Cartographie dense et compacte par vision RGB-D pour la navigation d'un robot mobile

Bruce Canovas

Université Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : SIGNAL IMAGE PAROLE TELECOMS

Arrêté ministériel : 25 mai 2016

Présentée par

Bruce CANOVAS

Thèse dirigée par **Michèle ROMBAUT**, Université Grenoble Alpes
et codirigée par **Amaury NEGRE**, GIPSA-LAB

préparée au sein du **Laboratoire Grenoble Images Parole Signal Automatique**
dans **l'École Doctorale Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)**

Cartographie dense et compacte par vision RGB-D pour la navigation dun robot mobile

Dense and compact mapping based on RGB-D vision for mobile robot navigation

Thèse soutenue publiquement le **6 octobre 2021**,
devant le jury composé de :

Denis PELLERIN
Professeur, Université Grenoble Alpes, Président
Guillaume CARON
Maître de conférences, Université de Picardie Jules Verne, Rapporteur
Cédric DEMONCEAUX
Professeur, Université de Bourgogne, Rapporteur
Roland CHAPUIS
Professeur, Université Clermont Auvergne, Examinateur
Michèle ROMBAUT
Professeur, Université Grenoble Alpes, Directrice de thèse
Amaury NEGRE
Ingénieur de recherche, CNRS, Co-encadrant, Invité

UNIVERSITÉ DE GRENOBLE ALPES

ÉCOLE DOCTORALE EEATS

Électronique, Électrotechnique, Automatique, Traitement du Signal

T H È S E

pour obtenir le titre de

docteur en sciences

de l'Université Grenoble Alpes

Mention : Signal, Image, Parole, Télécoms

Présentée et soutenue par

Bruce CANOVAS

Cartographie dense et compacte par vision RGB-D pour la navigation d'un robot mobile

Thèse dirigée par Michèle ROMBAUT
et co-encadrée par Amaury NÈGRE

préparée au laboratoire Grenoble Images Parole Signal Automatique (GIPSA-lab)

soutenue le 6 Octobre 2021

Jury :

Rapporteurs :	Guillaume CARON	-	Université de Picardie Jules Verne
	Cédric DEMONCEAUX	-	Université Bourgogne Franche-Comté
Directeur :	Michèle ROMBAUT	-	Université Grenoble Alpes
Encadrant :	Amaury NÈGRE	-	CNRS
Président :	Denis Pellerin	-	Université Grenoble Alpes
Examinateur :	Roland CHAPUIS	-	Université Clermont Auvergne

Remerciements

Si la thèse n'est pas une fin en soi, c'est une aventure riche qui permet de se découvrir, de se révéler et de s'exposer sans crainte. Je n'aurais pas pu parvenir au terme de ces trois années intenses sans le soutien de nombreuses personnes.

Je tiens à remercier en premier lieu ma directrice de thèse Michèle et mon co-directeur Amaury pour l'engagement, la gentillesse et la compréhension dont ils ont fait preuve durant ces années. Vous êtes toujours restés présents et, par vos idées, commentaires et suggestions, avez su me guider et me faire grandir en tant que chercheur. J'espère avoir été digne de la confiance que vous m'avez accordée en m'offrant la possibilité de monter mon propre sujet de thèse et en me laissant une grande liberté dans mes recherches.

J'ai ensuite une énorme pensée pour tous ceux qui ont contribué à faire de la vie au laboratoire (et en dehors) un plaisir quotidien. Je pars du GIPSA plein de bons souvenirs. Merci aux différents membres de "l'équipe robot compagnon" pour leur ouverture, disponibilité et bienveillance. Merci Denis pour ton optimisme par rapport à mes travaux. Cela m'a remotivé quand parfois j'en manquais. Je n'oublie pas non plus les copains du laboratoire, compagnons de galère mais surtout, entre autres, de pauses cafés, de conférences, de repas au RU, de soirées films/jeux, de bar, de grimpe, de basket... J'espère vous avoir donné autant que ce que vous avez pu m'apporter. En particulier, merci à Julien, Ludovic, Ivan et Thibaut. Le laboratoire m'a paru un peu plus vide après chacun de vos départs.

Je souhaite aussi remercier ma famille. Vous m'avez toujours soutenu, encouragé et fourni les moyens de réussir. Merci pour votre affection, votre patience et vos précieux conseils.

Enfin, merci à toi, Lauriane, de m'avoir tenu la main du début à la fin de cette épreuve. Ta présence à mes côtés et ton amour m'ont à chaque fois donnés de la force pour continuer. Je ne saurai jamais te remercier autant que je le devrais.

Table des matières

Table des figures

Liste des tableaux

Table des sigles et acronymes

ACP	*Analyse en Composantes Principales*
BRIEF	*Binary Robust Independent Elementary Features*
CPU	*Central Processing Unit*
FAST	*Features from Accelerated Segment Test*
FPS	*Frames Per Second*
GMS	*Grid-based Motion Statistics*
GPU	*Graphics Processing Unit*
GPS	*Global Positioning System*
ICP	*Iterative Closest Point*
LiDAR	*Light Detection And Ranging*
ORB	*Oriented FAST and Rotated BRIEF*
RANSAC	*Random Sample Consensus*
RGB-D	*Red Green Blue-Depth*
RRT	*Rapidly-Exploring Random Tree*
ROS	*Robot Operating System*
SIFT	*Scale-Invariant Feature Transform*
SLAM	*Simultaneous Localization and Mapping*
SURF	*Speeded Up Robust Features*
TSDF	*Truncated Signed Distance Function*
VO	*Visual Odometry*
2D	*Deux Dimensions*
3D	*Trois Dimensions*

Chapitre 1

Introduction générale

1.1 Contexte

1.1.1 Robotique mobile

La robotique mobile est un secteur au développement extrêmement rapide qui s'appuie sur de nombreuses disciplines, telles que l'informatique, la mécanique, l'automatique, l'électronique, les sciences cognitives et l'intelligence artificielle. Elle prend en compte les machines à base mobile, douées de capacités de perception, de décision et d'action, capables de se déplacer de manière autonome ou semi-autonome dans des milieux changeants, pour réaliser des missions spécifiques. Elle se différencie de la robotique industrielle traditionnelle, bien souvent stationnaire, qui s'effectue dans des environnements connus en reposant essentiellement sur des schémas exécutifs prédéfinis fixes et sur des processus déterministes. Au contraire de cette dernière, axée sur l'amélioration des vitesses et des capacités de production, ainsi que sur la répétition exacte de tâches de haute précision, la robotique mobile répond à des besoins de flexibilité et d'autonomie. Les systèmes robotiques mobiles doivent être capables de s'adapter "intelligemment" à des évènements spontanés non planifiés et de fonctionner sans l'assistance d'opérateurs humains dans des milieux inconnus. Ils utilisent pour cela de nombreux capteurs, comme des caméras, des GPS (Global Positioning System), des télémètres à ultrason, des scanners laser et des centrales inertielles, leur permettant de percevoir et d'analyser l'environnement qui les entoure.

En raison de sa flexibilité et de l'autonomie de ses solutions, le champ des applications de la robotique mobile est extrêmement vaste. Grâce à l'essor des méthodes probabilistes et les progrès technologiques au niveau des capteurs et des outils de traitement, il va bien au-delà du cadre de la production industrielle en visant à assister l'homme dans la réalisation de tâches pénibles ou même à le remplacer dans des environnements dangereux, voir inaccessibles. On trouve par exemple dans le domaine de l'aérospatial le robot terrestre Perseverance [MAR20] ayant pour mission l'exploration d'un cratère sur la planètre Mars. De nombreux prototypes de robots pour la recherche et le secours de victimes en milieux accidentés ont aussi été développés, comme le iRap Robot [Eng16], développé pour les environnements urbains, et EMILY d'Hydronalix [Hyd15], pour le sauvetage en mer (voir figure 1.1). Cette robotique de service s'étend aussi au cadre de la vie quotidienne, avec la présence sur le marché d'aspirateurs et de tondeuses autonomes, suivant des schémas de navigation programmés mais pouvant éviter des obstacles et rejoindre leur station de rechargement par eux-mêmes. Les transports,

l'agriculture, et les loisirs sont aussi d'autres domaines dans lesquels les solutions de robotique mobile sont très présentes.

FIGURE 1.1 – Exemples d'applications de la robotique mobile, avec à gauche la plateforme EMILY [Hyd15] pour le sauvetage en mer, et à droite le robot d'exploration de la planète Mars Perseverance [MAR20].

Cette thèse est réalisée dans le cadre d'un projet qui s'inscrit dans le domaine de l'assistance aux personnes à la santé vulnérable. Il vise à développer une plateforme robotique mobile de type robot compagnon, fonctionnant en milieu intérieur, destinée à participer à la surveillance d'un ensemble de personnes en situation de fragilité. Par rapport à des capteurs placés dans les bâtiments, comme des caméras et des micros par exemple, un robot compagnon présente l'avantage de pouvoir se déplacer dans plusieurs pièces et donc de couvrir plus de surface à moindre coût. Par ailleurs, la présence d'un robot à l'allure sympathique et bien visible par les personnes est généralement mieux acceptée que les capteurs de surveillance conventionnels, jugés trop intrusifs. Ce prototype de robot compagnon mobile attentif aux personnes en situation de fragilité a pour principales missions la vérification de l'état de santé, la prévention de risques et le lancement d'alerte en cas d'accidents.

1.1.2 Localisation et cartographie simultanée

Pour pouvoir réaliser des missions de haut niveau, comme de la livraison, de la recherche ou dans notre cas l'évaluation de l'état de santé de personnes, un robot doit être capable de naviguer à travers son environnement en planifiant ses mouvements et en évitant les obstacles. Les facultés énoncées requièrent toutes l'usage d'une carte permettant au robot de connaître le milieu dans lequel il évolue et de s'y repérer. En pratique, cette carte n'est pas disponible à l'avance, la plateforme robotique pouvant être déployée dans différents endroits et la structure de ces endroits pouvant évoluer au cours du temps. Le robot doit donc la construire lui même s'il veut être en mesure de se localiser et de s'en servir pour planifier ses déplacements. Néanmoins pour générer et étendre sa carte, il doit aussi savoir où ajouter les nouvelles informations mesurées par ses capteurs. Il existe donc une relation d'interdépendance entre la cartographie et la localisation. Pour la résoudre, il faut répondre au problème de cartographie et localisation simultanées (SLAM - Simultaneous Localization and Mapping), qui est un module de

perception primordial, présent au coeur de nombreux systèmes de robotique autonome.

Développés dès les années 1980 [SC86], les algorithmes de SLAM interprètent les mesures d'un ou plusieurs capteurs pour construire et mettre à jour une représentation interne d'un environnement inconnu observé, la carte, tout en estimant la pose (position et orientation) d'un système mobile dans cet environnement. Cadena et al. [Cad+16] fournissent un historique complet du problème de SLAM, en partant des premières approches développées à partir de formulations probabilistes basées sur l'inférence bayésienne (filtre de Kalman étendu, filtre particulaire...). Le processus conjoint de reconstruction incrémentale d'une carte de l'environnement et de suivi d'un capteur en mouvement est schématisé en figure 1.2. Les méthodes de SLAM sont particulièrement utiles dans les situations où un système GPS ne peut pas être déployé, à l'intérieur des bâtiments par exemple, ou lorsque sa précision n'est pas suffisante.

Un capteur populaire, fréquemment employé par les algorithmes de SLAM, est le LiDAR (Light Detection and Ranging) 2D, basé sur l'utilisation de la lumière laser, qui fournit à chaque instant une coupe étendue de la scène observée. Ses mesures sont précises, simples et rapides à traiter, mais ne permettent qu'une cartographie et une localisation 2D, inadaptées pour la navigation dans des milieux complexes. Son équivalent 3D est pour sa part généralement trop coûteux pour une utilisation grand public, ainsi que trop lourd pour être embarqué sur des plateformes robotiques de taille réduite, comme c'est souvent le cas pour les robots mo-

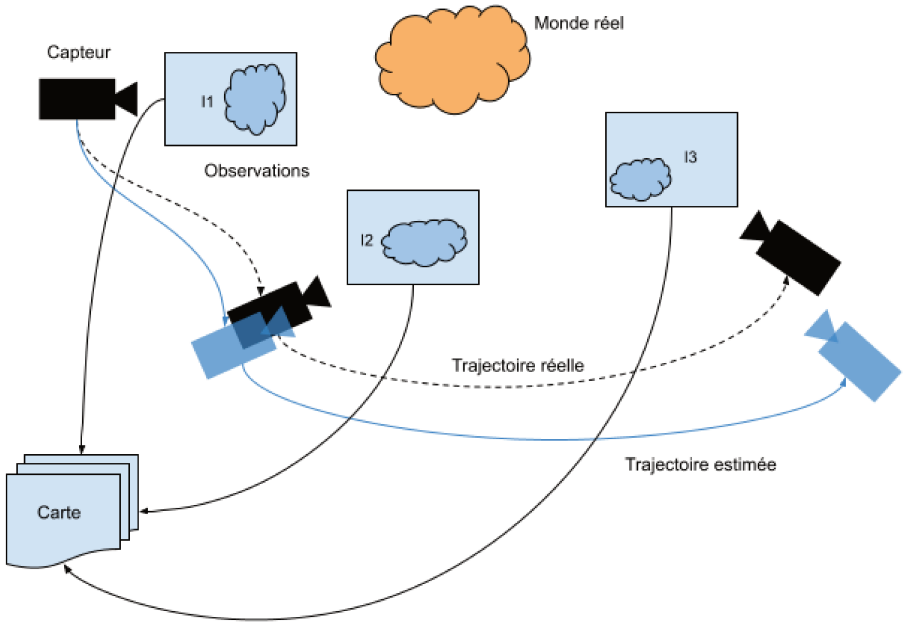

Figure 1.2 – Schéma simplifié du problème de SLAM. Un capteur est déplacé dans un environnement. Ses mesures permettent de construire et de mettre à jour une carte représentive de la structure de l'environnement et de calculer en même temps sa trajectoire. Les capteurs étant généralement soumis à différents bruits, la trajectoire et la carte estimée ont tendance dévier de la réalité.

biles évoluant en intérieur. Les caméras sont souvent préférées aux LiDARs dans les systèmes de SLAM, en raison de leur légèreté, leur petite taille, leur faible consommation énergétique et la richesse des informations qu'elles fournissent. Néanmoins, avec une caméra monoculaire, il n'est possible de construire une carte et de se localiser qu'à un facteur d'échelle près. Les caméras stéréopiques et RGB-D permettent de retrouver l'échelle de la scène filmée en donnant des mesures de distance.

Bien que le problème de SLAM soit résolu dans le cadre d'applications spécifiques, de nombreux défis entravent son adoption dans un cadre général. Ces défis et leurs solutions diffèrent en fonction du type de capteur employé, de l'application, de l'environnement et du matériel informatique utilisé pour traiter les données du capteur. Chaque capteur est cependant soumis à des bruits qui perturbent ses mesures et on retrouve donc dans toutes les méthodes le problème commun d'accumulation d'erreurs au niveau de la localisation, produisant des décalages dans la carte reconstruite et la trajectoire estimée. Les algorithmes de SLAM prennent souvent en compte l'incertitude sur les mesures du ou des capteurs pour maintenir ce décalage le plus petit possible ou tentent de le corriger avec des stratégies de fermeture de boucle. Un autre problème majeur vient du coût de calcul des méthodes de SLAM, en particulier celles basées sur des caméras, qui peuvent nécessiter d'importantes ressources pour l'extraction, le stockage et le traitement de données denses. Une implémentation minutieuse et le recours à des outils matériels et logiciels de programmation parallèle sont souvent nécessaires pour permettre une exécution temps-réel des algorithmes.

1.2 Objectifs et contributions

L'objectif principal de cette thèse est de concevoir et de mettre en oeuvre un algorithme de localisation et cartographie simultanées, basé sur le flux vidéo d'une caméra RGB-D, pour permettre la navigation en milieu intérieur d'une plateforme robotique mobile de type robot compagnon. Même si de nombreuses solutions de reconstruction 3D dense et de localisation ont émergé depuis l'introduction des caméras RGB-D, la plupart de ces méthodes ne considèrent que des milieux statiques et ne permettent pas de reconstruire de manière robuste une scène parcourue par des éléments mobiles. De plus, elles sont bien souvent trop lourdes et coûteuses pour être utilisées sur des plateformes embarquées (limitées en termes de puissance et d'espace mémoire) destinées à la robotique mobile. En effet, elles nécessitent du matériel (processeur, mémoire et carte graphique) coûteux et lourd pour stocker et traiter en temps réel l'important volume de données de couleur et de profondeur acquises par le capteur.

Le système de SLAM développé doit d'une part répondre aux critères et aux exigences de la navigation, et d'autre part être adapté à la puissance matérielle limitée de la carte embarquée du robot mobile. Les traitements effectués doivent être réalisés efficacement en ligne, avec une localisation robuste, ainsi qu'une cartographie 3D suffisamment précise et dense pour garantir un évitement d'obstacle et une planification de mouvements sans risque, mais aussi compacte et légère pour être utilisable rapidement et minimiser l'utilisation de la mémoire. Il faut en plus que le système soit capable de fonctionner correctement dans des environnements humains,

parcourus par des personnes pouvant perturber le champ de vision de la caméra RGB-D.

Les principales contributions de cette thèse sont :

— la mise en place d'une nouvelle forme de représentation 3D, dite *supersurfel*, pour la reconstruction basse résolution mais dense, rapide et légère ;

— la conception d'un algorithme de SLAM utilisant une caméra RGB-D déplacée à la main, appelé *SupersurfelFusion* et articulé autour de la nouvelle forme de représentation introduite ;

— la création d'une méthode de détection et de filtrage d'objets dynamiques, pour améliorer la robustesse du système de SLAM ;

— le portage de l'algorithme SupersurfelFusion sur la carte embarquée d'une plateforme robotique mobile terrestre et la fusion des données RGB-D avec celles des roues du robot ;

— la modification d'un système de navigation préexistant pour le connecter à la sortie de l'algorithme de SLAM développé ;

— l'application en ligne de SupersurfelFusion à la navigation autonome du robot.

1.2.1 Publications

Les travaux réalisés durant la thèse ont donné lieu aux publications suivantes :

— **Bruce Canovas**, Michèle Rombaut, Amaury Nègre, Serge Olympieff, Denis Pellerin. *A Coarse and Relevant 3D Representation for Fast and Lightweight RGB-D Mapping.* VISAPP 2019 - 14th International Conference on Computer Vision Theory and Applications, Feb 2019, Prague, Czech Republic.

— **Bruce Canovas**, Michèle Rombaut, Amaury Nègre, Serge Olympieff, Denis Pellerin. *Reconstruction 3D légère et basse résolution en environnements intérieurs à partir de caméra RGB-D.* GRETSI 2019 - XXVIIème Colloque francophone de traitement du signal et des images, Aug 2019, Lille, France.

— **Bruce Canovas**, Michèle Rombaut, Amaury Nègre, Denis Pellerin, Serge Olympieff. *Speed and Memory Efficient Dense RGB-D SLAM in Dynamic Scenes.* IROS 2020 - IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 2020, Las Vegas, United States.

— **Bruce Canovas**, Amaury Nègre, Michèle Rombaut. *Onboard Dynamic RGB-D SLAM for Mobile Robot Navigation.* ETRI Journal, 43 : 617-629, Aug 2021.

1.2.2 Logiciel libre

Le code source de l'algorithme de SLAM SupersurfelFusion, associé à la version utilisant une caméra RGB-D déplacée manuellement, a été mis à disposition au lien suivant : `https://github.com/BruceCanovas/supersurfel_fusion`.

1.3 Structure de la thèse

Ce manuscrit de thèse est organisé en six chapitres. Après cette introduction, le chapitre 2 introduit les notations utilisées, les principes élémentaires de la vision 3D et décrit l'architecture générale des méthodes de SLAM visuel.

Le chapitre 3 se focalise sur les formes de représentation utilisées pour la reconstruction 3D dense d'environnements à partir d'images acquises par une caméra RGB-D. Il met en avant les forces et les faiblesses des principales approches utilisées dans l'état de l'art, puis propose une forme de représentation basse résolution, flexible et compacte, permettant une reconstruction rapide et légère sur des plateformes de puissance réduite, ou pour des applications nécessitant un haut niveau d'efficacité plutôt qu'une précision importante. Cette représentation modélise l'environnement sous forme de patchs elliptiques appelés *supersurfels*.

Les différentes solutions de SLAM RGB-D dense sont présentées dans le chapitre 4. Un système de SLAM RGB-D complet basé sur la représentation supersurfel est aussi proposé. Il fonctionne à partir d'un capteur RGB-D déplacé à la main et comprend un module de fermeture de boucle permettant d'assurer la cohérence globale de la carte reconstruite, ainsi qu'un module de détection d'objets en mouvement pour accoître sa robustesse face aux éléments dynamiques. L'algorithme développé, nommé *SupersurfelFusion*, est comparé aux autres méthodes de l'état de l'art.

Dans le chapitre 5, l'approche SupersurfelFusion est portée sur la carte embarquée de pilotage d'une plateforme robotique mobile et adaptée pour pouvoir tirer profit des mesures des encodeurs des roues du robot. Le système est couplé à un planificateur de trajectoires afin d'obtenir une solution de cartographie et de navigation autonome simultanées, testée en conditions réelles.

Enfin, le chapitre 6 présente un bilan des travaux réalisés durant cette thèse et une discussion sur les directions futures à suivre.

Chapitre 2

Prérequis

Sommaire

Ce chapitre a pour vocation d'introduire des concepts élémentaires utilisés dans la suite du manuscrit et nécessaires à sa compréhension. Il est essentiellement destiné aux lecteurs novices dans le domaine de la vision 3D. Il s'agit premièrement d'aborder les systèmes de coordonnées et transformations géométriques impliqués dans la modélisation mathématique du processus de formation des images et dans l'extraction de nuages de points 3D. Ensuite, des méthodes pour l'optimisation numériques de systèmes de moindres carrés sont aussi abordées. Pour finir, un aperçu général des composants principaux des systèmes de SLAM visuel est exposé.

2.1 Principales notations

Dans le but de faciliter la lecture du document, nous précisons ici les principales notations utilisées pour décrire les processus de vision 3D présentés par la suite :

Mathématiques générales

$a, b, A, B...$	scalaire
$\mathbf{a}, \mathbf{b}, \mathbf{A}, \mathbf{B}...$	vecteur colonne, matrices
$a_i, \mathbf{a}_i, \mathbf{A}_i...$	i-ème élément
$f(.), g(.)...$	fonction
$\mathbf{A}^{-1}, f(.)^{-1}$	matrice inverse, fonction inverse
$\mathbf{a}^\intercal, \mathbf{B}^\intercal...$	transposée du vecteur, de la matrice
$\mathfrak{a}, \mathfrak{b}...$	référentiel
$\bar{\mathbf{a}}$	moyenne d'un ensemble $\{..., \mathbf{a}_i, ...\}$
$\hat{\mathbf{a}}$	valeur prédite
$\mathbf{a}^*$	valeur optimale
$\frac{\partial f}{\partial x}$	dérivée partielle de f par rapport à x
$\mathbb{R}^d$	espace réel de dimension d
$\mathbb{R}_+$	espace des réels positifs
$SO(2), SO(3)$	groupe spécial orthogonal (dimensions 2 et 3)
$SE(2), SE(3)$	groupe spécial eucliden (dimensions 2 et 3)
$se(3)$	algèbre de Lie du groupe spécial euclidien SE(3)
$\mathscr{M}_3(\mathbb{R})$	groupe des matrices réelles d'ordre 3

Caméra et géométrie 3D

C, D, I	images de couleur, de profondeur et d'intensité
Ω	plan image
$\pi(.)$, $\pi^{-1}(.)$	projection perspective et rétroprojection
f_x, f_y, c_x, c_y	paramètres intrinsèques de la caméra
$\mathbf{u}, \mathbf{v}...$	pixel, position 2D dans l'image
$\mathbf{p}, \mathbf{m}...$	point, position 3D
$\mathbf{p}', \mathbf{m}'...$	point issu d'une transformation de $\mathbf{p}$, $\mathbf{m}$...
$\mathbf{n}$	normal
$\mathbf{t}$	translation
$\mathbf{R}, \mathbf{O}, \mathbf{q}...$	matrice de rotation, orientation
$\mathbf{q}$	quaternion
$\mathbf{T}$	pose, transformation
$\mathbf{T}_\mathfrak{a}^\mathfrak{b}, \mathbf{t}_\mathfrak{a}^\mathfrak{b}, \mathbf{R}_\mathfrak{a}^\mathfrak{b}, \mathbf{q}_\mathfrak{a}^\mathfrak{b}$	transformation/translation/rotation du repère $\mathfrak{a}$ au repère $\mathfrak{b}$

SLAM RGB-D

$\mathcal{M}$	carte/modèle 3D global de supersurfels
$\mathcal{F}$	ensemble de supersurfels de la vue courante
$\mathcal{L}$	carte locale de points d'intérêts
$\mathfrak{c}$	repère de la caméra
$\mathfrak{r}$	repère de la base mobile du robot
$\mathfrak{w}$	repère du monde
$\mathcal{E}$	image clé
$\mathbf{\Sigma}$	matrice de covariance
$\mathbf{\Lambda}$	matrice d'information
$\rho(.)$	fonction de coût robuste
$\mathbf{J}$	matrice jacobienne
$\mathbf{e}, \mathbf{r}$	vecteur d'erreurs, de résidus

2.2 Modèle de caméra et géométrie projective

On présente brièvement ici le modèle d'acquisition de la caméra, qui permet de décrire la relation mathématique existant entre les coordonnées des points 3D de la scène observée et les coordonnées 2D de leur projection dans l'image.

2.2.1 Modèle de sténopé

Dans cette thèse, on suppose que les caméras utilisées possèdent des lentilles sans, ou avec très peu de distorsions, ou encore que les images reçues sont déjà rectifiées. Aucune correction de distorsion n'est donc nécessaire. Le processus de formation des images peut alors être décrit simplement par le modèle mathématique de sténopé (ou pinhole en anglais) [HZ03], schématisé en figure 2.1. Très fréquemment adopté en traitement d'image et vision par ordinateur pour sa simplicité et sa proximité avec la réalité physique, il représente la structure complète d'une caméra par un point $\mathbf{c}$, appelé centre optique, relié à un plan Ω, nommé plan image, par un axe perpendiculaire à ce plan : l'axe optique. Le centre optique $\mathbf{c}$ de la caméra est éloigné d'une distance f, appelée focale, du plan image Ω. Ce modèle est bâti sur l'approximation que tous les rayons lumineux issus de la scène observée sont projetés sur le plan image en passant par le centre optique, via la transformation $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$, créant ainsi l'image. La projection perspective $\pi(.)$, qui simule mathématiquement le processus de formation des images, dépend des paramètres intrinsèques de la caméra décrits dans la section suivante.

2.2.2 Paramètres intrinsèques

Comme le centre optique $\mathbf{c}$ n'est pas toujours aligné avec l'origine du repère image, le point principal image c', de coordonnées $[c_x, c_y]^\intercal$ en unité de pixels, est introduit pour matérialiser

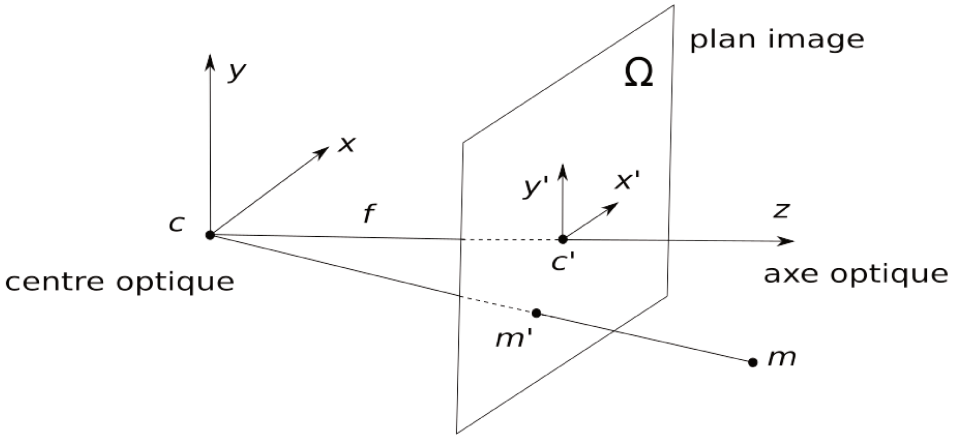

FIGURE 2.1 – Modèle de sténopé : un point $\mathbf{m}$ de l'espace 3D est projeté en $\mathbf{m}'$ sur le plan image Ω, le long d'une droite passant par le centre optique $\mathbf{c}$.

ce décalage. Il représente l'intersection du plan image Ω avec l'axe optique. De plus, les pixels du capteur n'étant pas nécessairement carrés, deux distances focales exprimées en pixels sont définies le long des axes d'abscisses et d'ordonnées : $f_x = s_x f$ et $f_y = s_y f$, avec s_x et s_y le nombre de pixels par unité de distance dans chaque direction. Les paramètres intrinsèques $\{f_x, f_y, c_x, c_y\}$ constituent les paramètres internes à la caméra. Ils peuvent être obtenus par calibrage [Zha00] et permettent de définir complètement la projection perspective $\pi : \mathbb{R}^3 \to \mathbb{R}^2$, associant les coordonnées spatiales d'une point 3D $\mathbf{m}[x, y, z]^\intercal$ dans le référentiel de la caméra $\mathfrak{c}$, avec le pixel associé $\mathbf{m}'[x', y']^\intercal$ dans l'image créée par la caméra :

$$\pi : [x, y, z]^\intercal \to [x', y']^\intercal$$

$$x' = \frac{x f_x}{z} + c_x \tag{2.1}$$

$$y' = \frac{y f_y}{z} + c_y \tag{2.2}$$

Cette transformation n'est pas bijective, on ne peut pas retrouver les coordonnées 3D $[x, y, z]^\intercal$ d'un point $\mathbf{m}$ correspondant à un pixel $\mathbf{m}'[x', y']^\intercal$ à partir d'une seule vue de la caméra et des paramètres intrinsèques. Néanmoins, si l'on connaît en plus la profondeur d du pixel, il est possible d'extraire les coordonnées 3D correspondantes, par une transformation "quasi-réciproque", qu'on appelle rétroprojection $\pi^{-1} : \mathbb{R}^3 \to \mathbb{R}^3$, telle que :

$$\pi^{-1} : [x, y, d]^\intercal \to [x, y, z]^\intercal$$

$$x = \frac{x' - c_x}{f_x} d \tag{2.3}$$

$$y = \frac{y' - c_y}{f_y} d \tag{2.4}$$

$$z = d \tag{2.5}$$

Les données de profondeur peuvent être fournies ou calculées en utilisant différents types de capteurs, dont les caméras RGB-D.

2.2.3 Paramètres extrinsèques

Les paramètres extrinsèques représentent la pose (position et orientation) d'un référentiel fixe par rapport au référentiel de la caméra, qui lui peut être mobile. On parle généralement d'espace global ou d'espace monde. Son origine est souvent définie par un marqueur extérieur visible par la caméra, dans le cas d'un système statique, ou fixée à l'emplacement initial de la caméra, dans le cas d'un système mobile. La pose du repère monde $\mathfrak{w}$ par rapport à celui de la caméra $\mathfrak{c}$ est définie par la transformation homogène $\mathbf{T}_{\mathfrak{w}}^{\mathfrak{c}} \in SE(3)$:

$$\mathbf{T}_{\mathfrak{w}}^{\mathfrak{c}} = \begin{bmatrix} \mathbf{R}_{\mathfrak{w}}^{\mathfrak{c}} & \mathbf{t}_{\mathfrak{w}}^{\mathfrak{c}} \\ 0_3 & 1 \end{bmatrix} \tag{2.6}$$

Elle est composée de la rotation $\mathbf{R}_{\mathfrak{w}}^{\mathfrak{c}} \in SO(3)$ et de la translation $\mathbf{t}_{\mathfrak{w}}^{\mathfrak{c}} \in \mathbb{R}^3$. Pour obtenir la pose de la caméra par rapport à l'espace monde $\mathfrak{w}$ il suffit de prendre la transformation homogène inverse $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} = (\mathbf{T}_{\mathfrak{w}}^{\mathfrak{c}})^{-1}$, calculée comme :

$$\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} = \begin{bmatrix} \mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}} & \mathbf{t}_{\mathfrak{c}}^{\mathfrak{w}} \\ 0_3 & 1 \end{bmatrix} = \begin{bmatrix} (\mathbf{R}_{\mathfrak{w}}^{\mathfrak{c}})^\intercal & -(\mathbf{R}_{\mathfrak{w}}^{\mathfrak{c}})^\intercal \mathbf{t}_{\mathfrak{w}}^{\mathfrak{c}} \\ 0_3 & 1 \end{bmatrix} \tag{2.7}$$

Pour simplifier la notation des formules, nous faisons abstraction des processus d'homogénéisation et de déshomogénéisation des coordonnées dans l'intégralité du manuscrit. On peut donc simplement noter la pose de la caméra $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}$ comme un tuple $\{\mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}} | \mathbf{t}_{\mathfrak{c}}^{\mathfrak{w}}\}$. Son application permet de transformer un point 3D $\mathbf{m}$ du référentiel caméra vers le référentiel global (cf. figure 2.2) :

$$(\mathbf{m})_{\mathfrak{w}} = \mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} (\mathbf{m})_{\mathfrak{c}} = \mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}} (\mathbf{m})_{\mathfrak{c}} + \mathbf{t}_{\mathfrak{c}}^{\mathfrak{w}} \tag{2.8}$$

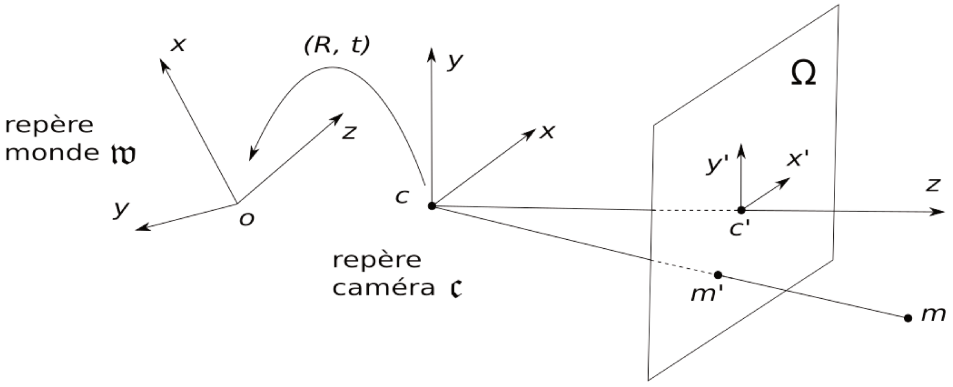

FIGURE 2.2 – Transformation du repère de la caméra $\mathfrak{c}$ vers le repère monde $\mathfrak{w}$, équivalente à l'expression de la pose de la caméra dans le repère monde.

