sélectionnons les sommets avec une entropie supérieure à λ . À chaque itération, nous calculons le nombre de points visibles à partir des sommets sélectionnés, colorés en violet. Seules les sphères dont l'union de leur visibilité est supérieure à βN sont sélectionnées. La fig 2.16 montre les différentes étapes de ce processus d'optimisation.

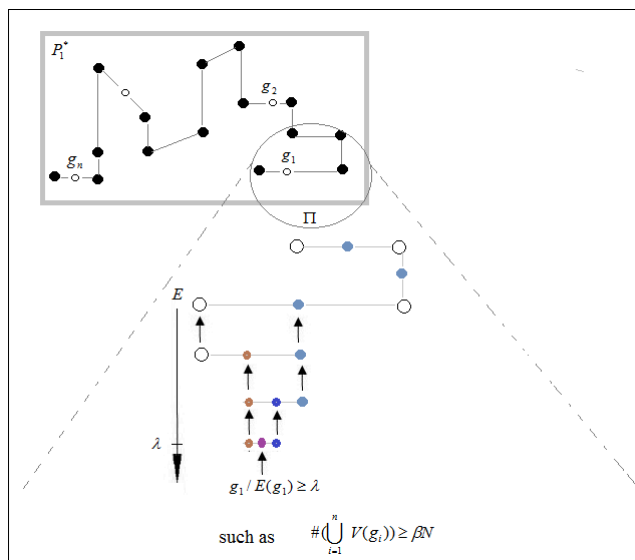


FIGURE 2.16 – Discrétisation et optimisation de l'Entropie sur un sous chemin du Watchman route

La dernière étape de l'optimisation est la minimisation de la taille n de \mathcal{P}_2^* qui représente le nombre de sphères dans g_n .

a.3 Minimisation du nombre de sphères

Dans cette section, nous présentons la dernière étape de notre algorithme per-

mettant de minimiser la taille n de \mathcal{P}_2^* . Le résultat de ce processus d'optimisation est l'ensemble optimal des sphères g_n d'une taille minimale n , ce qui permet une bonne visibilité de la scène complète et permet de capturer le nombre maximal de points saillants. Pour ce faire, nous avons proposé de calculer la similarité entre les sphères voisines pour éliminer les informations redondantes. La similarité entre deux sphères est calculée à l'aide de leur corrélation statistique. Cette corrélation exprime le taux de variation des informations photométriques, sémantiques et géométriques entre les sphères comparées. Dans [78], ce paramètre est calculé à l'aide de l'opérateur MAD (median absolute deviation) qui représente la différence (erreur) d'intensité entre les deux sphères S_{phot} . Théoriquement, deux sphères sont considérées comme similaires si la valeur de MAD est inférieure à un seuil σ . De la même manière, nous proposons d'ajouter une mesure sémantique de similarité S_{sem} entre deux sphères. Cette mesure consiste à calculer le nombre de pixels similaires portant les mêmes étiquettes. Nous avons également ajouté une mesure de similarité géométrique à l'aide du descripteur géométrique FPFH [98] (Annexe A). En comparant les histogrammes de chaque sphère, nous obtenons une mesure de similarité géométrique S_{geom} . Ces similarités sont calculées comme suit :

$$f_3 : \begin{cases} S_{phot} = \text{med}(|p(x) - \text{med}(p(x))|) \\ S_{geom} = \text{med}(|g(x) - \text{med}(g(x))|) \\ S_{sem} = \sum s(x) \\ S = \text{mean}(S_{phot} + S_{geom} + S_{sem}) \end{cases} \quad (2.14)$$

Dans ces équations, $p(x)$ et $g(x)$ sont respectivement les vecteurs contenant les erreurs d'intensité et géométriques. $s(x)$ est le vecteur indiquant la différence sémantique entre les deux images sphériques. Pour chaque pixel, une valeur binaire est stockée dans $s(x)$. Cette valeur est égale à 0 si les deux labels du même pixel dans les deux sphères sont identiques et 1 dans le cas contraire. Les similarités S_{phot} , S_{sem} et S_{geom} seront ensuite transformées en pourcentage puis additionnées pour fournir une seule valeur de similarité S . L'expression du problème de la minimisation du nombre de sphères est la suivante :

$$f_3 : \{ \min\#(\{P \in \mathcal{P}_2^* \mid \cup_P V(P) \geq \beta \cdot \Gamma, \beta \in [0..1]\}) \} \quad (2.15)$$

En supprimant les sphères similaires qui ont la même visibilité, nous obtiendrons un ensemble optimal de sphères. Nous détaillerons par la suite les résultats obtenus avec cet algorithme d'optimisation.

a.4 Résultats

Pour tester cette méthode d'optimisation, nous avons choisi un nuage de points

à grande échelle. Ce nuage contient plus de 50 millions de points 3D labellisés et représente une trajectoire de 80 *m*. Dans cet ensemble de données, nous avons 8 classes sémantiques : {1 : terrain artificiel, 2 : terrain naturel, 3 : végétation haute, 4 : végétation basse, 5 : bâtiments, 6 : paysage, 7 : artefacts, 8 : voitures }. Une étiquette supplémentaire {0 : points non étiquetés } est ajoutée pour les points qui n'ont pas été classés. Dans notre processus de synthèse, les points étiquetés "bâtiments" sont considérés comme les points les plus saillants et pertinents pour la localisation. Cet ensemble de données a été utilisé pour évaluer les performances de notre solution en utilisant toutes ses caractéristiques sémantiques, photométriques et géométriques. La fig 2.20 montre la décomposition du polygone de zones navigables en plusieurs polygones convexes. Chaque couleur représente un polygone différent. Le résultat de ce processus de positionnement des sphères est un ensemble compact de 20 images sphériques. La distance moyenne entre toutes les sphères est environ de 4*m*. Nous avons obtenu un taux de compression de 90 % pour cette carte.

Nous avons également testé notre algorithme sur notre propre base de données, qui décrit une partie de la ville de Rouen. Il couvre une trajectoire de 200*m* d'une zone urbaine (fig 2.18). Cette base de données est un nuage 3D texturé et étiqueté contenant plus de 7 millions de points 3D. Dans cet ensemble de données, nous avons 15 classes sémantiques : {0 : Divers, 1 : Route, 2 : Marquages de route, 3 : Trottoir de terrain, 4 : Bâtiment, 5 : Végétation, 6 : Arbre, 7 : Piéton, 8 : Voitures, 9 : Vélo, 10 : Moto, 11 : Feu de signalisation, 12 : Panneau de signalisation, 13 : pôle, 14 : ciel }. Chaque classe est représentée avec un code de couleur comme indiqué dans la fig 2.18. Le processus d'optimisation permet de résumer la carte initiale sous la forme de 54 images sphériques avec une distance moyenne entre toutes les sphères de 3,7 *m*. La fig 2.19 montre un exemple d'images sphériques RGB-D-L. Le taux de compression obtenu est de 98 %.

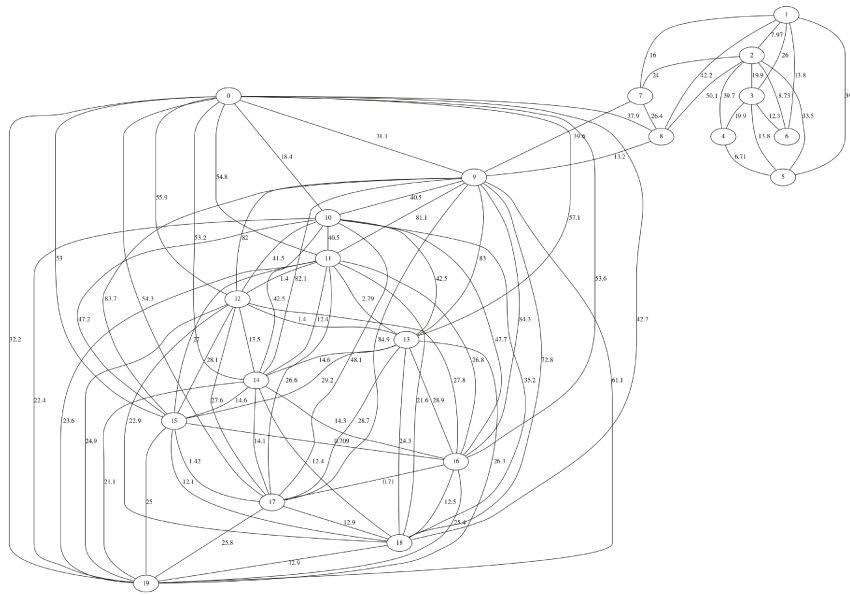


FIGURE 2.17 – Graphe de navigabilité : chaque nœud est une image sphérique et chaque arc représente la distance entre les sphères.

Nous avons comparé le taux de compression obtenu avec celui obtenu par Meiland et al. dans [78]. Nous avons constaté une amélioration de 2 % du taux de compression grâce au positionnement optimal des points de vue sphériques dans la scène. La dernière étape consiste à créer le graphe de navigabilité en reliant les sphères du polygone de navigabilité. Chaque arête est pondérée par la distance entre les deux sphères correspondantes. La fig 2.17 montre un exemple de ce graphe. Il peut être utilisé dans plusieurs applications telles que la navigation ou la localisation de robot mobile grâce à sa taille minimale et à la pertinence des données qu'il contient.

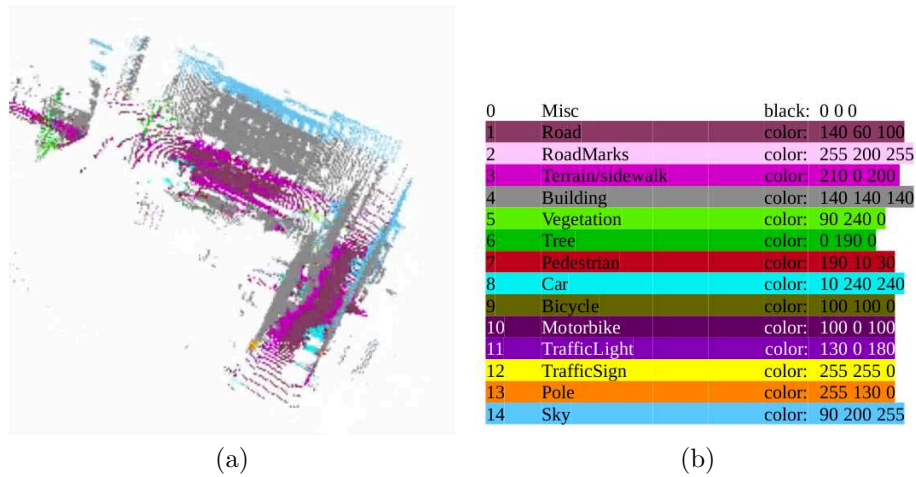


FIGURE 2.18 – Un nuage de points 3D de la ville de Rouen.(a) Nuage de points 3D sémantisé (b) Segmentation sémantique : 15 classes sémantiques

Toutes les sphères sont positionnées de manière à capturer le maximum de points possibles appartenant aux façades des bâtiments. Pour le premier jeu de données, l'ensemble des sphères obtenu permet de visualiser 80 % de la scène ($\beta = 0.8$). Le choix de β a un impact sur le taux de compression. En fait, fixer β à 1 favorisera la visibilité par rapport à l'entropie. Ainsi, le nombre de sphères sélectionnées augmentera pour couvrir l'ensemble de l'environnement. Une valeur de β supérieure ou égale à 0.8 permet de garantir un compromis entre la visibilité et le taux de compression.

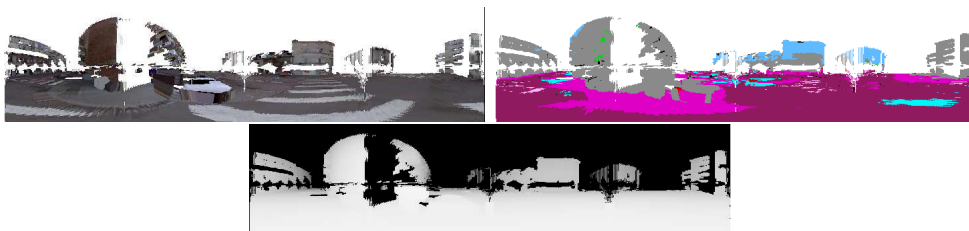


FIGURE 2.19 – Image sphérique RGBD-L. La sphère RGB contient l'information photométrique. La sphère D contient l'information profondeur. La sphère L contient l'information sémantique

Le choix du λ a un impact sur la pertinence des points 3D dans la carte de synthèse et par conséquent sur le rappel et la précision. En effet, si λ est fixé à 0.5, au moins la moitié des points seront saillants. Par conséquent, nous avons choisi $\lambda = 0.7$. Pour le deuxième jeu de données, nous avons calculé la localisabilité de chaque

image sphérique, telle que définie dans [131], toutes les sphères sont facilement localisables, car elles ont une valeur élevée de localisabilité (en moyenne 0.6). Cela garantit une bonne localisation dans le graphe de la sphère.

a.5 Conclusion

Nous avons détaillé la première stratégie d’optimisation appelée optimisation séquentielle. Cette première solution consiste à optimiser les deux objectifs simultanément : visibilité et entropie. L’algorithme développé permet de résumer efficacement un nuage de points à grande échelle. Le processus de synthèse est basé sur l’extraction de plusieurs vues sphériques représentant les sous-nuages –de la carte initiale. Cette représentation sphérique contient des informations sémantiques, photométriques et géométriques. Cette nouvelle méthode de synthèse de cartes 3D nous permet de faciliter plusieurs tâches de navigation dans les systèmes de transport intelligents (localisation, planification d’itinéraire, évitement d’obstacles, ...) en réduisant considérablement la taille de la mémoire requise.

Cependant, cette solution présente des limitations. En effet, jusqu’à maintenant, nous avons considéré que la zone de navigabilité permet de visualiser la totalité de la zone visible. Il serait intéressant de considérer que la zone de visibilité est différente de la zone de navigabilité. Dans ce cas, il serait intéressant d’adapter l’optimisation de la visibilité.

L’étape suivante consiste à tester une autre stratégie d’optimisation. Elle sera appliquée sur un autre type de carte 3D. La première stratégie a été appliquée sur des nuages de points 3D labellisés tandis que la deuxième stratégie sera appliquée sur des maillages texturés et sémantisés.

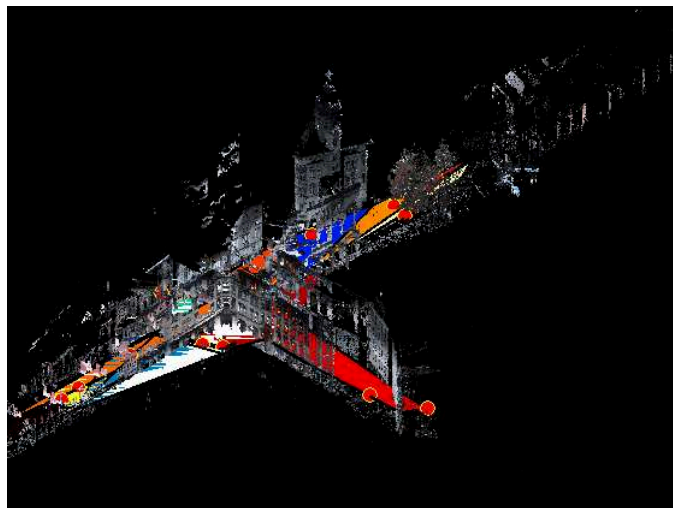


FIGURE 2.20 – Extraction de polygones et positionnement de sphères

b. Optimisation multiobjectif

Cette méthode consiste à optimiser simultanément les deux critères : Visibilité et Entropie. L'algorithme 1 permet de calculer l'ensemble optimal d'images sphériques $X = \{X_i, i = 1..n\}$. Pour ce faire, nous allons définir l'espace de recherche de meilleurs points de vue comme étant un graphe appelé graphe de barycentres. Dans cette section, nous détaillerons le principe de ce graphe de barycentres ainsi que les étapes du processus d'optimisation multiobjectif.

Algorithme 1 Construction du graphe de sphères $X = \{X_i, i = 1..n\}$

Calculer le graphe de sphères $X = \{X_i, i = 1..n\}$

Entrée:

$T = (C, \gamma)$ graphe des barycentres

$\Sigma = \{V_j, j = 1..k\}$ V_i sommets de Σ

Point de départ $C_s \in C$

Point d'arrivée $C_e \in C$

Sortie:

Marquer tous les sommets dans Σ comme non couverts

Marquer tous les sommets dans Σ visibles à partir de C_s

Marquer tous les sommets dans Σ visibles à partir de C_e

$X \leftarrow \{C_s, C_e\}$

$q \leftarrow C_s$

tant que $\exists x$ comme non couvert dans Σ **faire**

$\Pi(\gamma_k) =$ Discrétisation des arêtes γ_k

$p \leftarrow$ premier point $\Pi(\gamma_k)$ à droite de q tel que p couvre certains points non couverts $x \in \Sigma$ et $E(p) \geq \alpha$

$X \leftarrow X \cup \{q\}$

Marquer tous les points $\in x$ comme couvert

$q \leftarrow p$

si($q \in C$)

$\gamma_k \leftarrow$ arête dans γ dont son point de départ est q

fin tant que

return X

b.1 Graphe des barycentres

Le but principal de notre processus de synthèse est de réduire la taille d'une carte 3D tout en maintenant une vue globale de la scène. Ainsi, l'ensemble optimal d'images sphériques doit garantir une visibilité maximale. Toutes les sphères doivent être placées sur les zones navigables comme les rues pour les véhicules.

Pour simplifier l'espace de recherche, nous allons positionner ces sphères sur un graphe de départ appelé graphe des barycentres. Sur ce graphe $T = (C, \gamma)$, nous placerons nos images sphériques. C est l'ensemble des nœuds reliés par un ensemble des arcs γ . Les nœuds de ce graphe seront un ensemble de points de vue permettant de visualiser la totalité de la scène. Tous ces points de vue seront placés sur la zone navigable. Pour avoir une modélisation plus réaliste de l'environnement, nous considérons que la zone de navigabilité est incluse dans la zone de visibilité. Le polygone de visibilité Σ englobera le polygone de navigabilité Γ . Dans ce cas, nous devons trouver un chemin dans le polygone de navigabilité qui assure la visibilité totale du polygone de visibilité.

Pour ce faire, le polygone de visibilité Σ est décomposé en plusieurs zones convexes. Puisque le polygone de navigabilité est inclus dans le polygone de visibilité, nous allons calculer les polygones d'intersection (inter-polygones) entre chaque polygone convexe de Σ et le polygone de navigabilité. La fig 2.21 montre un exemple de la décomposition du polygone de visibilité en plusieurs polygones convexes et les intersections avec le polygone de navigabilité. Ce graphe sera construit à partir des points de vue C_k qui sont les centres des inter-polygones. Par la suite, tous les nœuds de T seront reliés par un ensemble d'arcs γ en respectant que ces arcs ne coupent pas le polygone de navigabilité. Ce graphe est une version adaptée de Watchman route qui traite la possibilité d'avoir une zone de navigabilité différente de la zone visible.

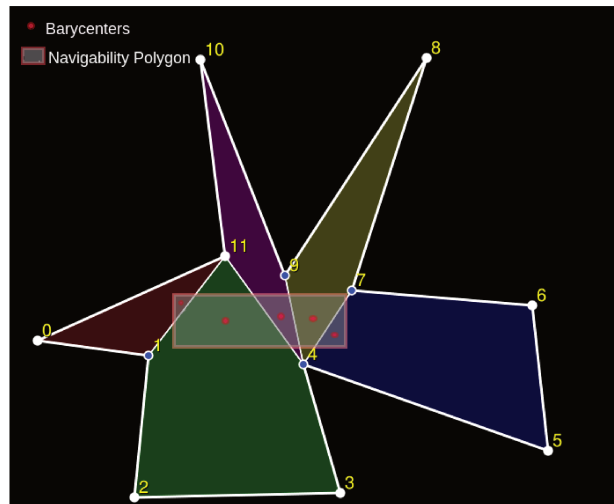


FIGURE 2.21 – Un exemple qui illustre l'intersection de polygones de navigabilité et de visibilité

b.2 Processus d'optimisation

Cette étape consiste à sélectionner sur le graphe de barycentre calculé précédemment T , un ensemble de points de vue qui maximisent l'entropie E et la Visibilité V . Comme indiqué précédemment dans l'algorithme 1, les sphères peuvent être placées sur les nœuds du graphe de départ T ou sur ses arêtes. En parcourant le graphe de barycentres T , nous sélectionnerons les meilleurs points de vue répondant aux critères d'entropie et de visibilité. L'algorithme d'optimisation 1 consiste à sélectionner un point de départ et un point d'arrivée sur le graphe initial T . À partir du point de départ, nous sélectionnerons des points de vue avec une entropie supérieure à un seuil α et couvrant tous les sommets du polygone de visibilité. Pour discrétiser les arêtes du graphe de barycentres T , nous proposons une fonction de bijection Π entre deux nœuds consécutifs. Cette fonction permet de discrétiser chaque arête en N points. Pour chaque arête γ_k reliant deux nœuds C_k et C_{k+1} , Π la bijection entre γ_k et l'intervalle $[0 \ 1]$ est définie comme suit :

$$\Pi : \begin{cases} [0 \ 1] \longrightarrow \gamma_k \\ r \longmapsto \Pi(r) \end{cases}$$

$\Pi(0)$ point de départ et $\Pi(1)$ un point d'arrivée de γ_k . Discrétiser γ_k consiste à le décomposer en N points tel que :

$$r_i = \frac{i}{N-1}, \quad i = 0 \quad \text{à} \quad N-1 \quad (2.16)$$

La fig 2.22 montre un exemple de discrétisation des arêtes.

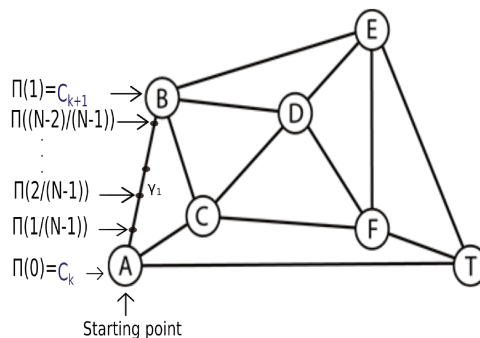


FIGURE 2.22 – Un exemple de discrétisation des arêtes dans un graphe de barycentres

Le processus d'optimisation sera répété jusqu'à couvrir toute la zone de visibilité. À la fin de ce processus, nous obtiendrons un graphe optimal des sphères. Dans la section suivante, nous détaillerons le processus d'évaluation des résultats obtenus.

b.3 Résultats Nous avons appliqué notre méthode sur deux bases de données qui décrivent deux parties de la ville de Rouen. Elles couvrent environ 600 m d'une zone urbaine contenant les mêmes structures que les données décrites précédemment (résultats d'optimisation séquentielle).



FIGURE 2.23 – Maillage 3D à grande échelle représentant une grande zone à Rouen

La première base de données est un maillage 3D texturé, triangulé et labellisé (fig 2.23) et contient plus de 5 millions triangles (faces). Avant de commencer l'étape de positionnement des sphères, une étape de prétraitement des données est effectuée. Cette étape consiste à extraire les polygones de visibilité et de navigabilité. Cette extraction se fait avec la technique d'Alpha Shape expliquée précédemment en utilisant l'information sémantique. Tous les points ayant les labels : route, marquages de route constituent les zones navigables.

La fig 2.24 montre l'extraction des polygones de navigabilité et de visibilité à partir d'un maillage.