Si l'on dispose de la pose de la caméra $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}$ à un instant précédent et qu'on connaît le déplacement relatif $\mathbf{T}_{rel} = \{\mathbf{R}_{rel} | \mathbf{t}_{rel}\}$ de la caméra entre l'instant courant et l'instant précédent, on peut composer ces deux transformations pour mettre à jour la nouvelle pose de la caméra dans le référentiel monde :

$$\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} \leftarrow \mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} \mathbf{T}_{rel} \quad \Leftrightarrow \quad (\mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}} \leftarrow \mathbf{R}_{rel} \quad \text{et} \quad \mathbf{t}_{\mathfrak{c}}^{\mathfrak{w}} \leftarrow \mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}} \mathbf{t}_{rel} + \mathbf{t}_{\mathfrak{c}}^{\mathfrak{w}}) \tag{2.9}$$

La concaténation des déplacements relatifs de la caméra, qui peuvent être estimés via différentes méthodes d'odométrie visuelle par exemple [AR18] ; [JGJ15] ; [EKC18], permet de construire sa trajectoire entière.

2.3 Caméra RGB-D

Les caméras RGB-D sont une alternative avantageuse aux techniques plus conventionnelles et plus lourdes de mesure de distances, qui utilisent des caméras stéréoscopiques ou de la télédétection par laser (technologie LiDAR). Un tel capteur fournit simultanément et en temps réel une image couleur $C : \Omega \to \mathbb{R}^3$ et une carte de profondeur $D : \Omega \to \mathbb{R}$, qui encode une estimation de la profondeur associée à chaque pixel de l'image. Connaissant les paramètres intrinsèques de la caméra, les mesures de profondeur peuvent être utilisées afin de calculer par rétroprojection les coordonnées 3D associées aux pixels de l'image et extraire un nuage de points coloré dense. Un résultat est visible sur l'image 2.4. Les capteurs RGB-D sont dits actifs, car ils émettent leur propre signal lumineux, par opposition aux caméras passives, qui récupèrent l'information utile seulement grâce à la lumière reçue. De ce fait, ils peuvent aussi fonctionner dans les environnements sombres. Légers, peu coûteux et à faible consommation énergétique, ces capteurs ont très vite été adoptés comme outil de perception embarqué de premier choix pour la robotique. Des exemples sont donnés en figure 2.3.

FIGURE 2.3 – Exemples de caméras RGB-D, de gauche à droite : Kinect V2, Asus Xtion Pro Live et Intel RealSense D435.

Les caméras RGB-D les plus utilisées sont les capteurs temps de vol (Time Of Flight en anglais), comme la Kinect V2 de Microsoft, et les capteurs à lumière structurée, comme la Asus Xtion Pro Live, qui utilisent une illumination infrarouge pour extraire les données de profondeur [Fre+10]. Les caméras temps de vol mesurent la durée entre l'émission d'une impulsion lumineuse et le retour de la lumière réfléchie pour estimer la distance, connaissant la vitesse de la lumière [SLK15]. Un capteur à lumière structuré se rapproche d'un capteur stéréoscopique dans son principe de fonctionnement qui repose sur la triangulation [GVR18]. Un motif infrarouge connu et régulier est projeté à la surface des objets observés et c'est l'observation et l'interprétation des déformations du motif, perçu d'un point de vue différent par un capteur infrarouge, qui permet de retrouver l'information de profondeur. Le soleil étant une source trop importante de lumière infrarouge, il cause des interférences qui rendent compliquée l'utilisation de ces caméras en extérieur. D'autres capteurs RGB-D, comme la

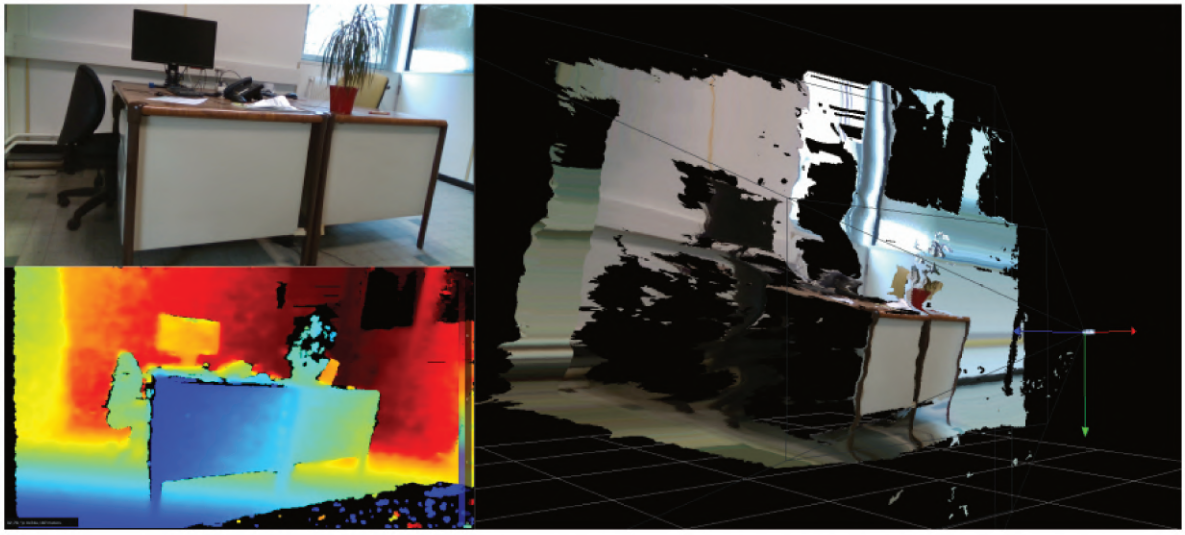

FIGURE 2.4 – Nuage de points 3D coloré et sa paire RGB-D correspondante. On peut noter la présence importante de bruit et d'artefacts dans les données 3D extraites par rétroprojection.

caméra Intel RealSense D435, combinent à la fois des technologies de stéréovision et de lumière structurée pour obtenir des mesures de profondeur plus précises et pouvoir fonctionner sous différentes conditions d'illumination, incluant les environnements extérieurs.

En dépit de leurs avantages, les capteurs RGB-D présentent des déficiences qu'il est nécessaire de prendre en compte lorsqu'on les utilise. Tout d'abord, ils fournissent des nuages de points plus denses que les caméras stéréo et les LiDARs, mais ont une portée bien plus faible. La distance qu'ils peuvent mesurer est limitée, autour de cinq à huit mètres suivant les modèles, et ils ne peuvent souvent rien mesurer en dessous de cinquante centimètres. De plus, des travaux comme [KE12] ont montré que ces capteurs sont très précis pour des distances proches, avec une incertitude sur les mesures évoluant de manière linéaire jusqu'à environ trois à quatre mètres, mais qu'au dessus de cette valeur, l'erreur augmente de façon quadratique. Enfin, ils sont sensibles aux réflexions lumineuses induites par les surfaces transparentes ou brillantes, ainsi qu'aux matériaux trop absorbants.

2.4 Optimisation non-linéaire

2.4.1 Moindres carrés

Les algorithmes de SLAM s'appuient principalement sur la méthode des moindres carrés non-linéaires [MNT04] pour estimer un vecteur d'état $\mathbf{x} \in \mathbb{R}^n$ à partir d'observations bruitées $\{x_i, y_i\}, i = 1, ..., m$. Le vecteur d'état peut par exemple contenir les paramètres d'une ou plusieurs poses de la caméra et/ou des points de la carte reconstruite. Les observations consistent quant à elles souvent en des correspondances de points d'intérêt, des intensités de pixels, des positions 3D... Le but est de trouver les paramètres $\mathbf{x}$ inconnus d'un modèle $\boldsymbol{y} = f(\boldsymbol{x}, \mathbf{x})$

permettant le meilleur ajustement aux observations $\{x_i, y_i\}$, ce qui revient à minimiser une fonction de coût prenant la forme d'un critère des moindres carrés :

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, E(\mathbf{x}) \tag{2.10}$$

où

$$E(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} r_i(\mathbf{x})^2 = \frac{1}{2}\|\mathbf{r}(\mathbf{x})\|^2 \tag{2.11}$$

avec $\mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m$, l'erreur résiduelle, une fonction à valeurs vectorielles dont les m composantes, nommées résidus, sont définies avec les valeurs de la fonction $r_i : \mathbb{R}^n \to \mathbb{R}$. Ils représentent les écarts entre les observations mesurées et les prédictions données par le modèle :

$$r_i(\mathbf{x}) = y_i - f(x_i, \mathbf{x}) \tag{2.12}$$

En vision par ordinateur, les $r_i(\mathbf{x})$ sont la plupart du temps non-linéaires (faisant intervenir des distorsions, des projections et des rotations par exemple) et non-convexes. La non-convexité des résidus, c'est-à-dire la présence d'un optimal global mais aussi d'optimaux locaux, pose problème pour la minimisation. Lors de la résolution numérique, il est généralement nécessaire de bénéficier d'une bonne estimation initiale des paramètres, c'est-à-dire proche de la valeur optimale, pour éviter que la solution finale ne corresponde à un minimum local.

Dans le cas où la fonction $\mathbf{r}(\mathbf{x})$ est linéaire, il est possible de trouver une formule explicite permettant de calculer directement les paramètres $\mathbf{x}$ à partir de techniques numériques comme la décomposition en valeurs singulières (SVD, Singular Value Decomposition) et la factorisation de Cholesky [Sol13]... Dans le cas où elle est non-linéaire, la stratégie répandue consiste à linéariser itérativement la fonction $\mathbf{r}(\mathbf{x})$ au voisinage de la valeur courante du vecteur d'état $\mathbf{x}$. A partir d'une estimation initiale $\mathbf{x}^0$ du minimum, les paramètres sont mis à jour itérativement par incrémentation d'un vecteur $\Delta\mathbf{x}$, tel que $\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}$ minimise la forme linéarisée à chaque itération k.

2.4.2 Algorithme de Gauss-Newton

Il s'agit d'une méthode itérative numérique pour la résolution de problèmes de moindres carrés non-linéaires. Elle utilise une approximation linéaire de la fonction vectorielle $\mathbf{r}(\mathbf{x})$ autour de la valeur des paramètres $\mathbf{x}$ avec la formule de Taylor au premier ordre :

$$\begin{aligned} E(\mathbf{x}+\Delta\mathbf{x}) &= \frac{1}{2}\|\mathbf{r}(\mathbf{x}+\Delta\mathbf{x})\|^2 \\ &\approx \frac{1}{2}\|\mathbf{r}(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x}\|^2 \\ &= \frac{1}{2}\mathbf{r}(\mathbf{x})^\intercal\mathbf{r}(\mathbf{x}) + \Delta\mathbf{x}^\intercal\mathbf{J}^\intercal\mathbf{r}(\mathbf{x}) + \frac{1}{2}\Delta\mathbf{x}^\intercal\mathbf{J}^\intercal\mathbf{J}\Delta\mathbf{x} \end{aligned} \tag{2.13}$$

$\mathbf{J}$ est la matrice jacobienne, ou gradient, contenant les dérivées partielles de la fonction $\mathbf{r}$ évaluées en la valeur courante du vecteur d'état $\mathbf{x}$:

$$\mathbf{J} = \left[\frac{\partial r_i(\mathbf{x})}{\partial \mathrm{x}_j}\right] \in \mathbb{R}^{m\times n} \tag{2.14}$$

Pour calculer la valeur optimale de $\Delta\mathbf{x}$ minimisant la forme linéaire obtenue dans l'équation 2.13, on la dérive et on regarde quand elle s'annule. On obtient alors un système linéaire avec les équations normales :

$$\mathbf{J}^\intercal\mathbf{J}\Delta\mathbf{x} = -\mathbf{J}^\intercal\mathbf{r}(\mathbf{x}) \tag{2.15}$$

qu'on peut résoudre en utilisant la factorisation de Cholesky par exemple pour obtenir $\Delta\mathbf{x}$. L'estimation des paramètres optimisés est alors incrémentée par $\Delta\mathbf{x}$:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x} \tag{2.16}$$

L'estimation finale $\mathbf{x}^*$ est calculée en itérant jusqu'à la convergence.

Remarque : on fait l'hypothèse $m \geq n$, sinon $\mathbf{J}^\intercal\mathbf{J}$ n'est pas inversible et les équations normales ne peuvent être résolues.

2.4.3 Algorithme de Levenberg-Marquardt

L'algorithme de Gauss-Newton converge rapidement si la solution initiale est proche du minimum global, mais peut se révéler très instable sinon. Pour résoudre ce problème et améliorer la stabilité de la convergence, l'algorithme de Levenberg-Marquardt introduit un facteur d'amortissement adaptatif $\lambda > 0$ dans l'équation de calcul du vecteur d'incrément $\Delta\mathbf{x}$:

$$(\mathbf{J}^\intercal\mathbf{J} + \lambda\ \mathrm{diag}(\mathbf{J}^\intercal\mathbf{J}))\Delta\mathbf{x} = -\mathbf{J}^\intercal\mathbf{r}(\mathbf{x}) \tag{2.17}$$

avec $\mathrm{diag}(\mathbf{J}^\intercal\mathbf{J})$ la matrice diagonale de $\mathbf{J}^\intercal\mathbf{J}$. Le facteur d'amortissement λ est ajusté à chaque itération pour contrôler le vecteur d'incrément $\Delta\mathbf{x}$. Si le critère E à minimiser décroît rapidement, on utilise une valeur de λ plus faible pour assurer la bonne convergence et ne pas passer au-delà du minimum. On se rapproche dans ce cas là de la méthode de Gauss-Newton. Au contraire, si une itération est peu efficace et ne fait pas décroître la fonction de coût E, ce qui se produit généralement lorsqu'on est loin du minimum, la valeur de λ est augmentée de manière répétée jusqu'à observer une diminution de l'erreur. On se rapproche alors du comportement d'un algorithme de descente de gradient. L'ajustement dynamique de λ peut nécessiter de résoudre l'équation 2.17 plusieurs fois à chaque itération, ce qui rend la méthode plus coûteuse en temps de calcul que celle de Gauss-Newton.

2.4.4 Fonctions de coût robustes

La norme euclidienne adoptée dans la fonction de coût (équation 2.11) présume que les observations sont distribuées suivant une loi normale, ce qui est rarement le cas en pratique. De plus, en raison de son terme quadratique, la formulation traditionnelle présente l'inconvénient de trop favoriser les valeurs de résidus importantes. Ces valeurs correspondent souvent à des observations aberrantes qui peuvent fausser l'optimisation des paramètres du modèle. Une façon de rendre la méthode des moindres carrés moins sensible aux aberrations, en réduisant

leur influence, consiste à remplacer le terme quadratique $(.)^2$ par une fonction de coût robuste $\rho(.)$:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, E(\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{m} \rho(r_i(\mathbf{x})) \tag{2.18}$$

On se place alors dans le cadre des techniques des M-estimateurs. Pour plus de détails sur les M-estimateurs, le lecteur est invité à se référer à l'ouvrage [MMY06].

Le minimum de l'équation 2.18 est obtenu en la dérivant par rapport à $\mathbf{x}$ et en annulant la dérivée :

$$\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} = \sum_{i=1}^{m} \frac{\partial r_i(\mathbf{x})}{\partial \mathbf{x}} \psi(r_i(\mathbf{x})) = 0 \tag{2.19}$$

avec $\psi(.)$ la dérivée de la fonction de coût robuste $\rho(.)$. En posant $w(x)x = \frac{\psi(x)}{x}x$ on obtient la formulation alternative suivante :

$$\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} = \sum_{i=1}^{m} \frac{\partial r_i(\mathbf{x})}{\partial \mathbf{x}} w(r_i(\mathbf{x})) r_i(\mathbf{x}) = 0 \tag{2.20}$$

qui est un minimiseur du problème de moindres carrés repondérés :

$$E(\mathbf{x}) = \sum_{i=1}^{m} w(r_i(\mathbf{x})) r_i(\mathbf{x})^2 \tag{2.21}$$

Ce problème peut être résolu simplement à partir des méthodes de Gauss-Newton et Levenberg-Marquardt présentées précédemment, en précalculant à chaque itération des poids fixés $w(r_i(\mathbf{x})) = w_i$ à partir de la précédente valeur de $\mathbf{x}$ et en les incorporant dans les équations normales :

$$\mathbf{J}^\intercal \mathbf{W} \mathbf{J} \Delta \mathbf{x} = -\mathbf{J}^\intercal \mathbf{W} \mathbf{r}(\mathbf{x}) \tag{2.22}$$

avec $\mathbf{W}$ une matrice diagonale de dimension $m \times m$ ayant les poids précalculés w_i pour éléments diagonaux.

A partir du chapitre 3, on utilise la fonction de coût robuste de Huber [Hub11], issue de la théorie des M-estimateurs :

$$\rho(x) = \begin{cases} \frac{x^2}{2} & \text{si } |x| \leq k \\ k\left(|x| - \frac{k}{2}\right) & \text{sinon} \end{cases} \tag{2.23}$$

où k est une constante déterminée à partir de l'estimation de l'écart type du bruit sur les mesures. Sa fonction de pondération associée est :

$$w(x) = \begin{cases} 1 & \text{si } |x| \leq k \\ \frac{k}{|x|} & \text{sinon} \end{cases} \tag{2.24}$$

2.5 Paramétrisation pour l'estimation de transformations rigides

Les méthodes d'optimisation classiques présument que les paramètres du modèle à estimer appartiennent à un espace vectoriel Euclidien. Dans le cas où le modèle à estimer est une transformation rigide $\mathbf{T} \in SE(3)$, c'est-à-dire une transformation géométrique Euclidienne qui préserve les distances entre les points, représentant par exemple le déplacement ou la pose d'une caméra, les méthodes de Gauss-Newton et Levenberg-Marquardt ne peuvent pas être exécutées exactement comme décrit auparavant. En effet, une transformation $\mathbf{T} \in SE(3)$ est composée d'un vecteur de translation $\mathbf{t} \in \mathbb{R}^3$ mais aussi d'une matrice de rotation $\mathbf{R} \in SO(3)$, qui ne fait pas partie d'un espace linéaire et est soumise à plusieurs contraintes ($\mathbf{R}^\intercal\mathbf{R} = \mathbf{I}$ et $\det(\mathbf{R}) = 1$), rendant sont utilisation en tant que variable d'optimisation compliquée. En particulier, l'opération d'addition n'est pas définie pour les éléments de $SO(3)$, donc le développement de Taylor (équation 2.13) et la mise à jour incrémentale des paramètres (équation 2.16) ne peuvent pas être réalisées. De plus, la représentation sous forme de matrice de transformation $\mathbf{T} \in SE(3)$ est sur-paramétrisée, puisqu'elle possède 12 paramètres pour seulement 6 degrés de liberté.

2.5.1 Algèbre de Lie de $SE(3)$

Une solution pour remédier à ces problèmes est de s'appuyer sur la théorie des groupes de Lie et de leurs algèbres de Lie correspondants. Ces concepts sont abordés ici de manière très succincte et plus de détails sur leur utilisation en robotique et en vision par ordinateur sont disponibles dans les documents [GZ21] ; [Bla10]. Le groupe des transformations rigides $SE(3)$ forme une variété lisse, c'est-à-dire un espace qui se comporte localement comme un espace Euclidien. On parle aussi de groupe de Lie. A chaque groupe de Lie correspond un algèbre de Lie, qui est un espace vectoriel représentant l'espace tangent du groupe en un point donné. En l'élément neutre (l'identité dans le cas du groupe $SE(3)$), cet espace tangent fournit une linéarisation du groupe de Lie, tout en maintenant ces propriétés. L'algèbre de Lie en l'identité associé à $SE(3)$ est noté $se(3)$. Il permet d'adopter une représentation compacte et non contrainte des mouvements rigides, en paramétrisant une transformation $\mathbf{T} \in SE(3)$ proche de l'identité par un vecteur de dimension 6 : $\boldsymbol{\xi}[\boldsymbol{\nu}, \boldsymbol{\omega}]^\intercal \in se(3)$, avec $[\nu_x, \nu_y, \nu_z]^\intercal$ les coordonnées de $\boldsymbol{\nu}$ et $[\omega_x, \omega_y, \omega_z]^\intercal$ celles de $\boldsymbol{\omega}$.

Les opérateurs $\log_{SE(3)}$ et $\exp_{SE(3)}$ permettent de passer du groupe de Lie $SE(3)$ à l'espace tangent $se(3)$ et inversement :

$$\exp_{SE(3)}(\boldsymbol{\xi}) = \begin{bmatrix} \exp(\boldsymbol{\omega}^\wedge) & \mathbf{V}\boldsymbol{\nu} \\ 0_3 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0_3 & 1 \end{bmatrix} = \mathbf{T} \in SE(3) \tag{2.25}$$

où l'opérateur $(.)^\wedge$ convertit $\boldsymbol{\omega}$ en une matrice antisymétrique :

$$\boldsymbol{\omega}^\wedge = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.26}$$

$\exp(\boldsymbol{\omega}^\wedge)$ est formulé par :

$$\exp(\boldsymbol{\omega}^\wedge) = \mathbf{I} + \frac{\sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|}\boldsymbol{\omega}^\wedge + \frac{1-\cos(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^2}(\boldsymbol{\omega}^\wedge)^2 \tag{2.27}$$

et

$$\mathbf{V} = \mathbf{I} + \frac{1-\cos(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^2}\boldsymbol{\omega}^\wedge + \frac{\|\boldsymbol{\omega}\| - \sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^3}(\boldsymbol{\omega}^\wedge)^2 \tag{2.28}$$

La transformation du logarithme $\log_{SE(3)}$ est obtenue à partir de l'équation 2.25 :

$$\log_{SE(3)}(\mathbf{T}) = \begin{bmatrix} (\log \mathbf{R})^\vee \\ \mathbf{V}^{-1}\mathbf{t} \end{bmatrix} = \boldsymbol{\xi} \in se(3) \tag{2.29}$$

où $(.)^\vee$ réalise l'opération inverse de $(.)^\wedge$, $(\boldsymbol{\omega}^\wedge)^\vee = \boldsymbol{\omega} \in \mathbb{R}^3$, et

$$\log \mathbf{R} = \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^\intercal) \quad \text{avec} \quad \theta = \cos^{-1}\left(\frac{\text{trace}(\mathbf{R}) - 1}{2}\right) \tag{2.30}$$

2.5.2 Optimisation sur $SE(3)$

Pour calculer les paramètres d'une transformation rigide à partir de méthodes d'optimisation comme Gauss-Newton et Levenberg-Marquardt, il est commode de représenter la transformation à estimer par $\exp_{SE(3)}(\Delta\boldsymbol{\xi})\mathbf{T}$, où $\exp_{SE(3)}(\Delta\boldsymbol{\xi})$ est une petite perturbation, un déplacement incrémental, appliquée à la valeur courante $\mathbf{T} \in SE(3)$ de la transformation. Le vecteur d'incrément $\Delta\boldsymbol{\xi}$ est représenté et optimisé à chaque itération dans l'espace tangent $se(3)$ de l'estimation courante $\mathbf{T}$. L'étape de mise à jour décrite par l'équation 2.16 et la jacobienne de la fonction de coût, définie à la formule 2.14, sont ensuite légèrement modifiées. La matrice de transformation $\mathbf{T}$ est mise à jour itérativement par composition avec la transformation incrémentale $\exp_{SE(3)}(\Delta\boldsymbol{\xi})$:

$$\mathbf{T}^{k+1} = \exp_{SE(3)}(\Delta\boldsymbol{\xi})\mathbf{T}^k \tag{2.31}$$

et la matrice jacobienne du problème linéarisé est exprimée par :

$$\mathbf{J} = \left.\frac{\partial \mathbf{r}(\exp_{SE(3)}(\Delta\boldsymbol{\xi})\mathbf{T})}{\partial \Delta\boldsymbol{\xi}}\right|_{\Delta\boldsymbol{\xi}=0} \tag{2.32}$$

2.6 Aperçu du SLAM visuel

Le travail précurseur de Klein et Murray sur PTAM [KM07] a fourni une architecture générale sur laquelle sont construites la plupart des approches de SLAM visuel modernes (cf. figure 2.5). Elle consiste en deux étapes : "front-end" et "back-end". Le front-end, aussi appelé odométrie visuelle, est responsable du calcul en temps réel de la pose de la caméra, tandis que le back-end est chargé des processus plus lents de mise à jour de la carte reconstruite, de son optimisation globale ainsi que de celle de la trajectoire. Dans la plupart des cas, front-end

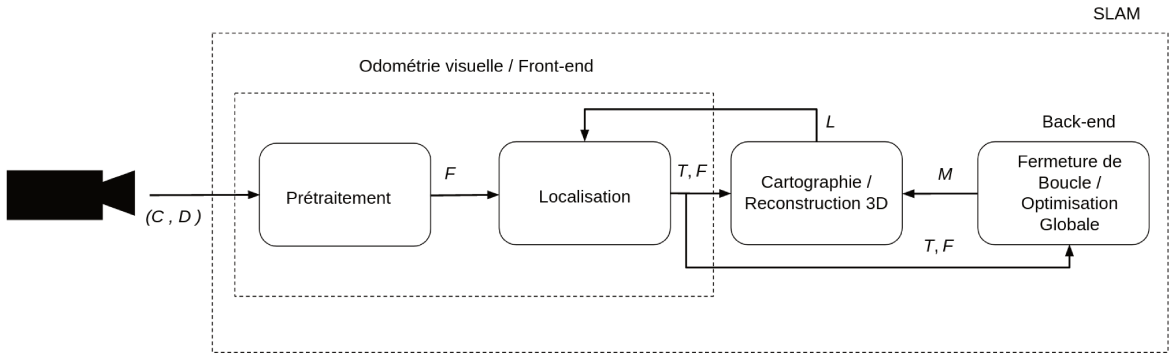

FIGURE 2.5 – Architecture générale des systèmes de SLAM visuel. Des données pertinentes F sont extraites de la dernière vue acquise, puis sont utilisées avec les mesures locales L de la carte M reconstruite, pour estimer la pose T de la caméra en mouvement. La carte M peut alors être mise à jour, localement ou globalement, à partir des nouvelles informations F, transformées dans son référentiel grâce à la pose T du capteur.

et back-end sont exécutés en parallèle, mais il existe certaines approches qui ne font pas de distinction explicite entre ces deux étapes et les exécutent séquentiellement. Le back-end peut aussi être effectué une seule fois à la fin, après acquisition de toutes les images.

Les blocs de front-end et de back-end englobent quatre principaux modules :

(i) un module de **prétraitement** optionnel dont le but est de filtrer et d'extraire les données pertinentes des images fournies par la caméra. Il peut s'agir d'images d'intensité ou de couleur seules dans le cas de caméras monoculaires, ou bien couplées à des images de profondeur si l'on utilise des capteurs RGB-D ou stéréoscopiques.

(ii) un bloc de **localisation** qui détermine la pose courante de la caméra à partir des nouvelles données acquises. Il est au coeur du front-end. Les approches les plus simples s'appuient sur des contraintes entre paires d'images successives pour réaliser le suivi de la pose de la caméra. Il est cependant courant d'utiliser plus de deux images pour obtenir une meilleure précision, après l'initialisation. Dans ces cas-là, le mouvement de la caméra est calculé à partir de contraintes entre l'image actuelle et une carte construite à partir des images précédentes.

(iii) une partie de **cartographie** qui reconstruit l'environnement observé. Elle est souvent à l'intersection des étapes de front-end et de back-end : le premier se focalisant plus sur une reconstruction locale, tandis que le second vise à assurer la cohérence de la carte à un niveau global. Les méthodes de cartographie varient beaucoup en fonction des applications.

(iv) un module d'**optimisation globale**, qui s'appuie généralement sur la détection d'une ou plusieurs fermetures de boucle, lorsqu'un endroit déjà vu est à nouveau visité, pour corriger les dérives accumulées et améliorer la carte reconstruite. Il est le principal élément du back-end et consiste la plupart du temps en une étape de reconnaissance de lieu, suivie d'une correction de l'erreur présente dans la carte et la trajectoire.

L'odométrie visuelle est utilisée comme un élément de base des méthodes de SLAM visuel,

dans lesquelles elle est synonyme de front-end. Le SLAM visuel est souvent confondu avec l'odométrie visuelle dans la littérature, car les deux se comportent généralement de façon similaire jusqu'à la fermeture d'une boucle. Néanmoins leurs objectifs sont bien différents. L'odométrie visuelle se concentre sur l'estimation locale de la trajectoire de la caméra et parfois aussi de la géométrie de la scène observée, tandis que les techniques de SLAM visent à reconstruire une carte globalement cohérente et à produire une estimation de la trajectoire fiable au niveau global.

Pour classifier les systèmes de SLAM visuel on peut soit se baser sur leur manière de cartographier l'environnement, soit sur la stratégie qu'ils utilisent pour suivre le mouvement de la caméra. Au niveau de la cartographie, on distingue les approches qui génèrent des cartes éparses de celles qui génèrent des cartes denses. Pour la localisation, ou odométrie visuelle, on distingue essentiellement deux types de méthodes : directes et indirectes. Les systèmes qui effectuent une cartographie éparse se servent généralement de techniques d'odométrie visuelle indirecte, tandis que ceux qui reconstruisent un modèle 3D dense de l'environnement observé reposent habituellement sur de l'odométrie visuelle directe. Nous nous focalisons ici sur ces deux classes d'associations, qui sont celles qu'on rencontre le plus souvent, même s'il existe des algorithmes de SLAM visuel hybrides (semi-directes, semi-denses...) mixant ou combinant les stratégies de localisation et de cartographie énoncées [Don+21] ; [Gal+21] ; [Liu+19].

2.6.1 SLAM éparse indirect

Les méthodes indirectes utilisent seulement un sous-ensemble réduit de pixels saillants issus des images en entrée pour se localiser et reconstruire une carte, afin de répondre à des exigences d'exécution en temps réel et de robustesse aux variations de point de vue et d'illumination. Elles sont basées sur l'extraction de points d'intérêt (aussi appelés amers visuels, points caractéristiques ou encore points clés) dans les images, qui correspondent à des zones présentant des propriétés locales remarquables et distinctives. Il s'agit bien souvent de coins (corners) positionnés sur les contours des objets, pour lesquels la texture, la couleur, ou l'intensité avoisinante varie brusquement dans toutes les directions. Les points d'intérêts extraits dans une image sont décrits par des descripteurs compacts qui peuvent être comparés à ceux obtenus dans d'autre images, dans le but de trouver des correspondances entre les points clés. Les correspondances trouvées permettent l'estimation des transformations relatives liant les images, dont la concaténation forme la trajectoire du capteur, ainsi que la triangulation des points d'intérêt 2D appariés en points 3D dans le cas de caméras monoculaires. La réalisation d'une mise en correspondance robuste nécessite le calcul de descripteurs présentant de bonnes propriétés d'invariance aux changements d'éclairage, d'échelle et d'orientation, induits par le déplacement de la caméra. Parmi les algorithmes d'extraction de points d'intérêt et de descripteurs associés les plus connus, on trouve SIFT (Scale-Invariant Feature Transform) [Low04], SURF (Speeded Up Robust Feature) [Bay+08] et ORB (Oriented FAST and Rotated BRIEF) [Rub+11].

Les systèmes de SLAM indirect reconstruisent généralement une carte éparse en identifiant des images clés et en calculant les positions 3D des points d'intérêt détectés dans chaque image

clé. La carte consiste alors en un ensemble d'images clés et de positions 3D associées. Le but du sous-échantillonnage du flux vidéo en images clés est de réduire les calculs en maintenant une carte pertinente de petite taille. La sélection d'image clés s'appuie souvent sur l'évaluation de la distance parcourue par la caméra depuis la précédente image clé [KM07], sur l'évaluation de la surface de recouvrement entre l'image courante et la précédente image clé [Leu+13], ou encore sur l'évolution du nombre de points d'intérêts suivis [QLS18].

Le suivi de la pose de la caméra se fait la plupart du temps en estimant le déplacement $\mathbf{T} \in SE(3)$ de la caméra entre la dernière image clé sélectionnée et l'image courante. Pour estimer la transformation relative $\mathbf{T}$, les méthodes indirectes minimisent l'erreur de reprojection entre les positions 3D $\mathbf{p}_i$ des points d'intérêts de la dernière image clé, et les positions pixels 2D $\mathbf{u}_j$ des points d'intérêts correspondants, détectés dans l'image courante :

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_{\mathcal{A}_{i,j}} \|\mathbf{u}_j - \pi(\mathbf{T}\mathbf{p}_i)\|^2 \tag{2.33}$$

avec $\mathcal{A}_{i,j}$ la liste d'associations entre les points de la carte et ceux de l'image actuelle. L'approche EPnP (Efficient Perspective-n-Point) [LMNF09] est généralement employée pour trouver le mouvement de la caméra à partir des appariements 2D-3D. Une procédure d'optimisation locale basée sur de l'ajustement de faisceaux [Tri+00] (décrit un peu plus loin en 2.6.3) peut ensuite être employée pour affiner la trajectoire estimée et les positions 3D des points de la carte, en minimisant l'erreur de reprojection sur toutes les images clés précédentes qui partagent une partie du champ de vision de la dernière. Un algorithme de SLAM visuel indirect et éparse très largement utilisé est ORB-SLAM [MAMT15], exposé sur la figure 2.6.

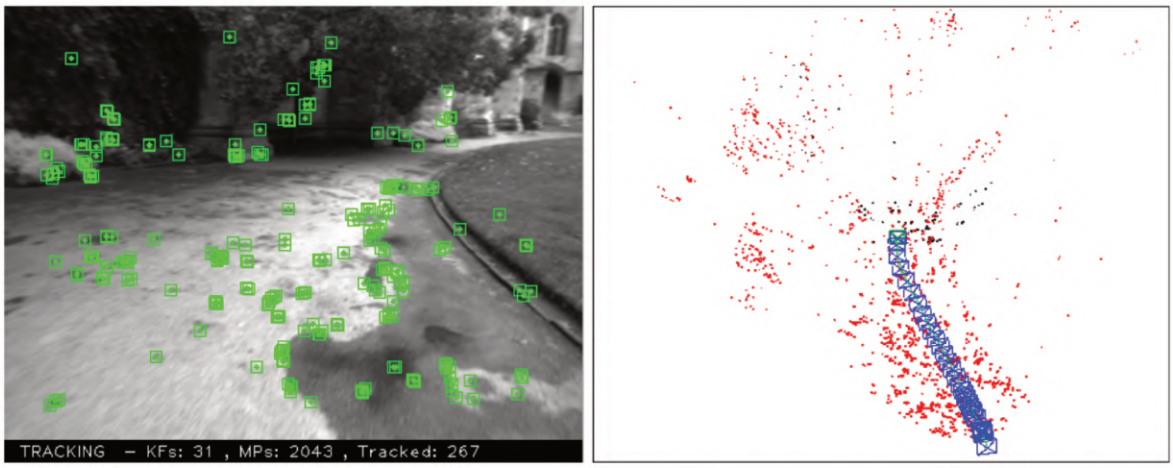

FIGURE 2.6 – Cartographie éparse et localisation indirecte exécutées par ORB-SLAM [MAMT15] à partir de points clés caractéristiques détectés dans les images.

L'inconvénient des méthodes indirectes est qu'elles ne peuvent pas suivre le mouvement de la caméra si le nombre de points clés détectés est insuffisant, ce qui peut se produire dans les environnements faiblement texturés. De plus, l'appariement des points d'intérêt à partir des descripteurs crée souvent de mauvaises correspondances qui peuvent perturber l'estimation

du déplacement de la caméra. L'algorithme RANSAC (RANdom SAmple Consensus) [FB81], présenté en annexe B, est généralement appliqué pour filtrer les paires aberrantes.

2.6.2 SLAM dense direct

Contrairement aux algorithmes de SLAM éparse indirect, qui n'utilisent qu'une infime partie des pixels des images, les systèmes denses directs s'appuient sur la quasi-totalité des pixels pour calculer la trajectoire de la caméra et reconstruire l'environnement. Ces méthodes nécessitent du matériel informatique plus puissant que les approches indirectes pour fonctionner correctement, à cause de la plus grande quantité d'informations qu'ils utilisent, mais produisent des cartes denses qui ouvrent la voie à un champ d'application (réalité virtuelle, navigation, modélisation 3D...) plus large que les cartes éparses, dont l'utilisation est souvent limitée à la localisation. La figure 2.7 montre un exemple de reconstruction 3D produite par un algorithme de SLAM dense direct.

FIGURE 2.7 – Scène reconstruite par un SLAM [McC+17] s'appuyant sur des méthodes de cartographie dense et de localisation directe, qui utilisent l'ensemble des pixels des images RGB-D en entrée.

Les méthodes de localisation directe présentent l'avantage de ne pas nécessiter de phases de détection d'amers visuels, de calcul de descripteurs et de mise en correspondance explicite. Elles reposent sur l'hypothèse que des pixels qui se correspondent sur deux images ont la même valeur d'intensité. La transformation relative $\mathbf{T} \in SE(3)$ entre deux images d'intensité successives I_t et I_{t+1} peut alors être calculée en minimisant la différence d'intensité entre leurs pixels [KSC13a] ; [NLD11] ; [ESC15] à l'aide de techniques d'optimisation numériques similaires à celles présentées en section 2.4) :

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}}\, E_{photo} = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_{i=1}^{N} (I_{t+1}(w(\mathbf{T}, \mathbf{u}_i)) - I_t(\mathbf{u}_i))^2 \tag{2.34}$$

avec N le nombre total de pixels dans les images et $w(.)$ la fonction dite de "warping", permettant de réaliser le transfert de points d'une image à l'autre. Si l'on dispose d'une estimation D_t de la profondeur de l'image précédente, la fonction de warping prend la forme suivante :

$$w(\mathbf{T}, \mathbf{u}_i) = \pi(\mathbf{T}\pi^{-1}(\mathbf{u}_i, D_t(\mathbf{u}_i))) \tag{2.35}$$

Dans le cas des caméras RGB-D ou stéréoscopiques, qui permettent d'obtenir à la fois des images de couleur et de profondeur, on peut aussi minimiser conjointement la différence entre les valeurs de profondeur des pixels :

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \left(E_{depth} + \lambda E_{photo}\right) \tag{2.36}$$

où

$$E_{depth} = \sum_{i=1}^{N} (D_{t+1}(w(\mathbf{T}, \mathbf{u}_i)) - D_t(\mathbf{u}_i))^2 \tag{2.37}$$

avec λ un scalaire permettant de pondérer l'influence des deux termes. Dans la pratique les mesures de profondeurs sont souvent très bruitées et les méthodes RGB-D de l'état de l'art [Iza+11] ; [Whe+15] ; [Whe+13] préfèrent s'appuyer sur la métrique point-plan de l'algorithme "Iterative Closest Point" (ICP) [CM91], qui permet une convergence plus robuste. L'énergie E_{depth} a minimiser est alors remplacée par un critère E_{icp} visant à aligner les surfaces 3D issues des cartes de profondeur D_t et D_{t+1} des deux images :

$$E_{icp} = \sum_{\mathcal{A}_{i,j}}^{N} [(\mathbf{T}\mathbf{p}_j - \mathbf{p}_i) \cdot \mathbf{n}_i]^2 \tag{2.38}$$

où $\mathbf{p}_i$ et $\mathbf{p}_j$ sont les positions 3D obtenues par rétroprojection des pixels des deux images de profondeurs et $\mathbf{n}_i$ les normales associées aux positions 3D de la profondeur D_t. $\mathcal{A}_{i,j}$ est la liste de correspondances entre les données 3D, calculée avec la méthode d'association par projection de données [BL95]. Par rapport aux stratégies de localisation indirecte, l'estimation du mouvement de la caméra et la mise en correspondance implicite des pixels sont réalisées simultanément, donc plus rapidement. Néanmoins, cette mise en correspondance implicite rend les approches directes plus sensibles aux variations d'éclairage et aux éléments dynamiques [BWC18].

2.6.3 Optimisation globale et fermeture de boucle

Les algorithmes de SLAM visuel accumulent des erreurs au cours de leur fonctionnement, en particulier à cause du bruit des capteurs et/ou d'approximations numériques. Ces erreurs se traduisent par un décalage de la trajectoire estimée et de la surface reconstruite, par rapport à la trajectoire réelle de la caméra et de la vraie géométrie de la surface observée. Les méthodes de SLAM visuel ont recours à des techniques d'optimisation globale pour améliorer et corriger la reconstruction et la trajectoire calculées. Les deux techniques les plus répandues sont l'ajustement de faisceaux [Tri+00] et l'optimisation d'un graphe de poses [LM97].

L'ajustement de faisceaux est une méthode qui optimise de façon combinée la trajectoire et la structure de la scène reconstruite. Elle est très largement employée par les systèmes de SLAM éparse et indirect, basés sur l'extraction de points d'intérêt, et est formulée ainsi :

$$\mathbf{T}_i^*, \mathbf{p}_j^* = \underset{\mathbf{T}_i, \mathbf{p}_j}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{M} \|\mathbf{u}_{ij} - \pi(\mathbf{T}_i \mathbf{p}_j)\|^2 \tag{2.39}$$

avec $\mathbf{T}_i \in SE(3)$ les poses décrivant la trajectoire de la caméra sur les N images, $\mathbf{u}_{ij}$ les positions pixels des M points d'intérêt détectés dans la $i-$ème image, et $\mathbf{p}_j$ les positions 3D des points de la carte appariés aux points d'intérêts.

L'optimisation d'un graphe de poses est une approximation, moins précise mais plus rapide, de l'ajustement de faisceaux, qui consiste à ignorer la structure de la scène reconstruite et à n'optimiser que la trajectoire. La différence entre les deux méthodes est explicité par le schéma 2.8. La trajectoire de la caméra est représentée par la liste de ses poses succesives $\mathbf{T}_1, ..., \mathbf{T}_N$ et un graphe de poses $\mathcal{G}$ est construit, dont les noeuds sont les poses de la caméra et les branches sont les transformations relatives calculées, reliant les poses qui partagent un rapport de covisibilité. Soit $\mathbf{T}_i, \mathbf{T}_j \in SE(3)$ deux poses de la caméra à deux instants distincts i, j, et $\mathbf{T}_j^i$ une mesure de la transformation relative liant les deux poses, telle que $\mathbf{T}_j = \mathbf{T}_i \mathbf{T}_j^i$. On peut définir l'erreur $\mathbf{e}_{i,j}$ sur la transformation relative mesurée :

$$\mathbf{e}_{i,j} = \log_{SE(3)}(\mathbf{T}_j^i \mathbf{T}_j^{-1} \mathbf{T}_i) \tag{2.40}$$

$\log_{SE(3)}$ désigne ici l'application logarithme permettant de passer de la variété $SE(3)$ à son espace tangent, qui constitue l'algèbre de Lie $se(3)$ (cf. 2.5.1). L'optimisation du graphe de poses est réalisée en minimisant la somme des erreurs $\mathbf{e}_{i,j}$ des transformations relatives sur l'ensemble du graphe $\mathcal{G}$, à partir de méthodes de moindres carrés non-linéaires :

$$E_{poses} = \sum_{(i,j) \in \mathcal{G}} \|\mathbf{e}_{i,j}\|^2 \tag{2.41}$$

L'utilisation d'un graphe de poses est généralement préférée à l'ajustement de faisceaux par les méthodes denses et directes, car l'optimisation de données denses, issues de la totalité des pixels des images, introduit un nombre trop important de variables pour traiter le problème sur du matériel informatique abordable et dans des délais raisonnables.

Les stratégies d'optimisation globales sont la plupart du temps exécutées lors de fermetures de boucles. La fermeture de boucle est cruciale pour garantir la cohérence globale des algorithmes de SLAM à long terme. Elle désigne la procédure consistant à détecter le fait que la caméra observe un endroit par lequel elle est déjà passée auparavant, à générer des contraintes entre l'observation courante et l'observation passée correspondante, puis à corriger les dérives du SLAM grâce à ces contraintes. Un exemple est affiché en figure 2.9. La détection de fermeture de boucle se fait en comparant les données de la vue courante avec celles des vues précédentes. Comme il est impossible de garder en mémoire toutes les images acquises et leurs poses associées pour des applications en temps réel, les systèmes de SLAM définissent habituellement des images clés (jugées les plus pertinentes selon divers critères) : seul un sous-ensemble représentatif des poses de la trajectoire et des observations est conservé afin de réduire l'empreinte mémoire et le coût des calculs lors de la détection et de la correction.

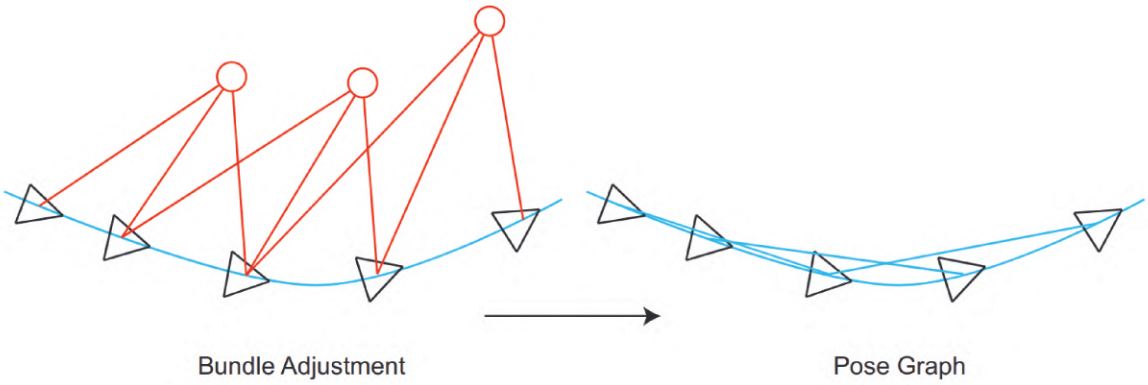

FIGURE 2.8 – Différences entre les structures des problèmes d'ajustement de faisceaux (gauche) et d'optimisation de graphe de poses (droite) (source [GZ21]). Les poses de la caméra sont marquées par les triangles noirs et les points de la cartes par les cercles rouges.

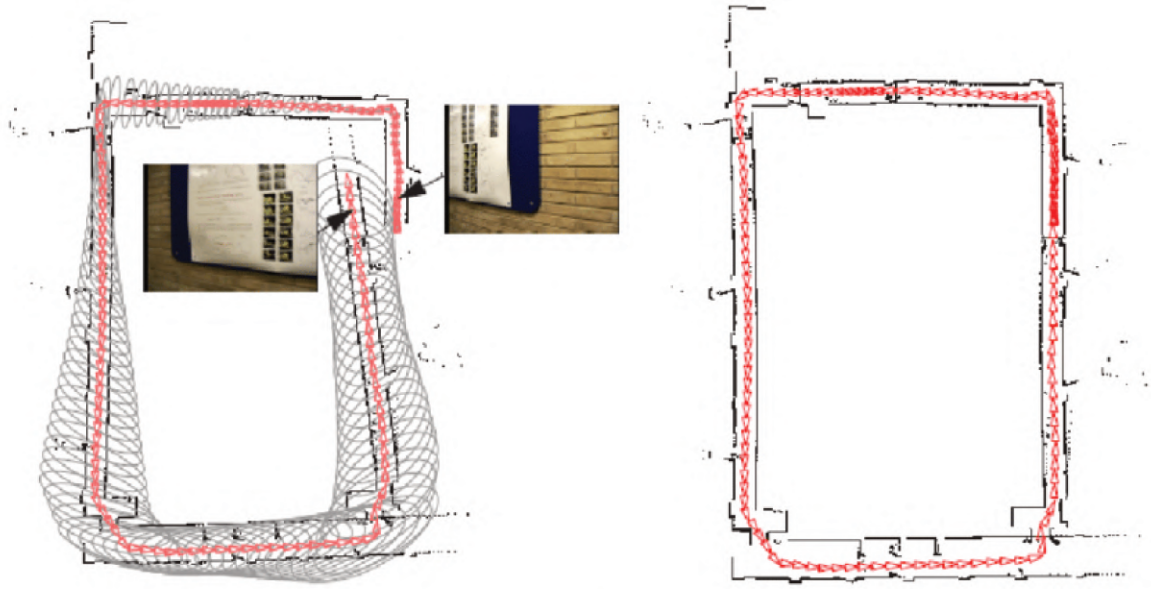

FIGURE 2.9 – Exemple de fermeture de boucle [HN07]. A gauche, la carte 2D et la trajectoire estimées par un système de SLAM juste avant une fermeture de boucle. Les ellipses grises représentent l'incertitude sur la localisation, qui augmente au cours du déplacement. La fermeture de boucle est détectée lorsque le capteur revient vers sa position initiale grâce à un algorithme de reconnaissance de lieu, qui parvient à mettre en correspondance la vue courante avec une image précédente. A droite, la carte et la trajectoire corrigées suite à la détection de la fermeture de boucle.

Chapitre 3

Modélisation 3D compacte à partir de caméra RGB-D

Sommaire

Nous nous concentrons ici sur les formes de représentation utilisées pour la reconstruction de cartes 3D denses à partir du flux vidéo d'un capteur RGB-D. Une carte peut être vue comme une description géométrique de l'environnement. L'objet de ce chapitre est d'introduire la nouvelle forme de représentation que nous avons développée pour la modélisation basse-résolution mais légère et rapide, qui sera la principale représentation utilisée pour la cartographie 3D dense de scènes dans nos travaux.

3.1 Introduction

Afin de pouvoir interagir dans et avec son milieu, un robot doit avoir accès en temps réel à une représentation explicite ou implicite de la géométrie de ce dernier. Cette information peut être fournie par des algorithmes de SLAM basés sur l'utilisation de capteurs RGB-D, particulièrement adaptés pour la reconstruction 3D dense de scènes d'intérieur. Les données de couleur et de profondeur de chacune des paires d'images RGB-D peuvent être combinées de manière incrémentale à l'aide de la connaissance des poses de la caméra, calculées par odométrie visuelle par exemple ou par recalage des cartes de profondeur, pour générer, mettre à jour et étendre un modèle 3D global de l'environnement observé. La forme de représentation choisie pour modéliser la surface de l'environnement doit être adaptée aux objectifs du robot et lui permettre de planifier au mieux ses actions.

De nombreux algorithmes de SLAM visuel reconstruisent et utilisent des cartes éparses ou semi-denses qui ne contiennent que les points les plus caractéristiques de l'environnement. Bien que ces solutions soient particulièrement légères et extensibles, elles ne peuvent être employées que dans des applications de localisation. En effet, la reconstruction d'un sous-ensemble parcimonieux de points clés de la surface réelle est insuffisante pour garantir une interaction sûre du robot avec son environnement à partir des cartes générées par ces systèmes. Pour effectuer des tâches de navigation autonome en limitant les risques de collision et de renversement, il est préférable d'utiliser une représentation 3D dense de la scène par rapport à une représentation éparse qui ne va pas forcément modéliser tous les obstacles. Si beaucoup d'approches de SLAM RGB-D dense ont été développées, la plupart de celles-ci se limitent à la reconstruction d'environnements de petite taille et/ou nécessitent du matériel lourd et coûteux (CPU et GPU de pointe) pour fonctionner en temps réel, à cause des types de représentation 3D qu'elles utilisent pour modéliser l'environnement. En effet, elles emploient pour la plupart des formes de représentation permettant un haut niveau de détail, mais trop coûteuses. Cela rend leur utilisation compliquée sur des systèmes embarqués de capacité mémoire et de puissance de calcul limitées, alors que dans beaucoup de situations une telle précision n'est pas nécessaire.

Dans ce chapitre nous présentons une nouvelle forme de représentation 3D nécessitant peu de ressources pour la reconstruction 3D dense d'environnements intérieurs à l'aide d'un capteur RGB-D. Contrairement aux approches existantes qui se focalisent sur la production de modèles 3D complexes avec un haut niveau de détail, la représentation développée, appelée représentation *supersurfel*, permet une cartographie basse-résolution mais consistente, légère et rapide. Son utilisation est particulièrement intéressante pour des plateformes de puissance réduite, n'ayant pas besoin d'une reconstruction très précise de leur milieu pour des tâches de localisation et planification, mais plutôt d'une reconstruction efficace, robuste, compacte et utilisable en temps réel.

Dans un premier temps, nous introduirons les différentes formes de représentation utilisées traditionnellement pour la reconstruction et cartographie 3D à partir de caméra RGB-D. Puis nous définirons la nouvelle forme de représentation mise en place et décrirons la méthode pour reconstruire une scène observée par une caméra RGB-D en utilisant cette représentation. La validité de l'approche sera évaluée et discutée en fin de chapitre.

3.2 Formes de représentations pour la reconstruction 3D dense

3.2.1 Reconstruction volumétrique

De nombreuses approches de reconstruction 3D à partir de caméra RGB-D se basent sur la représentation volumétrique introduite par [CL96]. Popularisées avec KinectFusion [Iza+11], l'un des premiers SLAM RGB-D, ces méthodes convertissent les cartes de profondeur en entrée en des fonctions implicites continues, qui sont ensuite discrétisées et fusionnées incrémentalement dans un modèle 3D unique prenant la forme d'une grille 3D régulière englobant la scène à reconstruire. Plus précisément, la représentation utilisée, appelée TSDF (Truncated Signed Distance Function), consiste en une fonction de distance signée discrétisée dans un volume subdivisé en éléments cubiques : les voxels. Un description est visible en figure 3.1. La TSDF signée encode en chaque voxel sa distance signée tronquée par rapport à la surface la plus proche, suivant la direction d'un rayon joignant le centre du capteur et le centre du voxel. La mise à jour du modèle se fait par simple moyenne pondérée avec les valeurs de la fontion de distance signée représentant la géométrie de la vue actuelle du capteur. Ces méthodes sont robustes aux différents bruits et produisent des résultats très convaincants, mais le fait de devoir prédéfinir la taille de la grille, sa résolution et la nécessité de représenter les espaces vides les rendent peu efficaces en terme de gestion de la mémoire. Une grille haute résolution de voxels est très lourde à stocker, ce qui limite grandement la taille de l'environnement que l'on peut reconstruire. La fonction de distance signée ne peut par ailleurs pas être visualisée directement via le biais d'un pipeline de rendu graphique standard et doit être convertie en une autre forme de représentation, comme un nuage de points ou un mesh, par des algorithmes de raycasting ou marching cubes par exemple [Iza+11].

Des solutions ont été proposées pour étendre la taille de la zone reconstructible ou diminuer l'empreinte mémoire nécessaire pour stocker la reconstruction. Par exemple [Whe+12] décale de manière dynamique la grille de voxels pour qu'elle suive le volume observé par la caméra et stocke séparément la partie du volume décalée hors de la grille pour libérer des voxels. La méthode [Nie+13] propose l'utilisation d'une table de hachage pour organiser plus efficacement les voxels du modèle, supprimant la nécessité de représenter les espaces vides. FastFusion [SSC14] exploite les bénéfices apportés par l'utilisation d'une structure hiérarchique de type octree à la place d'une grille régulière, pour accéder plus rapidement aux données. Une autre forme de représentation volumétrique célèbre, qui utilise aussi la structure hiérarchique d'octree est l'octoMap [Hor+13]. Il s'agit d'une grille d'occupation 3D spécifiquement conçue pour la navigation robotique. Elle encode en chaque voxel, non pas la distance à la surface la plus proche comme la TSDF, mais trois types d'informations sur l'espace qu'il contient : occupé, vide et inconnu. En plus de demander beaucoup de mémoire, une limite importante des représentations volumétriques est qu'elles ne sont pas directement déformables. Afin d'appliquer une transformation non rigide à la surface reconstruite, il est nécessaire d'extraire complètement ses mesures de la grille 3D (de les "dé-intégrer") pour obtenir un nuage de points sur lequel appliquer la déformation. Les points transformés doivent alors être à nouveau intégrés dans le volume. La surface reconstruite ne peut donc pas être immédiatement corrigée dans le cadre d'une fermeture de boucle par exemple et doit être complètement réintégrée pour

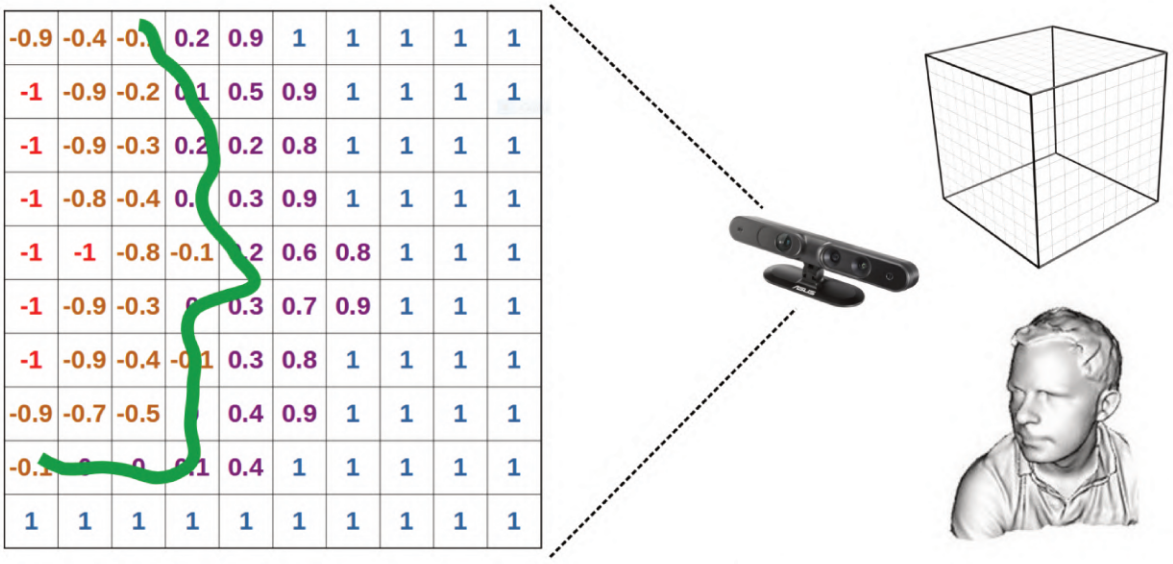

FIGURE 3.1 – A gauche une vue 2D schématique de la représentation volumétrique issue d'une image de profondeur. Chaque case représente un voxel stockant un échantillon de la fonction de distance signée. Le tracé vert est une coupe de la surface à reconstruire. A droite une grille 3D de voxels et un modèle généré par KinectFusion [Iza+11].

refléter la trajectoire rectifiée, ce qui peut prendre énormément de temps dans le cadre de la reconstruction de scènes larges.

3.2.2 Reconstruction basée nuages de points et surfels

Une manière plus directe de reconstruire une scène observée par un capteur RGB-D consiste à extraire de chaque image de profondeur ou d'un sous-ensemble d'images clés, un nuage de points 3D par rétroprojection des pixels dans l'espace. Connaissant les poses du capteur associées aux images, les nuages de points peuvent être alignés et combinés dans un même référentiel pour générer et étendre un modèle 3D global de la scène. L'absence de connectivité explicite rend la représentation flexible, facilement manipulable et les espaces libres n'ont pas à être représentés. Cependant la reconstruction résultante est discontinue, présente de très nombreuses redondances et est sensible au bruit, contrairement aux représentations volumétriques. Le rendu visuel est aussi très clairsemé. Dans la pratique, des nuages de points bruts issus de caméras RGB-D ne sont quasiment jamais utilisés directement en robotique. Après prétraitement, ils peuvent être utilisés pour de la localisation à partir de différentes méthodes de recalage de points 3D [BM92] ; [Rus19] ; [Mag09] ; [BTP13]. Pour de la planification, ils sont généralement fusionnés et intégrés a posteriori dans une OctoMap, comme dans [End+14].

Afin d'obtenir un rendu visuel plus dense et un modèle 3D moins bruité, tout en évitant les contraintes des représentations volumétriques, de nombreux SLAM RGB-D utilisent une représentation plus complète aussi basée sur des primitives ponctuelles extraites à partir des pixels d'images de pronfondeur : les surfels [Pfi+00] (pour surface elements). La scène recons-

truite est modélisée sous la forme d'un ensemble de petits patchs circulaires de tailles uniformes dans l'espace 3D, constituant une représentation compacte d'une séquence de nuages de points fusionnés. Le processus de fusion de données permet de réduire les redondances dans la surface reconstruite et l'impact du bruit de quantification des mesures de profondeur. L'utilisation de surfels comme primitives permet une meilleure représentation des surfaces qu'un simple nuage de points qui donne un rendu plus parcimonieux. L'un des avantage de la modélisation via les surfels par rapport aux grilles de voxels est aussi qu'ils s'adaptent à la résolution du capteur et à la profondeur de la scène. Un surfel loin de la caméra prend moins de place dans l'espace image lorsqu'il est projeté, qu'un élément proche de celle-ci. Chaque surfel encode comme principaux attributs une position, une couleur, une normale. Dans le cadre de la reconstruction 3D d'environnements, un attribut de confiance est ajouté pour quantifier la fiabilité du surfel.

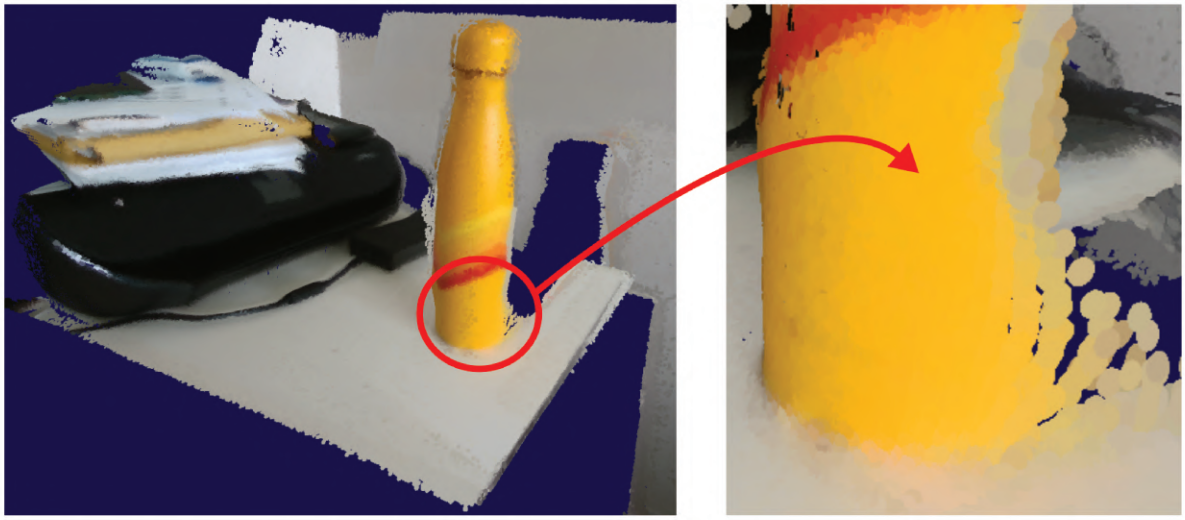

FIGURE 3.2 – Exemple d'un rendu obtenu avec une représentation à base de surfels. Les primitives circulaires sont clairement visibles sur l'image de droite.

Le système de cartographie RGB-D dense proposé par HENRY et al. [Hen+12] utilise la représentation surfel pour fusionner les mesures d'un flux vidéo RGB-D, à partir des poses optimisées de la caméra, une fois la numérisation terminée. [Kel+13] et [Whe+15] sont deux des méthodes les plus représentatives de SLAM RGB-D en ligne basées sur une représentation surfel de l'environnement. Le modèle 3D global est mis à jour à chaque nouvelle acquisition d'une paire RGB-D par association projective des surfels du modèle global avec ceux de la vue courante. A partir de la pose actuelle connue de la caméra, les surfels du modèle reconstruit sont reprojetés dans le plan image de la vue courante et il y a correspondance si la projection d'un surfel de la vue courante recouvre celle d'un surfel du modèle. Si des correspondances sont trouvées, les surfels sont fusionnés par moyenne pondérée des attributs et la valeur de confiance associée est augmentée. La mise en correspondance entre deux surfels est validée si ils sont suffisamment proches en terme d'orientation et de position. Si aucun point de correspondance viable n'est trouvé, le surfel de la vue courante est ajouté au modèle en tant que point instable. Un point ayant subi plusieurs fusions, donc observé de manière répétée, voit sa valeur de confiance augmentée et peut donc passer de l'état instable à l'état stable. Les

éléments trop longtemps instables sont supprimés. La figure 3.2 montre une reconstruction 3D sous la forme de surfels.