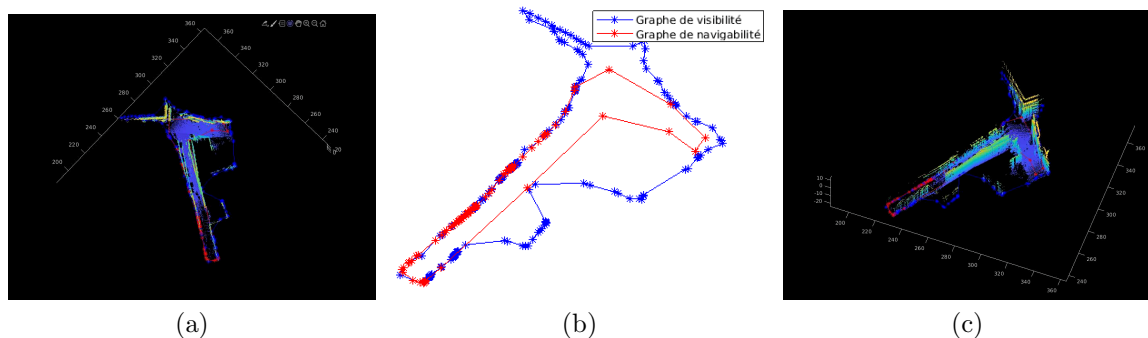


FIGURE 2.24 – Extraction de polygones de navigabilité et de visibilité : (a) et (c) : un nuage de points 3D, (b) : en rouge le graphe de navigabilité et en bleu le graphe de visibilité

Le graphe de barycentres sera calculé sur les deux polygones. Ensuite, l'algorithme d'optimisation multiobjectif est appliqué. La sortie de ce processus d'optimisation, illustrée dans la fig 2.25, est un ensemble compact d'images sphériques RGB-D-L (fig 2.26). Sur cette première base de données, nous avons obtenu 10 sphères RGBD-L résumant le maillage 3D. Nous avons obtenu un taux de compression de 99% de cette carte.

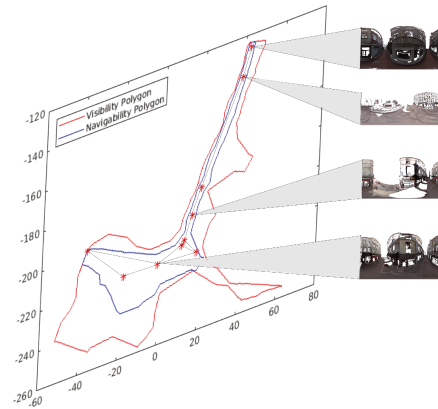


FIGURE 2.25 – Graphe optimal d'image sphérique RGB-D-L

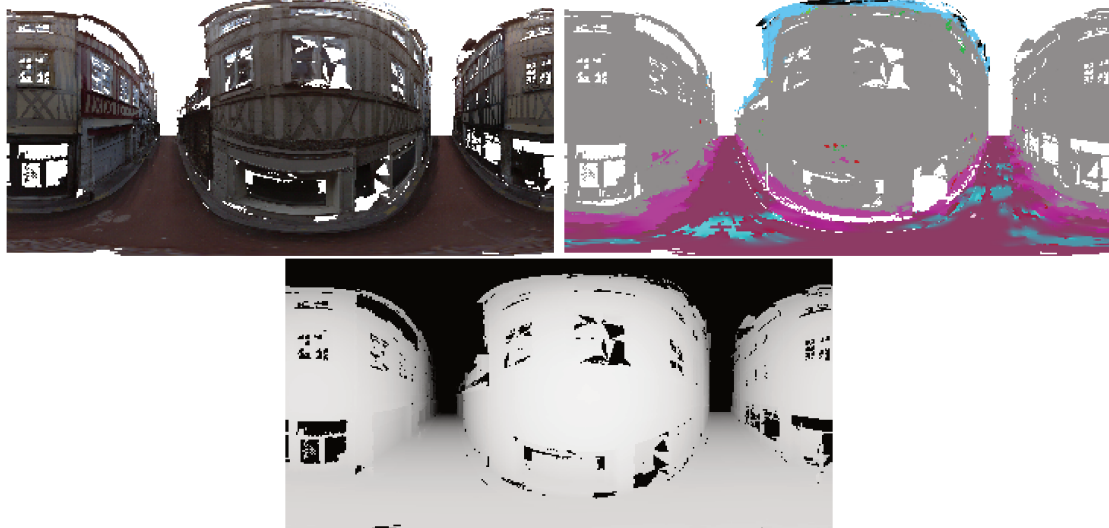


FIGURE 2.26 – Image sphérique RGB-D-L

Nous avons appliqué notre algorithme sur une base de données plus grande qui

contient plus de 30 millions triangles. Il représente une trajectoire de 500m. Fig 2.27 montre la première étape de notre algorithme qui consiste à calculer le graphe des barycentres.

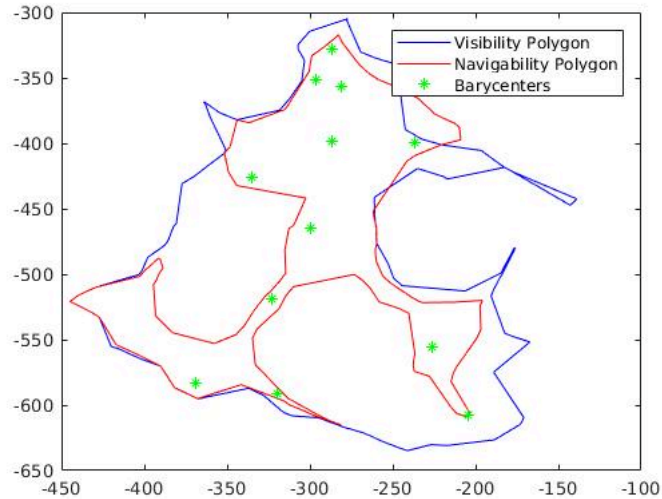


FIGURE 2.27 – Emplacement des nœuds du graphe de barycentres dans le deuxième maillage couvrant une trajectoire de 500 m

Le tableau 2 montre les taux de compression obtenus sur les deux bases de données.

TABLE 2 – Résultats finaux

Bas de données	1	2
Longueur de trajectoire	100m	500m
Nombre de triangles	5 millions	30 millions
Nombre de sphères	10	52
Taux de compression	99%	98.7%

Nous avons obtenu des taux de compression importants, toujours supérieures à 89%. Cependant, ces taux de compression sont fortement liés à la résolution des sphères RGBD-L choisie. Nous avons testé plusieurs résolutions en calculant le taux de compression à chaque fois. Pour ce faire, nous avons choisi 5 trajectoires de longueurs différentes. La fig 2.28 montre les trajectoires choisies.

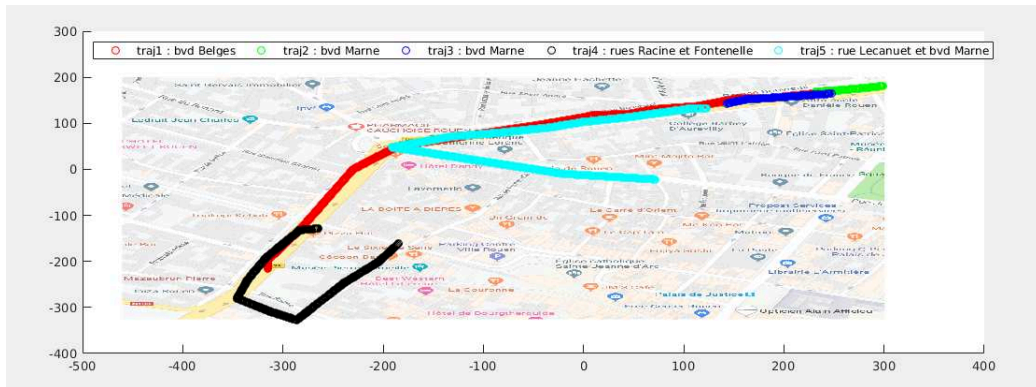


FIGURE 2.28 – Les trajectoires de test à Rouen

Comme il est indiqué dans la fig 2.29, nous pouvons voir l'influence de la résolution sphérique sur le taux de compression final de la carte initiale. Même, en choisissant une très haute résolution le taux de compression reste important (89 %). Pour choisir la bonne résolution, nous avons fait des tests de recalage 3D sur les sphères RGBD-L . Ce processus de recalage sera détaillé dans le chapitre suivant. Nous avons constaté qu'une résolution de 2000*1000 pixels est suffisante pour réussir le recalage et la localisation dans un graphe de sphères.

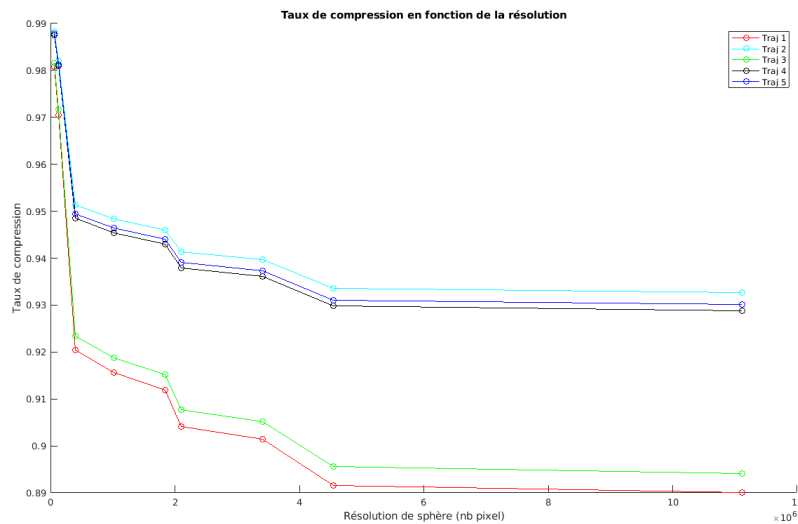


FIGURE 2.29 – Taux de compression en fonction de la résolution de la sphère

L'ensemble final d'images sphériques permet de résumer le maillage 3D initial. La plupart des points visibles depuis ces sphères appartiennent à des points pertinents

tels que les bâtiments et la végétation. En conséquence, nous avons considérablement réduit la taille de la carte. Néanmoins, nous avons réussi à conserver un maximum de points saillants. Pour évaluer notre solution, nous avons calculé la localisabilité de chaque image sphérique, comme nous l'avons définie dans la section précédente. Toutes les sphères ont une valeur de localisabilité entre 0,5 et 0.7. Ces valeurs indiquent que les sphères ont des caractéristiques géométriques permettant la localisation de ces sphères. Cela garantit une bonne localisation dans le graphe de navigabilité. Notre algorithme garantit une visibilité et une entropie maximales de l'environnement étudié, permettant ainsi une navigation précise et rapide. Réduire la taille de la carte initiale jusqu'à 1% permet de gagner en terme de mémoire et de temps de calcul. En effet, nous pouvons facilement intégrer ce graphe minimal sur un système de navigation à faibles ressources ou l'envoyer sur un réseau d'un serveur à un agent.

4. Conclusion

Le nouveau critère de sélection des emplacements des sphères augmentées RGBD-L proposé ici permet de résumer une carte 3D initialement volumineuse. Contrairement aux méthodes existantes, la technique développée utilise conjointement les informations photométriques, géométriques et sémantiques associées à chaque pixel. La méthode développée permet de résumer efficacement un maillage 3D ou un nuage 3D à grande échelle. Par rapport aux approches de cartographie précédentes, le graphe de navigabilité proposé permet de maximiser la visibilité tout en garantissant une présentation riche en information. En effet, le processus de synthèse est basé sur l'extraction de plusieurs vues sphériques représentant une partie de la carte initiale. Cette représentation sphérique contient des informations sémantiques, photométriques et géométriques. Cette nouvelle méthode de synthèse de cartes 3D nous permet de faciliter plusieurs tâches de navigation dans les systèmes de transport intelligents (localisation, planification d'itinéraire, évitement d'obstacles, ...). Elle permet de réduire considérablement le temps de calcul et la taille de la mémoire requise. Nous pensons également que l'utilisation des informations sémantiques permet d'élaborer une carte récapitulative précise en rejetant les données de localisation inutiles telles que les points appartenant à des objets dynamiques (voitures, piétons, ...). L'algorithme proposé est très simple à mettre en œuvre et adapté pour les environnements à grande échelle, cependant, il serait intéressant d'analyser l'utilité du graphe de navigabilité.

Pour ce faire, nous avons testé la localisation dans un graphe de sphères RGBD-L. Dans le chapitre suivant, nous présenterons les résultats de recalage 3D dans un graphe de navigabilité. Plusieurs scénarios ont été élaborés afin d'évaluer l'utilité

de ce résumé de carte.

Chapitre III.

Localisation en ligne et navigation dans un graphe de navigabilité



1. Introduction

Dans ce chapitre, nous allons aborder l'utilisation en ligne du graphe d'images sphériques augmentées RGBD-L. Ce graphe de navigabilité peut faciliter la réalisation de plusieurs tâches de navigation grâce à sa taille réduite et à la pertinence de son contenu. Parmi ces applications, nous allons détailler le processus de la localisation dans ce graphe. La localisation de la caméra agent dans un graphe de navigabilité est définie comme un problème de minimisation d'une fonction de coût entre une sphère de référence et l'image agent courante. La sphère de référence est définie comme étant le nœud du graphe le plus proche de l'agent. Plusieurs scénarios ont été élaborés afin de tester la localisation dans un graphe de sphères. Dans un premier temps, les images agent ont été simulées à l'aide des images synthétisées à partir du maillage. Ensuite, une séquence d'images réelles a été acquise à l'aide d'un système de caméras stéréoscopiques monté sur un véhicule navigant dans le voisinage du graphe de sphères. Pour une meilleure précision et un meilleur temps de calcul, une méthode de filtrage des données est utilisée pour éliminer les pixels inutiles à la localisation. Enfin pour obtenir une meilleure robustesse et une meilleure précision, nous avons proposé d'utiliser simultanément plusieurs nœuds du graphe dans la localisation.

Dans ce chapitre, nous allons voir comment choisir la sphère de référence RGBD-L pour localiser un agent dans un graphe de sphères. Nous allons présenter l'algorithme de recalage utilisé pour évaluer la pertinence et l'utilité du graphe dans la localisation. Nous détaillerons par la suite l'optimisation des algorithmes développés tout au long de cette thèse. Enfin, nous allons présenter les résultats de recalage obtenus avec tous les scénarios possibles.

2. Sélection de l'image de référence

Comme nous avons expliqué dans l'introduction du projet pLaTINUM, la localisation d'un agent se fait sur deux étapes. La première étape consiste à localiser l'agent grossièrement en trouvant la sphère de référence la plus proche. Tandis que la deuxième étape permet d'affiner la première localisation. Pour sélectionner l'image de référence la plus proche de l'agent, notre partenaire, le laboratoire Le2i, a proposé une méthode de localisation basée vision [93]. La localisation basée vision (LBV) est un sujet qui a été largement abordé dans le domaine de la vision par ordinateur. La LBV consiste à localiser une image requête par rapport à un ensemble d'images de référence. Plusieurs algorithmes ont été proposés dans la

littérature afin de résoudre ce problème. La plupart de ces méthodes utilisent une seule modalité (image) en se basant sur la description globale d'images [5] ou l'appariement de descripteurs locaux [104]. Cependant, aujourd'hui, il existe de plus en plus de modalités comme la profondeur et la sémantique. De ce fait, des nouvelles méthodes utilisent plusieurs caractéristiques pour localiser la requête. En s'inspirant de ces méthodes, une nouvelle approche d'apprentissage à partir de modalités auxiliaires pour la localisation basée vision a été proposée. Cette méthode consiste à entraîner un CNN à partir de données multimodales. Au moment du test, une unique modalité est utilisée pour évaluer le système. La fig 3.1 montre un exemple des données utilisées dans l'apprentissage ainsi qu'un exemple de la requête.

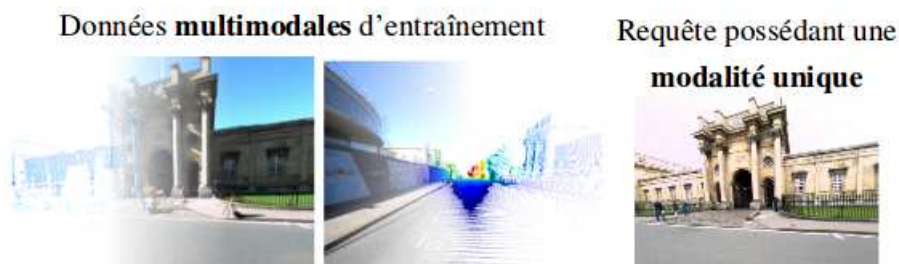


FIGURE 3.1 – Entraînement d'un système de localisation basé vision en utilisant des informations auxiliaires [93]

L'objectif de la localisation grossière est de sélectionner l'image sphérique la plus proche de l'agent dans un graphe de navigabilité. Pour ce faire, des adaptations sont nécessaires pour faire fonctionner le système avec des images sphériques au lieu des images perspectives. La sortie du système sera un ensemble contenant N images sphériques ordonnées à l'aide d'un score. Une fois que la localisation grossière est effectuée, nous allons affiner la position obtenue afin de localiser précisément l'agent.

3. Recalage 3D : estimation de la position d'une image agent

A. Introduction

Le recalage est un problème fondamental en vision par ordinateur. Ce problème a été largement étudié dans la littérature. Étant donné deux ensembles de points 3D, l'objectif de ce processus est de trouver la transformation qui aligne le mieux les deux nuages de points. Le recalage est utilisé dans de nombreuses applications comme la navigation, la mise à jour automatique de référentiel, la fusion des données et la localisation. Parmi les nombreuses méthodes de recalage proposées dans la littérature, l'algorithme ICP (Iterative Closest Point) [11, 129, 23], introduit dans les années 1990, est l'algorithme le plus connu pour le recalage de deux ensembles de points de référence.

Étant donné une transformation initiale : rotation et translation, l'algorithme alterne entre la recherche des correspondances des points les plus proches dans la transformation en cours et l'estimation de la transformation avec ces correspondances, jusqu'à la convergence.

Pour définir le problème du recalage nous considérons deux ensembles de points 3D : le nuage de points de référence $X = \{x_i, i = 1, N\}$ et le modèle $Y = \{y_j, j = 1, M\}$, où $\{x_i, y_j \in R^3\}$ sont des coordonnées de points 3D. Le but est d'estimer un mouvement rigide de rotation $R \in R_{3 \times 3}$ et de translation $T \in R^3$ entre X et Y . La fig 3.2 montre une illustration de recalage 3D.

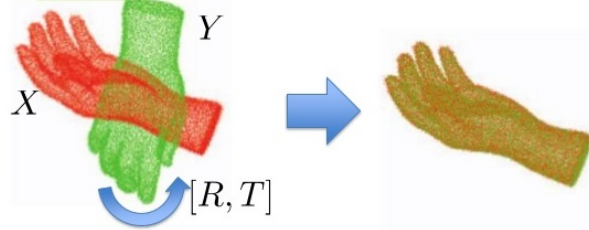


FIGURE 3.2 – Recalage 3D

Le problème du recalage est défini comme étant un problème de minimisation de l'erreur E comme indique l'équation 3.1 :

$$E(R, t) = \sum_{i=1}^M \|Rx_i + T - y_{j^*}\|^2 \quad (3.1)$$

Étant donné R et T , le point y_{j^*} est désigné par la correspondance optimale de x_i en Y , qui est le point le plus proche de x_i transformé dans Y (équation 3.4).

$$j^* = \arg \min_{j \in \{1, \dots, M\}} \|Rx_i + T - y_j\| \quad (3.2)$$

Comme montre la fig 3.3, ICP résout le problème de manière itérative en alternant entre l'estimation de la transformation avec eq.3.1 et la recherche de correspondances des points les plus proches avec l'équation 3.4.

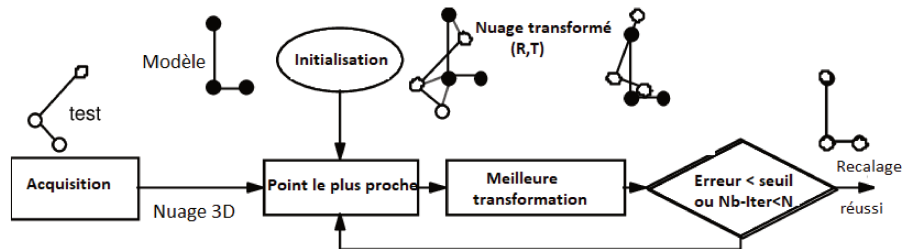


FIGURE 3.3 – ICP : Iterative Closest Point

ICP est une méthode itérative, elle nécessite donc une bonne initialisation, sans laquelle l'algorithme peut facilement converger vers un minimum local. Si cela se produit, la solution risque d'être loin de la solution optimale, ce qui entraînerait une estimation erronée de la position de l'agent. En plus, il n'existe aucun moyen fiable de savoir s'il a convergé ou non vers un minimum local. Pour remédier à ces problèmes, un nouvel algorithme appelé Go-ICP a été proposé [125]. Nous détaillerons par la suite cette méthode de recalage 3D.

B. Go-ICP : a globally optimal solution to 3D ICP point-set registration

Cet algorithme a été proposé en 2016 [125], il a été largement utilisé pour faire le recalage 3D. L'idée principale de cet algorithme est l'utilisation d'ICP classique avec la stratégie BnB : "Branch-and-Bound". L'utilisation de BnB permet de résoudre le problème de minimum local de l'ICP classique. Pour trouver la solution globale de recalage entre deux nuages de points 3D, Go-ICP résout le problème de manière itérative en alternant entre la stratégie BnB et ICP. Nous détaillerons par la suite les étapes de cet algorithme. Dans un premier lieu, nous allons voir comment l'espace de recherche de rotation et de translation a été paramétré.

L'auteur a représenté la rotation à l'aide de la représentation "angle-axis" sous la forme d'un vecteur 3D r . Le domaine de rotation est représenté à l'aide d'un cube $C_r \in [-\pi, \pi]$. Pour la translation, l'auteur suppose que la valeur optimale

de translation se trouve dans un cube C_t délimité par ϵ . La fig.3.4 montre le paramétrage de l'espace de rotation et translation.

L'algorithme Go-ICP se base principalement sur :

- La stratégie Branch-and-bound pour résoudre le problème de minimum local de l'ICP
- Le paramétrage de l'espace pour BnB (fig 3.4)

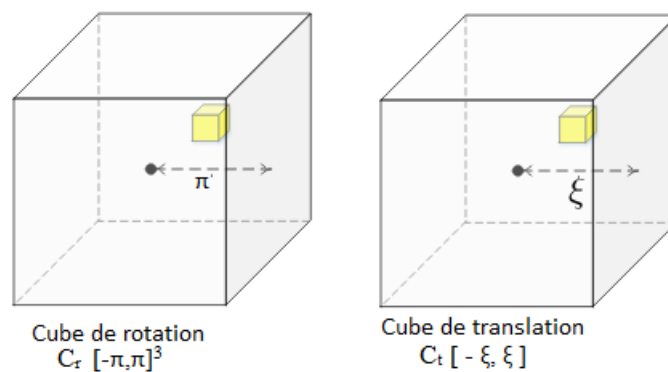


FIGURE 3.4 – Paramétrage de l'espace pour BnB. À gauche : l'espace de rotation. À droite : la translation est supposée être dans un cube 3D. Le cube jaune représente un sous-cube dans l'espace de recherche [125].