Bien que plus flexibles et légères que les représentations volumétriques, les représentations basées sur les points et les surfels présentent en général plus de bruits et de redondances. De plus, un modèle 3D sous la forme d'ensemble de surfels peut vite contenir plusieurs millions de primitives. En effet, chaque paire RGB-D fournit plus de 300 000 points pour une résolution de 640×480 pixels. La mémoire nécessaire pour stocker une telle quantité d'éléments reste importante et son maniement est lent et coûteux. Le traitement des données denses issues d'un capteur RGB-D force les méthodes citées précédemment à recourir à la programmation parallèle sur des CPU et des GPU de pointe pour fonctionner en temps réel. Il est plus difficile d'obtenir des performances semblables sur des plateformes robotiques mobiles dont les ressources sont souvent limitées. Sachant que des surfels voisins sont souvent extrêmement similaires en matière de couleur et de géométrie, il peut sembler inefficace voir inutile de stocker autant d'informations.

3.3 Reconstruction supersurfel

3.3.1 Définition du modèle

La solution basse-résolution, mais légère et cohérente, que nous proposons s'inspire des méthodes basées sur les surfels [Whe+15] ; [Kel+13]. Elle représente la scène sous la forme d'un ensemble non structuré de primitives appelées supersurfels, exprimées dans le référentiel global du monde $\mathfrak{w}$. Cet ensemble constitue le modèle global $\mathcal{M} = \{\mathcal{M}_k\}$, qu'on peut aussi désigner par carte globale dans le cadre de la navigation. Il s'agit d'une représentation simplifiée de la géométrie de la scène observée, assimilée à un ensemble de patchs elliptiques indépendants. Un supersurfel peut être vu comme l'approximation de la reprojection 3D d'un superpixel associé. Un superpixel [RM03] définit un groupe de pixels connectés similaires. Au lieu d'associer à chaque pixel d'une image un surfel, l'approche décrite propose de générer des supersurfels à partir de superpixels, découpant l'image en sous-ensembles homogènes. Le but est de réduire les mesures redondantes et le nombre de primitives 3D utilisées pour représenter la scène tout en préservant au mieux sa structure géométrique et sa texture, afin de gagner en efficacité et conserver un temps d'exécution relativement stable lorsque la taille de la reconstruction augmente. Un modèle 3D compact est aussi plus simple à manipuler à corriger. Il peut être récupéré rapidement pour servir à d'autres opérations comme la planification de trajectoires sans nécessiter de nombreux post-traitement. De plus la modélisation de primitives à partir de superpixels permet de diminuer fortement l'influence des valeurs aberrantes et bruitées dans les mesures de profondeur acquises. Le modèle $\mathcal{M}$ est complété et mis à jour incrémentalement pour chaque nouvelle image acquise par une caméra RGB-D, à partir de l'ensemble $\mathcal{F}$ de supersurfels $\mathcal{F}_i$ associés à cette image.

Chaque supersurfel $\mathcal{M}_k$ du modèle $\mathcal{M}$ est donc défini comme un patch elliptique de l'espace 3D qui encode plusieurs attributs :

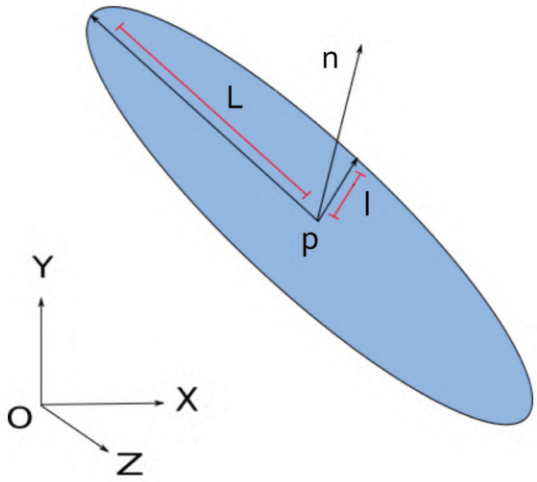

FIGURE 3.3 – Supersurfel défini dans le repère $(O; X, Y, Z)$, de position $\mathbf{p}$, normale $\mathbf{n}$, largeur l et longueur L.

- la position de son centre $\mathbf{p}_k \in \mathbb{R}^3$;
- sa couleur $\mathbf{c}_k \in \mathbb{R}^3$;
- son orientation $\mathbf{O}_k \in SO(3)$;
- son élongation latérale $l_k \in \mathbb{R}$;
- son élongation longitudinale $L_k \in \mathbb{R}$;
- un horodatage $h_{0,k} \in \mathbb{R}_+$ qui indique sa première apparition ;
- un horodatge $h_k \in \mathbb{R}_+$ qui stocke le dernier instant où le supersurfel a été observé ;
- un poids $w_k \in \mathbb{R}_+$ pour quantifier la confiance que l'on a par rapport à la validité du supersurfel (état stable ou instable) ;
- une matrice de covariance 3D $\mathbf{\Sigma}_k \in \mathscr{M}_3(\mathbb{R})$ décrivant sa forme.

Par souci de commodité, on désigne aussi la normale du supersurfel $\mathcal{M}_k$ par $\mathbf{n}_k$. Elle correspond à la troisième colonne de sa matrice d'orientation. Une vue schématique d'un supersurfel est donnée en figure 3.3.

3.3.2 Segmentation d'une paire RGB-D en superpixels

La première étape consiste à récupérer et à segmenter en superpixels la paire RGB-D $\{C, D\}$ de la vue courante. Les superpixels permettent de simplifier une image et donc d'accroître l'efficacité des calculs réalisés ensuite, tout en conservant l'information importante. Ici, les informations de couleur et de profondeur sont toutes les deux prises en compte. Nous avons utilisé une version GPU adaptée aux images RGB-D de l'algorithme "Topology Preserving Segmentation" (TPS) [YMU14] pour partitionner l'image courante en superpixels adaptés, c'est-à-dire préservant autant que possible les contours et les discontinuités géométriques, dans le but d'assurer une modélisation compacte mais cohérente.

L'approche TPS génère un ensemble de superpixels regroupant des pixels de couleur similaire et dont les rétroprojections 3D appartiennent à un même plan incliné. On note $\theta_i = \{A_i, B_i, C_i\}$ la paramétrisation du plan de disparité assigné au superpixel d'indice i. La disparité d'un pixel $\mathbf{u}[x, y]^\intercal$ appartenant à ce superpixel peut alors être calculée à partir du plan incliné du superpixel qui l'englobe comme :

$$d(\mathbf{u}, \theta_i) = A_i x + B_i y + C_i \tag{3.1}$$

Soit e_u le label du superpixel assigné par segmentation au pixel $\mathbf{u}$. L'algorithme TPS est une méthode itérative qui, à partir du découpage initial de l'image en une grille régulière de taille choisie, consiste à décaler les frontières des superpixels tout en préservant la topologie (superpixels connexes) par minimisation d'une énergie E_{tps}. L'algorithme est très efficace puisque seuls les pixels $\mathbf{u}$ sur les contours sont mis à jour à chaque itération. L'énergie E_{tps} est constituée de quatre termes :

$$E_{tps} = \sum_{\mathbf{u}} E_{col} + \lambda_{pos} \sum_{\mathbf{u}} E_{pos} + \lambda_{depth} \sum_{\mathbf{u}} E_{depth} + \lambda_{bou} \sum_{\{\mathbf{u},\mathbf{v}\} \in \mathcal{V}_8} E_{bou} \tag{3.2}$$

où λ_{pos}, λ_{depth}, λ_{bou} sont des constantes et $\mathcal{V}_8$ l'ensemble des paires des voisinages de 8-connexité des pixels.
Le premier terme encourage les pixels avec des couleurs similaires à être dans un même superpixel :

$$E_{col} = \| C(\mathbf{u}) - \mathbf{c}_{e_u} \|^2 \tag{3.3}$$

avec $C(\mathbf{u})$ la couleur du pixel $\mathbf{u}$ dans l'image C et $\mathbf{c}_{e_u}$ la couleur du superpixel contenant $\mathbf{u}$.
Le second permet de favoriser la proximité spatiale et la compacité dans la formation des superpixels :

$$E_{pos} = \| \mathbf{u} - \mu_{e_u} \|^2 \tag{3.4}$$

où μ_{e_u} est la position du centroïde du superpixel.
Le troisième contraint le plan de disparité estimé associé au superpixel à être en accord avec l'information fournie par l'image de profondeur, à la respecter du mieux possible :

$$E_{depth} = \begin{cases} (D(\mathbf{u}) - d(\mathbf{u}, \theta_{e_u}))^2 & \text{si } D(\mathbf{u}) \text{ est valide} \\ \gamma_d & \text{sinon} \end{cases} \tag{3.5}$$

γ_d est une constante utilisée pour les pixels qui n'ont pas de mesure de profondeur ou dont la valeur est aberrante (on se limite généralement à des valeurs de profondeur inférieures à cinq mètres en raison d'imprécisions trop importantes au-delà).
Enfin, la dernière composante sert à favoriser la constitution de bordures régulière pour les superpixels :

$$E_{bou} = \begin{cases} 0 & \text{si } e_u = e_v \\ 1 & \text{sinon} \end{cases} \tag{3.6}$$

avec e_u l'indice du superpixel contenant le pixel $\mathbf{u}$ et e_v celui du segment qui contient le pixel $\mathbf{v}$.

Une fois la segmentation terminée, un critère de coplanarité est utilisé pour lisser la carte de profondeur D à partir des superpixels générés afin de réduire l'influence du bruit. Pour

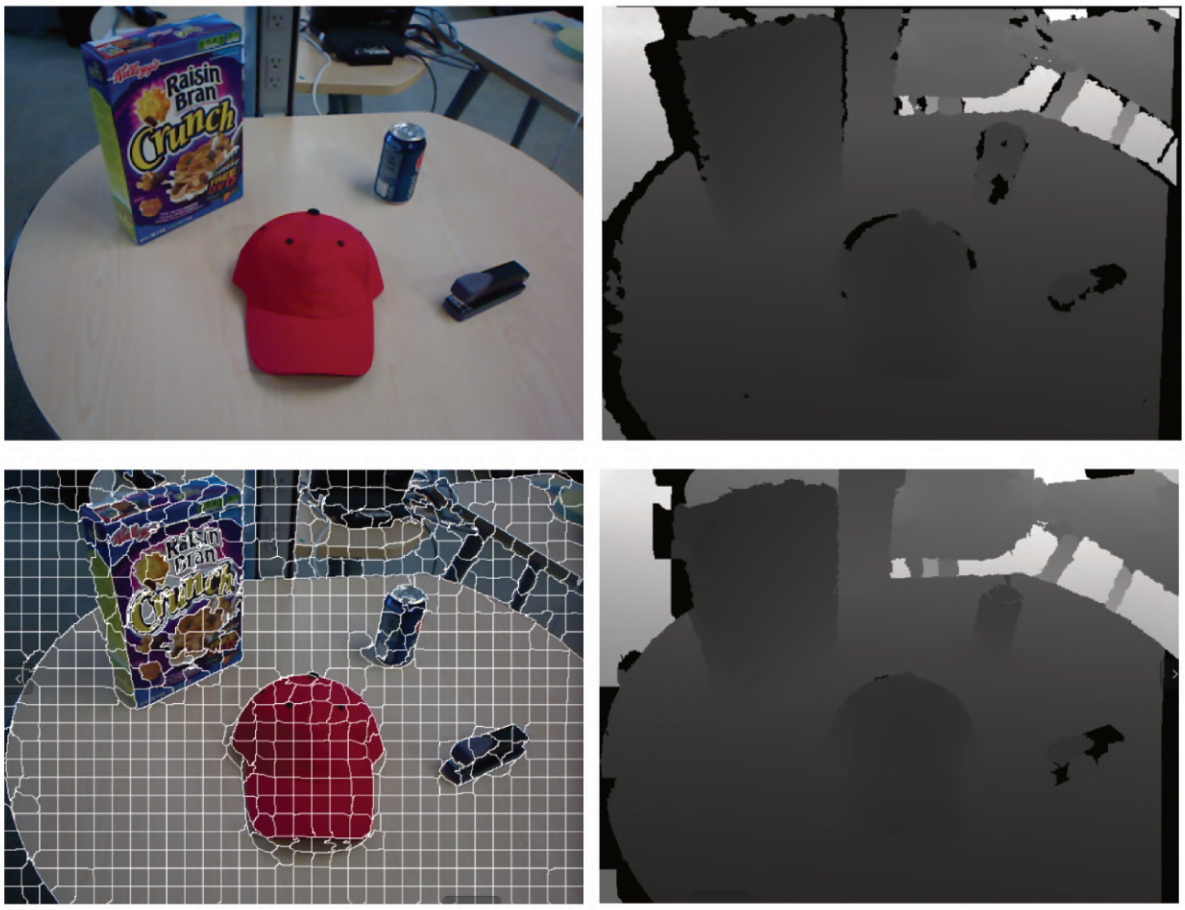

FIGURE 3.4 – Visualisation de superpixels RGB-D obtenus avec l'approche TPS [YMU14] en bas. La carte de profondeur filtrée en bas à droite, présentant les plans inclinés des superpixels, est moins bruitée que l'image initiale au dessus. La paire RGB-D originale est issue du jeu de données "RGB-D Scenes Dataset" [Lai+11].

chaque superpixel, on note d_i la disparité de son centre $[\bar{x}_i, \bar{y}_i]^\intercal$, obtenu par moyennage des positions des pixels qui le composent, et $\frac{\partial d_i}{\partial x}$, $\frac{\partial d_i}{\partial y}$ les dérivées de la disparité par rapport à la position. Pour lisser la carte de pronfondeur, on cherche les nouveaux états $\mathbf{X}_i$ des superpixels qui minimisent la distance aux observations $\mathbf{Z}_i[d_i, \frac{\partial d_i}{\partial x}, \frac{\partial d_i}{\partial y}]^\intercal$ ainsi que les variations entre les superpixels voisins. On peut exprimer l'énergie à minimiser par la formule suivante :

$$E_{smo} = \sum_i \left(\alpha \|\mathbf{Z}_i - \mathbf{X}_i\|^2 + \beta \sum_{j \in \mathcal{N}_i} \|\mathbf{X}_i - \mathbf{H}_{ij}\mathbf{X}_j\|^2 \right) \tag{3.7}$$

avec $\mathbf{H}_{ij} = \begin{bmatrix} 1 & \bar{x}_i - \bar{x}_j & \bar{y}_i - \bar{y}_j \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ et $\mathcal{N}_i$ les superpixels voisins du superpixel d'indice i. Le problème se ramène à un système de moindres carrés linéaires qu'on résout de manière itérative avec la méthode de Block-Jacobi [Saa03]. Le détail des calculs est fourni en annexe A. Un exemple de segmentation en superpixels RGB-D avec l'approche TPS est montré en figure

3.4.

3.3.3 Extraction de supersurfels

Les superpixels obtenus sont utilisés pour générer l'ensemble $\mathcal{F} = \{\mathcal{F}_i\}$ de supersurfels de la vue courante $\{C, D\}$ (cf. figure 3.5). Un supersurfel est extrait dans le référentiel de la caméra $\mathfrak{c}$ pour chaque superpixel. Soit un supersurfel $\mathcal{F}_i$ construit à partir du superpixel d'indice i correspondant, qui regroupe les pixels $\mathbf{u}_j, j = 1, 2, ..., N$ avec N la taille du superpixel. Il prend pour position $\mathbf{p}_i$ la moyenne des positions des retro-projections 3D $\pi^{-1}(\mathbf{u}_j, D(\mathbf{u}_j))$ des pixels $\mathbf{u}_j$ contenus dans le superpixel d'indice i associé. De même la couleur affectée au supersurfel est la moyenne des couleurs $C(\mathbf{u}_j)$ des pixels, calculée dans l'espace CIELab, plus conforme au système visual humain.

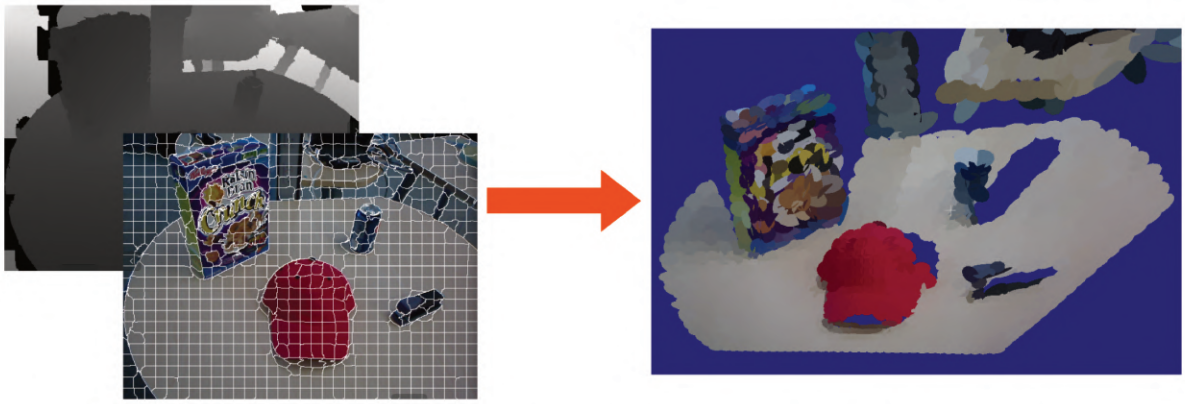

FIGURE 3.5 – Extraction de supersurfels à partir de la segmentation en superpixels d'une paire RGB-D.

Pour déterminer l'orientation ainsi que la taille du supersurfel, on effectue l'analyse en composantes principales (ACP) de la matrice de covariances de l'ensemble des rétro-projections 3D issues du superpixel. L'ACP est une méthode classique pour estimer la normale à une surface en un point donné à partir de ses voisins [Hop+92]. Chaque point est traité comme l'échantillon d'une loi normale multidimensionnelle, dont les coordonnées spatiales x, y, z sont trois variables indépendantes. La décomposition en éléments propres de la matrice de covariance de l'ensemble de points donné permet de trouver une base orthogonale qui représente au mieux l'ensemble des points, approximant la surface locale sous la forme de plan. Les vecteurs propres calculés associés aux deux plus grandes valeurs propres indiquent les directions dans lesquelles les données varient le plus tandis que les valeurs propres quantifient l'étalement des données suivant les directions associées. Le vecteur propre qui a la plus petite valeur propre pointe dans la direction de la normale ou dans la direction de son opposé. Dans l'espace 3D, il est alors possible de simplifier et de représenter un ensemble de points par une ellipsoïde, ou par une ellipse en 2D comme le montre la figure 3.6. Dans notre cas, la segmentation en superpixels regroupe les points voisins quasi-coplanaires. On approxime donc l'ellipsoïde par une ellipse. La matrice de covariance $\mathbf{\Sigma}_i$, associée au supersurfel $\mathcal{F}_i$ de position $\mathbf{p}_i$, est calculée

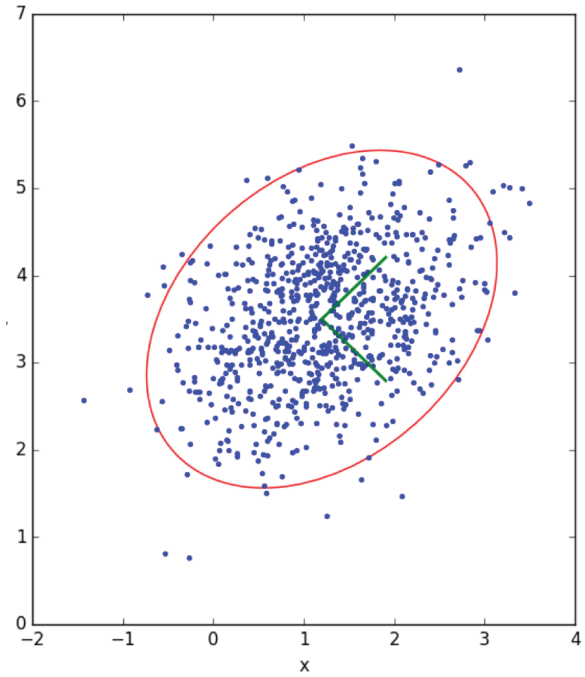

FIGURE 3.6 – Exemple d'ellipse de confiance à 95% d'un nuage de points 2D obtenue par Analyse en Composantes Principales.

à partir de la formule suivante :

$$\mathbf{\Sigma}_i = \frac{1}{N}\sum_{j=1}^{N}(\pi^{-1}(\mathbf{u}_j, D(\mathbf{u}_j)) - \mathbf{p}_i)(\pi^{-1}(\mathbf{u}_j, D(\mathbf{u}_j)) - \mathbf{p}_i)^\intercal \tag{3.8}$$

Cette matrice est carrée, symétrique et à coefficients réels, donc diagonalisables dans une base orthonormée d'après le théorème spectral. Les valeurs propres $\lambda_1, \lambda_2, \lambda_3 (\lambda_1 \geqslant \lambda_2 \geqslant \lambda_3)$ et vecteurs propres $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$ de la matrice de covariance $\mathbf{\Sigma}_i$ peuvent donc être obtenus par sa diagonalisation :

$$\mathbf{\Sigma}_i = \mathbf{PDP}^\intercal \quad \text{avec} \quad \mathbf{D} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \quad \text{et} \quad \mathbf{P} = \begin{bmatrix} e_{1x} & e_{2x} & e_{3x} \\ e_{1y} & e_{2y} & e_{3y} \\ e_{1z} & e_{2z} & e_{3z} \end{bmatrix} \tag{3.9}$$

Les trois vecteurs propres donnent l'orientation du supersurfel : $\mathbf{O}_i \leftarrow \mathbf{P}$. La normale $\mathbf{n}_i$ est représenté par le vecteur propre $\mathbf{e_3}$ associé à la plus petite valeur propre λ_3. En revanche, l'ACP ne permet pas de déterminer mathématiquement le signe de la normale, qui est orientée suivant le point de vue de la caméra (ici en $[0, 0, 0]^\intercal$) de façon à satisfaire la relation suivante :

$$\mathbf{n}_i \cdot \mathbf{p}_i < 0 \tag{3.10}$$

Les élongations longitudinale et latérale L_i, l_i sont respectivement obtenues par multiplication des racines carrées des valeurs propres λ_1, λ_2 avec un facteur d'échelle revenant à considérer une ellipse de confiance contenant 95% des retro-projections 3D :

$$L_i = 2.448\sqrt{\lambda_1} \quad \text{et} \quad l_i = 2.448\sqrt{\lambda_2} \tag{3.11}$$

Le marqueurs temporels $h_{0,i}, h_i$ sont initialisés avec le temps courant t et le poids de confiance w_i prend une valeur faible correspondant au ratio entre le nombre de pixels ayant une profondeur valide $N_v \in N$ et le nombre total de pixels N contenus dans le superpixel associé :

$$w_i \leftarrow N_v/N \tag{3.12}$$

Une mesure de profondeur est dite valide si elle est comprise dans l'intervalle de portée du capteur.

3.3.4 Mise à jour du modèle

Dans cette étape, basée sur l'approche de fusion de [Kel+13], l'ensemble de supersurfels $\mathcal{F} = \{\mathcal{F}_i\}$ extrait de la vue courante est intégré au modèle global $\mathcal{M} = \{\mathcal{M}_k\}$ pour le compléter et le parfaire. Les nouveaux supersurfels générés à partir de la paire RGB-D courante sont soit directement ajoutés dans le modèle, soit combinés avec des supersurfels du modèle pour mettre à jour leurs attributs. Un supersurfel $\mathcal{M}_k$ fusionné avec un supersufel $\mathcal{F}_i$ verra sa valeur de confiance w_k croître. Un supersurfel du modèle peut ainsi passer de l'état instable à l'état stable si il est observé de manière répétée durant la numérisation de l'environnement. Les supersurfels vérifiant $w_k > w_{stable}$, avec w_{stable} un seuil fixé, sont considérés comme stables.

La fusion des supersurfels de la vue courante $\mathcal{F}$ avec les supersurfels similaires du modèle reconstruit $\mathcal{M}$ a pour but d'affiner le modèle et de réduire les redondances. Les données sont fusionnées dans le référentiel global du monde $\mathfrak{w}$, dans lequel est exprimé le modèle $\mathcal{M}$. Il est donc nécessaire dans un premier temps de transformer les attributs des supersurfels de la vue courante $\mathcal{F}$, initialisés dans le repère de la caméra $\mathfrak{c}$, pour les exprimer dans le repère monde $\mathfrak{w}$. Pour cela il faut disposer de la pose de la caméra dans le repère monde, qu'on suppose connue ici. Elle peut être obtenue par différentes techniques de localisation comme [FCT20] ; [KSC13b] ; [Whe+13] par exemple. Nous exposons dans le chapitre 4 notre propre solution d'odométrie visuelle pour l'estimation de la pose de la caméra. Soit $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} \in SE(3)$ la pose de la caméra dans le repère global. Elle est composée de deux parties : une matrice de rotation $\mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}} \in SO(3)$, qui matérialise l'orientation du repère de la caméra dans le repère monde, et un vecteur de translation $\mathbf{t}_{\mathfrak{c}}^{\mathfrak{w}} \in \mathcal{R}^3$, pour la position du repère caméra dans le repère monde. La transformation d'une supersurfel $\mathcal{F}_i$ du système de coordonnées de la caméra $\mathfrak{c}$ au repère monde $\mathfrak{w}$ est réalisée par les opérations suivantes :

$$(\mathbf{p}_i)_{\mathfrak{w}} = \mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}}(\mathbf{p}_i)_{\mathfrak{c}} + \mathbf{t}_{\mathfrak{c}}^{\mathfrak{w}} \tag{3.13}$$

$$(\mathbf{O}_i)_{\mathfrak{w}} = \mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}}(\mathbf{O}_i)_{\mathfrak{c}} \tag{3.14}$$

$$(\mathbf{\Sigma}_i)_{\mathfrak{w}} = \mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}}(\mathbf{\Sigma}_i)_{\mathfrak{c}}\mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}\intercal} \tag{3.15}$$

Une fois le changement de repère effectué, on cherche pour chaque nouveau supersurfel $\mathcal{F}_i$, si un supersurfel ressemblant $\mathcal{M}_k$ est présent dans le modèle. Pour cela nous utilisons une stratégie d'association par projection de données. Les centres $\mathbf{p}_k$ des supersurfels du modèle $\mathcal{M}$ sont projetés dans le plan image Ω sur les superpixels de la vue courante et appariés avec

leurs supersurfels associés $\mathcal{F}_i$. On vérifie alors si les supersurfels $\mathcal{M}_k$ et $\mathcal{F}_i$ sont assez similaires pour être fusionnés. Trois critères sont évalués :

1. les pairs de supersurfels pour lesquelles la distance euclidienne entre les centres excède un seuil δ_{dist} sont rejetées ;
2. les supersurfels dont l'angle de divergence entre les normales dépasse un seuil δ_{norm} sont ignorés ;
3. la distance entre les couleurs des supersurfels dans l'espace colorimétrique CIELab, en ignorant le canal L^* pour rendre la méthode robuste aux changements d'intensité lumineuse et ne considérer que l'information purement chromatique, doit être inférieure à une valeur δ_{col}.

Dans le cas où plusieurs supersurfels du modèle sont associés à un même supersurfel de l'image courante, on conserve celui dont la valeur de confiance est la plus élevée. Si plusieurs ont la même valeur de confiance, le supersurfel du modèle qui minimise le deuxième critère est sélectionné.

Si aucun supersurfel correspondant $\mathcal{M}_k$ n'est trouvé, le supersurfel $\mathcal{F}_i$ est simplement ajouté dans le modèle. Sinon, pour chaque paire de correspondances $\{\mathcal{M}_k, \mathcal{F}_i\}$ trouvée, les attributs de $\mathcal{M}_k$ sont mis à jour par fusion de données à partir des attributs de $\mathcal{F}_i$. Les nouvelles positions et covariances sont calculées en appliquant la stratégie d'intersection de covariances [JU07], dans le but d'obtenir une mise à jour fiable et d'affiner les supersurfels lorsque la caméra se rapproche de la surface :

$$\alpha = \frac{w_k}{w_k + w_i}. \tag{3.16}$$

$$\mathbf{\Sigma}_k^{'} = (\alpha {\mathbf{\Sigma}_k}^{-1} + (1-\alpha){\mathbf{\Sigma}_i}^{-1})^{-1} \tag{3.17}$$

$$\mathbf{p}_k \leftarrow \mathbf{\Sigma}_k^{'}(\alpha {\mathbf{\Sigma}_k}^{-1}\mathbf{p}_k + (1-\alpha){\mathbf{\Sigma}_i}^{-1}\mathbf{p}_i) \tag{3.18}$$

$$\mathbf{\Sigma}_k \leftarrow \mathbf{\Sigma}_k^{'} \tag{3.19}$$

La couleur est mise à jour par simple moyenne pondérée dans l'espace CIELab :

$$\mathbf{c}_k \leftarrow \frac{w_k\mathbf{c}_k + w_i\mathbf{c}_i}{w_k + w_i}. \tag{3.20}$$

La même procédure que celle utilisée lors de l'étape de génération des supersurfels (Section 3.3.3) est appliquée pour calculer les nouvelles valeurs d'élongations L_k, l_k à partir de la covariance mise à jour. Le poids de confiance w_k est incrémentée :

$$w_k \leftarrow w_k + w_i \tag{3.21}$$

Enfin, l'horodatage h_k est modifiée avec la valeur du temps courant t.

Suite à la fusion, deux stratégies sont appliquées pour supprimer les aberrations du modèle. Premièrement, les supersurfels qui restent dans un état instable (sans être observés) pendant trop longtemps sont éliminés après une période Δt. Deuxièmement, les supersurfels du modèle qui se trouvent devant les supersurfels nouvellement mis à jour, par rapport à la caméra, sont détectés et supprimés. Ils représentent des violations de l'espace libre et correspondent probablement à des éléments de la scène qui ont changé de place ou à des données erronées introduites par une mauvaise estimation d'odométrie visuelle par exemple.

3.4 Évaluation

La reconstruction 3D à partir de la représentation supersurfel avec caméra RGB-D a été évaluée sur un ordinateur portable équipé d'une carte graphique Nvidia GTX 950M avec 2 Go de mémoire et d'un processeur Intel Core i5-6300HQ. Le système de reconstruction 3D a été implémenté en C++ et CUDA. Des essais sur une carte embarquée Nvidia Jetson TX1 et sur un second ordinateur portable doté d'une carte graphique Nvidia Quadro K610M plus ancienne et moins puissante ont permis l'observation des performances similaires à celles présentées dans cette partie.

Des séquences vidéos issues du jeu de données réelles du TUM RGB-D [Stu+12] (*fr1_xyz*, *fr1_room*, *fr2_rpy*, *fr3_office*), filmées dans des bureaux, et du jeu de données synthétiques d'ICL-NUIM [Han+14] (*kt0*, *kt1*, *kt2*, *kt3*), simulant un salon d'appartement, ont étés utilisées pour les différents tests réalisés. Chaque séquence consiste en une suite d'images de couleur et de profondeur, enregistrées à une fréquence de 30 Hz et à une résolution de 640×480 pixels, appariées en fonction de leurs horodatages et recalées. Une caméra Microsoft Kinect a été utilisée pour les enregistrements. Les vidéos sont toutes fournies avec des mesures de vérité terrain sur la trajectoire de la caméra. Un système de capture du mouvement a été utilisé dans le cas des données réelles du TUM pour faire l'acquisition des données de vérité terrain à une fréquence de 100 Hz. En plus des images et des trajectoires, le jeu de données synthétiques d'ICL-NUIM comprend aussi un modèle 3D vérité terrain, sous forme de mesh, de la surface à reconstruire, qui peut servir de référence pour mesurer la précision de la méthode de reconstruction. Les séquences sont décrites dans le tableau 3.1.

Séquence	**Durée (s)**	**Longueur de la trajectoire (m)**
fr1_xyz	30.09	7.112
fr1_room	48.90	15.989
fr2_rpy	109.97	7.029
fr3_office	87.09	21.455
kt0	51.00	6.540
kt1	33.00	2.050
kt2	30.00	8.430
kt3	42.00	11.320

TABLE 3.1 – Description des séquences vidéos utilisées pour l'évaluation de la représentation supersurfel.

Nous avons comparé notre méthode de reconstruction 3D basée sur les supersurfels avec celle de l'approche état de l'art ElasticFusion [Whe+15], plus précise puisqu'elle modélise la scène à partir de surfels, en générant une primitive 3D pour chaque pixel d'une image RGB-D. Les mesures de vérité terrain des trajectoires données avec chacune des vidéos ont été fournies aux deux systèmes pour permettre de focaliser les expériences sur le processus de reconstruction 3D et sur la représentation employée, sans prendre en compte pour l'instant

le suivi de la caméra, donc indépendamment du processus de localisation. La précision et l'efficacité de notre méthode ont été évaluées.

3.4.1 Choix de la taille des superpixels

Dans la méthode de segmentation en superpixels utilisée, les superpixels résultants possèdent tous environ la même taille (le même nombre de pixels contenus). Cette taille est fixée par le choix de la résolution de la grille initiale employée lors de la segmentation (en 3.3.2) et influence la densité de la reconstruction supersurfel. En effet, plus les superpixels sont petits, plus le nombre de supersurfels généré est grand et plus le modèle 3D reconstruit est précis mais lourd et lent à manipuler. Il est possible de se faire une idée de la densité voulue à une profondeur z en considérant un disque de rayon r_{3D} dans l'espace 3D et de rayon projeté r_{proj} dans le plan image Ω, avec la relation suivante :

$$r_{proj} = \frac{f}{z} r_{3D} \tag{3.22}$$

où f est la focale de la caméra de profondeur. Par exemple, en prenant $2r_{proj}$ pour côté de chacun des blocs de la grille initiale, on obtient des supersurfels dont les centres sont globalement séparés d'environs $2r_{3D}$ à la profondeur z. Inversement, connaissant la portée minimale d'un capteur RGB-D et l'espacement approximativement souhaité entre les supersurfels à cette distance minimale, on peut fixer les valeurs de z et r_{3D} pour déterminer la valeur de la taille des superpixels à utiliser.

Pour l'ensemble des expériences, nous avons utilisé des superpixels initialisés à partir d'une grille régulière de blocs de 20×20 pixels (donc $r_{proj} = 10$), permettant une compression importante des données tout en conservant une bonne précision et un bon niveau de détail (cf. 3.4.2). Avec la caméra Intel Realsense D435 cette valeur permet d'obtenir des supersurfels espacés au minimum d'approximativement 5 cm lors de leur génération. La figure 3.7 montre que la segmentation en superpixels initialisés à partir de blocs de 20×20 pixels préserve relativement bien la texture et la géométrie d'une paire RGB-D, contrairement à un sous-échantillonnage avec un pas régulier de 20 pixels, qui déteriore les mesures, introduit des discontinuités brutales et résulte en une perte d'information conséquente.

3.4.2 Précision de la reconstruction

Bien que la précision de la surface reconstruite ne soit pas la finalité première de notre méthode, qui met l'accent sur l'efficacité et la légèreté, le modèle 3D obtenu doit être suffisamment cohérent et proche de la réalité afin de servir pour des tâches de navigation par exemple. C'est pourquoi nous avons évalué la qualité de la surface produite par notre approche de reconstruction 3D sur les séquences *kt0*, *kt1*, *kt2*, *kt3*, du jeu de données synthétique ICL-NUIM et de son modèle 3D vérité terrain. Le modèle 3D sous forme d'ensemble de supersurfels reconstruit pour chaque séquence vidéo a été converti en un nuage de points dense par suréchantillonnage de la surface de chaque supersurfel, puis la distance moyenne entre les

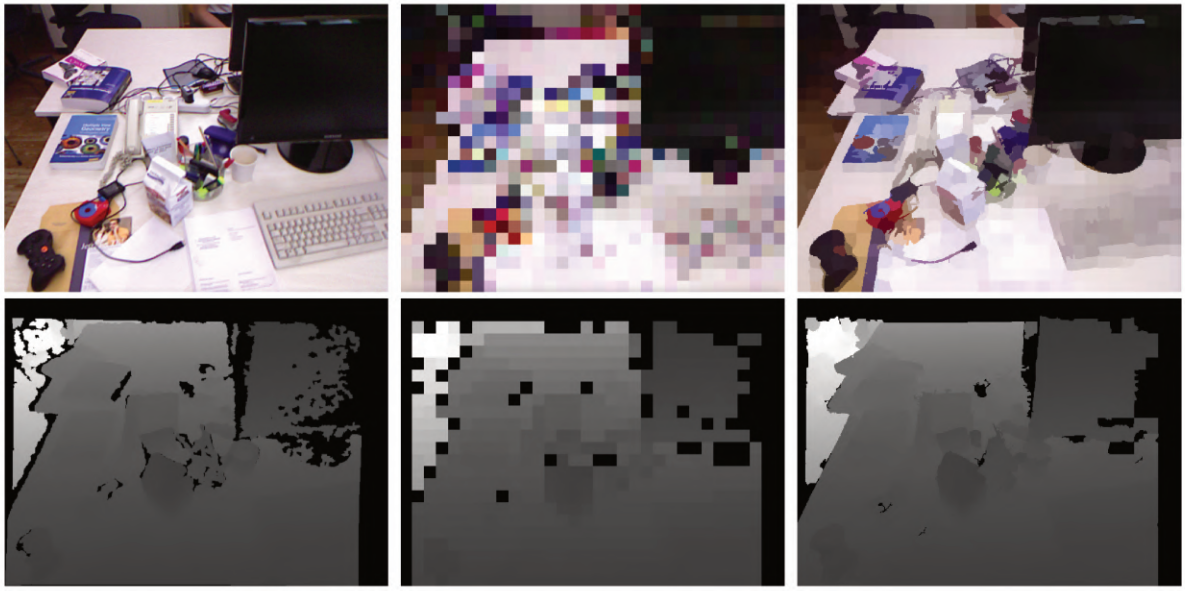

FIGURE 3.7 – Paire RGB-D originale à gauche, paire RGB-D obtenue par sous-échantillonnage avec un pas de 20 pixels au centre et paire RGB-D obtenue par segmentation en superpixels initialisés en blocs de 20×20 pixels à droite. Les superpixels préservent correctement la couleur et lissent même la profondeur. Les images sous-échantillonnées ne permettent pas de distinguer et de reconnaître correctement les éléments de la scène observée.

points suréchantillonnés et la surface la plus proche de la vérité terrain a été calculée. Les valeurs obtenues sont présentées dans le tableau 3.2. Elles montrent que même si la reconstruction (méthode *Supersurfel*) est faite à partir d'une forme de représentation plus grossière, en décomposant le milieu observé en un ensemble de patchs planaires, un niveau de précision proche de l'état de l'art (méthode *Surfel* de ElasticFusion) peut être atteint.

Méthode	**kt0**	**kt1**	**kt2**	**kt3**
Surfel	0.42	0.68	0.97	0.44
Supersurfel	0.60	0.76	1.09	0.54

TABLE 3.2 – Comparaison de la précision de la surface reconstruite (cm).

Le résultat de la reconstruction pour la séquence *kt1* est donné en figure 3.8, qui permet de visualiser l'erreur sur la surface reconstruite. De manière générale les zones reconstruites qui sont les plus éloignées de la caméra présentent une erreur plus importante car l'incertitude sur les mesures de profondeur augmente avec la distance. Les surfaces incurvées (plis de rideaux, coussins, abat-jour de lampe, fauteuil...) et les contours sont aussi moins fidèlement modélisés, puisque nous utilisons une représentation qui simplifie la scène en la découpant et en la décrivant localement par des patchs elliptiques. Les différents éléments reconstruits restent néanmoins reconnaissables et la modélisation est suffisamment précise pour être utilisée pour des tâches robotiques de localisation et de déplacements, comme nous le verrons dans les

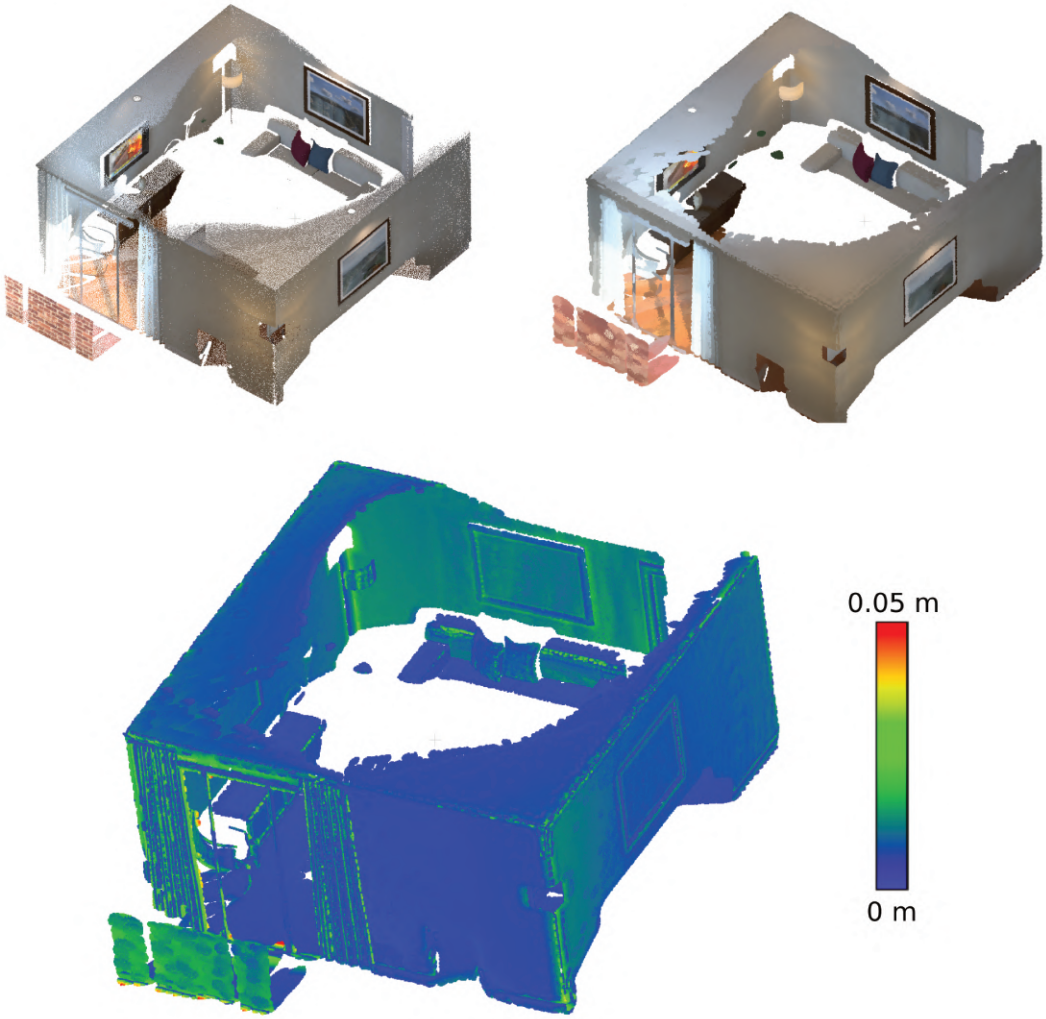

FIGURE 3.8 – Résultats de reconstruction sur *kt1*. En haut à gauche le modèle 3D produit par l'approche surfel ElasticFusion. En haut à droite le modèle reconstruit par notre méthode et en bas l'erreur mappée sur cette même reconstruction.

chapitres 4 et 5.

Des résultats qualitatifs supplémentaires de reconstructions denses obtenues par les méthodes basées sur les supersurfels et sur les surfels sont montrés sur la figure 3.9. On peut voir que les supersurfels ne permettent pas une reconstruction avec un niveau de détail aussi élevé que les surfels et ne sont pas particulièrement adaptés pour les petits objets, les éléments fortement incurvés ou encore les éléments fins comme les feuilles des plantes. En effet, ces éléments sont plus difficilement segmentables et représentables par des ensembles de surfaces elliptiques. La modélisation à base de supersurfels reste cependant convenable : elle préserve l'information importante et donne par ailleurs de très bons résultats pour les surfaces planes dont sont principalement composés les environnements d'intérieurs. De plus les supersurfels

permettent une reconstruction plus dense de l'environnement, moins parcimonieuse qu'avec ElasticFusion, à partir d'un nombre très inférieur de primitives 3D.

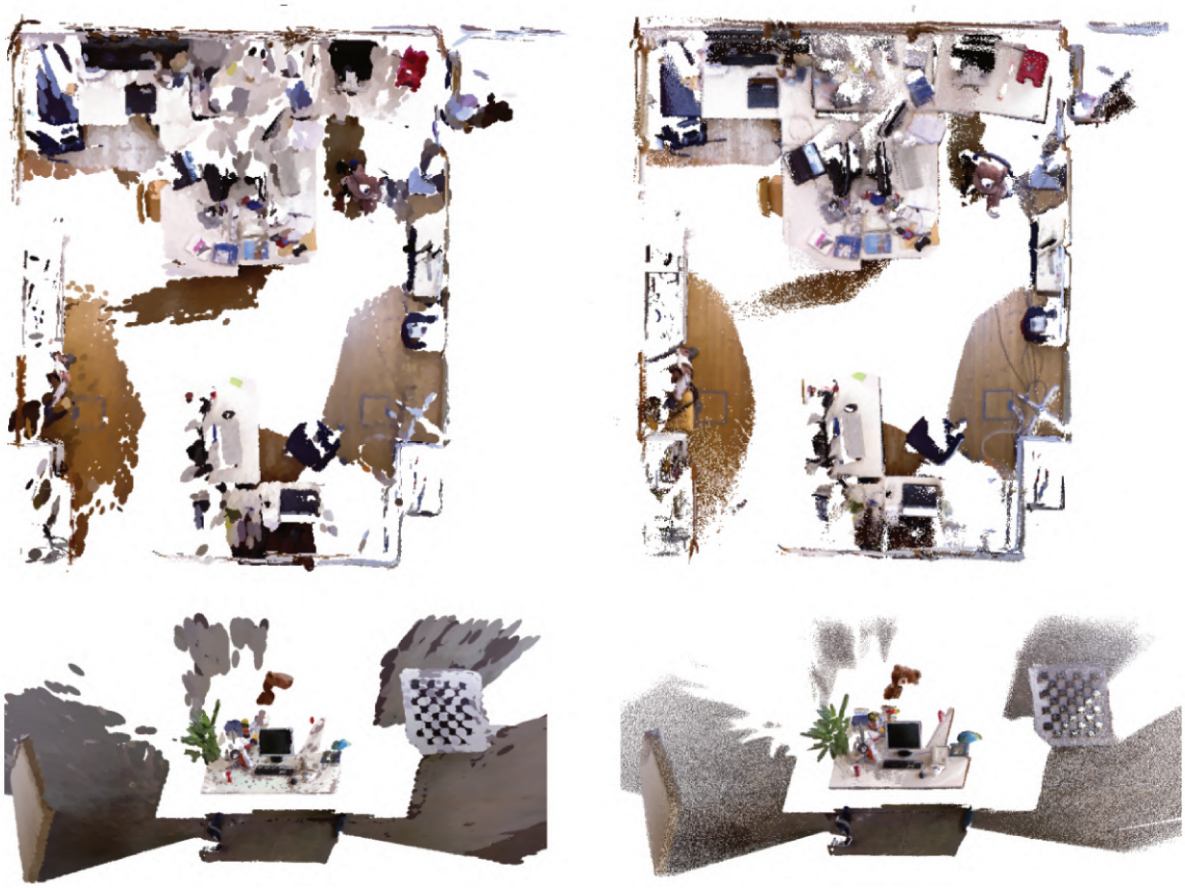

FIGURE 3.9 – Reconstruction 3D des séquences fr1_room (en haut) et fr2_rpy (en bas), réalisées avec notre méthode (à gauche) et ElasticFusion (à droite). Les petits éléments et les plantes sont moins précisément modélisés par les supersurfels, mais le rendu reste cohérent et consistent. Par ailleurs la reconstruction à base de supersurfels apparaît plus dense.

Nous avons aussi comparé la reconstruction supersurfel de la séquence *kt1* avec la surface obtenue à partir de surfels en diminuant la taille des images de la séquence par 20 selon la hauteur et la largeur. Les images redimensionnées sont calculées par sous-échantillonnage régulier avec un pas de 20 pixels. Les résultats obtenus sont affichées en figure 3.10. La précision moyenne de la surface est seulement de 2.65 cm avec les images réduites contre 0.76 cm en utilisant les supersurfels. Notre représentation est donc plus de 3 fois plus précise sur cet exemple. Il s'agit ici de souligner que même si la réduction de la taille des images RGB-D permet d'obtenir un niveau de compression des données 3D similaire à celui de l'approche supersurfel, elle produit une modélisation 3D bien plus inexacte. Cette méthode est trop imprécise pour être utilisable dans la pratique, d'autant plus que nous nous sommes placé dans un cas de localisation idéal en employant la vérité terrain de la trajectoire. En outre, on peut remarquer qu'il est compliqué, voir impossible, de pouvoir caractériser la scène 3D

reconstruite sous forme de surfels à partir des images sous-échantillonnées.

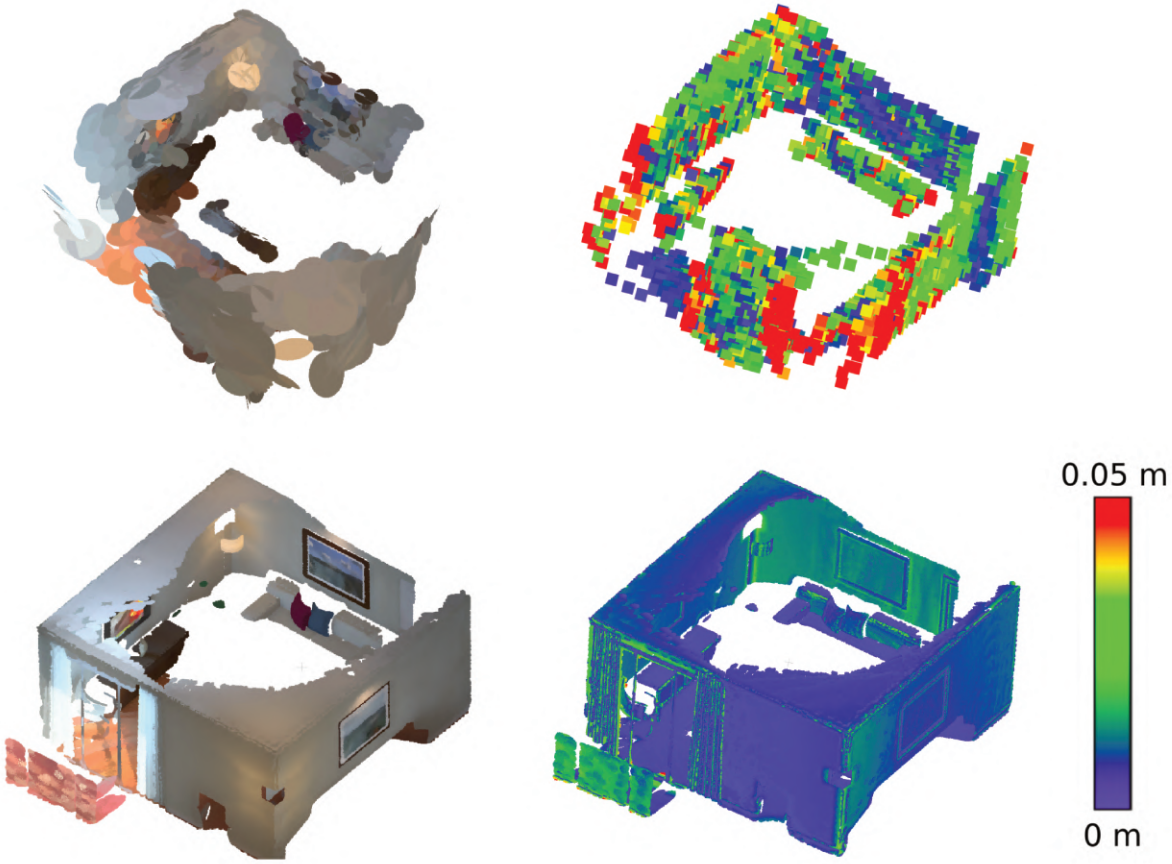

FIGURE 3.10 – Comparaison des reconstructions données sur *kt1* avec des surfels issus d'images dont la taille a été réduite par un facteur 20 (en haut) et des supersurfels générés à partir de superpixels d'environ 20×20 pixels (en bas). A gauche on retrouve les modèles 3D reconstruits et à droite l'erreur de précision représentée sur ces modèles.

3.4.3 Performances computationnelles

L'efficacité de la méthode de reconstruction 3D, utilisant notre nouvelle forme de représentation supersurfel de l'environnement, a été évaluée en termes d'empreinte mémoire et de vitesse d'exécution, sur les séquences vidéo réelles *fr1_xyz*, *fr1_room*, *fr2_rpy*, *fr3_office* issues du TUM. Dans *fr1_xyz* et *fr2_rpy*, la caméra observe des scènes plutôt petites, focalisée sur un bureau, tandis que dans *fr1_room* et *fr3_office* la caméra parcourt des environnements plus larges. Les tableaux 3.3 et 3.4 présentent le temps d'exécution de la méthode de reconstruction 3D et la taille mémoire maximale utilisée pour stocker le modèle pour chacune des vidéos. Le processus de reconstruction 3D basé sur les supersurfels est beaucoup plus ra-

pide (plus de deux fois pour la plupart des séquences) et léger (plus de vingt fois) que celui de ElasticFusion sur notre plateforme de tests. Ces résultats correspondent à nos attentes puisque ElasticFusion est une approche qui génère une primitive 3D pour chaque pixel d'une image RGB-D tandis que notre approche génère des données 3D à un niveau superpixel plus condensé. Ils marquent l'efficacité de la représentation utilisée par rapport aux surfels et aux représentations volumétriques, qui sont encore bien plus lourdes dans leur usage de la mémoire. En effet, il faut au moins 536 Mo pour stocker une grille régulière de $512 \times 512 \times 512$ voxels, permettant de reconstruire un volume d'environ 5 m^3 pour une résolution de 1 cm^3 par voxel, sans information de couleur ni de normale. L'utilisation d'une représentation basse résolution rend possible la reconstruction 3D dense d'environnement sur une plus large gamme de machines et d'étendre l'échelle de la scène reconstruite. La fréquence moyenne de fonctionnement du système de reconstruction 3D sur l'ensemble des séquences est inférieure à la fréquence d'acquisition de 30 Hz du capteur RGB-D, qui est donc sa seule limitation temporelle.

Méthode	**fr1_xyz**		**fr1_room**		**fr2_rpy**		**fr3_office**	
	Nb	Mo	Nb	Mo	Nb	Mo	Nb	Mo
Surfel	0.44×10^6	20.26	1.52×10^6	69.74	0.93×10^6	42.35	1.59×10^6	72.77
Supersurfel	4626	0.46	12574	1.25	7934	0.79	36015	3.57

Table 3.3 – Nombres (Nb) maximaux de primitives des modèles et empreintes mémoires maximales utilisée pour les stocker (Mo).

Méthode	**fr1_xyz**	**fr1_room**	**fr2_rpy**	**fr3_office**
Surfel	29.4 ± 2.9	38.3 ± 8.5	36.9 ± 4.3	46.2 ± 7.7
Supersurfel	17.2 ± 1.1	17.6 ± 1.5	15.1 ± 1.0	18.8 ± 1.2

Table 3.4 – Temps d'exécution moyens (ms).

Notre reconstruction génère des modèles compacts, contenant un nombre très faible de supersurfels (quelques milliers contre plus de 1 millions pour ElasticFusion), donc rapides à traiter. Cette propriété est intéressante car elle rend envisageable une utilisation quasi-immédiate de la carte reconstruite par des systèmes de planification de trajectoire ou d'évitement d'obstacle, sans avoir besoin d'appliquer des prétraitements comme du sous-échantillonnage uniforme par exemple. La forte compacité de la reconstruction 3D proposée est intéressante puisqu'elle permet un temps d'exécution relativement faible, constant et indépendant de l'évolution du nombre de supersurfels dans le modèle, même pour des scènes relativement grandes comme la pièce de *fr1_room*. Ce résultat est démontré par le graphique visible en figure 3.11 qui décrit l'évolution de la vitesse de l'algorithme de reconstruction 3D et du nombre de supersurfels dans le modèle durant la vidéo *fr1_room*. Un profilage des principales étapes de notre méthode de reconstruction 3D a aussi été réalisé avec le tableau 3.5. La partie la plus chronophage du système correspond à la segmentation des paires RGB-D en superpixels qui nécessite environ 80% du temps total, tandis que l'extraction de supersurfels de l'image courante à partir des

superpixels et la mise à jour du modèle varient autour de 10%.

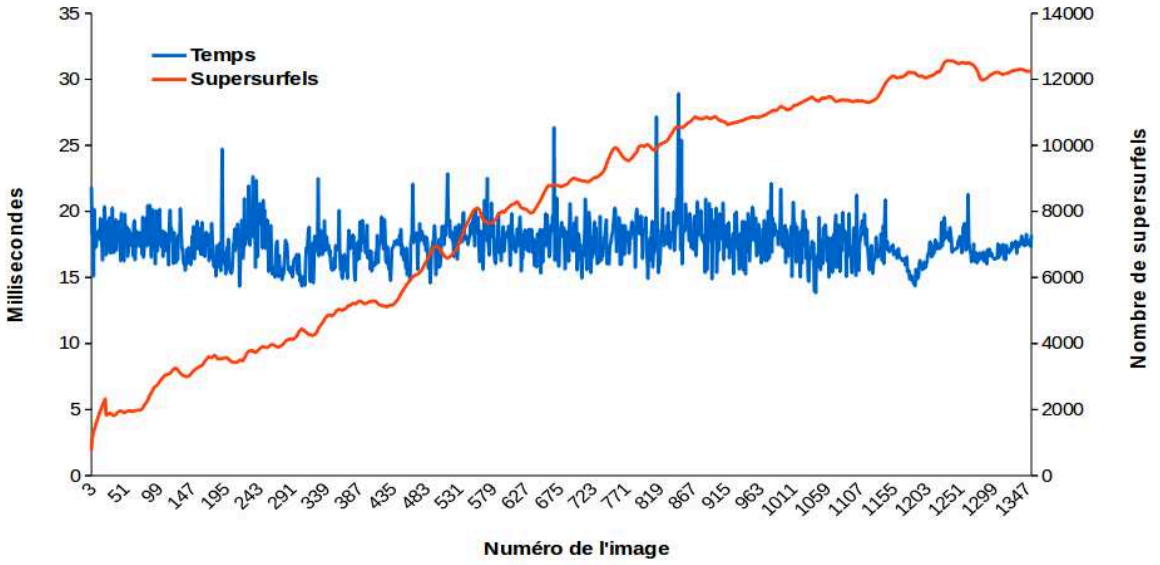

FIGURE 3.11 – Evolution du temps d'exécution et du nombre de supersurfels dans le modèle 3D reconstruit pour la séquence *fr1_room*. Le nombre de supersurfels contenus dans le modèle se stabilise sur la fin car la caméra retourne à son point de départ.

Processus	**fr1_xyz**	**fr1_room**	**fr2_rpy**	**fr3_office**
Segmentation	80.3%	77.8%	80.8%	73.3%
Extraction	13.3%	13.1%	11.9%	7.5%
Mise à jour	6.4%	9.1%	7.3%	19.2%

TABLE 3.5 – Profilage du temps requis pour les différentes étapes de la reconstruction 3D à partir des supersurfels.

3.5 Bilan

Une nouvelle forme de représentation basse résolution, pour la reconstruction 3D en environnement intérieur, a été introduite dans cette partie. Elle permet de modéliser une scène observée par une caméra RGB-D sous la forme d'un ensemble de patchs elliptiques appelés supersurfels, qui encodent la géométrie et la couleur locale d'une surface. Les supersurfels sont générés à partir de la segmentation en superpixels d'images RGB-D. L'utilisation de superpixels réduit fortement la charge mémoire nécessaire à la reconstruction 3D par rapport aux formes de représentation plus traditionnelles. De plus, ils préservent autant que possible les contours et les discontinuités géométriques ce qui garantit la fiabilité de notre représentation approximative mais compacte et légère de l'environnement. La génération de données 3D à

un niveau superpixel permet aussi un meilleur filtrage de la profondeur, éliminant les valeurs aberrantes et le bruit.

Un système de reconstruction 3D basé sur cette représentation a été proposé. Il effectue une cartographie rapide, grossière mais pertinente, d'une scène observée, en utilisant peu de mémoire. Le modèle 3D généré est léger et simple à mettre à jour à partir de données RGB-D reçues en temps réel. Les tests réalisés ont permis de valider le fonctionnement en ligne de la méthode et la faible capacité mémoire qu'elle nécessite, rendant son application envisageable sur des systèmes mobiles de puissance limitée, équipés de GPU non professionnels et bon marché. L'approche développée est intéressante pour des applications ne nécessitant pas un haut niveau de résolution dans la surface reconstruite, mais plutôt d'une cartographie 3D dense, moins précise mais moins coûteuse, permettant de conserver suffisamment de ressources de calcul pour réaliser d'autres tâches.

Chapitre 4

SLAM RGB-D dense, rapide et léger en milieu dynamique

Sommaire

Ce chapitre aborde le problème de localisation et cartographie 3D dense à partir de caméra RGB-D et présente nos travaux sur la conception d'un SLAM RGB-D temps réel robuste aux environnements dynamiques, appelé SupersurfelFusion. Nous nous focalisons sur les approches capables de fournir une reconstruction dense de l'environnement, car même si elles sont plus coûteuses que les systèmes épars et requièrent l'utilisation de GPU, elles produisent des modèles 3D utilisables dans un cadre plus large d'applications robotiques que la seule localisation.

4.1 Introduction

La recherche en SLAM RGB-D dense a réalisé des progrès importants et de nombreuses solutions capables de résultats impressionnants ont été proposées au cours des dix dernières années. Cependant la majorité des approches développées partagent des restrictions communes. Tout d'abord, nous avons vu précédemment (au chapitre 3) qu'elles se basaient souvent sur des formes de représentation très coûteuses pour produire des modèles 3D complexes, ce qui limite la taille maximale de la scène pouvant être reconstruite. Elles se focalisent essentiellement sur la précision de la modélisation alors que de nombreuses applications ne nécessitent pas un tel niveau de détail et bénéficieraient plutôt de cartes 3D moins précises mais plus adaptées et pratiques à utiliser. Les différentes recherches autour des capteurs RGB-D ont souvent considéré le SLAM comme une finalité en soi, plutôt que comme un bloc de perception visant à être intégré sur un système robotique, afin de permettre l'exécution de tâches de plus haut niveau.

Par ailleurs, les méthodes de SLAM RGB-D dense s'appuient surtout sur des stratégies d'odométrie visuelle directes ([KSC13a] ; [JGJ15] ; [GGMCG16]) pour se localiser. L'estimation du mouvement entre deux paires d'images RGB-D consécutives se fait par minimisation d'une erreur photométrique et/ou géométrique. Le processus d'optimisation numérique nécessite que la solution initiale soit suffisamment proche du minimum global pour converger correctement. Ces stratégies ont donc tendance à échouer pour des mouvements de caméra importants entre acquisitions successives et lorsque la surface de recouvrement entre les images est faible. De plus, les approches d'optimisation globale et de fermeture de boucle mises en place sont la plupart du temps trop lourdes pour corriger en temps réel les dérives accumulées dans la trajectoire de la caméra et dans le modèle 3D reconstruit. Elles sont bien souvent appliquées a posteriori afin d'obtenir une carte cohérente et fiable de l'environnement, qui ne peut donc pas être utilisée directement lors de l'acquisition, pour de la navigation et de l'exploration autonome.

Enfin, les algorithmes de SLAM RGB-D dense traditionnels considèrent que l'environnement est statique. Ils ne gèrent pas les éléments dynamiques qui peuvent se déplacer indépendamment de la caméra, corrompre la localisation et introduire des anomalies durant la construction de la carte 3D. Ceci empêche leur déploiement dans un grand nombre de scénarios réels, qui comprennent bien souvent une ou plusieurs personnes en mouvement. Des solutions robustes aux milieux dynamiques ont émergés plus récemment. Néanmoins elles reposent majoritairement sur l'hypothèse, parfois fausse, que la partie statique de l'environnement visible par la caméra est prépondérante dans l'image, ou sur l'utilisation de réseaux de neurones trop coûteux pour fonctionner en temps réel.

Ce chapitre introduit la nouvelle méthode de SLAM RGB-D dense temps réel que nous avons développée. Appelée *SupersurfelFusion*, elle s'articule autour de la représentation supersurfel décrite au chapitre 3 et a pour but de surmonter les contraintes partagées par les architectures de SLAM RGB-D dense classiques. Plus précisément, notre approche aborde les problèmes des environnements dynamiques, des mouvements inter-images importants, de la reconstruction et de la cohérence de scènes larges. Plutôt que la précision de la surface

reconstruite, nous favorisons l'efficacité en terme de mémoire et de vitesse d'exécution, qui constituent des critères primordiaux pour l'application d'algorithmes de SLAM dense sur des robots mobiles. Notre solution est temps réel, aucune étape n'est réalisée "hors-ligne", et fonctionne complètement en boucle fermée. En plus d'intégrer la représentation supersurfel pour une reconstruction rapide et légère, notre système effectue un suivi robuste de la pose de la caméra en combinant une stratégie d'odométrie visuelle éparse avec un procédé de recalage dense entre l'image courante et le modèle 3D généré. L'odométrie visuelle basée sur les points d'intérêts fonctionne correctement dans des scènes texturées plates. Elle est aussi plus robuste aux mouvements inter-images brusques ou larges que les méthodes directes, qui nécessitent des images RGB-D riches en terme de géométrie et proche les unes des autres. La fiabilité de la reconstruction 3D est assurée sur le long terme grâce à une fermeture de boucle basée sur une approche de déformation non-rigide de l'espace. La technique de fermeture de boucle utilisée est similaire à celle proposée par ElasticFusion [Whe+15]. Une autre contribution de notre SLAM est l'ajout d'une nouvelle stratégie complète de détection dense d'objets dynamiques, alliant apprentissage profond, compensation du mouvement de la caméra, flot optique et différence de profondeur.

Nous ferons tout d'abord un bref état de l'art des systèmes de SLAM RGB-D dense. Ensuite nous présenterons notre solution SupersurfelFusion et expliciterons ses différents modules. Enfin, le système implémenté sera testé et comparé à d'autres approches connues.