Go-ICP se base sur la stratégie BnB qui consiste à diviser l'espace de recherche en plusieurs sous-espaces. Chaque sous-espace sera évalué à part pour juger sa capacité à contenir la solution optimale. D'une itération à une autre, l'espace de recherche se rétrécit en convergeant vers la solution optimale. L'auteur propose de diviser les cubes de rotation et translation C_r, C_t en 8 sous-cubes chacun.

Pour évaluer chaque sous-cube, l'auteur propose de délimiter l'erreur avec une borne supérieure \overline{E}_r et une borne inférieure \underline{E}_r . Étant donné un cube de rotation C_r centré à r_0 et un cube de translation C_t centré à t_0 , les deux bornes de l'erreur seront calculées comme suit. D'abord, l'auteur affirme qu'avec une rotation $r \in C_r$ et une translation $t \in C_t$, un point transformé x_i se situera dans la boule centrée sur $R_{r_0}x_i + t_0$ de rayon σ ($\sigma = \sigma_r + \sigma_t$).

σ_r est le rayon d'incertitude de la rotation et σ_t est le rayon d'incertitude de la translation. La fig 3.5 montre la zone d'incertitude. Un point x perturbé par un mouvement rigide 3D composé d'une rotation $r \in C_r$ et d'une translation $t \in C_t$ se situe dans la boule d'incertitude (en jaune) centrée sur $R_{r_0}x + t_0$ avec le rayon

$\gamma = \gamma_r + \gamma_t$. Les points du modèle y_{j^*} et $y_{j_0^*}$ sont les plus proches de $R_r x + t$ et $R_{r_0} x + t_0$ respectivement. Il est clair que $a \leq b \leq c$ où $a = \underline{e}_i$

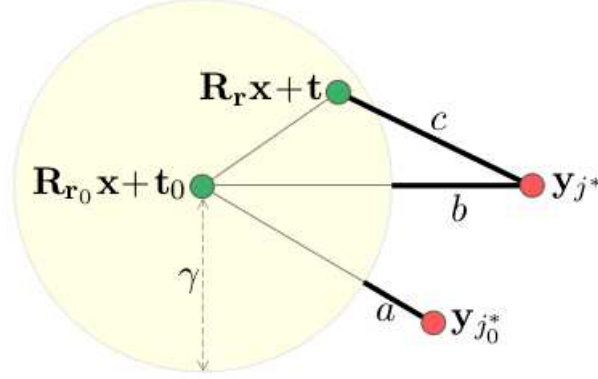


FIGURE 3.5 – Zone d'incertitude [125]

La fig 3.5 montre l'explication géométrique de \underline{e}_i . Puisque $y_{j_0^*}$ est le plus proche du centre $R_{r_0} x + t_0$ de la boule d'incertitude de rayon $\gamma = \gamma_r + \gamma_t$, il est également le plus proche de la surface de la boule et \underline{e}_i est la distance la plus proche entre le nuage de points Y et la boule. Ainsi, peu importe où se trouve le point de données transformé $R_r x + t$, sa distance la plus proche du Y ne sera pas inférieure à \underline{e}_i . En sommant les bornes supérieures et inférieures de tous les points, nous obtenons la borne supérieure \overline{E} et la borne inférieure \underline{E} de l'erreur de recalage E comme le montre l'équation 3.3 :

$$\overline{E} = \sum_{i=1}^M \overline{e}_i^2 = \sum_{i=1}^M e_i(R_{r_0}, t_0)^2 \quad (3.3)$$

$$\underline{E} = \sum_{i=1}^M \underline{e}_i^2 = \max\left(\sum_{i=1}^M e_i(R_{r_0}, t_0) - (\gamma_r + \gamma_t), 0\right)^2 \quad (3.4)$$

les bornes supérieure et inférieure sont calculées de la même manière pour la rotation et la translation.

L'algorithme 2 résume les étapes de recalage dans Go-ICP. Pour accélérer le calcul de la distance entre les points des deux nuages de points, une structure de données kd-tree et la transformation de distance euclidienne (DT) sont utilisées. L'avantage d'algorithme Go-ICP est sa capacité de converger vers un minimum global grâce à la stratégie BnB. À l'opposé de l'ICP, l'initialisation de la position initiale de la caméra n'a pas une grande influence sur la convergence de l'algorithme.

Algorithme 2 Go-ICP : a globally optimal solution to 3D ICP point-set registration

Entrée : nuages de points modèle/référence; seuil γ ; initialisation cubes CC_r, CC_t

Sortie : E^*, r^*, t^*

Sortie:

Mettre C_r en file d'attente prioritaire Q_r

Fixe $E^* = +\infty$

boucle

Lire un cube avec la limite inférieure la plus basse \underline{E}_r de Q_r

Quitter si $E^* - \underline{E}_r < \gamma$

Diviser le cube en 8 sous-cubes

Pour chaque sous cube C_r calculer \overline{E}_r et t

si $\overline{E}_r < E^*$ **alors**

Appliquer ICP avec (r_0, t) .

Mettre à jour $E^*, r^*,$ et t^* avec les résultats d'ICP.

fin si

Calculer \underline{E}_r de C_r

si $\underline{E}_r > E^*$ **alors**

jeter C_r

fin si Mettre C_r dans Q_r

fin si

fin

C. Résultats

Nous avons choisi l’algorithme Go-ICP pour tester la navigation dans un graphe de sphères RGBD-L . Un des points forts de cet algorithme est sa robustesse face à des changements de luminosité dans une scène urbaine. En effet, l’utilisation de la géométrie de la scène peut garantir une meilleure convergence vers une solution précise. Pour évaluer notre graphe de navigabilité, nous avons proposé plusieurs scénarios. Notre objectif est de tester si un agent peut se localiser précisément dans un graphe de sphères RGBD-L . Nous effectuons le recalage d’une image agent avec une ou plusieurs sphères RGBD-L . Ce recalage est fait séparément d’une image agent à une autre.

Dans cette expérience nous disposons de l’information sémantique pour chaque pixel de l’image sphérique. Nous proposons d’utiliser cette information sémantique pour filtrer les données inutiles à la localisation avant le recalage.

a. Filtrage sémantique

Le filtrage sémantique est une étape de prétraitement appliquée sur les sphères RGBD-L avant le recalage. La fig 3.6 montre les étapes de recalage 3D ainsi que l’étape de filtrage sémantique.

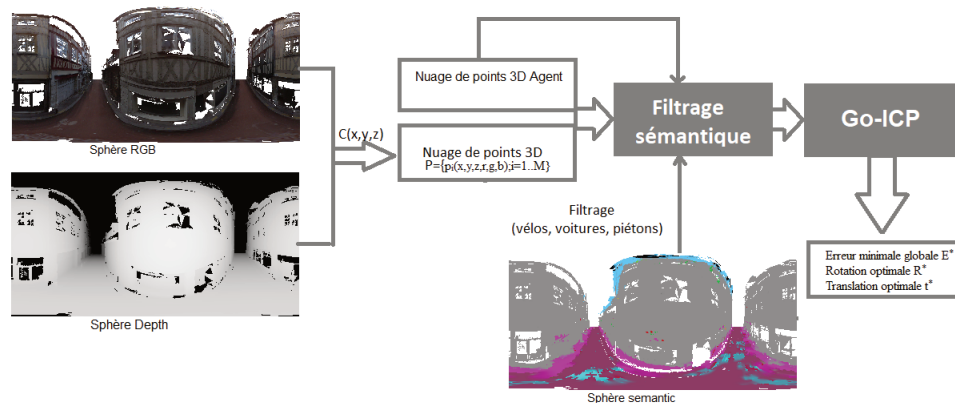


FIGURE 3.6 – Filtrage sémantique et recalage 3D à partir des images sphériques RGBD-L et un nuage de points 3D agent

Cette opération consiste à exploiter l’information sémantique présente dans les sphères. Le filtrage permet d’éliminer les pixels inutiles à la navigation. En effet, les pixels appartenant à des objets mobiles comme les classes voiture, moto, vélo et piéton sont considérés inutiles à la localisation à l’opposé des classes bâtiments et végétation. D’autres classes ont été enlevées comme la classe ciel.

Classes	Sans filtrage	Avec filtrage
Misc	✓	X
Road	✓	✓
RoadMarks	✓	✓
Terrain/sidewalk	✓	✓
Building	✓	✓
Vegetation	✓	✓
Tree	✓	✓
Pedestrian	✓	X
Car	✓	X
Bicycle	✓	X
Motorbike	✓	X
TrafficLight	✓	X
TrafficSign	✓	✓
Pole	✓	X
Sky	✓	X

TABLE 1 – Classes sémantiques supprimées lors du filtrage sémantique

Le tableau 1 résume les classes supprimées lors du filtrage sémantique.

Comme illustre la fig 2.18, nous avons 15 classes sémantiques au départ. Le filtrage sémantique permet d'éliminer 8 classes. En conséquence, nous avons éliminé environ 20% de pixels dans chaque image sphérique.

b. Premiers résultats de recalage

Le premier test effectué avec l'algorithme Go-ICP consiste à utiliser une sphère RGBD pour extraire le nuage de points 3D de référence. Le nuage 3D de l'agent sera un sous-nuage de la sphère de référence. Ce nuage sera perturbé par une transformation (R, t) . La fig 3.7 montre un nuage de points de référence et un sous nuage d'agent (jaune) ainsi que le résultat de recalage (en vert).

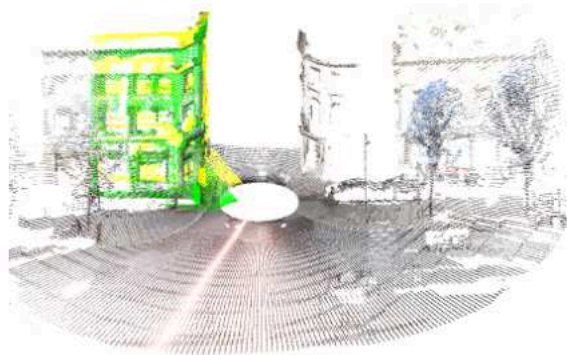


FIGURE 3.7 – Un exemple de recalage. Le nuage jaune représente l'image agent avant le recalage et le nuage vert le résultat de recalage

Pour évaluer les résultats obtenus, nous avons calculé l'erreur de rotation (deg), le RMS (root-mean square) ainsi que le temps d'exécution (s). Un des paramètres les plus importants de Go-ICP est le nombre N de points 3D utilisés dans le recalage. Pour voir l'influence de ce paramètre sur les résultats, nous l'avons fait varier entre 20 et 15.000 points.

Le tableau 5 montre la variation du nombre de pixels N et son influence sur le temps d'exécution, l'erreur en rotation et le RMS.

N	T (s)	E_R (deg)	RMS(m)	Résultat visuel
20	2,59	2,2	0,477	
100	3,44	2,0260	0,465	
1000	24.7399	1,1833	0,035	
1500	63,21	1,1818	0,009	
5000	274,861	1,1810	0,008	
10000	473	1.1808	0,002	
15000	636	1.1808	0,002	

TABLE 2 – Résultats de recalage 3D avec Go-ICP en fonction de nombre de points 3D N utilisés dans le recalage

Dans la littérature, ce paramètre a été souvent fixé empiriquement à 1000 pixels pour des nuages de points contenant 1 à 2 M de points. De la même manière suite à notre expérience, nous avons regroupé ces résultats dans la fig 3.8. Pour atteindre un compromis entre le temps d'exécution et la précision de localisation, nous avons choisi une valeur de 2000 points.

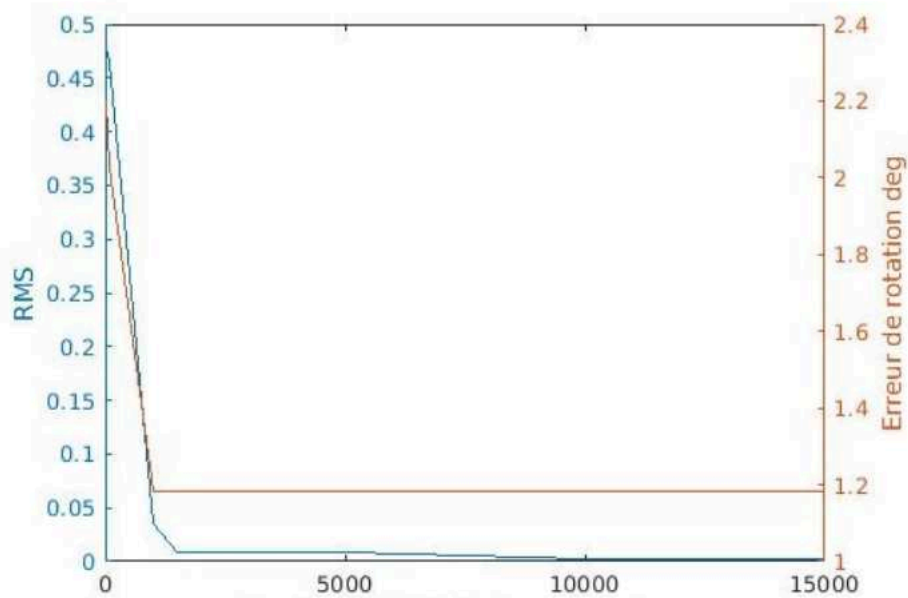


FIGURE 3.8 – L'évolution de l'RMS et l'erreur en rotation en fonction du nombre de pixels N utilisés dans le recalage avec Go-ICP

Dans la suite de ce manuscrit, nous allons détailler les différentes expériences réalisées pour tester les performances de recalage dans un graphe des sphères. La fig 3.9 illustre les différents cas de figure traités. Nous avons testé le recalage de deux types d'images agent. Nous avons utilisé des images agent réelles et synthétiques. Les images agent réelles ont été acquises selon trois méthodes : avec un banc stéréo-fish-eye, avec une caméra ZED ou avec le système multicaméra de l'IGN décrit dans le premier chapitre. D'un autre côté, nous avons proposé deux scénarios dans le cas où nous utilisons les images agent synthétisées à partir d'un maillage texturé. Nous détaillerons par la suite tous les résultats obtenus dans chacun des cas de figure.

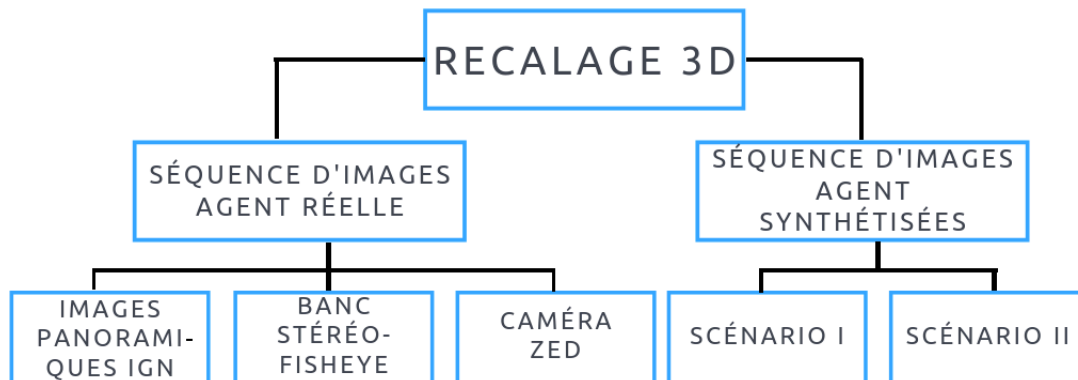


FIGURE 3.9 – Localisation dans un graphe

c. Localisation dans un graphe de panoramiques

Pour tester le recalage nous avons choisi une séquence complète d’images panoramiques acquises par l’IGN. Cette trajectoire contient 98 images panoramiques $\{S_i, i = 1..98\}$, séparées d’une distance entre 0.6123 m et 4.1303 m. La longueur de cette séquence est 282 m. Pour chaque couple d’images panoramiques consécutives (S_i, S_{i+1}) , une partie de l’image S_{i+1} est considérée comme étant une image agent et l’image S_i comme l’image de référence. La fig 3.10 suivante montre la localisation de l’agent dans cette séquence.

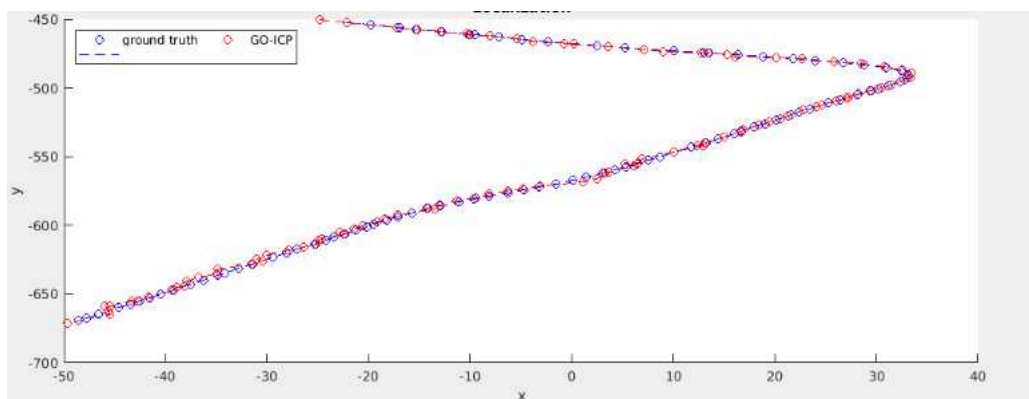


FIGURE 3.10 – Localisation dans un graphe d’images panoramiques : en bleu la trajectoire réelle, en rouge l’estimation de la position de l’agent

La fig 3.11 montre l’erreur de localisation (m) et la distance qui sépare l’agent et

la sphère de référence.

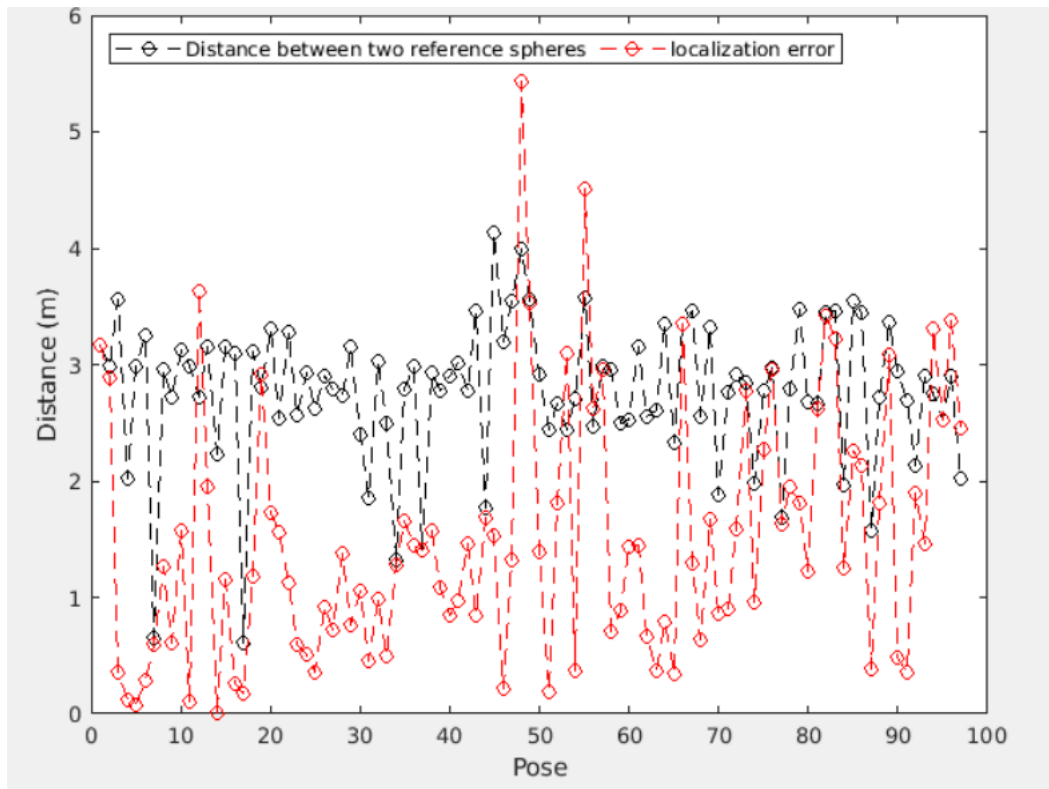


FIGURE 3.11 – Erreur de localisation et la distance entre l’agent et la sphère de référence en m

Nous pouvons constater que 78% d’images agent ont été localisées avec une erreur de localisation inférieure à 2 m et 40,81% ont été localisées avec une erreur inférieure à 1 m. Comme montre la fig 3.12, l’algorithme Go-ICP parvient à trouver la bonne transformation même avec une distance de 3,5 m par rapport à la sphère de référence.

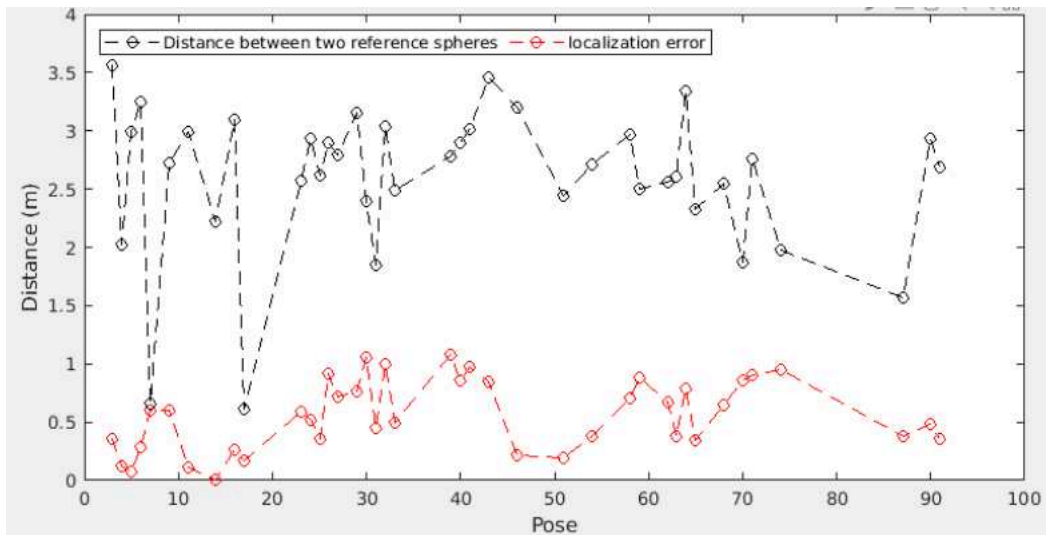


FIGURE 3.12 – Erreur de localisation inférieure à 1m et la distance entre l’agent et la sphère de référence en m

La fig 3.13 montre la trajectoire obtenue par l’algorithme de localisation : au point de départ, le véhicule effectue une marche en avant puis une marche arrière. On peut voir que l’utilisation d’images panoramiques permet une localisation dans deux directions de navigation différentes en utilisant les images de référence.