4.2 Systèmes de SLAM RGB-D dense

Cet état de l'art est centré sur la structure des différents algorithmes de SLAM avec caméra RGB-D produisant une modélisation dense de l'environnement, en milieux statiques puis dynamiques. La majorité de ces approches sont basées sur des méthodes d'odométrie visuelle dense et directe, employant l'ensemble des pixels des images pour se localiser. Les formes de représentation (volumétriques ou points/surfels) employées pour la reconstruction et leur utilisation ne sont pas abordées en détail puisqu'elles ont fait l'objet du chapitre 3 précédent.

4.2.1 Approches traditionnelles

KinectFusion [Iza+11], sorti en 2011, est l'un des premiers systèmes de SLAM RGB-D capable de réaliser la reconstruction 3D dense et en ligne d'une scène observée, gâce au parallélisme massif des GPU. La méthode intègre et fusionne les données de profondeur dans un unique modèle volumétrique représenté sous forme de TSDF, qui est aussi utilisé pour calculer la pose de la caméra. L'image de profondeur en entrée est pré-traitée avec un filtre bilatéral pour réduire le bruit sur les mesures, puis les points 3D et les normales associées à l'image de profondeur filtrée sont calculés. Les données 3D extraites sont ensuite alignées avec le modèle à partir de l'algorithme ICP (Iterative Closest Point) [CM91], en minimisant de manière itérative la distance point-plan entre les nouveaux points 3D et la surface du modèle,

obtenue en appliquant un algorithme de raycasting sur la TSDF. Le recalage global des données entre la vue courante et le modèle (alignement "frame-to-model") contribue à réduire fortement l'accumulation d'erreurs dans l'estimation de la trajectoire de la caméra, par rapport à un recalage entre images consécutives (alignement "frame-to-frame"), comme le souligne la figure 4.1. Plutôt que de se servir de l'ICP pour l'alignement des données, Bylow et al. [Byl+13] ont présenté une stratégie qui estime la pose de la caméra en minimisant une erreur directement sur la TSDF, sans avoir besoin d'extraire les données 3D par raycasting. En plus d'utiliser une représentation volumétrique coûteuse, réduisant la taille maximale possible de la scène à reconstruire, ces méthodes n'utilisent que les données de profondeur pour la localisation et sont amenées à échouer lorsque la scène observée manque d'informations géométriques. Elles ne disposent par ailleurs pas de mécanisme de fermeture de boucle, ce qui rend impossible la correction de la dérive accumulée lors du suivi progressif de la caméra. Pour améliorer la robustesse de la localisation, [SO+14] exploite à la fois les données de couleur (ou d'intensité) et de profondeur lors de l'estimation de la pose de la caméra. Kintinuous [Whe+12], par Whelan et al., est une extension de KinectFusion, capable d'opérer dans de plus grands environnements en transférant les zones reconstruites éloignées de la caméra du GPU au CPU grâce à un buffer circulaire et en déplaçant le volume de la TSDF avec la caméra. L'approche intègre aussi une fermeture de boucle, similaire à celle de InfiniTAM [Pri+17], basée sur la reconnaissance de place et l'optimisation d'un graphe de poses [KA08]. A chaque pose du graphe est associée un sous-volume de la reconstruction. Lorsqu'une fermeture de boucle est détectée, les sous-volumes sont décalés de manière rigide en fonction des poses optimisées.

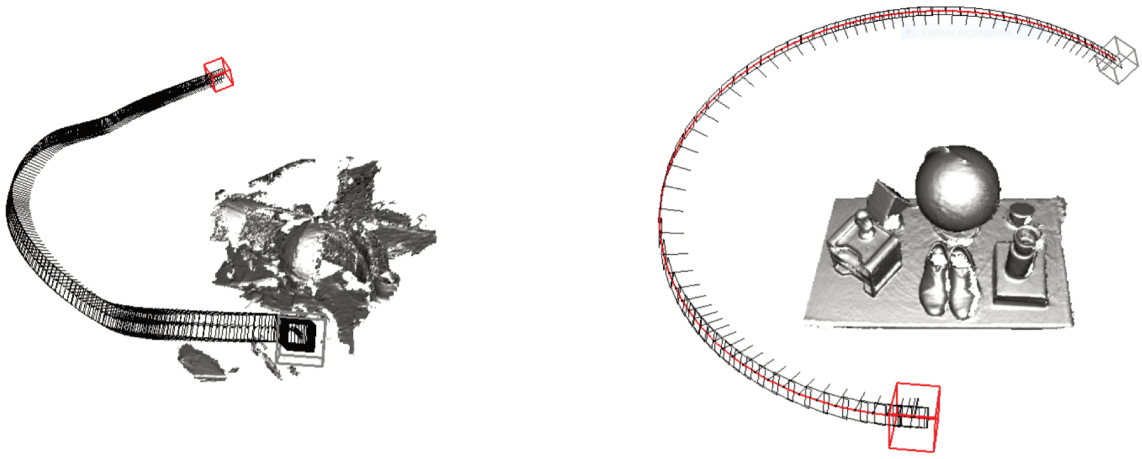

Figure 4.1 – Résultats obtenus par KinectFusion [Iza+11] sur un même environnement en employant une stratégie d'odométrie "frame-to-frame" (à gauche) et une stratégie "frame-to-model" (à droite). La trajectoire de la caméra est représentée autour de la reconstruction par ses différentes poses.

Le système [Kel+13] présente une architecture similaire à celle de KinectFusion, mais remplace la représentation volumétrique par des surfels pour gagner en flexibilité et en légèreté. Whelan et al. présentent ElasticFusion [Whe+15], un SLAM RGB-D temps réel, aussi basée sur la représentation surfel, capable de fournir une carte cohérente à un niveau global sans

graphe de poses. Il s'agit d'une approche centrée sur la carte, et non la trajectoire, qui ajuste le modèle reconstruit par application de déformations non-rigides à travers deux types de fermetures de boucle : locales et globales. Les fermetures de boucle locales sont réalisées le plus fréquemment possible entre les surfels actifs (récents) et les surfels inactifs (plus anciens) de la carte, pour rester proche de son mode de distribution. Les déformations ne sont ici appliquées que sur les surfels compris dans le champ de vision de la caméra. Les fermetures de boucle globales, détectées à l'aide d'un algorithme de reconnaissance visuelle de place, permettent la correction de dérives plus importantes. Dans le cas d'une fermeture de boucle globale, les déformations sont appliquées sur tous les surfels du modèle pour garantir la cohérence de la reconstruction à un niveau plus général. Cette stratégie centrée sur la carte reconstruite, plutôt que sur la trajectoire du capteur, permet à ElasticFusion de fonctionner complètement en boucle fermée : les corrections sont appliquées immédiatement, en direct, sans retard dû à l'optimisation coûteuse d'un graphe de poses et sans post-traitement. Néanmoins l'implémentation courante est limitée à des environnements de l'échelle d'une pièce intérieure car la complexité du système augmente de manière conséquente avec le nombre de surfels dans la carte. Xingyin et al. [Xin+18] améliore l'extensibilité de ElasticFusion en mettant en place deux cartes de surfels : une carte locale servant pour la plupart des calculs et une carte globale. La carte locale, dans laquelle sont intégrées les images RGB-D, est de taille limitée pour maintenir un temps d'exécution stable. Les surfels de la carte locale trop distants de la caméra sont transférés dans la carte globale.

Si les SLAM cités ci-dessus se localisent tous grâce à des techniques d'odométrie visuelle dense et directe, il en existe aussi qui s'appuient sur des techniques de localisation éparse et indirecte basées sur l'extraction et la mise en commun de points d'intérêt. Henry et al. [Hen+12] et Endres et al. [End+14] combinent les points d'intérêt SIFT [Low04] avec l'algorithme ICP pour une estimation plus robuste du mouvement de la caméra. L'appariement de points caractéristiques avec des descripteurs SIFT invariants aux orientations des images, à leurs résolutions, et peu sensibles à leurs expositions ainsi qu'aux changements de points de vue, contribue à garantir la bonne convergence de l'ICP. Ils optimisent un graphe de poses à la fin de l'enregistrement des données pour générer une carte globalement cohérente de l'environnement exploré, à l'aide de la trajectoire optimisée. La carte est modélisée sous forme de surfels pour [Hen+12] et d'OctoMap [Hor+13] pour [End+14], à partir d'un sous-ensemble d'images clés. Dans ces deux approches aucune optimisation globale ou correction de la carte par fermeture de boucle n'est réalisée en ligne, c'est-à-dire directement lors de l'acquisition des images. Dai et al. ont proposé BundleFusion [Dai+17] en 2017, un système de pointe en terme de précision, capable de générer et de maintenir en direct une reconstruction TSDF large avec une cohérence globale. Le haut niveau de précision repose sur l'application continue d'une stratégie hiérarchique d'optimisation globale de la trajectoire, à partir de correspondances de points d'intérêt SIFT puis de données pixels denses (intensités et profondeurs). BundleFusion propose aussi un procédé pour dé-intégrer et ré-intégrer les données 3D de la TSDF à la volée, de manière à ce que l'optimisation globale des poses de la caméra soit reflétée immédiatement dans la reconstruction. Ce procédé requiert cependant le stockage en mémoire de toutes les images précédemment acquises par le capteur, ce qui induit un coût mémoire considérable. L'usage pratique du système est très limité car il nécessite deux GPU hauts de gamme pour fonctionner en temps réel.

4.2.2 SLAM dynamiques

Les approches précédentes ne sont pas conçues pour gérer les éléments dynamiques, qui peuvent altérer les performances de la localisation et corrompre la carte 3D reconstruite. Elles dépendent du postulat que la scène observée est statique et sont incapables de distinguer les déplacements d'objets en mouvement de ceux de la caméra. Les caractéristiques visuelles associées aux objets dynamiques peuvent alors être prises en compte lors de l'estimation de la pose de la caméra et corrompre la localisation. On ne peut donc pas les utiliser dans la plupart des situations du monde réel qui impliquent la présence de personnes, d'animaux ou encore de véhicules. Les systèmes de SLAM RGB-D dense élaborés pour fonctionner en milieux dynamiques distinguent les objets en mouvement à partir de contraintes géométriques traditionnelles, ou plus récemment à l'aide de méthodes d'apprentissage profond. Ils peuvent être divisées en deux catégories.

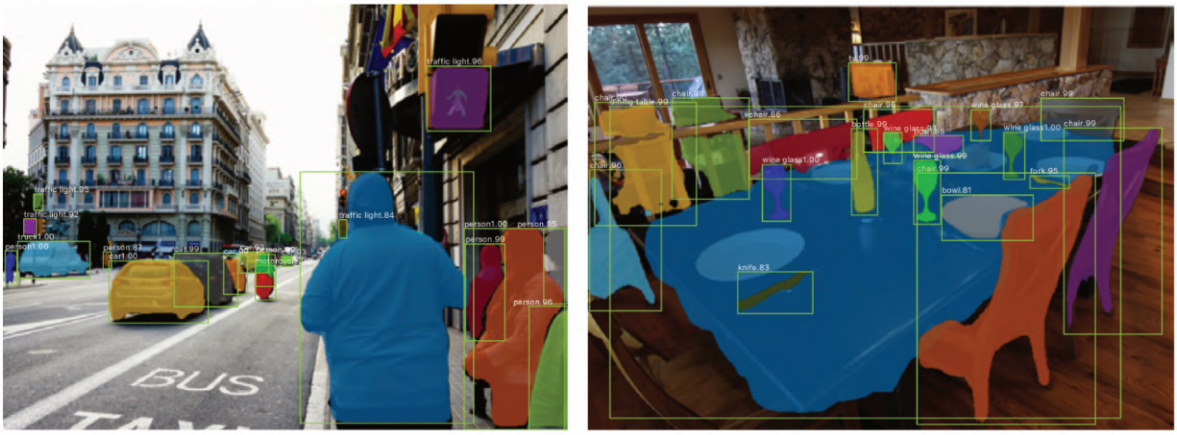

FIGURE 4.2 – Exemple de détection d'objets et segmentation sémantique réalisée par Mask R-CNN [HDG17].

La première catégorie de systèmes détecte les objets en mouvement et les élimine pour reconstruire une carte dense de la partie statique de l'environnement. StaticFusion [Sco+18] regroupe les points 3D extraits des images de profondeur en clusters avec l'algorithme de K-means [Mac67], puis applique une formulation pour l'estimation conjointe du déplacement de la caméra et de la segmentation "statique/dynamique" des clusters. Cette méthode nécessite par contre un temps d'initialisation pour fonctionner correctement. ReFusion [Pal+19] distingue les éléments dynamiques des éléments statiques en analysant les valeurs des erreurs obtenues pour chaque pixel lors d'une première étape d'estimation dense et directe de la pose de la caméra. Les pixels avec des valeurs résiduelles élevées sont ceux qui ne suivent pas le même mouvement que la caméra, ils sont supprimés et la pose est optimisée à partir des données statiques seules. ZHANG et al. [Zha+20] améliorent l'implémentation de StaticFusion en employant du flot optique pour une meilleure détection des éléments dynamiques. Le principal problème de ces solutions est qu'elles montrent un taux d'échec élevé lorsque la proportion d'éléments dynamiques dans les images est supérieure à celle d'éléments statiques. De plus elles sont incapables de détecter des éléments actifs immobiles et fonctionnent mal pour des mouvements lents. Pour plus de robustesse, WANG et al. [Wan+19] incorporent à leur solution

le réseau de neurones spécialisé dans la détection d'objets YOLOv3, réputé pour sa vitesse et sa légèreté. Ils suppriment tous les pixels associés à des éléments actifs connus par le réseau. Il est donc possible que des éléments inconnus puissent perturber la localisation et la reconstruction. SGC-VSLAM [Yan+20] associe YOLOv3 à une méthode basée sur des contraintes géométriques multi-vues pour extraire les points d'intérêt qui sont sur des objets en mouvement connus et inconnus par le réseau. L'algorithme ne fournit cependant pas de classification "statique/dynamique" dense de l'environnement.

Le deuxième type de méthodes suit chaque objet dynamique détecté, en plus de reconstruire la scène statique. Co-Fusion [RA17] effectue un ajustement multiple de modèles de mouvement pour identifier, reconstruire et suivre différents objets rigides indépendamment. Mais sa précision décroît fortement lorsque les objets se déplacent trop rapidement et la méthode ne fonctionne pas sur les éléments non rigides. De plus le temps de calcul augmente de manière importante pour chaque objet supplémentaire suivi. [Won+21] reprend l'idée de Co-Fusion en y ajoutant un algorithme d'apprentissage profond pour le masquage d'humains, afin d'éviter les interférences d'éléments non rigides. [Li+20] combine des méthodes de suivis rigide et non rigide pour pouvoir reconstruire tout type d'éléments. MaskFusion [RBA18] et MID-Fusion [Xu+19] utilisent Mask R-CNN [HDG17], un modèle deep-learning de pointe pour la segmentation sémantique d'objets dans les images (cf. figure 4.2), effectuant ainsi une reconstruction géométrique et sémantique multi-objets. Ces approches souffrent cependant d'un temps de calcul élevé dû à l'utilisation coûteuse du réseau de segmentation sémantique, précis mais lourd.

4.3 SupersurfelFusion

Nous proposons SupersurfelFusion : un SLAM RGB-D dense et robuste aux milieux dynamiques, articulé autour de la représentation supersurfel que nous avons développée (cf. chapitre 3). L'approche fonctionne complètement en boucle fermée. Elle est centrée sur la carte à la manière de [Whe+15] : aucun processus d'optimisation global n'est exécuté en arrière-plan, afin de refléter et d'incorporer les changements et les corrections dans la reconstruction 3D le plus vite possible. Le système prend en entrée le flux vidéo recalé d'une caméra RGB-D pour reconstruire incrémentalement un modèle 3D cohérent $\mathcal{M}$ de la partie statique de l'environnement observé, tout en opérant un suivi robuste de la pose $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} \in SE(3)$ de la caméra, définie par son référentiel $\mathfrak{c}$, dans le référentiel du monde $\mathfrak{w}$. A l'initialisation les repères $\mathfrak{c}$ de la caméra et $\mathfrak{w}$ du monde sont confondus.

4.3.1 Architecture globale

Deux cartes 3D sont entretenues par SupersurfelFusion dans le repère monde $\mathfrak{w}$: une carte locale éparse $\mathcal{L}$ de points d'intérêt et la carte globale dense $\mathcal{M}$ de supersurfels. La carte locale $\mathcal{L}$ est une carte secondaire, de taille limitée, qui recueille les rétroprojections de points clés ORB [Rub+11] statiques détectés dans les images précédentes et leurs descripteurs associés.

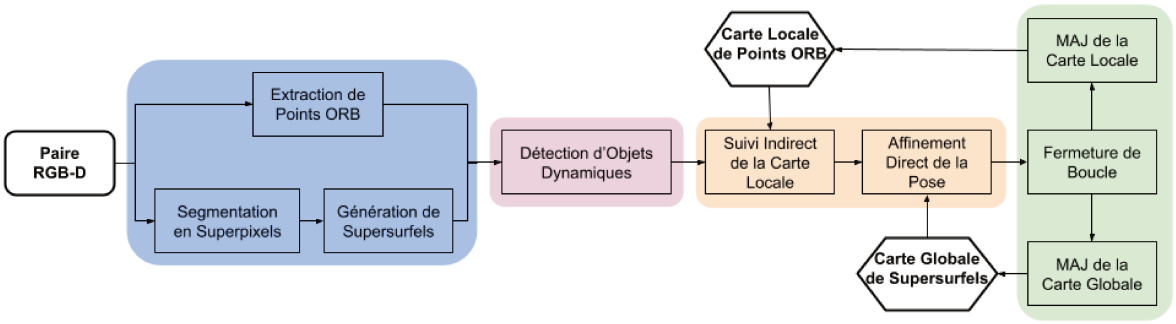

FIGURE 4.3 – L'architecture du SLAM présenté est composée de quatre modules : l'extraction de primitives (bleu), la détection d'objets dynamiques (violet), la localisation (jaune) et la cartographie (vert).

Elle n'est utilisée que pour améliorer le processus de localisation. La carte principale est la carte globale $\mathcal{M}$. Elle modélise la surface de l'environnement sous forme de supersurfels. La représentation supersurfel (définie en section 3.3) permet de produire une reconstruction basse-résolution fiable, résistante aux bruits, et de réduire la complexité algorithmique du SLAM ainsi que son empreinte mémoire. On rappelle que chaque supersurfel $\mathcal{M}_k$ de la carte globale $\mathcal{M} = \{\mathcal{M}_k\}$ est décrit par les attributs suivant : sa position $\mathbf{p}_k \in \mathbb{R}^3$, sa couleur $\mathbf{c}_k \in \mathbb{R}^3$, son orientation $\mathbf{O}_k \in SO(3)$, son élongation latérale $l_k \in \mathbb{R}$ et longitudinale $L_k \in \mathbb{R}$, une matrice de covariance décrivant sa forme $\mathbf{\Sigma}_k \in \mathscr{M}_3(\mathbb{R})$, sa valeur de confiance $w_k \in \mathbb{R}_+$ et enfin deux horodatages $h_{0,k}, h_k \in \mathbb{R}_+$ qui indiquent sa première apparition et sa dernière apparition. On désigne en plus la normale du supersurfel par $\mathbf{n}_k$, correspondant à la troisième colonne de la matrice d'orientation.

L'architecture de SupersurfelFusion est présentée en figure 4.3. Elle est composée de quatre modules exécutés à chaque acquisition d'une nouvelle paire RGB-D $\{C, D\}$:

Extraction de primitives : l'ensemble $\mathcal{F} = \{\mathcal{F}_i\}$ de supersurfels associés à la vue courante est généré et un nombre limité de points d'intérêt ORB est détecté dans l'image d'intensité $I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$, calculée à partir de l'image couleur C.

Détection d'objets dynamiques : les éléments dynamiques visibles par la caméra sont détectés au niveau superpixel et supprimés pour ne pas dégrader la localisation et la reconstruction. Notre stratégie combine un réseau de neurones léger avec une approche plus traditionnelle basée sur la compensation du mouvement entre images successives. Cette combinaison lui permet de fonctionner de manière robuste même dans les cas où les objets mobiles sont prédominants dans les images.

Localisation : ce module calcule la pose de la caméra $\mathbf{T}_c^w$ en deux étapes successives : une étape d'initialisation, avec le suivi de la carte locale $\mathcal{L}$ par odométrie visuelle indirecte à l'aide des points ORB, puis une étape d'affinement dense de la pose estimée basée sur un algorithme de type ICP.

Cartographie : les deux cartes $\mathcal{M}$ et $\mathcal{L}$ sont mises à jour. Une stratégie de fermeture

de boucle fondée sur la reconnaissance visuelle de lieu et l'application de déformations non rigides est employée pour garantir la consistance générale de la reconstruction supersurfels sur le long terme.

4.3.2 Extraction de primitives

Des points d'intérêt ORB [Rub+11] (environ 1000) et leurs descripteurs binaires associés sont détectés dans l'image d'intensité I de la vue courante. Le niveau de gris d'un pixel $\mathbf{u}$ est calculé à partir de l'image de couleur C comme : $I(\mathbf{u}) = 0.299R(\mathbf{u}) + 0.587V(\mathbf{u}) + 0.114B(\mathbf{u})$, où $[R, V, B]$ sont les canaux rouge, vert et bleu de C. Nous avons choisi d'utiliser les points ORB car ils sont très rapides à détecter, à apparier et possèdent de bonnes propriétés d'invariance aux changements de point de vue. Il peuvent être mis en correspondance même pour des images assez éloignées. Dans le but d'assurer une répartition assez uniforme des points clés ORB extraits dans l'image courante, nous suivons la stratégie de ORB-SLAM2 [MAT17]. L'image est d'abord divisée en une grille régulière, puis on essaye d'extraire un nombre minimum de points clés dans chaque cellule de la grille, en ajustant la valeur du seuil de détection. L'image de profondeur D sert de masque pour rejeter les points d'intérêt ayant une profondeur invalide. Un exemple de points ORB détectés dans une image est présenté sur la figure 4.4.

FIGURE 4.4 – Points d'intérêt ORB.

En parallèle de l'extraction des points ORB, la nouvelle paire RGB-D $\{C, D\}$ acquise est segmentée en superpixels. Un supersurfel $\mathcal{F}_i$ est généré pour chaque superpixel afin de former l'ensemble $\mathcal{F} = \{\mathcal{F}_i\}$ de supersurfels de la vue courante. Les coordonnées des supersurfels sont exprimées dans le repère de la caméra $\mathfrak{c}$. Les détails de la segmentation en superpixels et du calcul des supersurfels ont été présentés dans le chapitre précédent, en section 3.3.

4.3.3 Détection d'éléments dynamiques

La détection d'objets mobiles se fait à partir de l'image courante et de l'image précédente. Elle contient trois étapes principales :

1. une étape de détection préalable d'éléments potentiellement actifs (humain, animal...) dans l'image courante basée sur un modèle d'apprentissage profond léger. Elle permet une meilleure robustesse dans des scénarios dynamiques complexes.
2. une deuxième étape de compensation du mouvement ayant pour but de calculer une prédiction de la vue courante à partir de la vue précédente, sous l'hypothèse d'une scène statique.
3. et une dernière étape de classification statique/dynamique au niveau superpixel, obtenue en comparant l'image courante observée avec l'image prédite. La comparaison repose sur l'analyse de vecteurs de flot optique et de différences de mesures de profondeur.

Les supersurfels et les points d'intérêt ORB de l'image courante, associés à des superpixels classifiés dynamiques, sont rejetés du reste du système de SLAM. La figure 4.5 schématise le principe de fonctionnement de la méthode proposée.

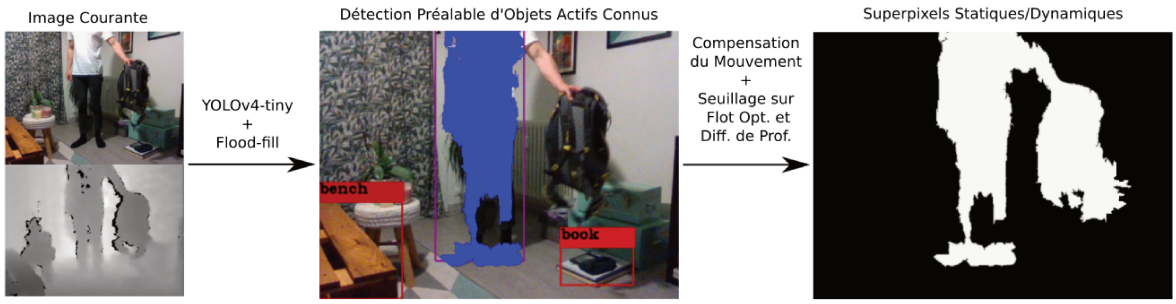

FIGURE 4.5 – Aperçu de la détection d'éléments dynamiques. Les éléments actifs connus par YOLOv4 [BWL20] (boîte englobante violette) sont détectés et segmentés plus précisément par algorithme de "flood-fill" (masque bleu). Une compensation de mouvement robuste est ensuite effectuée sur le reste de l'image. Finalement, un seuillage adaptatif est appliqué sur les modules de vecteurs de flot optique ainsi que sur les différences de profondeur compensée, pour filtrer les éléments mobiles restants (ici le bras gauche tenant le sac à dos).

4.3.3.1 Détection préalable d'objets actifs

Nous nous appuyons sur YOLOv4-tiny, la version compacte de YOLO-v4 [BWL20], pour préfiltrer les objets dans l'image de couleur actuelle C qui sont vraisemblablement dynamiques, tels que les humains et les chiens par exemple. YOLOv4 est un réseau de neurones convolutifs (CNN) rapide et précis, capable de performances temps réel sur plateforme embarquée. Le réseau a été pré-entraîné pour détecter et identifier une vingtaine de types d'objets. Les détections résultantes sont fournies sous la forme de boîtes englobantes. Si la classe d'une détection correspond à un objet actif, c'est-à-dire capable de se déplacer par lui même, alors il

faut rejeter les points d'intérêt et les supersurfels qui lui sont liés. Il est cependant problématique de rejeter la totalité des données contenues dans une boîte englobante. En effet, celle-ci n'entoure que très grossièrement l'objet, donc beaucoup des pixels qu'elle contient peuvent appartenir à des parties statiques de la scène. Pour éviter de supprimer des données statiques, nous appliquons un algorithme de segmentation par remplissage par diffusion ("flood-fill") classique sur les mesures de profondeur à l'intérieur de chacune des boîtes englobantes, dans le but d'extraire un masque correspondant mieux à la forme réelle de l'objet. La segmentation est pratiquée au niveau superpixel afin d'accélérer son exécution. Nous utilisons une implémentation itérative au moyen d'une pile de voisins, détaillée dans l'algorithme 1, pour séparer les superpixels appartenant à l'objet détecté de ceux appartenant au fond de l'image. Pour chaque superpixel labélisé, on vérifie tous ses voisins et on place ceux qui doivent être identifiés comme appartenant à l'objet dans la pile. Le calcul s'arrête quand la pile est vide. Le superpixel avec la profondeur la plus faible parmi les superpixels situés dans le voisinage du centre de la boîte englobante, est sélectionné comme point de départ pour la diffusion. C'est un moyen simple et rapide de garantir dans la plupart des cas que le germe initial appartienne bien à l'objet détecté et non à un élément d'arrière-plan.

Algorithme 1 FLOOD-FILL

Entrées: superpixels $\mathcal{S}$, germe initiale $\mathbf{s}_{init}$, boîte englobante $\mathbf{B}$, pile $\mathbf{P}$
Sorties: classification $\mathcal{C}$ attribuant une classe objet/fond à chaque superpixel

$\mathcal{C} \leftarrow$ initialiser la segmentation en assignant la classe fond à tous les superpixels, $\mathbf{P} \leftarrow \emptyset$
PUSH($\mathbf{P}, \mathbf{s}_{init}$)
tant que P est non vide **faire**
  $\mathbf{s} \leftarrow$ POP($\mathbf{P}$)
  mettre à jour $\mathcal{C}$ en classifiant $\mathbf{s}$ comme objet
  pour tout superpixel $\mathbf{v}$ voisin de $\mathbf{s}$ **faire**
    si $\mathbf{v} \subset \mathbf{B}$ **et** $\mathbf{v}$ non visité **et** PROFONDEUR($\mathbf{v}$) $\approx$ PROFONDEUR($\mathbf{s}$) **alors**
      PUSH($\mathbf{P}, \mathbf{v}$)
      marquer $\mathbf{v}$ comme visité
    fin si
  fin pour
fin tant que

4.3.3.2 Compensation du mouvement

L'utilisation seule de YOLOv4-tiny n'est souvent pas suffisante pour filtrer complètement les éléments dynamiques. En effet, le réseau de neurones a été entraîné pour reconnaître un nombre limité d'objets et est enclin à manquer des détections dans les cas de conditions d'illumination difficiles et d'occultations. Nous avons recours à une approche plus traditionnelle, basée sur la compensation du mouvement entre images successives, pour détecter les éventuels éléments mobiles restants. Le processus de cette approche est décrit sur le schéma 4.6.

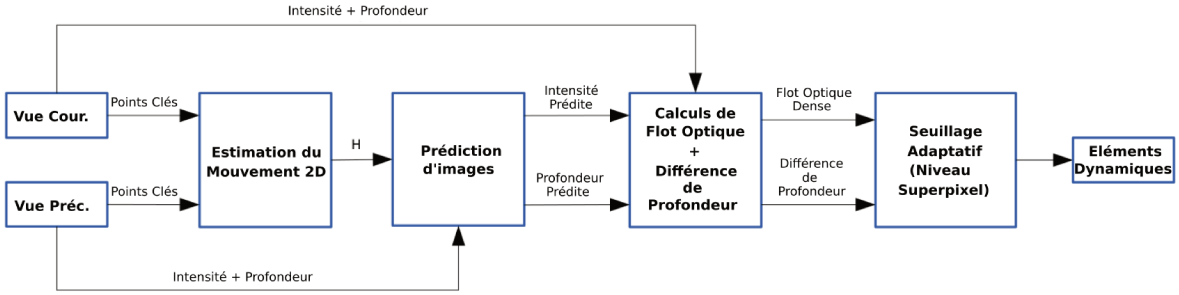

FIGURE 4.6 – Schéma des différentes étapes de l'approche de segmentation des objets dynamiques au niveau superpixel. Dans un premier temps, la compensation de mouvement entre deux images successives permet de prédire une estimation de l'image courante. Un seuillage adaptatif de vecteurs de flot optique et de différences de profondeur entre l'image courante et la prédiction permet ensuite de différencier les superpixels associés à des éléments statiques.

On estime la transformation affine 2D $\mathbf{H} \in SE(2)$ modélisant le déplacement des pixels de la vue précédente à la vue courante. Le travail est réalisé en 2D car la compensation du mouvement dans l'espace 3D est plus coûteuse en terme de calculs et moins robuste à cause du bruit dans les mesures de profondeur. Les points d'intérêt ORB détectés dans les images d'intensité courante I et précédente I_{prev} sont appariés grâce à leurs descripteurs. L'algorithme RANSAC (RANdom SAmple Consensus) [FB81] est utilisé pour calculer de manière robuste la transformation $\mathbf{H}$ en minimisant la distance entre les positions pixels des correspondances établies. RANSAC est une méthode permettant l'estimation robuste des paramètres d'un modèle mathématique parmi un ensemble de données bruitées, donc contenant des aberrations, par échantillonnages aléatoires répétés d'un sous-ensemble minimal des données observées. L'algorithme est explicité en annexe B. On suppose que la partie statique de la scène vue par la caméra est prépondérante par rapport aux éléments dynamiques. Dans le cas contraire, RANSAC peut être amené à sélectionner des points clés ORB associés à des objets dynamiques pour calculer la transformation $\mathbf{H}$, qui ne reflèterait alors plus le vrai mouvement de la caméra. La véracité de cette supposition est très fortement renforcée par l'utilisation de YOLOv4 au cours de l'étape précédente, pour supprimer les points appartenant à des objets potentiellement actifs. La transformation $\mathbf{H}$ calculée est appliquée à I_{prev} et D_{prev} pour générer deux nouvelles images : la prédiction de l'intensité courante I_{pred} et la prédiction de la profondeur courante D_{pred}. Ces images prédites, dans le cas d'une estimation correcte de $\mathbf{H}$ et d'une scène complètement statique sont censées être similaires à l'intensité courante I et à la profondeur courante D.

4.3.3.3 Segmentation des objets dynamiques

Le flot optique dense entre l'intensité courante I et l'intensité prédite I_{pred} est ensuite calculé. La méthode de flot optique DIS (Dense Inverse Search) [Kro+16] est employée car elle fonctionne jusqu'à 600 Hz sur un seul coeur de CPU et atteint des performances état de l'art même dans le cas de déplacements importants. Un vecteur de déplacement 2D $\mathbf{\Delta u}$ est

alors assigné à chaque superpixel de centre $\mathbf{u}$ dans l'image courante en faisant la moyenne des vecteurs de flot optique des pixels qu'il contient. L'utilisation des superpixels comme éléments se déplaçant de manière rigide permet d'accélérer la classification et de réduire l'influence des valeurs aberrantes. Un seuillage adaptatif est ensuite appliqué sur la norme des vecteurs de déplacement $\mathbf{\Delta u}$ des superpixels ainsi que sur leurs valeurs des différences de profondeur compensée, calculées à partir de D et D_{pred}, pour détecter les éléments dynamiques. Un superpixel de centre $\mathbf{u}$ est classé comme dynamique si :

$$\|\mathbf{\Delta u}\| > \tau_f \quad \text{et} \quad |D_{pred}(\mathbf{u}) - D(\mathbf{u})| > \tau_d \tag{4.1}$$

où τ_f et τ_d sont deux seuils adaptatifs. τ_d prend en compte l'incertitude du capteur de profondeur et est défini par :

$$\tau_d = k\sigma_d(\mathbf{u}) \tag{4.2}$$

avec k une constante mise à 10 et $\sigma_d(\mathbf{u})$ l'écart type du bruit sur la mesure de profondeur du centre du superpixel $\mathbf{u}$. Le seuil τ_f est défini d'après [Hua+18] comme :

$$\tau_f = \alpha + \beta\sqrt{\mathbf{H}_{t_x}^2 + \mathbf{H}_{t_y}^2} \tag{4.3}$$

où α sert à compenser la déstabilisation causée par la résolution du capteur ou la précision du flot optique et β permet de pondérer la norme des composantes $\mathbf{H}_{t_x}$ et $\mathbf{H}_{t_y}$ de la partie translation de la transformation affine $\mathbf{H}$, reflétant la vitesse du mouvement de la caméra. Des étapes de traitement supplémentaires permettent de corriger les mauvaises détections en se basant sur le voisinage des superpixels. Par exemple, un superpixel classé comme dynamique mais dont les voisins sont tous classés comme statiques verra son label changé en statique. Finalement, les nouveaux points clés ORB et supersurfels associés à des objets dynamiques sont supprimés pour ne pas perturber la localisation et la mise à jour de la reconstruction.

4.3.4 Localisation

Après le filtrage des points d'intérêt et des supersurfels dynamiques, les primitives statiques de la vue courante sont utilisées pour estimer la nouvelle pose $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}$ de la caméra dans le réferentiel du monde $\mathfrak{w}$. On rappelle que cette pose est composée d'une matrice de rotation $\mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}}$ et d'un vecteur de translation $\mathbf{t}_{\mathfrak{c}}^{\mathfrak{w}}$. On procède en deux étapes consécutives pour mettre à jour la pose :

1. la nouvelle valeur de la pose est initialisée en faisant le suivi de la carte locale $\mathcal{L}$, grâce à une stratégie d'odométrie visuelle indirecte basée sur la mise en correspondances des nouveaux points d'intérêt ORB statiques détectés dans l'image courante avec ceux de la carte locale.
2. la pose initiale est raffinée par recalage dense des données 3D de la vue courante avec les supersurfels de la carte globale $\mathcal{M}$, en utilisant une variante de l'algorithme ICP.

Ce processus en deux temps, qui combine odométrie éparse indirecte et recalage dense, permet une estimation plus robuste de la pose de la caméra dans des conditions variées et complexes. En effet, les méthodes indirectes basées sur les points d'intérêt sont mieux capables de faire

face aux mouvements abrupts de la caméra et aux larges surfaces planes tant que ces surfaces sont suffisamment texturées, tandis les approches denses basées sur l'ICP fonctionnent correctement dans les scènes peu texturées du moment qu'elles sont géométriquement riches. La première étape permet en outre de fournir à l'ICP une estimation initiale suffisamment proche de la valeur optimale nécessaire à sa bonne convergence. Finalement, il est aussi important de noter que la seconde étape effectue un recalage directement par rapport à la carte globale, similaire aux stratégies "frame-to-model" (cf. figure 4.1). Ceci contribue à diminuer le cumul d'erreurs sur le long terme, comparé aux méthodes d'odométrie qui alignent les données d'images consécutives.

4.3.4.1 Initialisation de la pose de la caméra par suivi indirect de la carte locale

Les nouveaux points ORB statiques détectés sont utilisés pour aligner la nouvelle image RGB-D $\{C, D\}$ par rapport à la carte 3D locale $\mathcal{L}$ de points caractéristiques construite à partir des points clés ORB statiques des images précédentes. On minimise l'erreur de reprojection entre les nouveaux points clés 2D et leurs correspondances 3D dans la carte locale, afin d'obtenir une estimation initiale de la pose de la caméra :

$$ {}^{*}\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} = \underset{\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}}{\operatorname{argmin}} \, E_{proj} \tag{4.4} $$

avec

$$ E_{proj} = \sum_{j} \rho(\|\mathbf{u}_j - \pi({\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}}^{-1}\mathbf{p}_j)\|^2) \tag{4.5} $$

où $\mathbf{u}_j$, $\mathbf{p}_j$ sont respectivement les positions 2D des points clés de l'image courante et les points 3D de la carte locale appariés. $\pi(.)$ est le modèle de projection pinhole et $\rho(.)$ est le M-estimateur de Huber : une fonction de coût qui permet d'amoindrir l'impact des données aberrantes (mauvaises associations). De manière générale les M-estimateurs ont pour but de réduire la sensibilité des systèmes de moindres carrés en remplaçant la "somme des erreurs au carré" par un critère plus robuste. La minimisation est faite avec l'algorithme de Levenberg-Marquardt. Le lecteur est invité à se reporter à l'annexe C pour le calcul du jacobien de E_{proj}.

Les points ORB 2D détectés sont appariés avec les points clés 3D de la carte locale en réalisant tout d'abord une mise en correspondance exhaustive ("brute-force") à partir des descripteurs binaires, puis en employant la méthode GMS (Grid Based Motion Statistics) [Bia+17]. GMS repose sur une formulation statistique pour distinguer les bons des mauvais appariements en tenant compte du nombre d'associations voisines. Cette méthode de mise en correspondance robuste présente de bons résultats même dans des environnements peu texturés. Dans le cas où le nombre de correspondances 2D-3D serait trop faible, l'estimation initiale de la pose de la caméra est prédite avec un modèle de mouvement uniforme en utilisant le déplacement de la caméra calculé à l'itération précédente du SLAM.

Notre gestion de la carte locale repose sur l'implémentation de LVT (Lightweight Visual Tracking) [AR18], un algorithme d'odométrie visuelle éparse temps réel nécessitant peu de

ressources informatiques. L'utilisation d'une carte locale permet de réduire fortement l'accumulation d'erreurs et d'augmenter le nombre d'appariements possibles durant la mise en correspondance, par rapport aux approches standards qui alignent les points clés d'images consécutives. Les nouveaux points d'intérêt détectés sont intégrés à la carte locale et les points qui ne sont plus suivis pendant un certain temps sont supprimés afin de ne garder en mémoire qu'un nombre limité de points caractéristiques les plus robustes. Plus de détails sur la gestion de la carte locale sont fournis dans la suite du manuscrit, en section 4.3.5.1.

4.3.4.2 Amélioration de la pose par alignement dense entre la vue courante et la carte globale

Une fois le suivi indirect de la carte locale réalisé, l'estimation initiale est passée en paramètre d'entrée d'un algorithme de type ICP afin d'obtenir une estimation plus précise de la pose de la caméra et plus proche de la carte globale. Cet algorithme accomplit un recalage dense entre les points 3D de la vue courante et les supersufels $\mathcal{M}_k$ du modèle global $\mathcal{M}$. Le recalage est effectué dans le repère monde $\mathfrak{w}$. Les positions 3D $\mathbf{p}_l$ des pixels $\mathbf{u}_l$ de la vue courante sont donc calculés dans le référentiel $\mathfrak{w}$ en leur appliquant la rétroprojection $\pi^{-1}(.)$ et l'estimation initiale de la pose de la caméra : $\mathbf{p}_l = \mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} \pi^{-1}(\mathbf{u}_l, D(\mathbf{u}_l))$. On assigne à chacun de ces points une normale $\mathbf{n}_l$, calculée en transformant dans le repère monde $\mathfrak{w}$ la normale $\mathbf{n}_i$ du supersurfel $\mathcal{F}_i$ de la vue courante $\{C, D\}$ contenant le point $\mathbf{p}_l$: $\mathbf{n}_l = \mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}} \mathbf{n}_i$, avec $\mathbf{R}_{\mathfrak{c}}^{\mathfrak{w}}$ la partie rotation de l'estimation actuelle de la pose de la caméra. Les points $\mathbf{p}_l$ sont ensuite alignés aux supersurfels $\mathcal{M}_k$ de la carte globale $\mathcal{M}$, de positions $\mathbf{p}_k$ et normales $\mathbf{n}_k$, dans le champ de vision de la caméra.

La version symétrisée de l'ICP point-plan [Rus19] est appliquée pour l'alignement. Il s'agit d'une variante de l'ICP traditionnel, qui scinde la transformation géométrique à estimer sy-

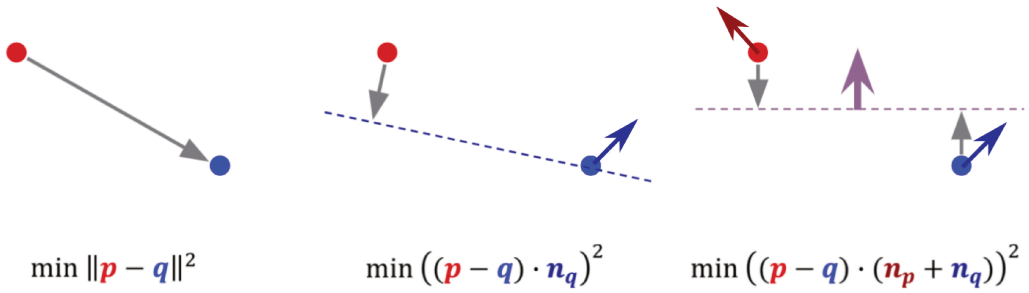

FIGURE 4.7 – Schéma des métriques employés par l'ICP standard [BM92] et ses variantes point-plan [CM91] et point-plan symétrisée [Rus19] (de gauche à droite). L'ICP standard minimise la distances entre les points des modèles 2D ou 3D à recaler. La variante point-plan permet une meilleure convergence en minimisant la distance d'un point au plan contenant l'autre point et perpendiculaire à sa normale. La variante [Rus19] remplace la métrique point-plan par une distance symétrique impliquant les normales des deux points, en les additionnant et en minimisant la distance dans la direction de la somme des normales.

métriquement pour produire une convergence plus rapide et plus sûre. La métrique point-plan symétrisée est explicitée par la figure 4.7. Un ensemble $\mathcal{A}_{k,l}$ des plus proches correspondances entre les supersurfels $\mathcal{M}_k$ de la carte globale et les points 3D $\mathbf{p}_l$ de la vue courante est calculé à chaque itération de l'ICP en utilisant une association par projection de données : le centre $\mathbf{p}_k$ de chaque supersurfel de la carte globale dans le champ de vision de la caméra est projeté dans le plan image Ω de la vue courante en une position pixel $\mathbf{u}_l$ et le point 3D $\mathbf{p}_l$ associé au pixel $\mathbf{u}_l$ est sélectionné pour appariement. Le processus d'association par projection de données est schématisé en figure 4.8. Les paires pour lesquelles les couleurs dans l'espace CIE-Lab, les positions ou les orientations des normales sont trop différentes sont rejetées pour ne pas prendre en compte de correspondances excessivement hétérogènes. Les appariements $\mathcal{A}_{k,l}$ trouvés servent au calcul du terme E_{sym}, bâti sur la métrique point-plan symétrisée, formulé dans l'équation ci-dessous :

$$E_{sym} = \sum_{\mathcal{A}_{k,l}} [(\mathbf{R}_{sym}\mathbf{p}_l - \mathbf{R}_{sym}^{-1}\mathbf{p}_k + \mathbf{t}_{sym}) \cdot (\mathbf{n}_l + \mathbf{n}_k)]^2 \tag{4.6}$$

où $\mathbf{R}_{sym}$, $\mathbf{t}_{sym}$ sont la rotation et translation issues de la scission symétrique de $\mathbf{T}_{rel}$, telles que $\mathbf{R}_{rel} = \mathbf{R}_{sym}\mathbf{R}_{sym}$ et $\mathbf{t}_{rel} = \mathbf{R}_{sym}\mathbf{t}_{sym}$.

Un second terme E_{con} basé sur la distance entre les positions 3D des points d'intérêt est ajouté au terme symétrisé. Inspiré par Xie et al. [Xie+15], il utilise 30% des paires 2D-3D correctes de points d'intérêt ORB ($\mathbf{u}_j$, $\mathbf{p}_j$) calculées lors du suivi de la carte locale (étape précédente 4.3.4.1) ayant l'erreur de reprojection la plus petite, comme contraintes fixes pour mieux guider la convergence de l'ICP. Si il n'est pas toujours utile, ce terme de contraintes permet néanmoins d'ajouter un niveau de robustesse supplémentaire, par exemple dans les cas où la caméra bouge rapidement devant des surfaces plates, sans reliefs. La rotation $\mathbf{R}_{rel}$ et la translation $\mathbf{t}_{rel}$ de la transformation relative $\mathbf{T}_{rel}$, liant l'estimation courante de la pose de la caméra à l'estimation raffinée finale, sont calculées en minimisant la fonction de coût totale suivante :

$$E_{icp} = E_{sym} + \lambda E_{pt} \tag{4.7}$$

avec

$$E_{pt} = \sum_j \|\mathbf{T}_{rel}\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}\pi^{-1}(\mathbf{u}_j, D(\mathbf{u}_j)) - \mathbf{p}_j\|^2 \tag{4.8}$$

λ est un scalaire qui pondère l'influence des contraintes fixes point-point. La pose finale de la caméra est obtenue après l'ICP par concaténation avec la transformation relative $\mathbf{T}_{rel}$: $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} \leftarrow \mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}\mathbf{T}_{rel}$. Les annexes D et E fournissent le détail de l'algorithme de l'ICP et la procédure employée pour dériver la métrique point-plan symétrisée.

4.3.5 Cartographie

Une fois la localisation effectuée, les nouvelles primitives statiques extraites de la paire RGB-D courante sont intégrées aux cartes locale et globale. Une approche de fermeture de boucle, inspirée de celle de ElasticFusion [Whe+15], est aussi appliquée pour corriger les dérives accumulées au niveau de la localisation et de la reconstruction lorsque la caméra

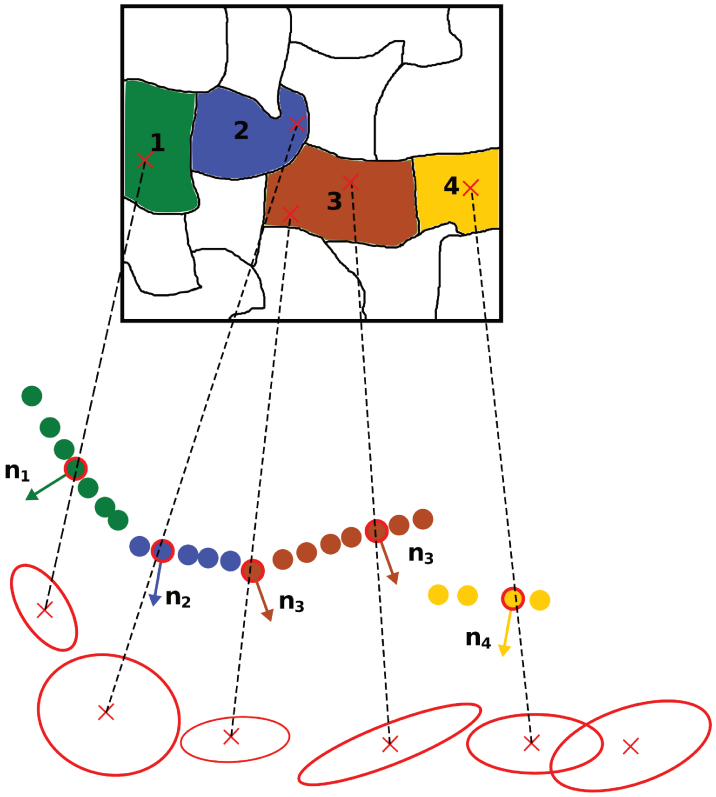

FIGURE 4.8 – Schéma de l'association par projection de données, entre les points 3D de la vue courante et les supersurfels de la carte globale. Les centres des supersurfels (ellipses rouges) sont projetés dans l'image courante segmentée en superpixels et appariés aux rétroprojections 3D (points multicolores) des pixels sur lesquels ils tombent. Les points appariés sont entourés en rouge. Les rétroprojections des pixels contenus dans un même superpixel (de couleur identique sur le dessin) ont tous la même normale.

repasse par un endroit déjà visité. En reconnaissant un lieu déjà observé, les données acquises et estimées depuis la première observation sont corrigées si nécessaire, en mettant en commun les informations issues des première et seconde observations. Un ensemble de déformations non rigides est appliqué à la carte globale afin de joindre les supersurfels récents qui ont dérivé aux supersurfels plus anciens.

4.3.5.1 Mise à jour des cartes locale et globale

L'ensemble de supersurfels $\mathcal{F}$, extrait de l'image courante, est ajouté à la carte globale $\mathcal{M}$ en suivant le processus de fusion décrit au chapitre 3 section 3.3.4, pour étendre la reconstruction, l'améliorer et réduire les redondances. Les éléments instables ou anormaux de la carte globale sont aussi supprimés.

Les points d'intérêt ORB détectés dans l'image courante sont quant à eux transformés dans le référentiel du monde $\mathfrak{w}$ et ajoutés à la carte locale $\mathcal{L}$, en tant que points 3D avec

descripteurs binaires associés, si le nombre d'appariements trouvés durant la localisation est trop petit ou si la taille de la carte local passe en dessous d'un seuil choisi (800 dans notre cas). Lorsqu'un point de la carte locale ne peut plus être suivi, c'est-à-dire quand il n'est pas mis en correspondance un certain nombre d'itérations (fixé à 10 pour assurer le suivi de points sur plusieurs images tout en limitant leur accumulation), il est supprimé. La carte locale garde donc une taille réduite, ne conservant en mémoire que les points d'intérêt traçables récents et proches de la caméra. De cette façon, le temps d'exécution pour l'initialisation de la pose de la caméra reste stable et rapide.

4.3.5.2 Détection de fermeture de boucle

L'implémentation de la détection de fermeture de boucle utilise la méthode de Glocker et al. [Glo+15] basée sur l'encodage d'images clés sous la forme de chaînes de codes binaires appelées *ferns*. Une fern b_F est définie comme un bloc de quatre bits $b_F = f_R f_V f_B f_D$, où chaque bit correspond à un canal de la vue RGB-D. Soit X un canal de la paire RGB-D à encoder (rouge R, ou vert V, ou bleu B, ou profondeur D), $\mathbf{u}$ une position pixel échantillonnée aléatoirement et τ_X un seuil fixé aléatoirement, différent pour chaque canal X. La valeur de chaque bit représente un test binaire sur les données images, évalué de la manière suivante :

$$f_X(\mathbf{u}, \tau_X) = \begin{cases} 1 & \text{si } X(\mathbf{u}) \geq \tau_X \\ 0 & \text{sinon} \end{cases} \tag{4.9}$$

La figure 4.9 schématise le processus d'encodage. Une image RGB-D est décrite comme un unique code global b_G, obtenu par concaténation de m ferns différentes : $b_F = b_{F_1} ... b_{F_m}$. On génère le code b_G en réduisant tout d'abord la taille de l'image RGB-D aux dimensions 80×60 puis en calculant $m = 500$ ferns, correspondant à 500 positions pixels choisies aléatoirement et uniformément distribuées dans l'espace image. Les seuils $\{\tau_R, \tau_V, \tau_B, \tau_D\}$ pour les 500 valeurs de m sont assignés par échantillonnage uniforme, sur la plage $[0, 255]$ pour les couleurs et $[z_{min}, z_{max}]$ pour la profondeur. Cet encodage très compact permet d'évaluer rapidement et simplement la dissimilarité entre deux vues RGB-D, nommées par exemple A et B, de codes

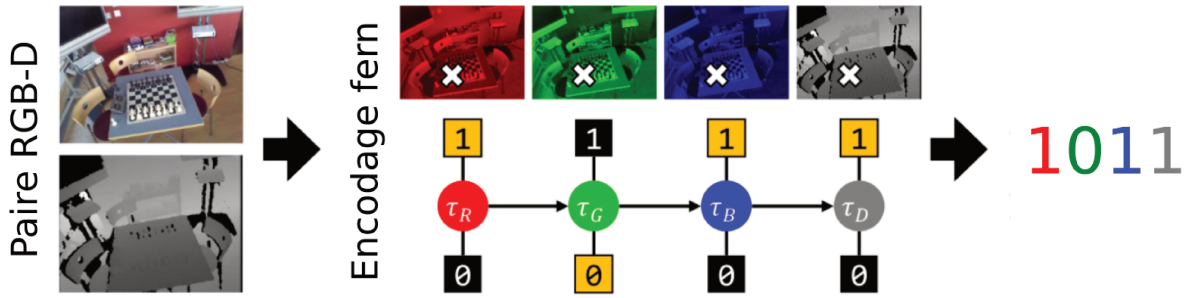

Figure 4.9 – Exemple d'encodage sous forme de fern [Glo+15]. Etant donné une image RGB-D, une position pixel (croix blanche) et des seuils (τ_R, τ_G, τ_B, τ_D) sont choisis aléatoirement, puis la valeur du pixel est comparée aux différents seuils pour former un code binaire sur quatre bits.

respectifs b_G^A, b_G^B, en appliquant la distance de hamming par bloc (BlockHD) :

$$\text{BlockHD}(b_G^A, b_G^B) = \frac{1}{m} \sum_{k=1}^{m} b_{F_k}^A \Leftrightarrow b_{F_k}^B \tag{4.10}$$

où l'opérateur d'équivalence $\Leftrightarrow$ retourne 0 si les deux ferns sont complètement identiques et 1 si au moins 1 de leurs bits est différent.

SupersurfelFusion construit au fur et à mesure une base d'images clés. Pour chaque nouvelle paire RGB-D $\{C, D\}$ acquise on calcule le score de dissimilarité avec toutes les images clés précédemment stockées, à partir de la distance BlockHD (equation 4.10). Si le score minimum est supérieur à un seuil τ_{kf}, la nouvelle vue est enregistrée dans la base d'images clés. Si le score est inférieur au seuil τ_{kf}, cela veut dire que la nouvelle vue est similaire à une vue précédemment enregistrée sous forme d'image clé $\mathcal{E}$. Une image clé $\mathcal{E}$ est une structure contenant plusieurs attributs :

— un code binaire $\mathcal{E}_{b_G}$ calculé avec la méthode des ferns ;

— une pose de caméra $\mathcal{E}_{\mathbf{T}}$;

— un ensemble de supersurfels $\mathcal{E}_{\mathcal{S}}$, dans le repère défini par la pose de caméra $\mathcal{E}_{\mathbf{T}}$;

— un ensemble de points ORB 3D $\mathcal{E}_{\mathcal{X}}$, dans le repère défini par la pose de caméra $\mathcal{E}_{\mathbf{T}}$;

— un marqueur temporel $\mathcal{E}_t$.

Quand une image clé $\mathcal{E}$ similaire à la vue courante $\{C, D\}$ est détectée, il s'agit peut-être d'une fermeture de boucle. On essaye alors d'aligner la vue courante à l'image clé afin de la valider, puis corriger la pose actuelle de la caméra $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}$. Pour se faire, les points ORB de la nouvelle paire RGB-D et de l'image clé candidate sont appariés avec la méthode GMS [Bia+17]. Si il y a suffisamment de correspondances, la transformation $\mathbf{T}_{LC}$ entre les deux vues est calculée par minimisation de l'erreur de reprojection entre les points ORB 3D de l'image clé et les points ORB 2D de la nouvelle image, comme au paragraphe 4.3.4.1. Le calcul de la transformation est associé à l'algorithme RANSAC [FB81] pour plus de robustesse. Si RANSAC donne moins de 30% d'appariements pertinents vis-à-vis de la transformation estimée, la fermeture de boucle est rejetée. Sinon, la fermeture de boucle est acceptée et la transformation $\mathbf{T}_{LC}$ est affinée par recalage dense des supersurfels $\mathcal{F}$ de la vue courante et de ceux de l'image clé $\mathcal{E}_{\mathcal{S}}$. La méthode utilisée, reposant sur la variante symétrisée de l'ICP point-plan [Rus19], est la même que celle proposée en section 4.3.4.2.

4.3.5.3 Génération du graphe de déformation

Dans le cas d'une fermeture de boucle acceptée, une approche de déformation de l'espace basée sur la technique de [SSP07] est utilisée pour refléter la correction de la pose de la caméra au niveau des supersurfels de la carte globale $\mathcal{M}$. Cette déformation de l'espace est effectuée à travers l'application d'un ensemble de transformations isométriques. Les attributs géométriques des supersurfels sont déformés de manière non rigide à partir d'un graphe de déformation incrusté dans la surface reconstruite. Sumner, Schmid et Pauly [SSP07] définissent un graphe de déformation $\mathcal{G}$ comme un ensemble de noeuds $\mathcal{G}_j$ connectés, distribués

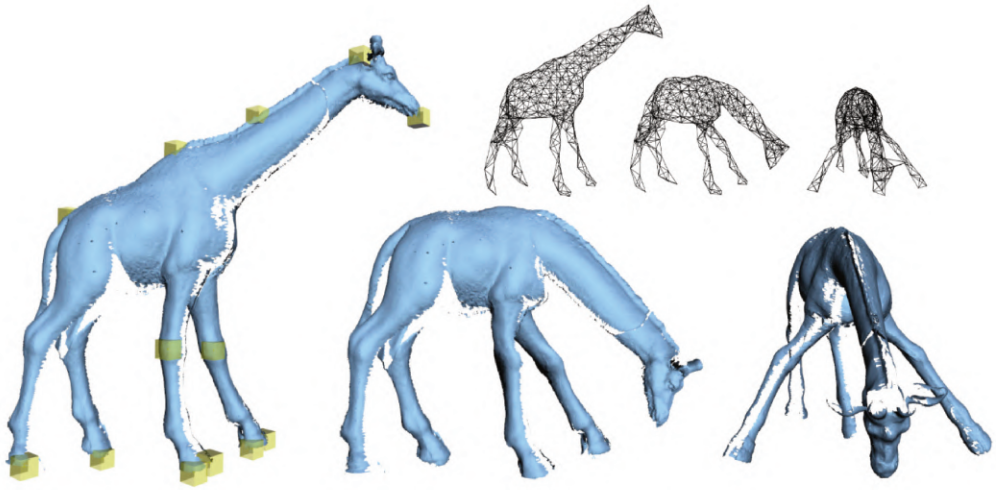

FIGURE 4.10 – Exemple de graphe de déformation (en haut) associé au modèle 3D d'une giraffe sous forme de nuage de points [SSP07]. Les cubes jaunes sont des contraintes manipulables par l'utilisateur, lui permettant de modifier la posture du modèle 3D en agissant sur la structure du graphe sous-jacent.

uniformément à travers le modèle 3D à déformer et ayant chacun une influence sur la surface locale environnante. Une visualisation d'un graphe de déformation est affiché figure 4.10. Dans notre cas les noeuds sont définis à partir d'un sous-ensemble des supersurfels du modèle 3D $\mathcal{M}$, obtenu par sous-échantillonnage par intervalles, tel que le nombre de noeuds du graphe soit fortement inférieur au nombre de supersurfels de la reconstruction.

Chaque noeud $\mathcal{G}_j$ du graphe $\mathcal{G}$ est doté d'une position $\mathbf{g}_j$ qui est celle du supersurfel échantillonné correspondant, et d'un horodatage h_j qui prend la valeur du marqueur temporel d'initialisation du supersurfel. L'ensemble des noeuds est ensuite trié par ordre décroissant de marqueurs temporels. La connectivité du graphe est alors calculée en connectant de manière séquentielle les noeuds ordonnés par ordre décroissant d'horodatages, afin de définir pour chacun son ensemble de voisins $\mathcal{N}(\mathcal{G}_j)$. Ici nous utilisons 4 voisins par noeud. L'intérêt de définir l'ensemble des noeuds voisins à partir des données d'horodatage et non spatiales est d'éviter de mettre en relation des zones temporellement non corrélées de l'environnement. Un noeud $\mathbf{g}_j$ stocke aussi une transformation isométrique tridimensionnelle, sous la forme d'une matrice de rotation $\mathbf{R}_j$ et d'un vecteur de translation $\mathbf{t}_j$, initialisée par défaut à l'identité. Les paramètres $\mathbf{R}_j$ et $\mathbf{t}_j$ des noeuds sont optimisés en fonction d'un ensemble de contraintes, établies grâce à la pose corrigée de la caméra, pour déformer la carte globale $\mathcal{M}$ de la façon la plus naturelle et continue possible.

4.3.5.4 Déformation non rigide de la reconstruction

Après optimisation du graphe de déformation (le processus est explicité dans la section suivante), les noeuds $\mathcal{G}_j$ sont utilisés pour déformer les supersurfels $\mathcal{M}_k$ de la carte globale $\mathcal{M}$.

Chaque supersurfel $\mathcal{M}_k$ est influencé par un ensemble noeuds qui lui sont proches $\mathcal{I}(\mathcal{M}_k, \mathcal{G})$ et se voit appliquer une combinaison linéaire de leurs transformations isométriques. Le jeu de noeuds environnants $\mathcal{I}(\mathcal{M}_k, \mathcal{G})$ d'un supersurfel $\mathcal{M}_k$ est déterminé en prenant dans un premier temps les α noeuds les plus proches dans le temps, avec $\alpha > n$. Parmis ces α noeuds, les n plus proches voisins en terme de distance euclidienne sont retenus et constituent l'ensemble de noeuds de déformation $\mathcal{I}(\mathcal{M}_k, \mathcal{G})$, influençant le supersurfel $\mathcal{M}_k$. L'effet de la transformation isométrique d'un noeud sur un supersurfel est centré en la position du noeud. La position $\mathbf{p}_k$ déformée d'un supersurfel de la carte globale $\mathcal{M}_k$, influencé par l'ensemble $\mathcal{I}(\mathcal{M}_k, \mathcal{G})$ de noeuds du graphe, est donnée par la formule suivante :

$$\mathbf{p}_k \leftarrow \sum_{j \in \mathcal{I}(\mathcal{M}_k, \mathcal{G})} w_j(\mathcal{M}_k)[\mathbf{R}_j(\mathbf{p}_k - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j] \tag{4.11}$$

où

$$w_j(\mathcal{M}_k) = (1 - \|\mathbf{p}_k - \mathbf{g}_j\|/d_{max})^2 \tag{4.12}$$

avec d_{max} la distance euclidienne entre le centre du supersurfel à déformé et le $n+1$ (cinquième dans notre cas) plus proche voisin parmi les noeuds du graphe. Pour transformer l'orientation du supersurfel et son attribut de covariance, on calcule d'abord une approximation $\bar{\mathbf{R}}$ de la moyenne des rotations $\mathbf{R}_j$ des noeuds de $\mathcal{I}(\mathcal{M}_k, \mathcal{G})$, à partir de la représentation quaternion $\bar{\mathbf{q}}$ de $\bar{\mathbf{R}}$ et des représentations quaternion $\mathbf{q}_j$ des rotations des noeuds :

$$\bar{\mathbf{q}} = \frac{\sum\limits_{j \in \mathcal{I}(\mathcal{M}_k, \mathcal{G})} w_j(\mathcal{M}_k)\mathbf{q}_j}{\| \sum\limits_{j \in \mathcal{I}(\mathcal{M}_k, \mathcal{G})} w_j(\mathcal{M}_k)\mathbf{q}_j \|} \tag{4.13}$$

L'orientation et la covariance peuvent alors être transformées :

$$\mathbf{O}_k \leftarrow \bar{\mathbf{R}}\mathbf{O}_k \quad \text{et} \quad \mathbf{\Sigma}_k \leftarrow \bar{\mathbf{R}}\mathbf{\Sigma}_k\bar{\mathbf{R}}^\intercal \tag{4.14}$$

Les positions 3D des points ORB de la carte locale $\mathcal{L}$ sont aussi déformées en appliquant une formule identique à celle de 4.11. Finalement, la pose courante de la caméra $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}$ peut alors être corrigée à partir de la transformation $\mathbf{T}_{LC}$ calculée en 4.3.5.2, liant la nouvelle paire RGB-D et l'image clé, et la pose $\mathcal{E}_{\mathbf{T}}$ de l'image clé : $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} = \mathcal{E}_{\mathbf{T}}\mathbf{T}_{LC}$. L'application de la déformation non rigide sur la carte globale $\mathcal{M}$ est réalisée sur GPU.

4.3.5.5 Optimisation du graphe de déformation

L'optimisation du graphe permet de trouver les transformations isométriques correctives $\{\mathbf{R}_j|\mathbf{t}_j\}$ des noeuds $\mathcal{G}_j$ du graphe. L'application de ces transformations sur les supersurfels permet de corriger les dérives et les décalages présents dans la carte globale $\mathcal{M}$. Cette optimisation est réalisée en minimisant les coûts de trois termes décrits ci-dessous.

1. *Energie de rotation* :

$$E_{rot} = \sum_{\mathcal{G}_j} \|\mathbf{R}_j^\intercal\mathbf{R}_j - \mathbf{I}\|_F^2 \tag{4.15}$$

avec $\|.\|_F$ la norme de Frobenius. L'énergie de rotation assure que les variables optimisées pour les rotations $\mathbf{R}_j$ des noeuds soient orthonormales et constituent donc des matrices de rotation valides.