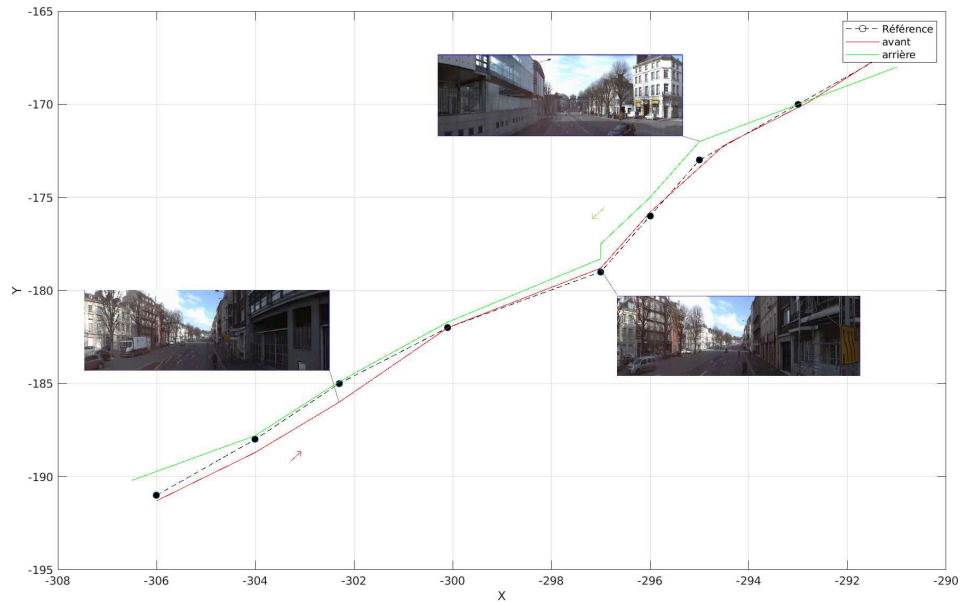


FIGURE 3.13 – Un exemple de localisation dans un graphe d’images panoramiques. En rouge : le véhicule navigue en marche avant. En vert : le véhicule effectue une marche arrière, des exemples d’images agent

Malgré ces résultats encourageants, le pourcentage d’images d’agent qui n’ont pas été bien localisées est considérable. En effet, les images de profondeur présentent quelques imperfections. Dans quelques panoramiques, il existe un décalage entre le RGB et la profondeur et parfois l’information de profondeur est incomplète. En effet, les cartes de profondeurs sont estimées à partir d’information laser du LiDAR.

De plus, la profondeur est encodée avec un seuil de 30 m. Ce qui explique l’absence de quelques zones dans l’image de profondeur comme les bâtiments situés à plus de 30 m comme dans les deux images suivantes (fig 3.14).

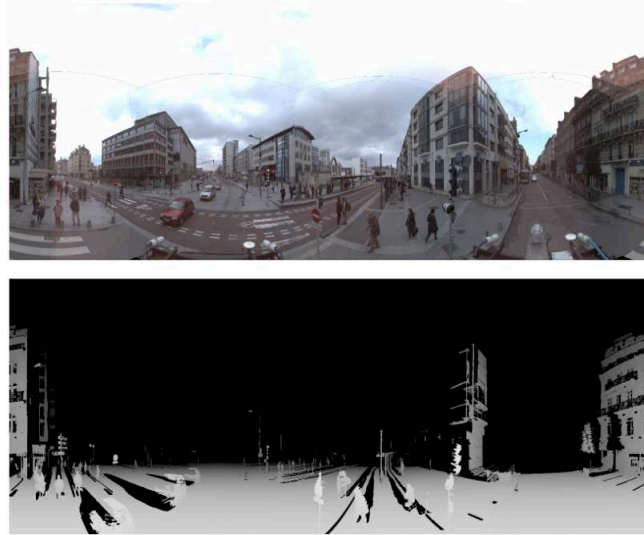


FIGURE 3.14 – Un exemple d’image panoramique acquise par l’IGN

Toutes ces imperfections peuvent expliquer l’erreur importante de localisation. À partir de maintenant, nous allons fixer un seuil pour l’erreur de localisation. Les images ayant une erreur de localisation inférieure à $1m$, seront considérées bien localisées.

d. Localisation dans un graphe de sphères synthétisées

Nous avons choisi un trajet de longueur $150 m$ (fig 3.15).

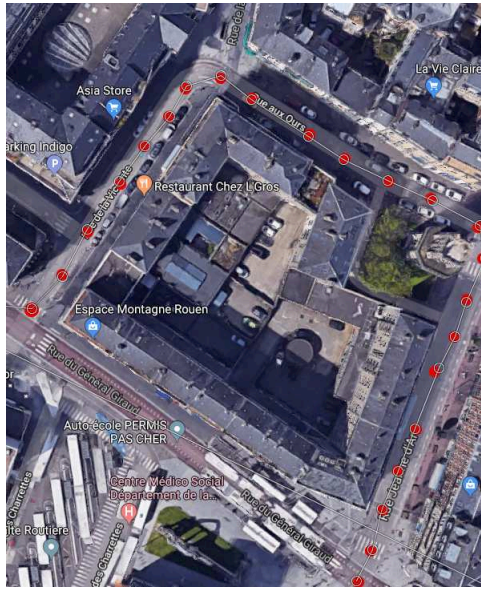


FIGURE 3.15 – Trajectoire d’essai à Rouen de longueur 150 m résumée avec 22 sphères RGBD-L

Ce trajet comporte 45 images panoramiques. Ces images ont été acquises par l’IGN en utilisant le système multi-caméras expliqué dans le premier chapitre. La fig 3.14 montre un exemple de ces images. Ces images panoramiques sont espacées d’une distance qui varie entre 3 et 4 m. Elles sont placées d’une manière systématique sans prendre en compte les caractéristiques de la scène.

Le premier scénario consiste à générer les sphères RGBD-L de référence dans les mêmes positions que les panoramiques de l’IGN $\{S_i, i = 1..45\}$. Pour chaque couple des sphères consécutives (S_i, S_{i+1}) , la sphère S_{i+1} est considéré comme étant une image agent et l’image S_i comme la sphère de référence.

Cependant, dans le deuxième scénario, les sphères de référence ont été générées suite à l’application de l’algorithme de recherche du graphe optimal. La trajectoire est résumée sous la forme d’un graphe de 22 sphères RGBD-L $\{S_i, i = 1..22\}$. Les sphères ont une résolution $2000 * 1000$.

Dans les deux scénarios, la séquence d’images agent contient 45 images perspectives extraites des sphères RGBD-L générées. Nous allons tester l’influence du filtrage sémantique sur la précision de la localisation.

d.1 Scénario i Comme expliqué précédemment, nous avons, 45 images agent et 45 sphères RGBD-L de référence. Le premier scénario consiste à effectuer le recalage de chaque image agent avec la sphère RGBD-L de référence la plus proche.

Nb images agent	Sans filtrage	Avec Filtrage
45	66.6667 %	88.8889 %

TABLE 3 – Score de recalage réussi avec et sans filtrage sémantique

Cette sphère est sélectionnée grâce à la localisation grossière. Pour chaque image, nous calculons l'erreur de localisation entre la position estimée et la position réelle de l'agent. Nous testons le même scénario sans ou avec le filtrage sémantique. Ce filtrage est appliqué sur la sphère de référence ainsi que sur l'image agent.

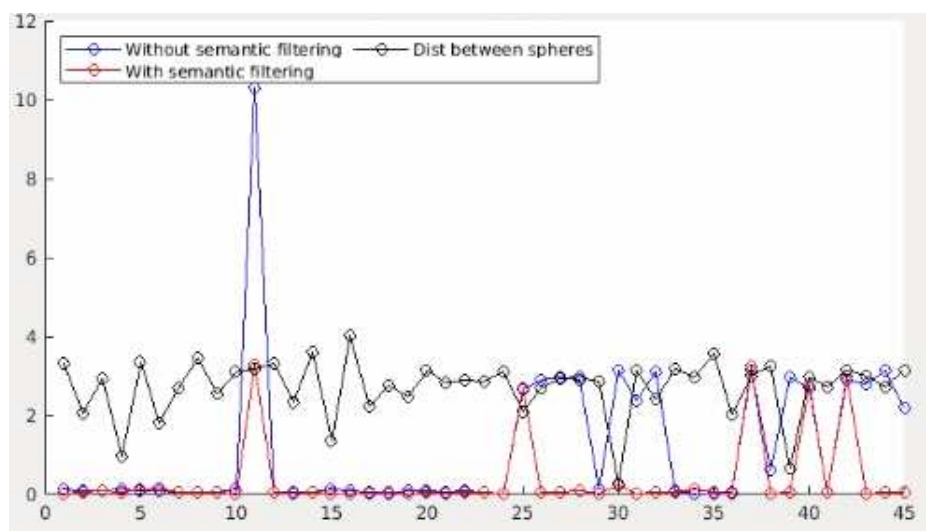


FIGURE 3.16 – Scénario i : erreur de localisation obtenu avec/sans filtrage

La fig 3.16 résume les résultats obtenus dans les deux cas de figure. On peut voir que le filtrage sémantique permet d'améliorer la précision de localisation. En effet, la fig 3.16 montre l'erreur de localisation avec filtrage sémantique en rouge et en bleu l'erreur de localisation sans filtrage sémantique. Nous constatons que la plupart du temps la courbe rouge est au-dessous de la courbe bleu ce qui indique une précision de localisation plus élevée. Comme nous avons précisé précédemment, le recalage est considéré comme réussi si l'erreur de localisation est inférieure à 1 m. En appliquant le même seuil, nous avons calculé le pourcentage d'images bien localisées comme il est indiqué dans le tableau 3.

D'autre part, le filtrage sémantique permet de réduire le temps nécessaire pour le recalage. Cette amélioration se traduit par une réduction de 20 (s) en moyenne dans le temps de calcul.

d.2 Scénario ii Dans le deuxième scénario, les sphères de référence ont été générées suite à l'application de l'algorithme de recherche du graphe optimal. La trajectoire de test a été résumée sous la forme d'un graphe qui contient 22 sphères RGBD-L. L'agent est simulé avec les 45 images agent générées précédemment. La fig 3.17 montre l'erreur de localisation obtenue dans les deux alternatives : avec ou sans le filtrage sémantique. On peut voir que le filtrage sémantique prouve son utilité à améliorer la précision de recalage.

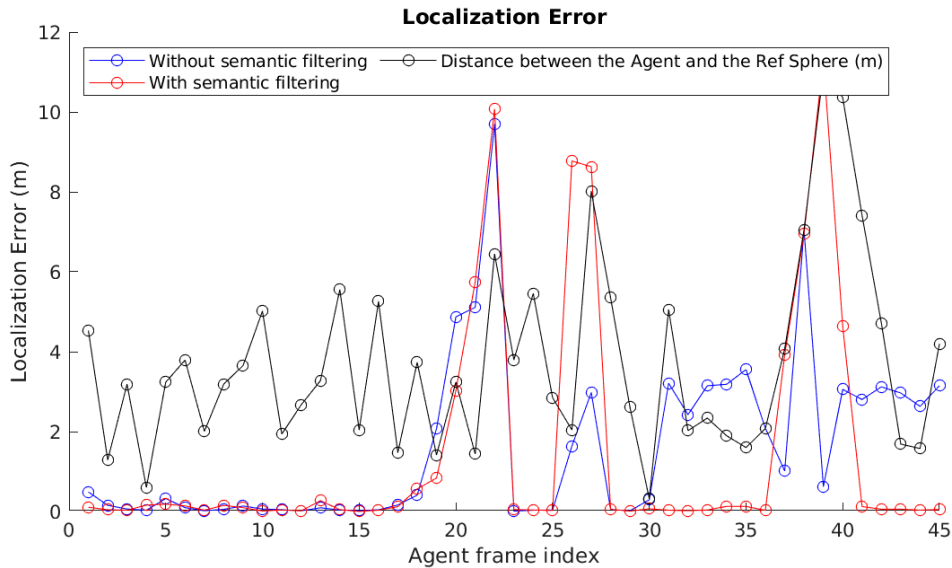


FIGURE 3.17 – Erreur de localisation avec et sans filtrage sémantique. 60 % des cas le filtrage sémantique améliore la précision de localisation

Nous avons analysé l'évolution de l'erreur de localisation en fonction de la distance séparant l'agent et la sphère de référence. Cette étude est illustrée dans la fig 3.18. Nous pouvons constater que l'agent parvient à se recalibrer par rapport à une sphère de référence située à 5 m. Les résultats de localisation les plus précis sont concentrés dans un rayon de 3 m par rapport à l'agent. On peut dire que la distance optimale entre les sphères de référence est égale à 3m. Cette distance sera intégrée dans le processus d'extraction du graphe de navigabilité comme une contrainte supplémentaire. L'objectif de l'ajout de cette contrainte est de garantir la réussite de navigation dans le graphe optimal.

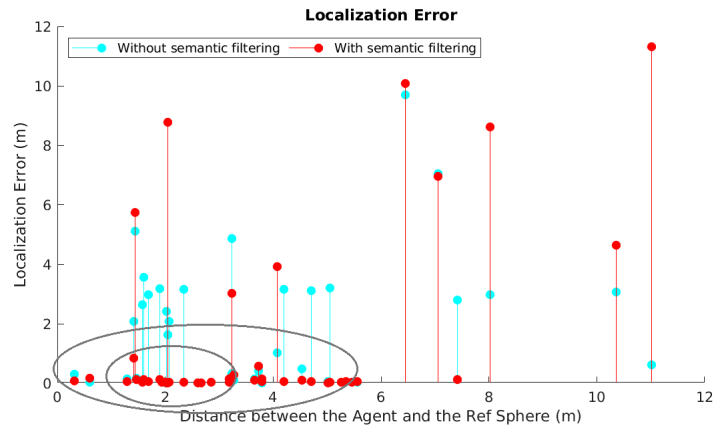


FIGURE 3.18 – Erreur de localisation avec et sans filtrage sémantique en fonction de la distance entre l’agent et la sphère de référence : au-delà de 5m l’erreur de localisation augmente

La fig 3.19 montre la trajectoire estimée avec et sans filtrage sémantique.

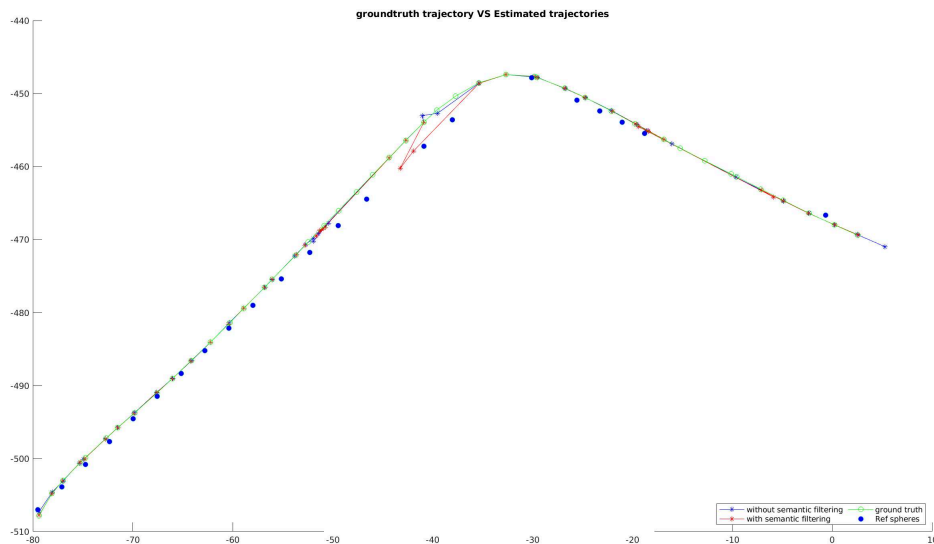


FIGURE 3.19 – Trajectoire d’essai : 150 m

Nous pouvons constater que dans 80 % des cas, Go-ICP a réussi à estimer la position avec une erreur inférieure à 1 m. En comparant les résultats obtenus dans les deux scénarios (tab 4), nous pouvons prouver l’utilité de filtrage sémantique. De plus, même en diminuant le nombre de sphères RGBD-L , l’agent parvient à suivre la trajectoire réelle.

Pourcentage de recalage réussi	Sans filtrage	Avec Filtrage	Taux de compression
Scénario i	66.6667	88.8889	95,64
Scénario ii	55.5556	80	97,75

TABLE 4 – Score de recalage réussi avec et sans filtrage sémantique, le taux de compression dans les scénarios

Jusqu'à maintenant, nous avons utilisé des images d'agent extraites à partir des images RGBD-L synthétisées. Nous avons décidé d'acquérir une séquence d'images d'agent réelles avec un système stéréoscopique. Pour réaliser cette acquisition, un banc stéréo-fisheye calibrée et synchronisé avec une IMU Xsens et un GPS (non synchronisé) ont été utilisés. Cette séquence permet de tester les performances de Go-ICP et la capacité de se localiser dans un graphe de sphères RGBD-L . Pour ce faire, nous avons obtenu une séquence des paires d'images stéréo sur le même trajet de test. Nous disposons uniquement des positions GPS peu précises pour chaque paire qui va nous servir de réalité terrain. Avant d'exploiter ces images, une étape de calibrage est nécessaire. Pour ce faire, nous avons utilisé un outil appelé "Stereo Camera Calibrator" développé par MATLAB basée sur la méthode décrite dans [130]. La fig 3.20 montre le résultat de calibrage.

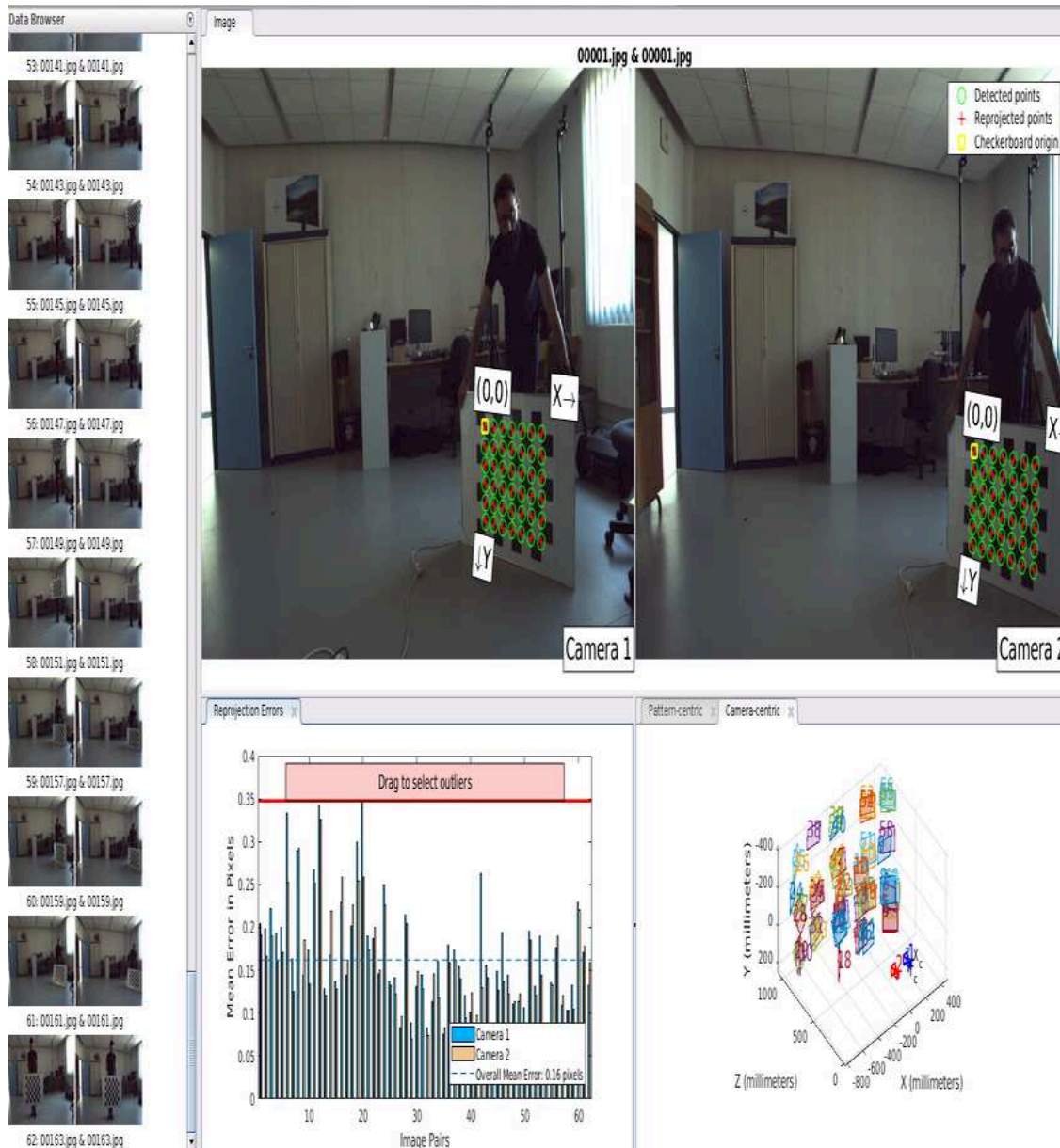


FIGURE 3.20 – Calibrage des caméras stéréoscopiques

Pour pouvoir récréer une scène 3D à partir des images d'agent nous avons besoin de rectifier chaque paire d'images stéréo. Ensuite, nous extrairons la carte de disparité. Un exemple de rectification des images stéréoscopiques est illustré dans la fig 3.21. La fig 3.22 montre la carte de disparité correspondante.

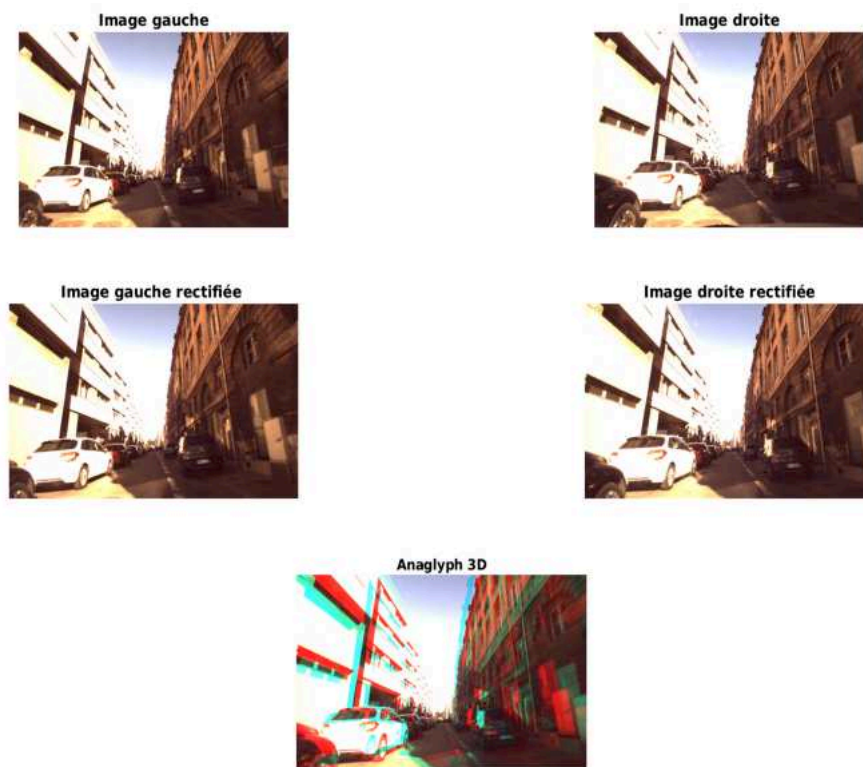


FIGURE 3.21 – Rectification des images stéréoscopiques

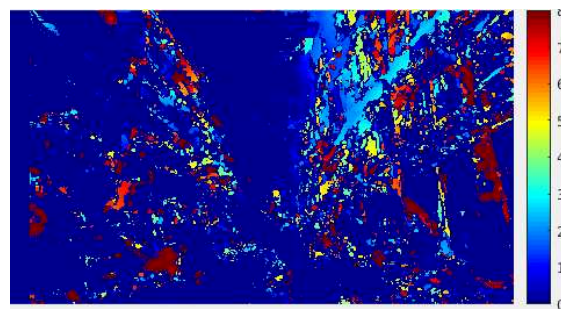


FIGURE 3.22 – Extraction de la carte de disparité

Nous avons testé le recalage de quelques images d'agent réelles dans un graphe de sphères RGBD-L . Nous avons obtenu des résultats précis lors de la localisation des images agent synthétisées. Cependant, les résultats obtenus avec des images agent réelles sont moins précis. Le tableau 5 montre quelques exemples des résultats obtenus. L'erreur de localisation obtenue varie entre 4 et 7 m. Ces résultats peuvent

être causés par plusieurs facteurs. Nous ne disposons pas d'une localisation précise pour chaque image d'agent pour pouvoir la comparer avec la position estimée. De plus, le système d'acquisition utilisé dans cette expérience présente quelques imperfections. Des erreurs de calibrage, des erreurs dans la synchronisation entre les deux caméras stéréoscopiques et la faible rigidité de support des caméras sont tous des facteurs qui peuvent influencer la qualité des images acquises.


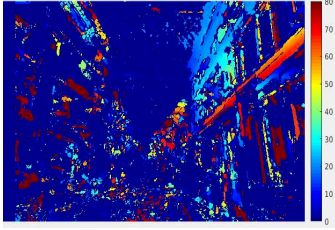

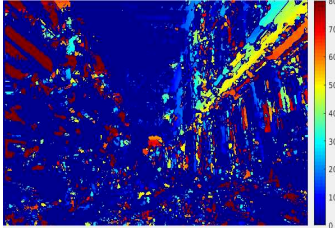

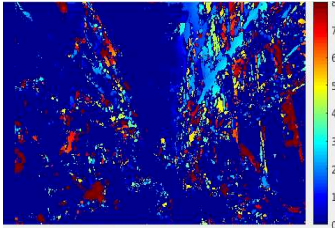
Images Rectifiées	Carte de disparité	Erreur de localisation
		5.8
		4
		7

TABLE 5 – Recalage dans un graphe de sphère RGBD-L et une séquence d'images agent réelles

Nous avons testé l'algorithme de recalage sur une nouvelle séquence d'images agent réelle. Elle couvre une trajectoire de 150 m. Dans cette séquence, nous avons utilisé la caméra ZED portée sur la poitrine pour réaliser une trajectoire piétonne. La caméra ZED est un outil développé en 2015, qui permet de réaliser des modèles 3D à partir d'une simple prise de vue en vidéo. L'avantage de cet outil est sa simplicité d'utilisation, ainsi que son coût. Les images ont été enregistrées à 15 *fps* sur un ordinateur hp Zbook. La fig 3.23 montre un exemple d'image agent et la construction du nuage 3D correspondant.

L'acquisition des images de l'agent s'est faite de deux façons. La première séquence est dédiée aux piétons et la seconde aux véhicules. Pour la trajectoire piétonne, nous avons acquis des images sur les deux trottoirs de la rue comme le montre la



FIGURE 3.23 – (a) La rectification d’une image agent droite et une image gauche
 (b) La construction d’un nuage 3D agent à partir de deux images gauche-droite

	LS	MLE
Image synthétisée	LS_{wf} : 56% LS_f : 80%	MLE_{wf} : 1.36 MLE_{wf} : 0.7
Fisheye	LS_{wf} : 40%	MLE_{wf} : 4.2
Caméra ZED (véhicule)	LS_{wf} : 71%	MLE_{wf} : 1.5
Caméra ZED (piéton)	LS_{wf} : 63%	MLE_{wf} : 1.8

TABLE 6 – Résultats de recalage : Le score de localisation LS (%), l’erreur moyenne de localisation MLE (m). Le test est effectué avec filtrage f ou sans filtrage sémantique wf .

figure 3.24. Pour tester le recalage, nous avons produit deux graphes de sphères. Un graphe dédié aux véhicules et le second aux piétons.

Pour évaluer les résultats obtenus nous avons proposé de calculer les critères suivants :

- **LS** : le score de localisation LS (
 - Une image agent est considérée comme bien localisée si l’erreur de localisation est inférieure ou égale à 1 m.
- **MLE** : erreur de localisation moyenne (m).
 1. **f** : recalage effectué avec filtrage sémantique
 2. **wf** : recalage effectué sans filtrage sémantique

Les résultats d’enregistrement obtenus avec cette séquence sont meilleurs que ceux de la séquence précédente. Le tableau 3.16 montre que le taux de réussite

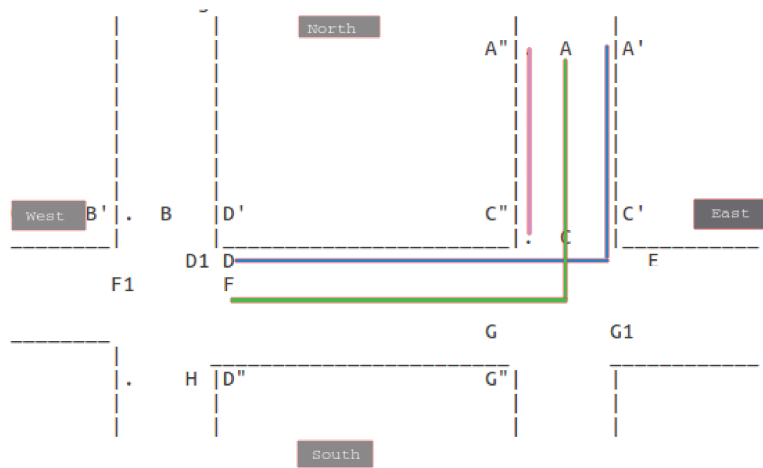


FIGURE 3.24 – Les séquences de la caméra ZED, vert : trajectoire du véhicule, bleu : trajectoire du piéton premier trottoir, rose : trajectoire du piéton deuxième trottoir

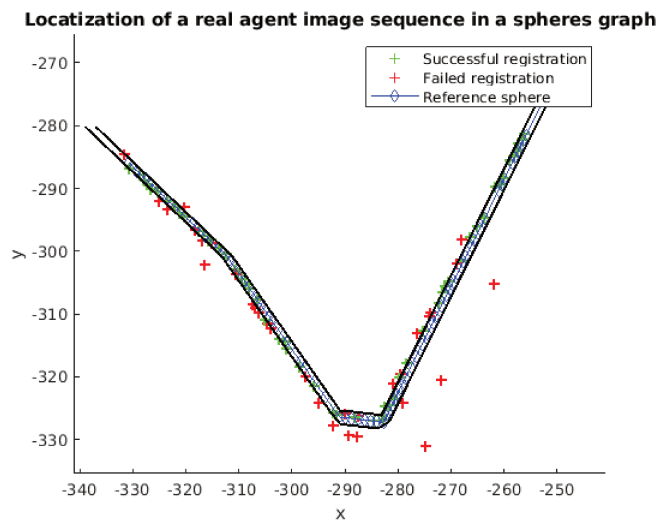


FIGURE 3.25 – Résultats de recalage obtenus avec la séquence de l'agent sur la trajectoire piétonne : la sphère de référence (bleu), la zone piétonne (noire), le recalage réussi (vert), recalage échoué (rouge)

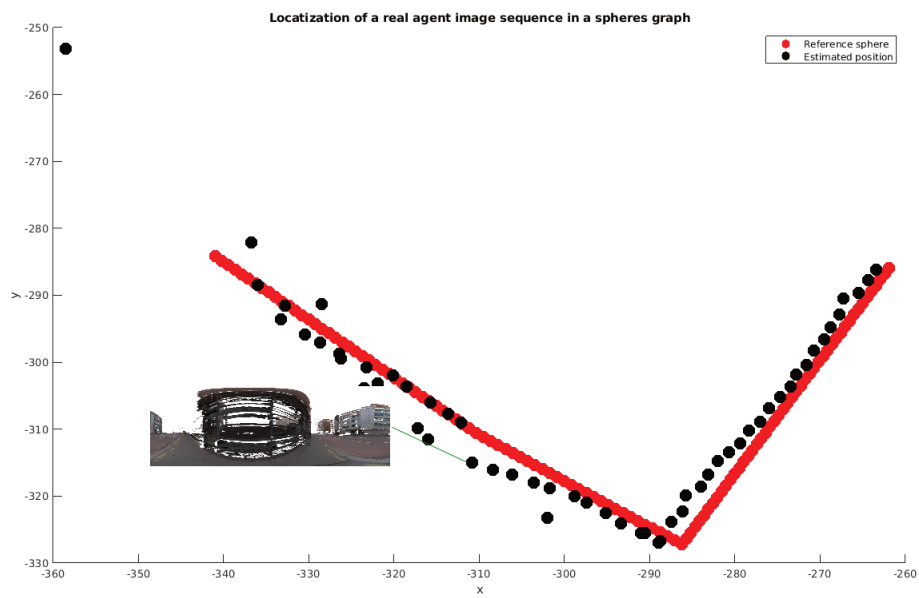


FIGURE 3.26 – Résultats de recalage obtenus avec la séquence d’agent sur la trajectoire du véhicule : la sphère de référence (rouge), les résultats de localisation (noir), un exemple de sphère où le recalage a échoué

de la localisation s'est amélioré par rapport à l'ancienne séquence, le système d'acquisition utilisé dans cette expérience est plus robuste et stable que l'ancien système. Ceci peut expliquer l'amélioration de la précision de localisation. Toutefois, l'évaluation des résultats reste difficile. En effet, pour évaluer les résultats obtenus, nous avons utilisé des coordonnées GPS qui ne sont pas très précises dans les rues étroites. Mais nous savons que dans la trajectoire piétonne l'ensemble des images est nécessairement sur les trottoirs ou sur le passage piétonnier. Pour évaluer les résultats, nous avons défini une zone de marge. Cette zone représente le trottoir sur toute la trajectoire. La largeur de cette zone est d'environ 1,2 m. En respectant cette hypothèse, toutes les images agent appartenant à cette région sont considérées comme bien localisées. La figure 3.25 montre les images qui ont été localisées dans la zone piétonne (en vert) et celles qui ont été localisées à l'extérieur de cette zone. Ces erreurs peuvent être causées par le choix de la sphère la plus proche. Les résultats de la localisation dans le graphe de sphères dédié aux véhicules semblent être meilleurs que la trajectoire piétonne. On peut expliquer cela par le fait que dans la séquence des piétons, le phénomène de masquage par les voitures est très fréquent. En effet, les images ont été prises à une hauteur moyenne de 1,5 m mais malgré cela, une grande partie de la scène est cachée par les voitures garées des deux côtés de la rue. La figure 3.26 montre un exemple de résultats obtenus où l'on peut voir que l'agent peut se localiser. Dans certains cas particuliers, nous constatons que l'erreur de localisation augmente considérablement. Par exemple, comme le montre la figure 3.26, dans le cas de bâtiments avec façades vitrées, l'algorithme ne converge pas vers la bonne solution. En effet, en cartographiant la scène, le Lidar ne capture pas toute la surface.

e. Utilisation de plusieurs nœuds du graphe

Lors de la phase de localisation, l'image d'agent est en permanence recalée sur la sphère de référence la plus proche. La même sphère de référence est alors conservée pendant un certain temps jusqu'à ce qu'une nouvelle sphère soit sélectionnée comme étant la plus proche de la position courante.

Cependant, plus la distance absolue entre la sphère de référence et l'image courante augmente, moins la capacité à se localiser diminue. De ce fait, l'estimation de la pose devient de moins en moins précise. Ces erreurs, ajoutées à l'incertitude sur la sélection de la sphère la plus proche, peuvent générer des discontinuités sur les trajectoires estimées. Cela peut conduire aussi à effectuer le recalage de l'image d'agent avec une sphère trop loin de la solution, ce qui peut empêcher l'algorithme de converger.

Afin d'obtenir des trajectoires les plus lisses possible, il est souhaitable d'utiliser plusieurs sphères de référence simultanément. Nous notons le nombre de sphères

de référence utilisées dans le recalage par N . L'image courante d'agent sera recalée sur une fenêtre de N sphères les plus proches dans le graphe comme illustré sur la fig 3.27.

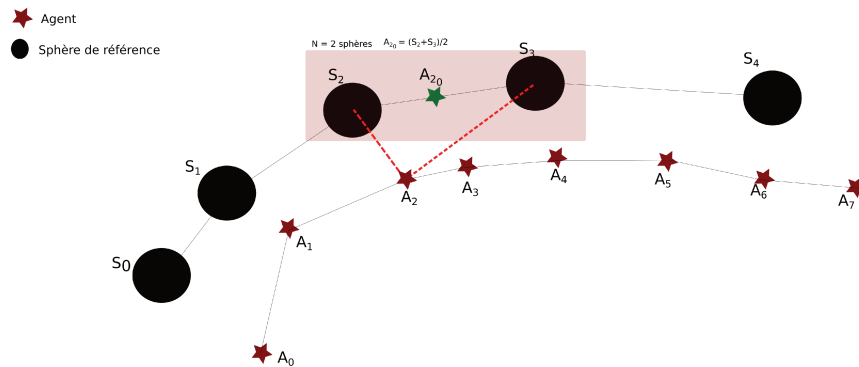


FIGURE 3.27 – Recalage avec N sphères de référence

Dans ce cas, le problème de localisation consiste à minimiser la distance entre le nuage de points de l'agent et les N sphères.

La position initiale de l'agent est initialisée à l'aide d'une valeur choisie au hasard, à l'intérieur d'un cercle qui englobe les N sphères (fig 3.28).

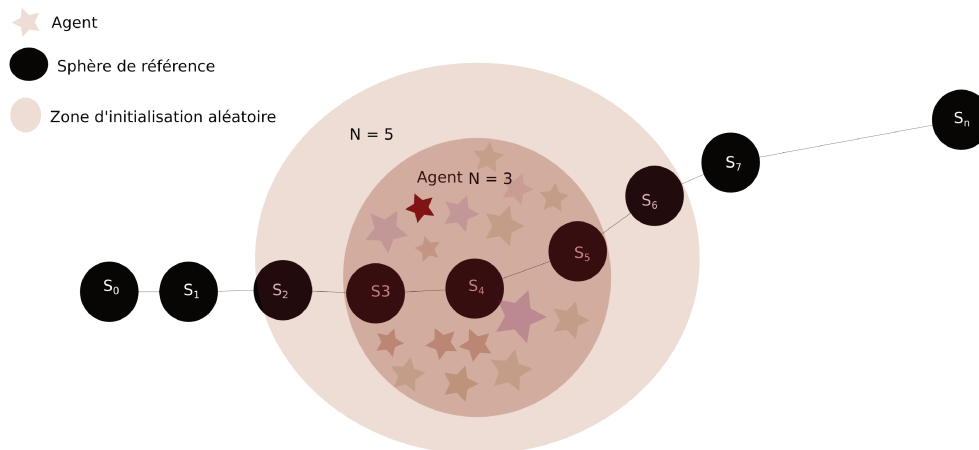


FIGURE 3.28 – Recalage avec N sphères de référence : initialisation de la position de l'agent

Pour tester cette proposition, nous avons choisi une trajectoire de 111 m. Le recalage sera effectué sur une fenêtre contenant les N sphères les plus proches de l'image agent. Nous avons testé deux valeurs de N : 3 et 5. La fig3.29 suivante

montre l'estimation de la position de l'agent (en rouge) superposée à la trajectoire réelle de l'agent. La courbe en bleu montre l'initialisation de la position de l'agent.

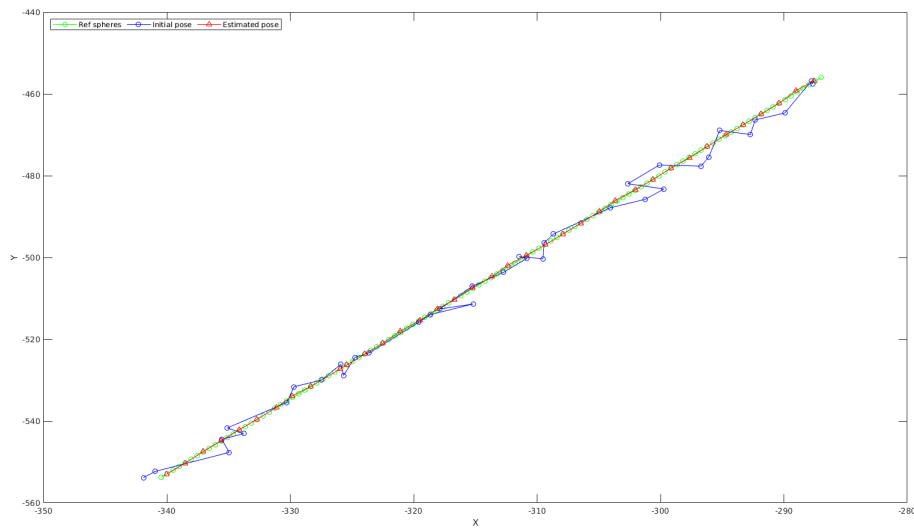


FIGURE 3.29 – Recalage d'image agent par rapport à $N = 5$ sphères de référence les plus proches

Nous avons testé le recalage avec une fenêtre de 3 sphères de référence. Les résultats obtenus avec une fenêtre plus large ($N=5$) permettent d'améliorer légèrement les résultats de la localisation. La fig 3.30 montre les résultats obtenus, dans les deux cas l'erreur de localisation est inférieure à 1m.

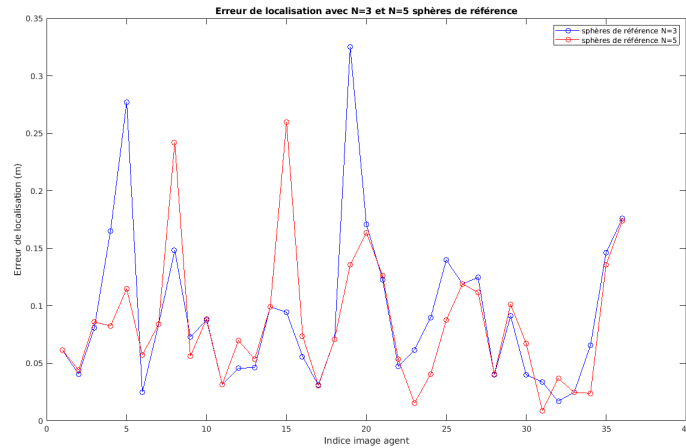


FIGURE 3.30 – L’erreur de localisation avec $N=5$ et $N=3$

Dans tous les scénarios testés, le temps de recalage est en moyenne 40 s qui reste un temps raisonnable pour un recalage en temps réel.

4. Outils utilisés

Nous proposerons dans cette partie, les outils logiciels et matériels utilisés tout au long de cette thèse. Ces outils nous ont aidé à bien implémenter et tester notre solution proposée. Par la suite, nous prêterons attention à l’optimisation et l’organisation du code. Enfin, nous arriverons à la conclusion du chapitre.

.1 Outils logiciels

Pour implémenter notre algorithme d’extraction du graphe de sphères RGBD-L , nous avons utilisé des bibliothèques libres. Elles sont largement utilisées dans le domaine de la vision par ordinateur.

PCL

Point Cloud Library (ou PCL) est une bibliothèque « open source » très utilisée pour l’imagerie 2D/3D et le traitement de nuage de points. Elle contient de nombreux algorithmes concernant le filtrage, les fonctions d’estimation, la reconstruction de la surface, l’enregistrement, l’ajustement du modèle et de la segmentation. Ces algorithmes peuvent être utilisés, par exemple, pour filtrer les valeurs aberrantes à partir de données bruitées, nuages de points 3D, extraire les points-clés

et calculer les descripteurs associés à des objets dans l'environnement étudié en se basant sur leur apparence géométrique, et aussi pour créer des surfaces à partir de nuages de points et de les visualiser. PCL est multiplateforme, et a été compilé et déployé sur Linux, MacOS, Windows et Android/iOS avec succès. Pour simplifier le développement, PCL est divisé en une série de petites bibliothèques de code, qui peuvent être compilées séparément. Cette modularité est importante pour la distribution de PCL sur les plateformes avec des contraintes de calcul ou de taille réduite.

OpenCV

OpenCV (Open Source Computer Vision Library) est une bibliothèque de logiciels « open source » spécialisée en vision par ordinateur et en apprentissage automatique. OpenCV a été conçue pour faciliter et accélérer l'utilisation de plusieurs outils de perception. OpenCV est sous licence BSD, il est facile pour les utilisateurs d'utiliser et de modifier le code. OpenCV est écrite nativement en C ++. La bibliothèque contient plus de 2500 algorithmes optimisés, qui incluent un ensemble complet d'algorithmes classiques et avancés de vision par ordinateur et d'apprentissage automatique.

Cette bibliothèque peut être utilisée dans plusieurs applications comme la détection des visages, l'identification des objets, la classification, calibrage des caméras et aussi la navigation.

Logiciel de gestion de versions

Nous avons utilisé git pour la gestion de versions. Cette technique nous permet de stocker un ensemble de fichiers(code, bibliothèque, documents...)en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. Cette solution nous permet notamment de retrouver les différentes versions d'un lot de fichiers connexes.

.2 Outils matériels

Les données d'entrée de notre algorithme de résumé de carte sont très volumineuses. Le test des différentes méthodes sur ces données devient impossible avec une machine ayant une capacité de calcul limitée. De ce fait, nous avons un autre partenaire dans le projet pLaTINUM qui est le Centre Régional Informatique et d'Applications Numériques de Normandie (CRIANN).

Le CRIANN a pour mission d'aider les organismes publics et privés normands à développer des activités d'enseignement, de recherche et de développement basé sur l'utilisation des nouvelles technologies de communication et sur l'informatique.

Pour cela, le CRIANN déploie des infrastructures informatiques à haut niveau de performance au service de l'enseignement supérieur, de la recherche et de l'innovation en Normandie.

Le CRIANN nous a fourni une machine puissante. Ce supercalculateur contient 40 cœurs de calcul et 16x 16 Go de RAM. Cette machine nous a permis d'accélérer l'exécution de nos algorithmes sur les données volumineuses de l'IGN. Les supercalculateurs du CRIANN comportent un environnement logiciel approprié pour le développement et le portage des codes sur une architecture parallèle.

5. Optimisation et organisation du code

L'algorithme de résumé de carte 3D développé tout au long de cette thèse permet d'extraire un graphe de navigabilité. Cet algorithme se décompose en plusieurs étapes résumées dans la fig 3.31. L'entrée sera un maillage texturé et sémantisé. Dans un premier temps, les zones navigables et visibles seront extraites en se basant sur l'information sémantique. Ensuite, elles seront transformées en polygones. Puis, le graphe de barycentres sera calculé. Enfin, l'étape d'optimisation permet d'extraire le graphe de navigabilité et construire les sphères RGBD-L. Cet algorithme peut s'adapter facilement à des environnements de grande taille. Bien que ce processus sera appliqué hors-ligne, le temps de calcul de ce graphe doit être raisonnable. En effet, ce système doit être capable de traiter des cartes très volumineuses représentant des villes entières. De plus, des mises à jour de ce graphe seront effectuées d'une manière périodique afin d'ajouter des éventuels changements dans la scène 3D. Enfin, l'algorithme peut fournir des sphères générées en ligne pour faciliter la navigation dans le graphe. Cela implique l'optimisation de l'algorithme pour accélérer le temps de construction du graphe. Plusieurs étapes de cet algorithme sont rapidement exécutées. Cependant, la construction de sphère augmentée à partir d'un maillage de point est une opération coûteuse en temps de calcul. C'est pour ces raisons, que nous avons consacré une partie de cette thèse à optimiser le temps de calcul de notre algorithme. Dans la suite de ce manuscrit, nous allons détailler les étapes d'optimisation et les résultats obtenus.

A. Étapes de génération d'un graphe de navigabilité : organisation du code

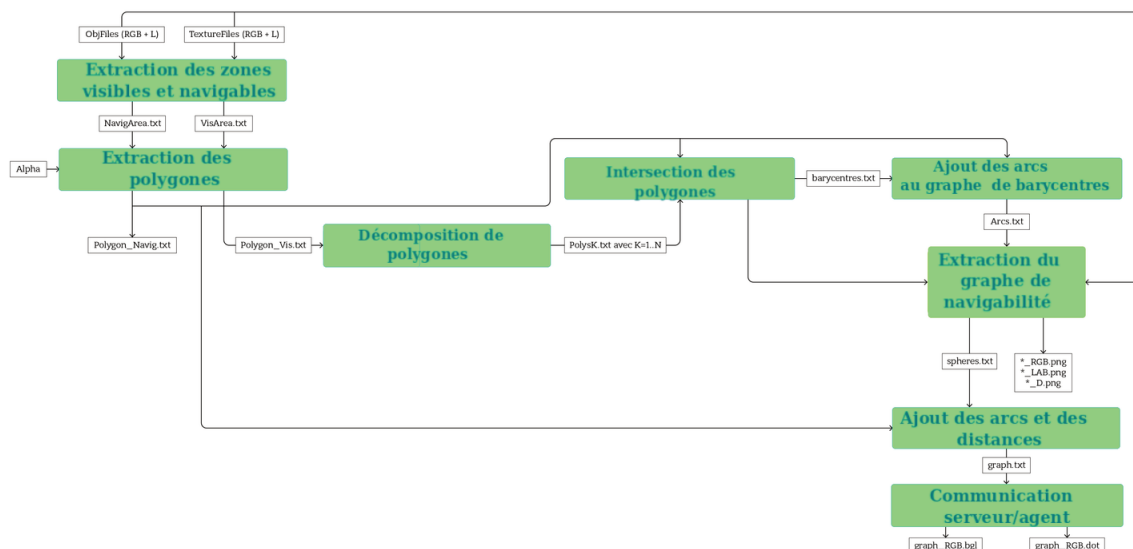


FIGURE 3.31 – Diagramme de construction d'un graphe de navigabilité

Un graphe de navigabilité est composé de plusieurs images sphériques reliées par des arcs qui représentent les distances entre les sphères. La génération d'un graphe de navigabilité nécessite l'exécution de plusieurs étapes. L'entrée de ce processus est un maillage triangulé, texturé et sémantisé. La sortie sera un graphe de navigabilité complet qui résume la carte initiale tout en gardant les informations nécessaires à la navigation. La fig 3.31 montre un diagramme qui résume les étapes de création d'un graphe de navigabilité. Ce diagramme montre aussi les différentes relations entre les composantes ainsi que les entrées/sorties de chaque étape. Les étapes de construction d'un graphe de navigabilité sont ordonnées comme suit :

- Étape 1 : extraction de la zone de navigabilité et de la zone de visibilité à partir d'un maillage texturé et sémantisé. Ce maillage est stocké sous la forme d'un fichier OBJ, un fichier MTL et plusieurs fichiers PNG pour la texturation (Annexe B).
- Étape 2 : extraction des polygones correspondant respectivement à la zone de navigabilité et à la zone de visibilité.
- Étape 3 : décomposition du polygone de visibilité en plusieurs polygones convexes.

- Étape 4 : calcul des barycentres qui représentent les intersections entre le polygone de navigabilité et les polygones convexes de visibilité.
- Étape 5 : construction d'un graphe des barycentres en ajoutant les arcs qui relient les barycentres entre eux tout en restant à l'intérieur de polygone de navigabilité.
- Étape 6 : calcul et génération des sphères dans les points de vue optimaux en se basant sur l'optimisation de deux critères : Entropie et Visibilité.
- Étape 7 : la génération d'un graphe de sphères complet en ajoutant les arcs entre les sphères et leurs distances respectives.

B. Optimisation du code

a. Calcul parallèle

Le (Centre Régional Informatique et d'Applications Numériques de Normandie) nous a fourni une machine puissante (40 cœurs de calcul, 16x 16Go RAM) pour tester notre algorithme avec les données de l'IGN. Le calcul parallèle nous permet de réduire le temps de construction d'une sphère. En utilisant les coordonnées sphériques, nous avons divisé l'image sphérique en 40 secteurs. Chaque secteur sera traité par un thread séparé. Le rôle de chaque thread consiste à lancer des rayons dans son secteur. Un secteur est défini par deux valeurs min_θ et max_θ . L'angle θ est compris entre 0 et 2π .

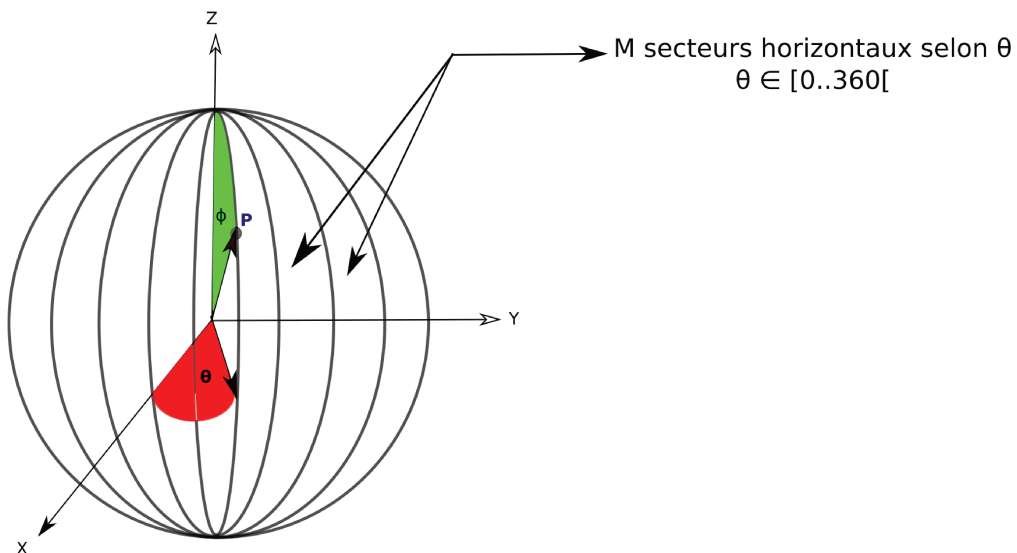


FIGURE 3.32 – Découpage d'une sphère selon l'angle θ en M secteurs horizontaux

b. Décomposition du maillage

Pour chaque pixel de chaque secteur de l'image, nous devons lancer un rayon. Une recherche exhaustive doit être effectuée à chaque lancer de rayon pour trouver le ou les triangles intersectés par ce rayon. Ceci provoque un temps de calcul important à chaque lancer du rayon. Prenons l'exemple d'un maillage contenant $2.7 M$ de triangles, un lancer du rayon prend environ $0.03 s$. Ce temps de calcul augmentera d'une façon significative lorsqu'on veut construire une sphère de résolution 2000×1000 pixels.

Pour optimiser le temps de construction de chaque secteur, nous avons proposé de réduire la zone de recherche dans le maillage. Pour construire une sphère de centre (X, Y, Z) nous avons décomposé le maillage projeté dans la géométrie de la sphère en M secteurs horizontaux et N secteurs verticaux selon les angles $\theta \in [0, 360[$ et $\phi \in [0, 180[$. Cette décomposition permet de chercher les triangles dans une zone limitée et éviter les zones de recherche inutiles.

La fig 3.32 et la fig 3.33 montre la décomposition d'une sphère ou d'un maillage en plusieurs secteurs suivant l'angle θ .

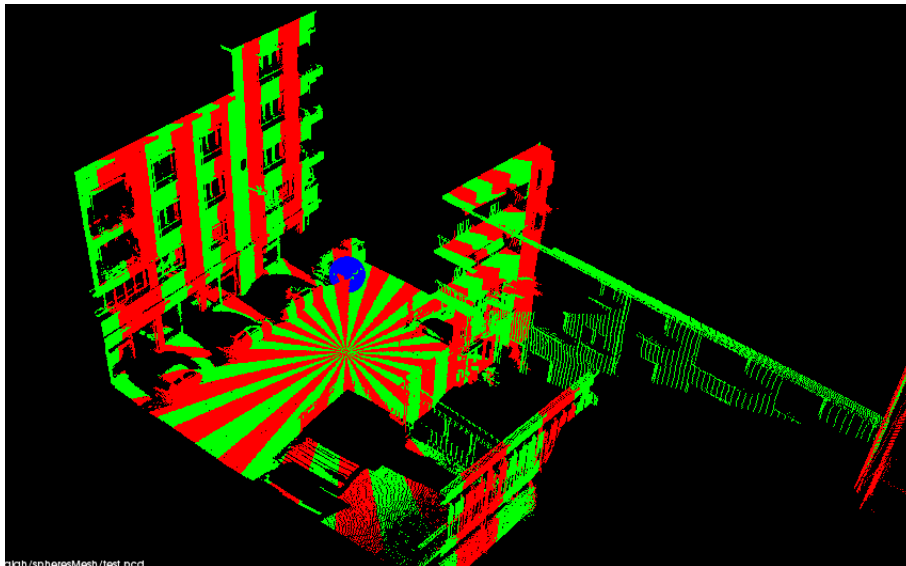


FIGURE 3.33 – Découpage d'un maillage selon l'angle θ

La décomposition du maillage en M secteurs horizontaux permet de réduire le temps de construction de l'image sphérique. Néanmoins, le temps obtenu reste important comme montre le tableau Tab 7. Nous avons proposé de continuer l'optimisation jusqu'à la décomposition du maillage selon l'angle ϕ pour diminuer le

temps d'exécution. Pour ce faire, nous avons proposé de décomposer chaque secteur horizontal en N secteurs verticaux selon l'angle $\phi \in [0, 180[$. La fig 3.34 montre la décomposition d'une sphère en $N * M$ secteurs.

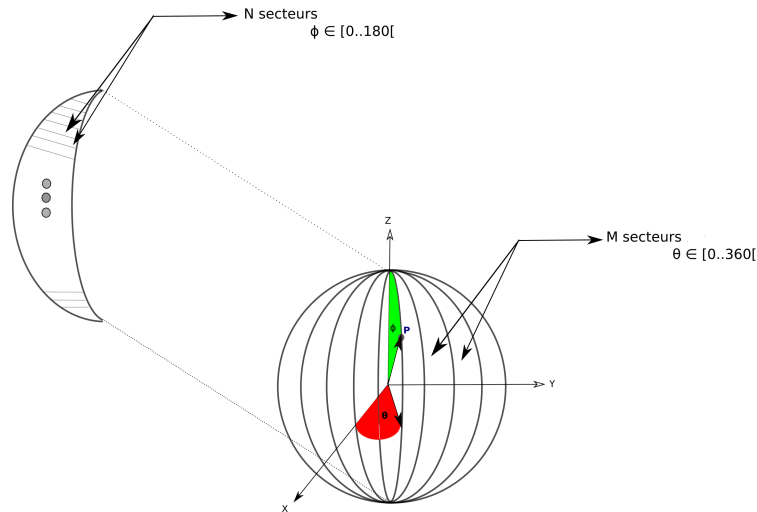


FIGURE 3.34 – Découpage d'une sphère selon l'angle ϕ et θ en $N * M$ secteurs verticaux

Pour chaque lancement de rayon R , une première étape consiste à sélectionner le secteur horizontal (M) qui correspond à la direction du rayon ainsi que les deux secteurs horizontaux voisins ($M + 1$) et ($M - 1$). La section des secteurs voisins permet de garantir le traitement des rayons qui se trouvent sur les bords des secteurs. En deuxième lieu, de la même manière, nous allons sélectionner les secteurs verticaux N , $N - 1$ et $N + 1$. La recherche du triangle intersecté par le rayon R sera réalisée progressivement sur une grille de taille $3 * 3$ formée par les intersections entre les secteurs sélectionnés précédemment. La recherche d'intersection est effectuée progressivement. En effet, la recherche commence dans la case au centre de la grille. Si aucune intersection n'a été trouvée la recherche cible la case suivante et ainsi de suite jusqu'à parcourir toutes les cases de la grille. La fig 3.35 montre la sélection des secteurs formant une grille de recherche pour un lancement d'un rayon R .

Nous avons testé la construction d'une image sphérique sur une partie des données de l'IGN. Ce maillage contient $1,7 M$ sommets formant $2,7 M$ triangles. Nous avons testé deux résolutions de l'image sphérique de sortie. Le premier test consiste à construire une sphère en utilisant la recherche exhaustive dans tout le maillage. Dans le deuxième test, nous avons appliqué la décomposition en secteurs horizontaux comme nous l'avons décrit auparavant. Le dernier test consiste à construire

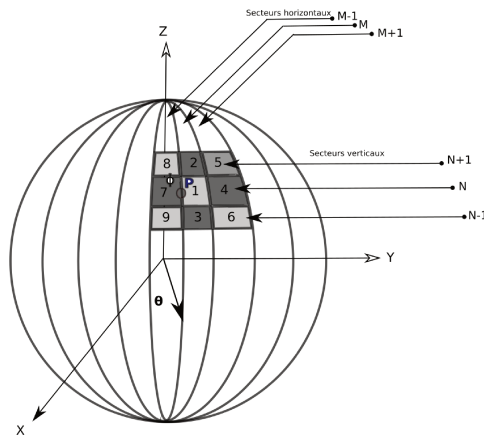


FIGURE 3.35 – Grille de recherche pour un rayon donné

Résolution	T initial (s)	T optimisé étape 1 (s)	T optimisé étape 2 (s)	Facteur de réduction
360*180	1320	900	70	18
2000*1000	43200	7200	220	196

TABLE 7 – Temps de calcul (s) de construction d'une sphère avant et après l'optimisation du code

une sphère en appliquant la décomposition en secteurs horizontaux puis verticaux. Les résultats sont présentés dans le tableau 7. Le temps de construction d'une sphère de haute résolution (2000 * 1000) est passé de 12 heures d'exécution à 220 secondes. Cette optimisation du temps d'exécution permet d'accélérer le calcul du graphe de navigabilité et la génération des sphères en temps réel.

6. Conclusion

Dans ce chapitre, les algorithmes de résumé de carte et de localisation en ligne ont été utilisés dans le cadre d'une application de navigation autonome. Plusieurs scénarios possibles ont été évoqués. Une séquence d'images d'agent a été acquise afin de tester la localisation dans des conditions plus réalistes. La robustesse des algorithmes a pu être mise à l'épreuve sur des séquences d'images synthétisées et réelles. Pour améliorer la robustesse et la précision de la localisation visuelle, nous exploitons l'information sémantique. En appliquant un filtrage sémantique avant la localisation, nous avons amélioré la précision de recalage. Il serait possible de tester la robustesse de ces algorithmes face aux changements de luminosité ou des

changements dans la structure de l'environnement. Cela implique l'utilisation de la brique mise à jour de référentiel présentée dans la première partie de ce manuscrit. Lors de la phase de localisation en ligne, le graphe de navigabilité est utilisé de manière efficace pour sélectionner un nombre réduit de sphères RGBD-L . Cela permet de réduire le nombre de pixels utilisés dans la fonction de recalage et d'effectuer les calculs en temps-réel.

Chapitre IV.

Conclusion générale et perspectives

1. Conclusion

Dans ces travaux, une représentation sphérique complète de l'environnement contenant plusieurs types d'information a été présentée. Ce modèle est constitué d'un graphe d'images sphériques augmentées par la profondeur et la sémantique. Chaque nœud du graphe contient une sphère augmentée RGBD-L disponible en une résolution suffisante pour la navigation. Les nœuds du graphe sont connectés par une pose 3D précise dans un repère global. Les arcs du graphe sont pondérés par les distances entre les sphères. Les données denses contenues dans les sphères permettent d'utiliser des techniques de localisations visuelles directes. Pour construire ce modèle, deux nouveaux critères ont été définis et utilisés pour placer précisément et d'une façon optimale les nœuds du graphe dans l'espace.

La méthode proposée permet de modéliser et résumer automatiquement, avec une représentation dense et riche, de larges environnements urbains. Elle permet de résumer une carte volumineuse tout en maintenant un maximum d'information utile à la navigation. Le graphe de sphères obtenu lors de cette phase de résumé est utilisé pour localiser en temps réel un agent naviguant dans le voisinage du graphe.

Pour accomplir cette tâche, l'algorithme Go-ICP a été adapté et utilisé pour effectuer le recalage 3D et localiser l'agent. L'utilisation d'un graphe réduit de sphères permet de simplifier ces tâches de navigation. En fait, le temps de calculs de l'alignement est accéléré suite à l'utilisation d'un nombre réduit de pixels, tout en maintenant l'observabilité complète de l'environnement.

L'avantage de cette approche est que la sélection des pixels est effectuée hors-ligne lors de la construction du graphe de navigabilité. Lors de la localisation en ligne, seules les sphères les plus proches sont utilisées ce qui permet d'obtenir une localisation en temps-réel.

Pour améliorer la robustesse de la méthode de localisation visuelle en ligne, une approche de filtrage sémantique a été proposée. La localisation d'un agent dans le graphe de sphères se fait sur deux étapes. La localisation grossière permet de trouver la sphère la plus proche de l'agent en utilisant plusieurs modalités. La localisation fine consiste à minimiser l'erreur géométrique, ce qui permet d'améliorer la robustesse aux changements de luminosité.

2. Perspectives

Les méthodes de résumé et de localisation ont été validées sur un certain nombre d'expérimentations, qui ont permis d'identifier plusieurs éléments améliorables. Il serait intéressant, d'ajouter d'autres critères d'optimisation dans le positionnement des sphères. En effet, nous pouvons ajouter un critère de recouvrement entre les sphères pour améliorer la précision de la localisation. Il pourrait être intéressant de tester différentes valeurs des poids α_i dans le calcul d'Entropie afin de favoriser ou pénaliser certaines parties de l'équation.

Il serait possible aussi d'ajouter d'autres types d'information sur les arcs dans le graphe pour faciliter les mises à jour de référentiel. Pour améliorer la robustesse de la méthode de localisation en ligne, il serait intéressant d'envisager la création des sphères RGBD-L supplémentaires en ligne. En effet, lorsque l'agent ne parvient pas à se localiser par rapport à la sphère de référence, nous pouvons créer en ligne d'autres sphères.

Pour ce faire, il est possible d'améliorer encore le temps de construction d'une sphère RGBD-L. Par exemple, nous pouvons exécuter l'algorithme de lancer de rayons sur le GPU. Il serait intéressant de proposer plusieurs types de graphe de navigabilité suivant le type de système de navigation.

Nous pouvons proposer un parcours adapté à suivre par l'agent en fonction de ses capacités de mobilité : voiture, piéton, vélo ou robot mobile. Cela impose l'amélioration de la sémantisation du maillage ou nuage de départ. En effet, il va falloir ajouter d'autres classes sémantiques comme les passages piétons. Pour évaluer le graphe de navigabilité, il serait intéressant de l'intégrer dans une application de navigation autonome. Nous pouvons le tester avec un véhicule équipé de plusieurs types de caméra en conditions réelles dans un environnement urbain.

Un autre axe d'amélioration pourrait être l'évaluation du graphe de navigabilité avec d'autres algorithmes de recalage en utilisant l'information photométrique. En effet, cela peut nous aider à mieux évaluer la qualité de résumé obtenu. Il serait intéressant d'effectuer le recalage d'une manière continue et d'intégrer l'aspect temporel. En effet, jusqu'à maintenant le recalage est effectué séparément pour chaque image agent. Pour ce faire, il est possible d'utiliser l'odométrie visuelle et de mémoriser la dernière localisation de l'agent. Cette localisation sera utilisée pour initialiser la position de l'agent pour l'image suivante.

Chapitre V.

Annexe A : détecteurs et descripteurs de points d'intérêt

1. Harris

La méthode de Harris [46] est une méthode basée sur la détection des coins et des contours (bords d'objets). Ce type de détecteur se focalise sur la détection des changements d'intensité dans les directions horizontale et verticale. Ces caractéristiques peuvent être utilisées dans l'analyse de la forme et du mouvement des objets. Les points d'intérêt peuvent être détectés directement à partir de la variation des niveaux de gris dans les images. Cet algorithme utilise la matrice d'auto-corrélation qui décrit la distribution du gradient du voisinage d'un point afin de chercher les maximums locaux supérieurs à un certain seuil. Ce détecteur est parmi les détecteurs les plus robustes dans la littérature. En effet, il est invariant aux transformations de translation et de rotation et stable aux changements d'illuminations.

2. Harris3D

Une adaptation de Harris 2D permet de détecter les points d'intérêts dans un modèle 3D. Pour le cas d'un nuage de points 3D, l'ajustement de cette méthode est assuré par la bibliothèque PCL (Point Cloud Library) en remplaçant les gradients des images (2D) par les normales de surface (3D). Ce détecteur calcule la matrice de covariance autour de chaque point dans un voisinage 3×3 . La réponse mesurée à chaque pixel de coordonnées (x, y, z) est alors définie par l'équation 5.1 :

$$r(x, y, z) = \det(\text{Cov}(x, y, z)) - k(\text{trace}(\text{Cov}(x, y, z)))^2 \quad (5.1)$$

Dans la bibliothèque PCL, nous pouvons trouver deux variantes du détecteur Harris3D : ceux-ci sont appelés "Lowe" et "Noble". La différence entre elles se manifeste par les fonctions qui définissent la réponse du détecteur :

Pour la variante "Lowe", la réponse est donnée par :

$$r(x, y, z) = \frac{\det(\text{Cov}(x, y, z))}{(\text{trace}(\text{Cov}(x, y, z)))^2} \quad (5.2)$$

Pour la variante "Noble", la réponse est donnée par :

$$r(x, y, z) = \frac{\det(\text{Cov}(x, y, z))}{\text{trace}(\text{Cov}(x, y, z))} \quad (5.3)$$

3. SUSAN

Le détecteur SUSAN (Smallest Univalve Segment Assimilating Nucleus) [111], est un des algorithmes qui se basent directement sur les valeurs d'intensité de l'image sans la segmenter à l'avance. Cet algorithme consiste à détecter les coins dans une image. Il mesure la similitude d'intensité entre un pixel appelé "Nucleus" ou "noyau" et son voisinage. Le voisinage est défini par un cercle centré autour du noyau. Les pixels du voisinage dont la luminosité est à peu près équivalente à celle du noyau forment la zone appelée USAN «Univalve Segment Assimilating Nucleus». Pour décider si un pixel est un coin ou non, il faut comparer la différence de luminosité entre le noyau et ses voisins. Pour ce faire, il faut estimer cette différence pour chaque pixel du masque (équation 5.4) :

$$c(r, r_0) = \begin{cases} 1 & \text{si } \|I(r) - T(r_0)\| \leq t \\ 0 & \text{sinon} \end{cases} \quad (5.4)$$

où t est un seuil de différence d'intensité. r est le noyau et r_0 est un pixel du son voisinage. c représente la sortie de la comparaison et $I(x)$ est l'intensité de tout pixel x .

4. FAST

Le détecteur FAST (Features from Accelerated Segment Test) est un détecteur de coins qui a été introduit en 2006 [97]. Ce détecteur se base sur la même idée que le détecteur SUSAN décrit précédemment.

L'algorithme FAST est le suivant :

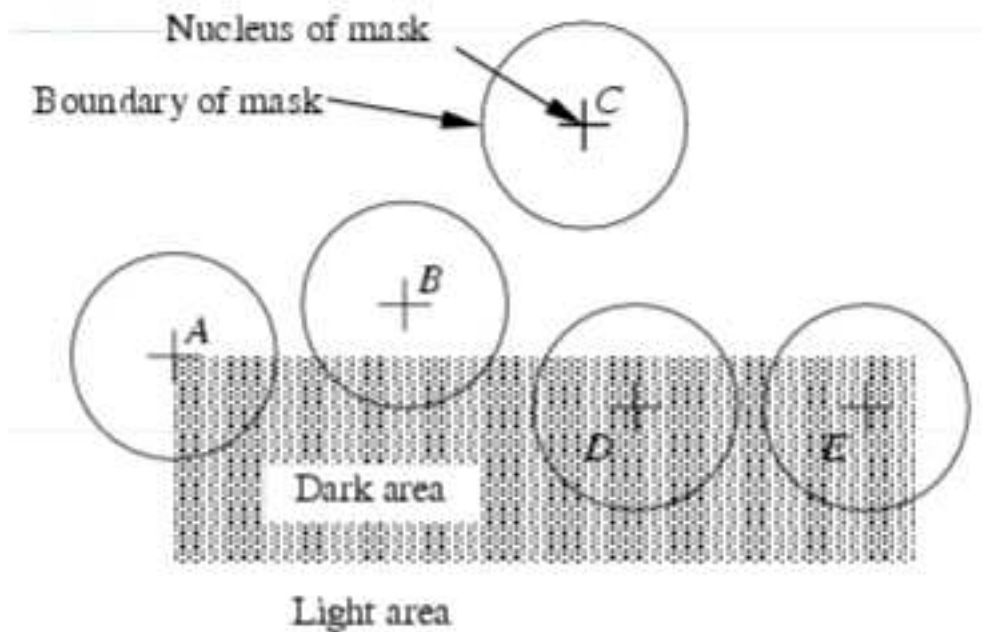


FIGURE 5.1 – Un exemple de détecteur de coins : SUSAN

- Sélectionner un pixel p dans l'image d'intensité I_p .
- Définir un seuil d'intensité T
- Considérer un cercle de 16 pixels entourant le pixel p en utilisant l'algorithme de Bresenham [15].
- Les «N» pixels adjacents parmi les 16 doivent être supérieurs ou inférieurs à I_p de la valeur T , si le pixel doit être détecté comme un point d'intérêt. (Les auteurs ont utilisé $N = 12$ dans la première version de l'algorithme)
- Pour que l'algorithme soit rapide, comparons d'abord l'intensité des pixels 1, 5, 9 et 13 du cercle avec I_p . Comme le montre la fig 5.2, au moins trois de ces quatre pixels doivent satisfaire aux critères de seuil correspondant au point d'intérêt. Si au moins trois des quatre valeurs de pixels I_1, I_5, I_9 et I_{13} ne sont pas supérieures ou inférieures à $I_p + T$, alors p n'est pas un point d'intérêt. Sinon, si au moins trois des pixels sont supérieurs ou inférieurs à $I_p + T$, alors une vérification du critère 12 pixels adjacents est effectuée.
- Répéter la procédure pour tous les pixels dans l'image.

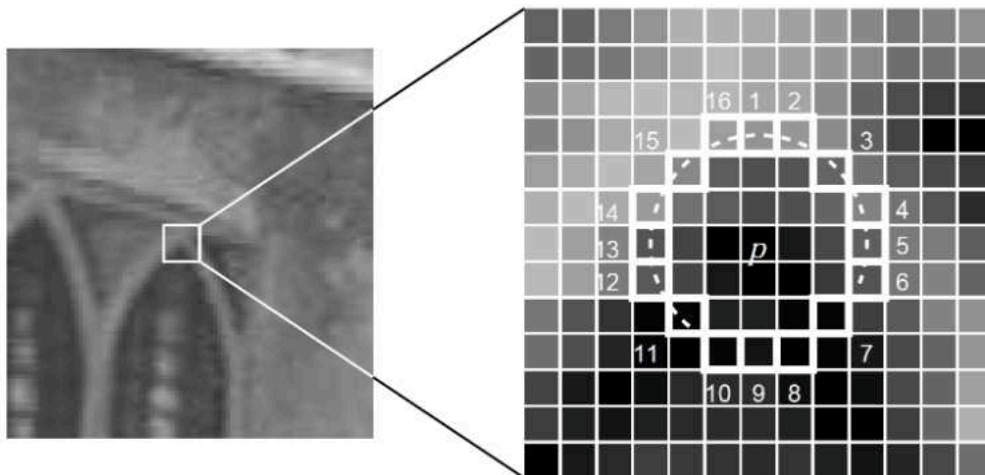


FIGURE 5.2 – Un exemple de détecteur de coins : FAST

5. SIFT

Nous avons vu des détecteurs de coin tels que Harris. Ils sont invariants en rotation. Cependant, ils sont sensibles à l'échelle. Un coin peut ne pas être un coin si l'image est redimensionnée. Le détecteur SIFT (Scale Invariant Feature Transform) a été initialement proposé par Lowe et al. [73, 71, 72] pour détecter les points d'intérêt dans une image 2D et résoudre le problème de l'invariance en échelle. Pour ce faire, un espace d'échelle est construit pour garantir l'invariance d'échelle. Un point p de l'image est caractérisé par ses coordonnées (x, y) et par son facteur échelle σ . La première étape de cet algorithme est la détection d'extremum dans l'espace des échelles (fig 5.3). Pour ce faire, une étape de lissage d'image I est effectuée afin d'estomper les détails trop petits dans I . Ce lissage L est le résultat de la convolution d'une image I par un filtre gaussien G de paramètre σ (équation 5.5) :

$$L(x, y, \lambda) = G(x, y, \sigma) \otimes I(x, y) \quad (5.5)$$

La détection des points d'intérêt se fait en étudiant l'image appelée différences de gaussiennes (DoG) définie comme suit équation 5.6 :

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (5.6)$$

où k est un paramètre fixe de l'algorithme qui dépend de la finesse de la discrétisation de l'espace des échelles.

Dans cette image ne persistent plus que les objets observables dans des facteurs d'échelle qui varient entre σ et $k\sigma$. Un point d'intérêt est défini comme un extre-

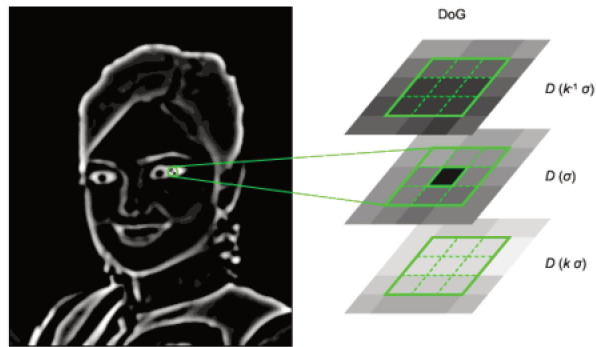


FIGURE 5.3 – Exemple de détection d’extrémums dans l’espace des échelles

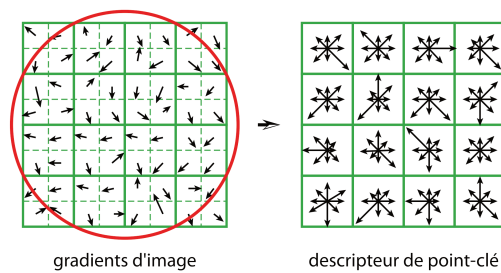


FIGURE 5.4 – Construction du descripteur SIFT

mum du DoG par rapport à ses voisins. Un traitement supplémentaire est effectué pour éliminer les points d’intérêt de faible contraste.

L’étape suivante consiste à affecter d’une orientation à chaque point d’intérêt en se basant sur la direction des gradients dans le voisinage du point.

Une fois les points d’intérêt trouvés et associés avec ses orientations, leurs descripteurs sont ensuite calculés (fig 5.4).

Les descripteurs associés aux points d’intérêt SIFT sont des histogrammes des orientations locales autour du point d’intérêt :

- On divise l’espace autour de chaque point d’intérêt (x, y) en N^2 carrés 4×4 .
- On calcule le gradient (G_x, G_y) pour les $4 \times 4 \times N^2$ points
- Pour chaque carré 4×4 , on calcule un histogramme des orientations quantifiées en 8 directions
- Pour être invariant en rotation : l’orientation locale du point d’intérêt (x, y) est utilisée comme origine (orientation nulle) des histogrammes.

6. SIFT3D

L'évolution de l'algorithme original SIFT (2D) vers le nouveau SIFT (3D) a été réalisée dans [39].

7. SURF

Le détecteur SURF, proposé par Bayet al. [8], il est partiellement inspiré par le descripteur SIFT. SURF est plus rapide et robuste pour différentes transformations d'images que le détecteur SIFT. Cela est dû au fait que le détecteur SURF utilise un calcul d'approximation des dérivées gaussiennes partielles de second ordre dans SIFT. Pour calculer les descripteurs, SURF utilise des réponses d'ondelettes de Haar.

8. MSER

Le détecteur MSER (Maximally stable extremal regions) a été proposé en 2004 par Matas et al. [76]. MSER est une méthode de détection des régions dans l'image. L'algorithme MSER extrait d'une image un certain nombre de régions covariantes, appelées MSER. Un MSER est une composante connectée stable de certains ensembles de niveaux de gris de l'image. Le mot "extremal" fait référence à la propriété selon laquelle tous les pixels du MSER ont une intensité supérieure (régions lumineuses) ou inférieure (régions sombres) par rapport à tous les pixels sur son contour. Les régions détectées doivent être stables quand on augmente ce seuil.

9. CenSurE

Le détecteur CenSurE (Center Surround Extrema) a été proposé en 2008 [3]. Une version modifiée de ce détecteur a été proposée par la bibliothèque OpenCV appelée STAR. La principale motivation derrière le développement de ce détecteur était d'obtenir une résolution spatiale complète dans un détecteur multi-échelles. Lorsque les détecteurs SIFT et SURF effectuent un sous-échantillonnage, la précision de la localisation des points d'intérêt est affectée. Le détecteur CenSurE utilise une approximation à deux niveaux de filtre LoG. Les formes circulaires du masque sont approximées par des formes permettant de combiner la simplicité et l'invariance en rotation. Le détecteur CenSurE utilise des masques de filtre de forme octogonale,

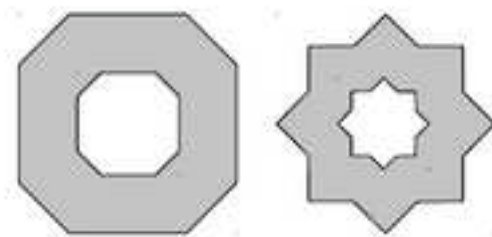


FIGURE 5.5 – Filtres utilisés par CenSurE (à gauche) et par STAR (à droite)

tandis que le masque utilisé dans STAR est constitué de deux carrés, dont l'un est pivoté de 45 degrés comme le montre la fig 5.5.

10. AGAST

Le détecteur AGAST (Adaptive and Generic Corner Detection Based on the Accelerated Segment Test) a été proposé en 2010 [74]. AGAST est basé sur le même critère de fonctionnalité que FAST, mais utilise un arbre de décision différent. AGAST est entraîné sur une base de données incluant toutes les combinaisons possibles de 16 pixels sur le cercle. Cela garantit que l'arbre de décision fonctionne dans tous les environnements. De plus, AGAST introduit un algorithme de changement d'arborescence dynamique, qui modifie automatiquement les arborescences de décision. Un arbre est pour les zones homogènes et l'autre pour les zones hétérogènes. De cette manière, la performance d'AGAST augmente pour les scènes aléatoires. En combinant ces deux améliorations, AGAST fonctionne dans tous les environnements arbitraires sans aucune étape d'apprentissage.

11. PFH

Le descripteur PFH (Point Feature Histograms) a été proposé en 2008 [99] pour déterminer les caractéristiques géométriques dans un nuage de points 3D. PFH permet de fournir une représentation de la géométrie autour d'un point spécifique en se basant sur les normales de surface et les estimations de courbure. Le but de la formulation PFH est de coder les propriétés géométriques de k voisins d'un point p . Pour ce faire, le descripteur consiste à calculer la courbure moyenne autour du point p à l'aide d'un histogramme multidimensionnel de valeurs. Cet histogramme est une signature informative pour la représentation des points 3D, il

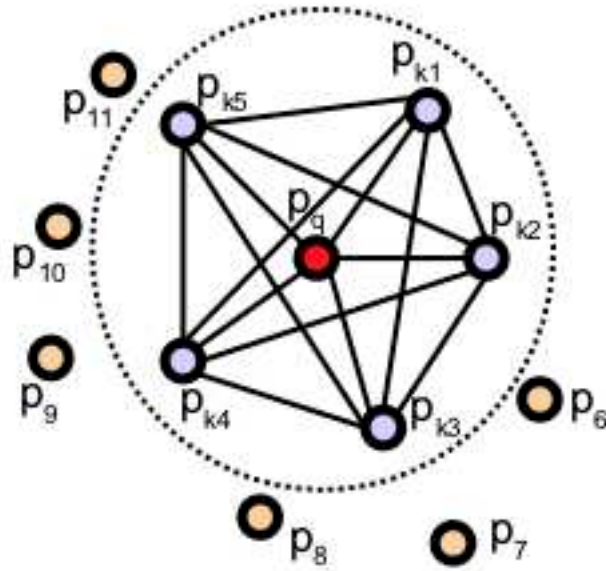


FIGURE 5.6 – Voisinage d'un point requête p_q (en rouge) et ses voisins (en bleu) sont entièrement interconnectés dans un maillage

est invariant par rapport à la pose $6D$ et s'adapte parfaitement aux différentes densités d'échantillonnage ou niveaux de bruit présents dans le voisinage. Cette représentation est basée sur les relations entre les points dans son voisinage de k points et leurs normales de surface estimées. Ce descripteur consiste de calculer les variations de surface en prenant en compte toutes les interactions entre les directions des normales estimées. L'hyperespace résultant dépend donc de la qualité des estimations de la surface normale en chaque point.

La fig 5.6 présente le voisinage d'un point marqué en rouge p_q et le calcul de la signature PFH. Le point p_q est placé au centre d'un cercle (sphère en 3D) de rayon r , ainsi que de ses k voisins dont les distances sont inférieures au rayon r . Les points sont entièrement interconnectés dans un maillage. Le descripteur PFH est calculé comme un histogramme de relations entre toutes les paires de points du voisinage et présente donc une complexité de calcul $O(k^2)$.

Pour calculer la différence relative entre deux points p_i et p_j et leurs normales associées n_i et n_j , un repère est défini comme le montre la fig 5.7.

$$\left\{ \begin{array}{l} u = n_s \\ v = u * \frac{p_t - p_s}{\|p_t - p_s\|} \\ w = u * v \end{array} \right. \quad (5.7)$$

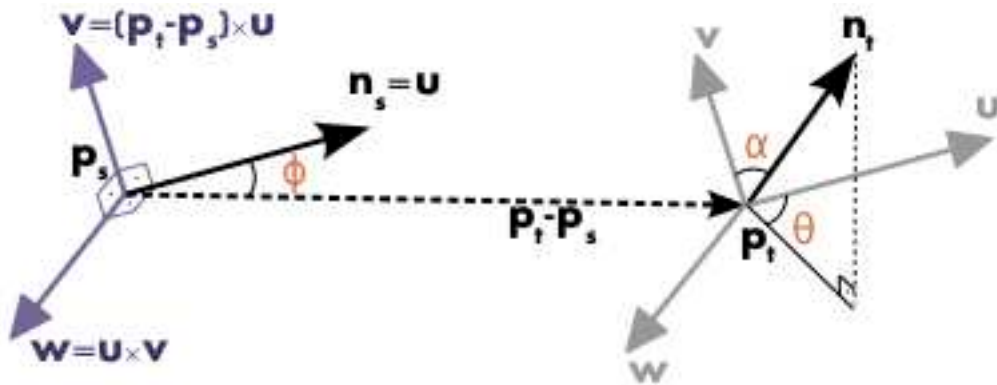


FIGURE 5.7 – Repère (u, v, w) pour le calcul des histogrammes PFH

En utilisant le repère uvw (équation 5.7), la différence entre les deux normales n_s et n_t peut être exprimée sous la forme d'un ensemble de caractéristiques angulaires, comme suit (équation 5.8) :

$$\begin{cases} \alpha = v.n_t \\ \psi = u.\frac{(p_t - p_s)}{d} \\ \theta = \arctan(w.n_t, u.n_t) \end{cases} \quad (5.8)$$

où d est la distance euclidienne entre les deux points p_s et p_t . Le quadruplet $(\alpha, \psi, \theta, d)$ est calculé pour chaque paire de deux points dans le voisinage, réduisant ainsi les 12 valeurs (x, y, z) et les informations normales des deux points à 4 paramètres.

12. FPFH

La complexité du calcul de l'histogramme avec PFH pour un nuage de points donné P avec n points est $O(nk^2)$, où k est le nombre de voisins pour chaque point p dans P . Pour les applications en temps réel, le calcul d'histogramme dans un environnement dense peut présenter une limitation de cet algorithme. Pour accélérer le temps de calcul de PFH, une nouvelle version FPFH (Fast Point Feature Histograms) a été proposée en 2009 [98]. Cette nouvelle version propose une simplification de la formulation de PFH pour réduire la complexité de calcul de l'algorithme à $O(nk)$, tout en conservant l'essentiel du pouvoir discriminant de la PFH.

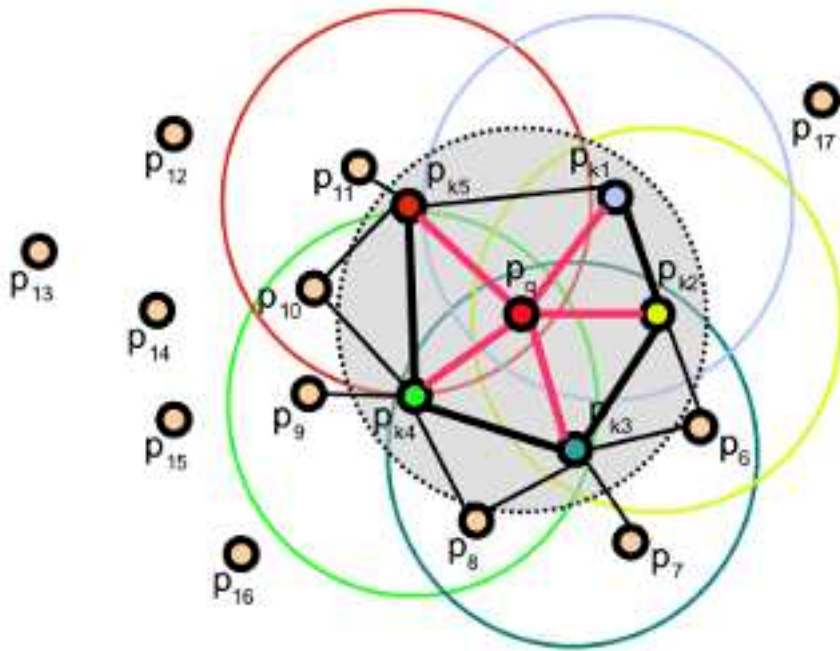


FIGURE 5.8 – Voisinage d'un point requête p_q (en rouge) et ses voisins directs (cercle gris)

La principale différence entre PFH et FPFH est que FPFH n'interconnecte pas complètement tous les voisins de p_q comme le montre la fig 5.8. En effet chaque point requête p_q (en rouge) est connecté uniquement à ses voisins directs (cercle gris). Chaque voisin direct est connecté à ses propres voisins et les histogrammes résultants sont pondérés avec l'histogramme du point de requête pour former le FPFH.

Chapitre VI.

Annexe B : organisation des données IGN

1. Organisation des données IGN

Les données de l'IGN sont sous la forme d'un maillage triangulé et texturé. Ce maillage est stocké dans des fichiers OBJ. OBJ est un format de fichier contenant la description d'une géométrie 3D. Les fichiers OBJ sont au format ASCII. Le format OBJ sert à représenter un modèle 3D texturé. Chaque fichier OBJ est accompagné par un seul fichier MTL cependant un fichier MTL peut être utilisé par plusieurs fichiers OBJ.

- Fichier .OBJ : qui donne toutes les informations sur les sommets et les faces.
- Fichier .MTL : (Material Template Library) qui contient la définition des matériaux utilisés pour texturer le modèle 3D (la coloration, la texture, les paramètres de réflexion optique...)

A. Organisation d'un fichier OBJ

Un fichier OBJ est composé de 5 sections :

- Section 1 : nom du fichier MTL
- Section 2 : les coordonnées des sommets (X, Y, Z)
- Section 3 : les normales des sommets
- Section 4 : les coordonnées de texture des sommets (V, T)
- Section 5 : les triplets d'indices de sommets qui forment les triangles

B. Organisation d'un fichier MTL

Un fichier MTL est composé de 5 sections :

- Section 1 : *newmtl* indique le début d'un nouveau matériel
- Section 2 : *Ka* nous donne la couleur ambiante (la couleur de l'objet sans lumière directe)
- Section 3 : *Kd* est utilisé pour la couleur diffuse (la couleur de l'objet sous lumière blanche)
- Section 4 : *Ks* pour la couleur spéculaire (specular)
- Section 5 : *Ke* pour la couleur émissive (emissive)
- Section 6 : *Ni* pour la densité optique
- Section 7 : *Ns* pour le specular entre 0 et 100
- Section 8 : *d* pour la transparence entre 0 et 1 (aucune transparence)
- Section 6 : *illum* pour les paramètres de lumières
- Section 10 : *map_kd* (ks, ka) pour la texture utilisé diffuse (specular, ambiante)

C. Atlas de texture

Plusieurs Atlas de texture ont été utilisés pour texturer le maillage 3D. Ces atlas sont une sorte de cartes où chaque morceau de surface (image) est aplati et positionné de façon à minimiser son espace total. L'intérêt derrière, c'est que pour des raisons d'efficacité et de rapidité, il est plus judicieux de passer à la carte graphique un ensemble des "patch" de texture à la fois que les passer un par un. On les sauvegarde donc dans une structure d'Atlas de texture. Chaque sommet est caractérisé par ces coordonnées (X, Y, Z) dans un repère terrain local défini comme du Lambert93 (x vers l'est, y vers le nord, z vers le haut) par rapport au pivot E=561400 N=6929000 (en mètres), ces coordonnées de texture (V, T) qui correspondent à la ligne et la colonne dans l'atlas de texture. Une paire de coordonnées comprises entre 0 et 1 est associée à chaque vertex (sommet de triangle), qui correspond à une zone de l'atlas de texture. La fig 6.2 montre un exemple de texturation d'un triangle. Les données sémantiques sont stockées de la même façon dans un fichier OBJ séparé. Chaque sommet de triangle possède un code couleurs de la classe sémantique. Fig 6.1 montre un exemple d'organisation des données IGN. L'inconvénient de cette organisation est le volume important de mémoire nécessaire pour stocker toutes les données. Cette organisation est très coûteuse en terme de mémoire en effet, l'information sur les sommets est stockée doublement. Pour chaque zone de l'environnement nous avons deux modèles OBJ. Le premier

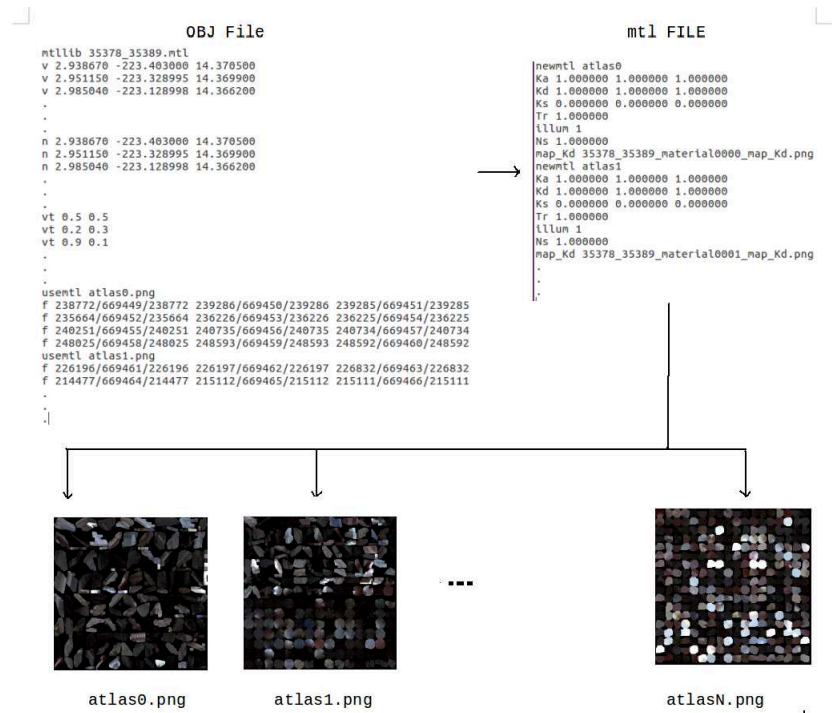


FIGURE 6.1 – Organisation des données IGN

permet de stocker l'information couleur pour chaque sommet et le deuxième permet de stocker l'information sémantique. C'est pourquoi il est devenu nécessaire de trouver une solution d'indexation de données permettant de limiter la zone de recherche à chaque lancé de rayon afin de gagner en mémoire ainsi en temps de calcul.

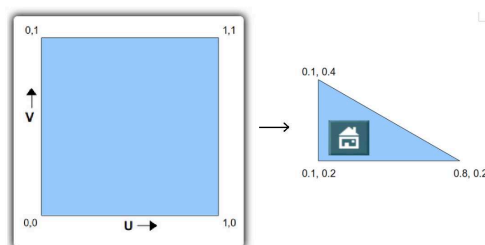


FIGURE 6.2 – Exemple de texturation

Chapitre VII.

Glossaire

LITIS : Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes
INSA : Institut National des sciences appliquées
ADAS : Advanced Driver Assistance Systtemes
LIDAR : Light Detection and Ranging
GPS : Global Positionning System
RTK : Real Time Kinematic GPS
SLAM : Localisation et Cartographie Simultanées
IMU : Inertial Measurement Units
UR : Universite de Rouen
ANR : Agence Nationale de la Recherche
pLaTINUM : porject Long Term MappINg for Urban Mobility
IGN : Institut géographique national
DOF : Degrees of freedom
MAD : median absolute deviation
SIFT : Scale Invariant Feature Transform
SUSAN : Smallest Univalue Segment Assimilating Nucleus
PCL : Point Cloud Library
CRIANN : Centre Regional Informatique et d'Applications Numeriques de Normandie

Chapitre VIII.

Bibliographie

Bibliographie

- [1] flir. <https://www.flir.com/iis/machine-vision/spherical-vision-systems>.
- [2] Ign. <http://www.ign.fr/institut/innovation/stereopolis>.
- [3] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure : Center surround extremas for realtime feature detection and matching. *European Conference on Computer Vision*, pages 102–115, 2008.
- [4] Jae-Kyun Ahn, Kyu-Yul Lee, Jae-Young Sim, and Chang-Su Kim. Large-scale 3d point cloud compression using adaptive radial distance prediction in hybrid coordinate domains. *IEEE Journal of Selected Topics in Signal Processing*, 9(3) :422–434, 2015.
- [5] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad : Cnn architecture for weakly supervised place recognition. pages 5297–5307, 2016.
- [6] Clemens Arth, Daniel Wagner, Manfred Klopschitz, Arnold Irschara, and Dieter Schmalstieg. Wide area localization on mobile phones. *IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2009*, pages 73–82, 2009.
- [7] Simon Baker and Iain Matthews. Equivalence and efficiency of image alignment algorithms. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1 :I–1090, 2001.
- [8] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. *European conference on computer vision*, pages 404–417, 2006.
- [9] Mark Bayazit. Decomposing polygons into convex regions. <https://mpen.ca/406/bayazit>.
- [10] Selim Benhimane, Alexander Ladikos, Vincent Lepetit, and Nassir Navab. Linear and quadratic subsets for template-based tracking. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, 2007.
- [11] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. *Sensor Fusion IV : Control Paradigms and Data Structures*, 1611 :586–607, 1992.

- [12] Ali Borji. Boosting bottom-up and top-down visual features for saliency estimation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 438–445, 2012.
- [13] Ali Borji, Dicky N Sihite, and Laurent Itti. What stands out in a scene? a study of human explicit saliency judgment. *Vision research*, 91 :62–77, 2013.
- [14] Mohamed Boussaha, Eduardo Fernandez-Moral, Bruno Vallet, and Patrick Rives. On the production of semantic and textured 3d meshes of large scale urban environments from mobile mapping images and lidar scans. *Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP), Marne-la-Vallée, France*, 2018.
- [15] Jack Bresenham. A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM*, 20(2) :100–106, 1977.
- [16] Neil Bruce and John Tsotsos. Saliency based on information maximization. *Advances in neural information processing systems*, pages 155–162, 2006.
- [17] Guy Thomas Buswell. How people look at pictures : a study of the psychology and perception in art. 1935.
- [18] Svante Carlsson, Håkan Jonsson, and Bengt J Nilsson. Finding the shortest watchman route in a simple polygon. *Discrete & Computational Geometry*, 22(3) :377–402, 1999.
- [19] Alexandre Chapoulie, Patrick Rives, and David Filliat. Topological segmentation of indoors/outdoors sequences of spherical views. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4288–4295, 2012.
- [20] Alexandre Chapoulie, Patrick Rives, and David Filliat. Appearance-based segmentation of indoors/outdoors sequences of spherical views. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1946–1951, 2013.
- [21] Raja Chatila and Jean-Paul Laumond. Position referencing and consistent world modeling for mobile robots. *IEEE International Conference on Robotics and Automation*, 2 :138–145, 1985.
- [22] Xiaobai Chen, Abulhair Saparov, Bill Pang, and Thomas Funkhouser. Schelling points on 3d surface meshes. *ACM Transactions on Graphics (TOG)*, 31(4) :29, 2012.
- [23] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3) :145–155, 1992.
- [24] Dana Cobzas, Hong Zhang, and Martin Jagersand. Image-based localization with depth-enhanced image map. *IEEE International Conference on Robotics and Automation, ICRA '03.*, 2 :1570–1575, 2003.

- [25] James L Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. *IEEE International Conference on Robotics and Automation*, pages 674–680, 1989.
- [26] Olivier Devillers and P-M Gandoin. Geometric compression for interactive transmission. *Visualization Research*, pages 319–326, 2000.
- [27] Zilong Dong, Guofeng Zhang, Jiaya Jia, and Hujun Bao. Keyframe-based real-time camera tracking. *2009 IEEE 12th international conference on computer vision*, pages 1538–1545, 2009.
- [28] Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph SB Mitchell. Touring a sequence of polygons. *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 473–482, 2003.
- [29] Romain Drouilly, Patrick Rives, and Benoit Morisset. Fast hybrid relocation in large scale metric-topologic-semantic map. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1839–1845, 2014.
- [30] Marcin Dymczyk, Simon Lynen, Titus Cieslewski, Michael Bosse, Roland Siegwart, and Paul Furgale. The gist of maps-summarizing experience for lifelong localization. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2767–2773, 2015.
- [31] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on information theory*, 29(4) :551–559, 1983.
- [32] Alberto Elfes. A tessellated probabilistic representation for spatial robot perception and navigation. *Proc.NASA Conference on Space Telerobotics*, 1989.
- [33] Miquel Feixas, Mateu Sbert, and Francisco González. A unified information-theoretic framework for viewpoint selection and mesh saliency. *ACM Transactions on Applied Perception (TAP)*, 6(1) :1, 2009.
- [34] Juan A Fernandez and Javier Gonzalez. A general world representation for mobile robot operations. *Seventh conference of the Spanish association for artificial intelligence (CAEPIA-97)*, 1997.
- [35] Michael P Fourman. Compaction of symbolic layout using genetic algorithms. *1st International Conference on Genetic Algorithms*, pages 141–153, 1985.
- [36] Friedrich Fraundorfer, Christopher Engels, and David Nistér. Topological mapping, localization and navigation using image collections. *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. IROS 2007.*, pages 3872–3877, 2007.

- [37] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics (TOG)*, 25(1) :130–150, 2006.
- [38] Gabriela Gallegos, Maxime Meilland, Patrick Rives, and Andrew I Comport. Appearance-based slam relying on a hybrid laser/omnidirectional sensor. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*,, pages 3005–3010, 2010.
- [39] Afzal Godil and Asim Imdad Wagan. Salient local 3d features for 3d shape retrieval. *Three-Dimensional Imaging, Interaction, and Measurement*, 7864 :78640S, 2011.
- [40] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1) :34–46, 2007.
- [41] Jose E Guivant and Eduardo Mario Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE transactions on Robotics and Automation*, 17(3) :242–257, 2001.
- [42] Chenlei Guo, Qi Ma, and Liming Zhang. Spatio-temporal saliency detection using phase spectrum of quaternion fourier transform. pages 1–8, 2008.
- [43] Gideon Guy and Gérard Medioni. Inference of surfaces, 3d curves, and junctions from sparse, noisy, 3d data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11) :1265–1277, 1997.
- [44] Dirk Hähnel, Wolfram Burgard, and Sebastian Thrun. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1) :15–27, 2003.
- [45] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. *Advances in neural information processing systems*, pages 545–552, 2007.
- [46] Chris Harris and Mike Stephens. A combined corner and edge detector. *Alvey vision conference*, 15 :50, 1988.
- [47] Xiaodi Hou and Liqing Zhang. Saliency detection : A spectral residual approach. *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, pages 1–8, 2007.
- [48] Andrew Howard. Real-time stereo visual odometry for autonomous ground vehicles. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3946–3952, 2008.
- [49] Xun Huang, Chengyao Shen, Xavier Boix, and Qi Zhao. Salicon : Reducing the semantic gap in saliency prediction by adapting deep neural networks. *IEEE International Conference on Computer Vision*, pages 262–270, 2015.

- [50] Arnold Irschara, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. From structure-from-motion point clouds to fast location recognition. *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, pages 2599–2606, 2009.
- [51] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3) :194, 2001.
- [52] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11) :1254–1259, 1998.
- [53] Matjaz Jogan and Ales Leonardis. Robust localization using panoramic view-based recognition. *15th International Conference on Pattern Recognition*, 4 :136–139, 2000.
- [54] Håkan Jonsson. An approximative solution to the zookeeper’s problem. *Information Processing Letters*, 87(6) :301–307, 2003.
- [55] Ran Ju, Ling Ge, Wenjing Geng, Tongwei Ren, and Gangshan Wu. Depth saliency based on anisotropic center-surround difference. *IEEE International Conference on Image Processing (ICIP)*, pages 1115–1119, 2014.
- [56] Julius Kammerl, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Betz, and Eckehard Steinbach. Real-time compression of point cloud streams. *IEEE International Conference on Robotics and Automation (ICRA), 2012*, pages 778–785, 2012.
- [57] J Mark Keil. Polygon decomposition. *Handbook of Computational Geometry*, 2 :491–518, 2000.
- [58] Mark Keil and Jack Snoeyink. On the time bound for convex decomposition of simple polygons. *International Journal of Computational Geometry & Applications*, 12(03) :181–192, 2002.
- [59] Wolf Kienzle, Matthias O Franz, Bernhard Schölkopf, and Felix A Wichmann. Center-surround patterns emerge as optimal predictors for human saccade targets. *Journal of vision*, 9(5) :7–7, 2009.
- [60] Hansung Kim and Adrian Hilton. Environment modelling using spherical stereo imaging. *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1534–1541, 2009.
- [61] Srinivas SS Kruthiventi, Kumar Ayush, and R Venkatesh Babu. Deepfix : A fully convolutional neural network for predicting human eye fixations. *IEEE Transactions on Image Processing*, 26(9) :4446–4456, 2017.
- [62] Chang Ha Lee, Amitabh Varshney, and David W Jacobs. Mesh saliency. *ACM transactions on graphics (TOG)*, 24(3) :659–666, 2005.

- [63] George Leifman, Elizabeth Shtrom, and Ayellet Tal. Surface regions of interest for viewpoint selection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 414–421, 2012.
- [64] Julien Leroy, Nicolas Riche, Matei Mancas, and Bernard Gosselin. 3d saliency based on supervoxels rarity in point clouds. *Hamburg, Germany*, 2015.
- [65] Fayin Li and Jana Kosecka. Probabilistic location recognition using reduced feature set. *IEEE International Conference on Robotics and Automation, ICRA*, pages 3405–3410, 2006.
- [66] Hyon Lim, Sudipta N Sinha, Michael F Cohen, and Matthew Uyttendaele. Real-time image-based 6-dof localization in large-scale environments. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1043–1050, 2012.
- [67] Kaiwen Lin and Toby P Breckon. Real-time low-cost omni-directional stereo vision via bi-polar spherical cameras. *International Conference Image Analysis and Recognition*, pages 315–325, 2018.
- [68] Tie Liu, Zejian Yuan, Jian Sun, Jingdong Wang, Nanning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. *IEEE Transactions on Pattern analysis and machine intelligence*, 33(2) :353–367, 2011.
- [69] Yufeng Liu, Rosemary Emery, Deepayan Chakrabarti, Wolfram Burgard, and Sebastian Thrun. Using em to learn 3d models of indoor environments with mobile robots. *ICML*, 1 :329–336, 2001.
- [70] Steven Lovegrove and Andrew J Davison. Real-time spherical mosaicing using whole image alignment. *European Conference on Computer Vision*, pages 73–86, 2010.
- [71] David G Lowe. Local feature view clustering for 3d object recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR*, 1 :I–682, 2001.
- [72] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2) :91–110, 2004.
- [73] David G Lowe et al. Object recognition from local scale-invariant features. *iccv*, 99(2) :1150–1157, 1999.
- [74] Elmar Mair, Gregory D Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. *European conference on Computer vision*, pages 183–196, 2010.
- [75] Ran Margolin, Lihi Zelnik-Manor, and Ayellet Tal. How to evaluate foreground maps? *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2014.

- [76] Jiri Matas, Ondrej Chum, Martin Urban, and Tomas Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10) :761–767, 2004.
- [77] Maxime Meilland, Andrew I Comport, and Patrick Rives. A spherical robot-centered representation for urban navigation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5196–5201, 2010.
- [78] Maxime Meilland, Andrew I Comport, and Patrick Rives. Dense omnidirectional rgb-d mapping of large-scale outdoor environments for real-time localization and autonomous navigation. *Journal of Field Robotics*, 32(4) :474–503, 2015.
- [79] Emanuele Menegatti, Takeshi Maeda, and Hiroshi Ishiguro. Image-based memory for robot navigation using properties of omnidirectional images. *Robotics and Autonomous Systems*, 47(4) :251–267, 2004.
- [80] Hamza Merzic, Elena Stumm, Marcin Dymczyk, Roland Siegwart, and Igor Gilitschenski. Map quality evaluation for visual localization. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3200–3206, 2017.
- [81] Sven Middelberg, Torsten Sattler, Ole Untzelmann, and Leif Kobbelt. Scalable 6-dof localization on mobile devices. *European Conference on Computer Vision*, pages 268–283, 2014.
- [82] Joseph Modayil, Patrick Beeson, and Benjamin Kuipers. Using the topological skeleton for scalable global metrical map-building. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, 2 :1530–1536, 2004.
- [83] Tomas Moller and Ben Trumbore. Fast, minimum storage ray/triangle intersection. *ACM SIGGRAPH 2005 Courses*, page 7, 2005.
- [84] Hans P Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. *Perception*, 1996.
- [85] Peter Muhlfellner, Mathias Burki, Michael Bosse, Wojciech Derendarz, Roland Philippsen, and Paul Furgale. Summary maps for lifelong visual localization. *Journal of Field Robotics*, 33(5) :561–590, 2016.
- [86] Shree K Nayar. Catadioptric omnidirectional camera. *IEEE computer society conference on computer vision and pattern recognition*, pages 482–488, 1997.
- [87] Jose Neira and Juan D Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on robotics and automation*, 17(6) :890–897, 2001.

- [88] Eli Packer. Computing multiple watchman routes. *Lecture Notes in Computer Science*, 5038 :114–128, 2008.
- [89] Nicolas Paparoditis, Jean-Pierre Papelard, Bertrand Cannelle, Alexandre Devaux, Bahman Soheilian, Nicolas David, and Erwann Houzay. Stereopolis ii : A multi-purpose and multi-sensor 3d mobile mapping system for street visualisation and 3d metrology. *Revue française de photogrammétrie et de télédétection*, 200(1) :69–79, 2012.
- [90] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. 22(3) :281–289, 2003.
- [91] Houwen Peng, Bing Li, Weihua Xiong, Weiming Hu, and Rongrong Ji. Rgb-d salient object detection : a benchmark and algorithms. *European Conference on Computer Vision*, pages 92–109, 2014.
- [92] Jonas Pfeil, Kristian Hildebrand, Carsten Gremzow, Bernd Bickel, and Marc Alexa. Throwable panoramic ball camera. *SIGGRAPH Asia*, pages 1–1, 2011.
- [93] Nathan Piasco, Désiré Sidibé, Valérie Gouet-Brunet, and Cédric Demonceaux. Apprentissage de modalités auxiliaires pour la localisation basée vision. 2018.
- [94] Dimitri Plemenos and Madjid Benayada. Intelligent display in scene modeling. new techniques to automatically compute good views. *International Conference GraphiCon*, 96 :1–5, 1996.
- [95] Keith Rayner. Eye guidance in reading : Fixation locations within words. *Perception*, 8(1) :21–30, 1979.
- [96] Anthony Remazeilles, François Chaumette, and Patrick Gros. Robot motion control from a visual memory. *IEEE International Conference on Robotics and Automation, ICRA '04*, 5 :4695–4700, 2004.
- [97] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *European conference on computer vision*, pages 430–443, 2006.
- [98] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. *IEEE International Conference on Robotics and Automation, 2009. ICRA '09*, pages 3212–3217, 2009.
- [99] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Learning informative point classes for the acquisition of object model maps. *2008 10th International Conference on Control, Automation, Robotics and Vision*, pages 643–650, 2008.
- [100] Imeen Ben Salah, Sebastien Kramm, Cédric Demonceaux, and Pascal Vasseur. Summarizing large scale 3d point cloud for navigation tasks. *2017*

- IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2017.
- [101] Imeen Ben Salah, Sebastien Kramm, Cédric Demonceaux, and Pascal Vasseur. Navigability graph extraction from large-scale 3d point cloud. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3030–3035, 2018.
- [102] Imeen Ben Salah, Sebastien Kramm, Cédric Demonceaux, and Pascal Vasseur. Summarizing large scale 3d mesh. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018.
- [103] Imen Ben Salah, Pascal Vasseur, Cédric Demonceaux, and Sebastien Kramm. Extraction d’un graphe de navigabilité à partir d’un nuage de points 3d enrichis. *16èmes Journées francophones des jeunes chercheurs en vision par ordinateur (ORASIS 2017)*, 2017.
- [104] Torsten Sattler, Akihiko Torii, Josef Sivic, Marc Pollefeys, Hajime Taira, Masatoshi Okutomi, and Tomas Pajdla. Are large-scale 3d models really necessary for accurate visual localization? *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1637–1646, 2017.
- [105] Francesco Savelli and Benjamin Kuipers. Loop-closing and planarity in topological map-building. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, 2 :1511–1517, 2004.
- [106] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. *Spbg*, pages 111–120, 2006.
- [107] Heiko G Seif and Xiaolong Hu. Autonomous driving in the icity—hd maps as a key challenge of the automotive industry. *Engineering*, 2(2) :159–162, 2016.
- [108] Philip Shilane and Thomas Funkhouser. Distinctive regions of 3d surfaces. *ACM Transactions on Graphics (TOG)*, 26(2) :7, 2007.
- [109] Elizabeth Shtrom, George Leifman, and Ayellet Tal. Saliency detection in large point sets. *IEEE International Conference on Computer Vision*, pages 3591–3598, 2013.
- [110] Elizabeth Shtrom, George Leifman, and Ayellet Tal. Saliency detection in large point sets. *IEEE International Conference on Computer Vision*, pages 3591–3598, 2013.
- [111] Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1) :45–78, 1997.

- [112] Dmitry Sokolov, Dimitri Plemenos, and Karim Tamine. Viewpoint quality and global scene exploration strategies. *GRAPP*, 2006 :184–191, 2006.
- [113] Ran Song, Yonghuai Liu, Ralph R Martin, and Paul L Rosin. 3d point of interest detection via spectral irregularity diffusion. *The Visual Computer*, 29(6-8) :695–705, 2013.
- [114] Ran Song, Yonghuai Liu, Ralph R Martin, and Paul L Rosin. Mesh saliency via spectral processing. *ACM Transactions on Graphics (TOG)*, 33(1) :6, 2014.
- [115] Hyun Soo Park, Yu Wang, Eriko Nurvitadhi, James C Hoe, Yaser Sheikh, and Mei Chen. 3d point cloud reduction using mixed-integer quadratic programming. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 229–236, 2013.
- [116] Ted J Steiner, Guoquan Huang, and John J Leonard. Location utility-based map reduction. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 479–486, 2015.
- [117] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5(3-4) :253–271, 1998.
- [118] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. Viewpoint selection using viewpoint entropy. *VMV*, 1 :273–280, 2001.
- [119] Jonathan Ventura, Clemens Arth, Gerhard Reitmayr, and Dieter Schmalstieg. Global localization from monocular slam on a mobile phone. *IEEE transactions on visualization and computer graphics*, 20(4) :531–539, 2014.
- [120] Jonathan Ventura and Tobias Höllerer. Wide-area scene mapping for mobile visual tracking. *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 3–12, 2012.
- [121] Jan Oliver Wallgrün. Robot mapping. *Hierarchical Voronoi Graphs*, pages 11–43, 2010.
- [122] Junle Wang, Matthieu Perreira Da Silva, Patrick Le Callet, and Vincent Ricordel. Computational model of stereoscopic 3d visual saliency. *IEEE Transactions on Image Processing*, 22(6) :2151–2165, 2013.
- [123] Eric W Weisstein. Spherical coordinates. *Wolfram Research, Inc.*, 2005.
- [124] Andreas Wendel, Arnold Irschara, and Horst Bischof. Natural landmark-based monocular localization for mavs. *IEEE International Conference on Robotics and Automation (ICRA), 2011*, pages 5792–5799, 2011.
- [125] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp : a globally optimal solution to 3d icp point-set registration. *arXiv preprint arXiv :1605.03344*, 2016.

- [126] Jae-Seong Yun and Jae-Young Sim. Supervoxel-based saliency detection for large-scale colored 3d point clouds. *IEEE International Conference on Image Processing (ICIP)*, pages 4062–4066, 2016.
- [127] Jae-Seong Yun and Jae-Young Sim. Supervoxel-based saliency detection for large-scale colored 3d point clouds. *2016 IEEE International Conference on Image Processing (ICIP)*, pages 4062–4066, 2016.
- [128] Bernhard Zeisl, Kevin Koser, and Marc Pollefeys. Automatic registration of rgb-d scans via salient directions. *IEEE international conference on computer vision*, pages 2808–2815, 2013.
- [129] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2) :119–152, 1994.
- [130] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.
- [131] Weikun Zhen, Sam Zeng, and Sebastian Soberer. Robust localization and localizability estimation with a rotating laser scanner. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6240–6245, 2017.