2. *Energie de régularisation* :

$$E_{reg} = \sum_{\mathcal{G}_j} \sum_{n \in \mathcal{N}(\mathcal{G}_j)} \|\mathbf{R}_j(\mathbf{g}_n - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j - (\mathbf{g}_n + \mathbf{t}_n)\|^2 \tag{4.16}$$

Cette énergie impose des transformations homogènes localement (entre les noeuds $\mathcal{G}_j$ et leurs voisins $\mathcal{N}(\mathcal{G}_j)$), afin de garantir une déformation lisse, régulière à travers le graphe.

3. *Energie de contrainte* : ce dernier terme est employé pour minimiser l'erreur de position sur un jeu de contraintes $\mathcal{Q}$ données. Une contrainte $\mathcal{Q}_l \in \mathcal{Q}$ est un tuple contenant une position d'origine $\mathbf{o}_l \in \mathbb{R}^3$ et une position de destination $\mathbf{d}_l \in \mathbb{R}^3$. Les transformations isométriques des noeuds sont calculées de manière à ce que les positions d'origine des contraintes rejoignent les positions de destination désirées, par application des déformation correctives :

$$E_{con} = \sum_{\mathcal{Q}_l} \|\Phi(\mathbf{o}_l) - \mathbf{d}_l\|^2 \tag{4.17}$$

La fonction $\Phi : \mathbb{R}^3 \to \mathbb{R}^3$ consiste en une transformation identique à la formule 4.11 :

$$\Phi(\mathbf{o}_l) = \sum_{j \in \mathcal{I}(\mathcal{O}_l, \mathcal{G})} w_j(\mathcal{Q}_l)[\mathbf{R}_j(\mathbf{o}_l - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j] \tag{4.18}$$

où $\mathcal{I}(\mathcal{O}_l, \mathcal{G})$ désigne l'ensemble des noeuds du graphe ayant une influence sur la position $\mathbf{o}_l$ de la contrainte $\mathcal{Q}_l$ et les $w_j(\mathcal{Q}_l)$ sont les poids associés, calculés similairement à l'expression 4.12 de la section précédente. Les contraintes sont obtenues en prenant un sous échantillonnage aléatoire des supersurfels de la vue courante $\mathcal{F}$. Pour chaque supersurfel $\mathcal{F}_i$ aléatoirement échantillonné on génère deux types de contraintes :

— une contrainte générique, qui utilise $\mathbf{o}_l = \mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} \mathbf{p}_i$ et $\mathbf{d}_l = \mathbf{T}_{LC} \mathcal{E}_{\mathbf{T}} \mathbf{p}_i$ comme positions d'origine et de destination. On rappelle que $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}$ est la pose, encore non corrigée, de la caméra dans le repère monde, $\mathbf{p}_i$ la position du centre du supersurfel $\mathcal{F}_i$, $\mathcal{E}_{\mathbf{T}}$ la pose de l'image clé extraite par la reconnaissance de place basée sur les ferns (voir section 4.3.5.2) et $\mathbf{T}_{LC}$ la transformation alignant la vue courante à l'image clé. Les contraintes génériques garantissent lors d'une fermeture de boucle que la carte soit déformée à partir de la zone reconstruite actuellement dans le champ de vision de la caméra (origine) vers une région précédemment observée correspondant à l'image clé (destination).

— une contrainte fixe qui utilise la même position $\mathbf{T}_{LC} \mathcal{E}_{\mathbf{T}} \mathbf{p}_i$ pour origine $\mathbf{o}_l$ et destination $\mathbf{d}_l$. Les contraintes fixes servent à préserver les zones viables, plus anciennes, de la surface reconstruite ayant accumulées moins de dérives. Elles ont pour but de fixer les parties de la carte 3D qui n'ont pas à être déformées.

La fonction de coût totale pour une fermeture de boucle est donc définie par :

$$E_{total} = w_{rot} E_{rot} + w_{reg} E_{reg} + w_{con} E_{con} \tag{4.19}$$

avec $w_{rot} = 1$, $w_{reg} = 10$ et $w_{con} = 100$. Elle est minimisée par rapport aux transformation $[\mathbf{R}_j|\mathbf{t}_j]$ de tous les noeuds du graphe avec l'algorithme de Gauss-Newton. La matrice Jacobienne du problème étant creuse, la factorisation de Cholesky creuse est utilisée pour une résolution efficace du système sur CPU. Une visualisation des contraintes générées lors d'une fermeture de boucle est fournie sur l'image de droite de la figure 4.11.

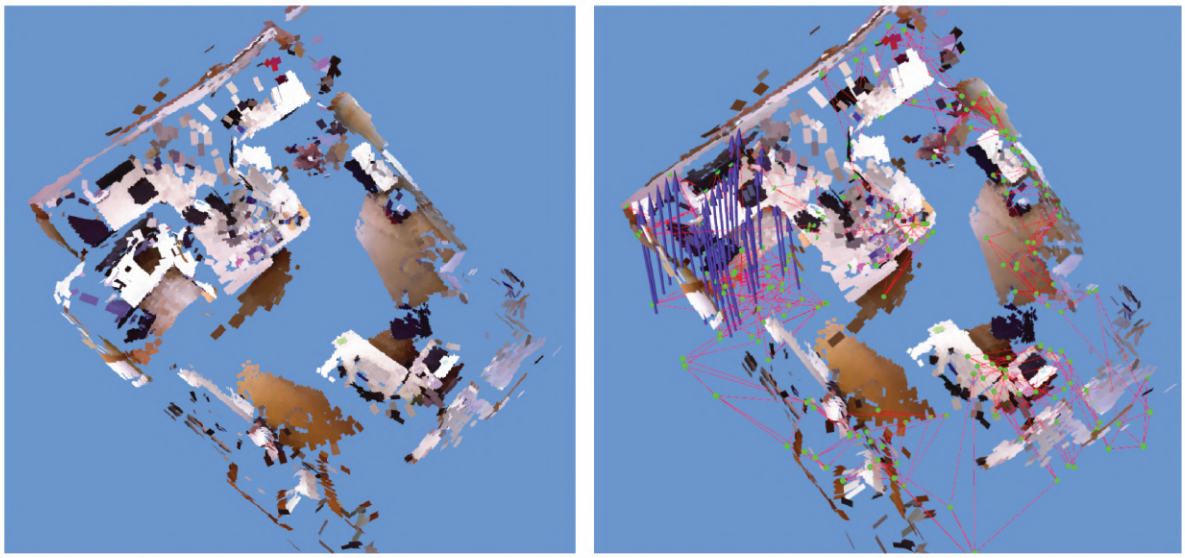

FIGURE 4.11 – Visualisation d'une fermeture de boucle. La scène vue de dessus avant fermeture de boucle est montrée à gauche. On remarque que l'espace en haut à gauche est décalé. A droite, la carte a été corrigée grâce à la fermeture de boucle. Les noeuds du graphe de déformation sont visibles en vert, les branches en rouge et les contraintes sont indiquées par des flèches bleues.

4.4 Évaluation

Nous avons conduit un ensemble d'évaluations quantitatives et qualitatives sur notre SLAM SupersurfelFusion (SSF). Il a été comparé à quatre autres systèmes denses RGB-D de l'état de l'art : Co-Fusion (CF) [RA17], StaticFusion (SF) [Sco+18], ReFusion (RF) [Pal+19], tous conçus pour les scènes dynamiques, et ElasticFusion (EF) [Whe+15], élaboré seulement pour les milieux statiques. ElasticFusion, StaticFusion et Co-Fusion sont des méthodes basées sur une modélisation surfel, tandis que ReFusion emploie une représentation volumétrique basée sur les voxels pour cartographier l'environnement. Les séquences vidéo du jeu de données TUM RGB-D [Stu+12] et les vérités terrains des trajectoires associées ont été employées pour mesurer la précision de la localisation du système dévelopé ainsi que son efficacité en terme de temps d'exécution et d'utilisation de la mémoire. La précision de la surface reconstruite est quant à elle évaluée sur les vidéos synthétiques de jeu de données ICL-NUIM [Han+14]. Des détails sur les séquences sont donnés dans le tableau 4.1.

Les tests ont été effectués sur le même matériel que ceux du chapitre 3, soit un ordinateur

Séquence	Durée (s)	Longueur de la trajectoire (m)
fr1_xyz	30.09	7.112
fr1_room	48.90	15.989
fr3_office	87.09	21.455
fr3_sit_xyz	42.50	5.496
fr3_walk_xyz	28.83	5.791
fr3_walk_half	35.81	7.686
kt0	51.00	6.540
kt1	33.00	2.050
kt2	30.00	8.430
kt3	42.00	11.320

TABLE 4.1 – Description des séquences vidéos utilisées pour l'évaluation de SupersurfelFusion.

portable équipé d'une carte graphique Nvidia GTX 950M avec 2 Go de mémoire et d'un processeur Intel Core i5-6300HQ. La détection de points d'intérêt, l'odométrie éparse indirecte, la mise à jour de la carte locale, la génération du graphe de déformation et son optimisation, ainsi que la compensation du mouvement et le flot optique dans la détection d'éléments dynamiques ont été implémentées en C++ sur CPU. Tout le reste est codé sur GPU avec CUDA. Des tests ont aussi été réalisés en direct dans les bureaux de notre laboratoire et dans le salon d'un appartement en utilisant une caméra RGB-D Intel RealSense D435, avec une fréquence de 30 Hz et une résolution de 640×480 pixels. On notera que pour accélérer et faciliter la visualisation en direct, nous avons choisi d'afficher les supersurfels avec une forme rectangulaire plutôt qu'elliptique.

4.4.1 Analyse qualitative et précision de la reconstruction

La figure 4.12 présente des vues de différentes cartes reconstruites au sein de notre laboratoire avec la représentation supersurfel en déplaçant le capteur RGB-D à la main.

Elle montre que le système est capable de reconstruire des pièces complètes en préservant leur intégrité. Les modèles produits sont denses, compacts et l'on peut constater que les supersurfels sont suffisamment adapatés pour reconstruire bon nombre d'éléments présents dans les milieux de travail scannés (chaises, bureaux, ordinateurs, murs, sol...).

Nous avons évalué la qualité de la surface produite par SupersurfelFusion (SSF) sur les séquences *kt0*, *kt1*, *kt2*, *kt3*, du jeu de données synthétique ICL-NUIM. Similairement à la section 3.4.2, la carte 3D de supersurfels a été convertie en un nuage de points dense pour chaque séquence vidéo. Puis la distance moyenne entre les points et la surface la plus proche de la vérité terrain a été calculée. Les valeurs obtenues sont présentées dans le tableau 4.2 et comparées à celles du SLAM ElasticFusion (EF). Notre méthode donne des résultats moins bons que ElasticFusion sur la plupart des séquences évaluées, principalement à cause des bords des objets qui ne sont pas modélisés aussi précisément avec les supersurfels qu'avec les surfels. Les séquences synthétiques utilisées sont de plus très peu bruitées et ne permettent pas de

FIGURE 4.12 – Exemples de cartes reconstruites avec SupersurfelFusion, caméra à la main, dans les bureau au laboratoire GIPSA-lab.

rendre compte des capacités de filtrage et de lissage des supersurfels. On note cependant que le SLAM proposé permet d'atteindre un niveau de précision proche de celui de ElasticFusion, le dépassant même sur la séquence *kt3*. Dans cette dernière la caméra fait face à des zones très pauvres en termes de géométrie et de couleur, ce qui déteriore la localisation de ElasticFusion basée sur la totalité des pixels des images. SupersurfelFusion parvient néanmoins à détecter suffisamment de points d'intérêt ORB pour guider correctement l'estimation du mouvement de la caméra. Notre système bénéficie d'une odométrie moins précise dans les situations idéales mais il est plus robuste dans des situations d'acquisition dégradées.

Système	**kt0**	**kt1**	**kt2**	**kt3**
EF	**0.7**	**0.7**	**0.8**	2.8
SSF	1.1	1.9	1.0	**1.3**

TABLE 4.2 – Comparaison de la précision de la surface reconstruite (cm). EF : ElasticFusion ; SSF : SupersurfelFusion.

Un exemple de fermeture de boucle lors de la reconstruction d'un salon d'appartement est montré sur la figure 4.13. La fermeture de boucle, appliquée en temps réel sur une carte relativement large, permet de corriger les dérives introduites lorque le système reconnaît un endroit précédemment parcouru par la caméra. Il est important de noter que, si l'approche de détection de fermeture de boucle utilisée par SupersurfelFusion est très efficace en terme de calculs, elle peut manquer des détections dans des cas particuliers en raison de sa simplicité. Par exemple, la reconnaissance visuelle de lieu est susceptible d'échouer si de nouveaux objets sont introduits dans la scène ou si les conditions lumineuses varient entre deux passages de

la caméra dans un même endroit. Pour gérer les éléments dynamiques pouvant perturber la détection de fermeture de boucle, on pourrait envisager d'utiliser des techniques d'*inpainting* pour remplacer les occultations, dues à des objets en mouvement, par le fond statique, à la manière de [Bes+18].

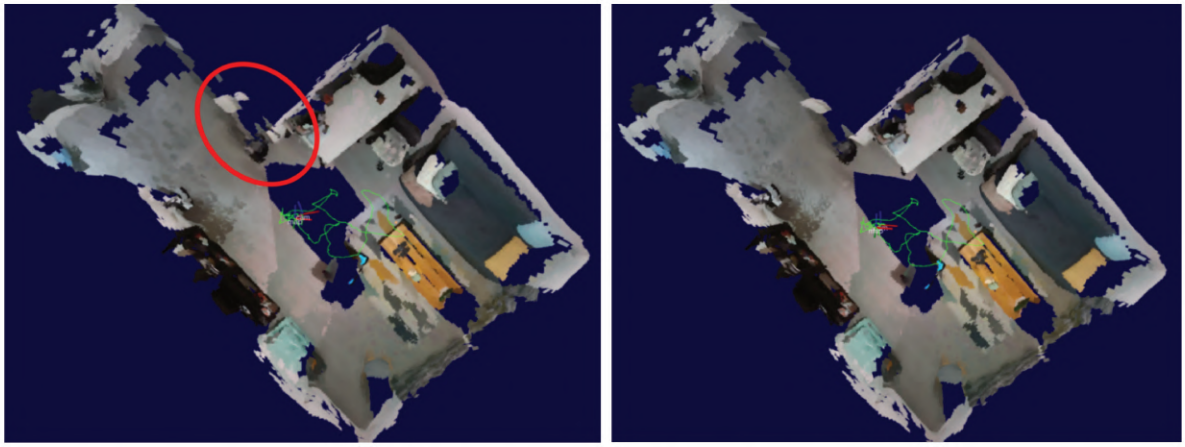

FIGURE 4.13 – Exemple de fermeture de boucle, avec à gauche la carte avant la correction et à droite la carte corrigée. L'inconsistance de la reconstruction (surface dupliquée) avant fermeture de boucle est entourée en rouge. Elle est due à l'accumulation d'erreurs dans l'estimation incrémentale de la trajectoire de la caméra.

4.4.2 Precision de la localisation

Pour mesurer la précision de notre localisation nous utilisons l'erreur quadratique moyenne de position entre les points de chaque trajectoire estimée et ceux de la vérité terrain dans 6 séquences vidéo différentes : trois séquences en milieu statique (*fr1_xyz*, *fr1_room*, *fr3_office*) et trois séquences en environnement dynamique (*fr3_sit_xyz*, *fr3_walk_xyz*, *fr3_walk_half*). *fr3_walk_xyz* et *fr3_walk_half* sont des séquences hautement dynamiques dans lesquelles deux personnes sont pratiquement constamment en mouvement dans le champ de vision de la caméra. Au contraire, la séquence *fr3_sit_xyz* est faiblement dynamique : les deux personnes sont assises à un bureau et gesticulent légèrement. Les poses de la trajectoire estimée et celles de la vérité terrain sont associées à partir d'horodatages, puis les deux trajectoires sont alignées, en se basant sur les appariements, par décomposition en valeurs singulières. La différence entre chaque paire de poses est alors calculée et on affiche l'erreur quadratique moyenne de ces différences. Les mesures sont répertoriées dans le tableau 4.3 et les trajectoires tracées sur la figure 4.14.

Les résultats donnés par notre système (SSF) sont globalement bons, dans les meilleurs en milieux statiques, surpassant même ElasticFusion (EF) sur la séquence *fr1_room*. Dans cette dernière la caméra se déplace irrégulièrement et fait face à un moment à une surface plane faiblement texturée. La localisation de SupersurfelFusion permet dans ce cas là une plus grande robustesse grâce à la combinaison d'odométrie visuelle éparse basée sur les points d'intérêt,

	Séquence	EF	CF	SF	RF	SSF sans YOLO	SSF
Milieu Statique	fr1_xyz	**1.2**	1.4	1.5	2.4	1.9	1.9
	fr1_room	22.6	26.9	51.8	76.5	**10.4**	**10.4**
	fr3_office	**2.3**	68.3	31.0	11.6	8.8	8.8
Milieu Dynamique	fr3_sit_xyz	**2.4**	2.9	3.9	3.7	4.1	4.5
	fr3_walk_xyz	73.8	69.8	9.3	**8.7**	21.0	11.8
	fr3_walk_half	55.0	82.0	68.0	**10.8**	16.7	13.5

TABLE 4.3 – Erreurs quadratiques moyennes de position (cm) des trajectoires estimées. EF : ElasticFusion ; CF : Co-Fusion ; SF : StaticFusion ; RF : ReFusion ; SSF : SupersurfelFusion.

moins sensible aux grands mouvements, avec le recalage dense s'appuyant sur l'ICP. Lorsque la caméra se déplace de manière lente et continue, la structure pyramidale de la procédure d'odométrie d'ElasticFusion et le grand nombre de pixels qui se chevauchent entre deux vues successives, lui permettent d'atteindre une précision plus importante.

En environnement hautement dynamique (*fr3_walk_xyz* et *fr3_walk_half*) nous pouvons voir les bénéfices apportés par la détection d'objets en mouvement pour supprimer les éléments dynamiques. La stabilité du système est fortement augmentée, contrairement à ElasticFusion, qui ne gère pas les éléments dynamiques et voit ses performances se déteriorer. Co-Fusion (CF), pourtant conçu pour les scènes dynamiques, échoue lorsque les objets bougent trop vite. SupersurfelFusion affiche des résultats globaux proches des meilleurs avec et sans utilisation de YOLO-tiny. On peut voir que l'utilisation de YOLOv4-tiny améliore la précision de SupersurfelFusion dans les séquences très dynamiques. Sans YOLO, l'algorithme donne tout de même un niveau de précision convenable par rapport aux autres approches de SLAM dynamiques. Notre méthode n'apporte pas d'amélioration en environnement très faiblement dynamique comme dans la séquence *fr3_sit_xyz*. On remarque aussi que dans celle-ci le système proposé fonctionne mieux sans le réseau de neurone. En effet, l'emploi de YOLO entraine la détection de la totalité des corps des deux personnes qui occupent une place importante dans l'image et la suppression des éléments qui leur sont associés. Sans YOLO, les parties immobiles des corps ne sont pas détectées et l'algorithme dispose de plus de données caractéristiques et saillantes pour calculer la pose de la caméra.

4.4.3 Efficacité de calcul

La vitesse d'exécution et l'empreinte mémoire de la carte globale reconstruite par SupersurfelFusion ont été examinées avec et sans utiliser YOLOv4-tiny lors de la détection d'objets en mouvement (section 4.3.3). Le réseau de neurone renforce la résistance du SLAM aux éléments dynamiques mais ajoute un coût mémoire de 23.1 Mo et un temps d'inférence autour des 30 ms. Les résultats du tableau 4.4 soulignent l'efficacité de notre méthode qui est bien moins exigeante que les autres en mémoire pour stocker la reconstruction. La mémoire sur GPU étant souvent assez limitée, l'utilisation d'une représentation 3D basse résolution permet à SupersurfelFusion d'être utilisable par une plus large gamme de machines et d'étendre l'échelle de la

scène reconstruite. En réduisant les temps de calculs pour la cartographie, elle permet aussi à SupersurfelFusion d'atteindre une vitesse d'exécution proche de celle de ElasticFusion, malgré son module de détection d'éléments dynamiques supplémentaire. Si l'on désactive YOLOv4-tiny, notre système est le plus rapide en fonctionnant à environ 20 images par seconde, tout en conservant une précision de localisation très compétitive vis-à-vis de l'état de l'art (comme le montrent les évaluations sur les trajectoires estimées en 4.4.2). Nous pouvons noter que StaticFusion est aussi au dessus des autres méthodes en terme de performances car elle utilise des images de taille réduite, tandis que ReFusion donne les pires résultats à cause de la représentation volumétrique qu'elle emploie, plus précise mais très coûteuse.

Séquence	**Taille de la carte (Mo)**					**Temps d'exécution (ms)**					
	EF	CF	SF	RF	SSF	EF	CF	SF	RF	SSF sans YOLO	SSF
fr1/xyz	14.4	13.2	4.5	166.5	**1.8**	72.6	83.0	64.2	297.6	**48.4**	79.6
fr3/office	57.8	42.5	18.6	785.6	**4.8**	84.9	125.0	77.3	480.6	**53.5**	86.3

TABLE 4.4 – Temps d'exécution moyens et empreintes mémoires maximales des cartes reconstruites. EF : ElasticFusion ; CF : Co-Fusion ; SF : StaticFusion ; RF : ReFusion ; SSF : SupersurfelFusion.

4.5 Bilan

Nous avons proposé dans ce chapitre un système de SLAM RGB-D dense, appelé SupersurfelFusion, pour la cartographie 3D de pièces et de salles intérieures. Il s'articule autour de la représentation supersurfel introduite dans le chapitre précédent pour gagner en efficacité et associe des méthodes éparse et dense d'odométrie visuelle pour mieux résister aux déplacements brusques du capteur. Le problème des environnements dynamiques a été abordé en intégrant au SLAM un module de détection des éléments dynamiques à base de compensation de mouvement, de calcul de flot optique et de différence de profondeur. Ce module est couplé avec un réseau de neurones rapide, placé en amont, pour assurer la robustesse de l'algorithme aux objets mobiles, même lorsque la partie dynamique de la scène observée occupe plus de la moitié des pixels d'une image. Les expériences ont démontré le faible coût mémoire et la rapidité du SLAM développé ainsi que ses performances proches de celles des autres systèmes de l'état de l'art pour la précision de la reconstruction et de la localisation. Sa vitesse, mesurée sur un GPU bon marché, est comprise entre 22 et 11 FPS (sans et avec le réseau de neurones), ce qui est suffisant pour une utilisation en ligne.

SupersurfelFusion présente cependant plusieurs faiblesses qui pourraient donner lieu à de futurs travaux. Par exemple, la détection de fermeture de boucle peut échouer car la méthode de reconnaissance de lieu actuelle est sensible aux éléments dynamiques. Une méthode d'*inpainting* pourrait permettre de reconstruire le fond derrière les parties dynamiques filtrées des images. De plus, des images trop plates et uniformes au niveau de la couleur et de la profondeur risquent d'entraîner une perte de localisation. Pour accroître la robustesse de SupersurfelFusion dans les environnements difficiles, on pourrait considérer l'utilisation d'une centrale inertielle qui n'est pas soumise aux mêmes limites que les caméras RGB-D, et faire de

la fusion de données multi-capteurs. Enfin, si SupersurfelFusion permet d'obtenir des cartes 3D cohérentes à l'échelle d'une pièce, pour cartographier des environnements beaucoup plus larges, il serait intéressant de réfléchir à un partitionnement de la carte de supersurfels en sous-ensembles et d'introduire une méthode pour optimiser ces sous-ensembles à un niveau global.

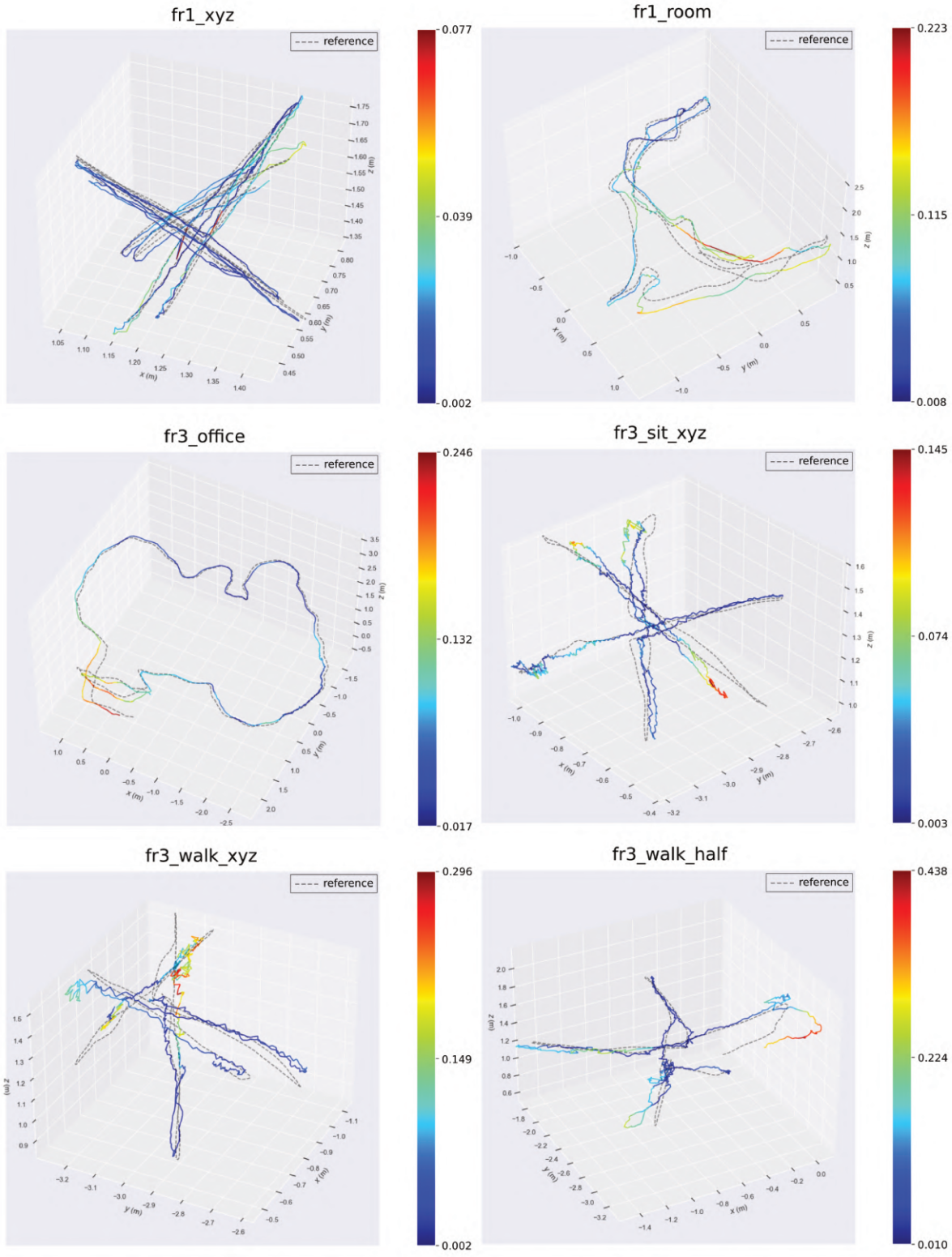

FIGURE 4.14 – Graphiques des trajectoires obtenues sur les différentes séquences d'évaluation. L'erreur (en m) est modélisée sous forme de couleur sur les trajectoires. Les vérités terrains sont tracées en pointillés.

Chapitre 5

Application de SupersurfelFusion à la navigation autonome d'un robot mobile

Sommaire

Dans cette partie, le système SLAM RGB-D dense SupersurfelFusion précédemment décrit est porté sur la carte embarquée de pilotage d'une plateforme robotique mobile et adapté pour pouvoir tirer profit des mesures des encodeurs des roues du robot. Nous proposons l'application du système conçu au cas de la navigation autonome afin d'obtenir une solution de cartographie et d'exploration simultanées basée sur la représentation supersurfel, permettant de se passer d'un opérateur humain pour téléopérer le robot.

5.1 Introduction

La tâche de navigation requiert d'une part une carte représentative de l'environnement pour décider du chemin à suivre et d'autre part un moyen de se situer dans la carte. Elle constitue une application potentielle majeure des systèmes de SLAM visuel, qui peuvent être employés afin de remplacer ou d'améliorer la localisation et la navigation GPS, inapplicable en intérieur et dont la précision de positionnement est de l'ordre de plusieurs mètres. Malgré l'essor rapide de la recherche sur les véhicules et robots autonomes ces dernières années, la navigation se limite la plupart du temps à des environnements déjà visités et se fait à partir d'une carte préexistante, qui peut être générée grâce à un système de SLAM recupérant les mesures acquises par un robot téléopéré. Le déploiement de robots autonomes dans des espaces inconnus ou changeants pour des applications d'exploration diverses (comme de la recherche et du secours en terrain risqué par exemple) nécessite des algorithmes de SLAM spécifiques capables de garantir l'intégrité des robots. Ils doivent fournir une cartographie suffisamment dense et rapide pour éviter des obstacles, ainsi qu'une localisation robuste maintenue à chaque instant pour ne pas faire échouer la navigation.

En dépit de la maturité atteinte par l'état de l'art des techniques de SLAM RGB-D, elles ne sont pas encore assez robustes pour garantir à elles seules la navigation sans risque d'un robot autonome. En effet, la portée des caméras RGB-D est limitée (entre cinq et dix mètres), ce qui est problématique pour la reconstruction d'environnements étendus, et l'incertitude sur leurs mesures est importante pour les surfaces éloignées. De plus, le bon fonctionnement de la plupart des algorithmes de SLAM visuel repose sur la présence de variations de couleur ou de géométrie dans les images. La rencontre de large surfaces planes de textures homogènes peut provoquer l'échec de la localisation, entraînant un arrêt critique du système et/ou un crash du robot. La plupart des techniques de l'état de l'art sont développées dans des environnements contrôlés et destinées à être utilisées à partir d'une caméra guidée par un opérateur humain qualifié. Elles fonctionnent généralement correctement lorsque le capteur est manipulé avec précaution et régularité par une personne qui a conscience des faiblesses des caméras RGB-D et des limites des algorithmes utilisés pour le suivi de la pose. Les mouvements d'un robot présentent des particularités qui peuvent entraîner un échec du système de perception. Par exemple les robots au sol sont susceptibles de démarrer et de s'arrêter brusquement, ou encore d'accomplir des rotations sur place pour avoir une vision globale. Les méthodes d'odométrie visuelle sont sensibles à ces différents mouvements qui génèrent du flou dans les images. La caméra montée sur le robot est de plus soumise à des vibrations. Nous avons aussi vu dans le chapitre précédent les détériorations que peuvent engendrer des personnes et objets en mouvement sur les performances d'algorithmes de SLAM visuel.

Pour pallier les limites des caméras RGB-D et garantir le bon fonctionnement des algorithmes de SLAM en milieux difficiles ou quand la qualité des images reçues est mauvaise, une solution courante est d'effectuer de la fusion de données avec des capteurs complémentaires. Combiner différents capteurs permet d'améliorer les performances d'une méthode de SLAM. Les configurations les plus fréquemment employées consistent à coupler la caméra RGB-D avec une centrale inertielle ou avec des encodeurs de roues. Une centrale inertielle, plus com-

munément appelée IMU (Inertial Measurement Unit) en anglais, est un capteur qui associe des accéléromètres et des gyroscopes pour mesurer l'accélération et la vitesse angulaire sur les trois axes. Ce type de capteur est pratiquement toujours présent sur les drones pour effectuer la stabilisation et permet une estimation de pose à 6 six degrés de liberté (espace 3D). Les encodeurs qui équipent les roues de robots terrestres permettent quant à eux de compter les tours de roues effectués. Si l'on connaît la taille des roues et de l'entraxe du chassis du robot, il est alors possible d'effectuer de l'odométrie en intégrant ces mesures pour extraire une pose à trois degrés de liberté (plan 2D). Les capteurs IMU et les encodeurs présentent l'avantage de n'utiliser aucune information extérieure au robot mobile et offrent donc une grande robustesse contre les éléments dynamiques, auxquels sont sensibles les caméras RGB-D. Cependant, l'odométrie des roues n'est précise que sur de courtes distances, à cause d'écarts introduits par des pertes d'adhérence au niveau des roues, tandis que les mesures inertielles dérivent fortement avec le temps et sont sensibles à la température. On peut néanmoins utiliser ces capteurs de manière fiable pour calculer une première estimation du mouvement relatif du robot entre deux acquisitions d'images, puisqu'ils fournissent des données à bien plus haute fréquence que les caméras RGB-D.

Outre les contraintes inhérentes aux capteurs RGB-D, il existe aussi un écart significatif entre les cartes générées par les approches de SLAM et les cartes souhaitées pour la planification de trajectoires. Les systèmes de SLAM RGB-D permettent une reconstruction dense de l'environnement qui représente tous les obstacles possibles, contrairement aux méthodes éparses, mais le niveau de détail trop élevé n'est généralement pas adapté à la navigation. Ils monopolisent de nombreuses ressources matérielles et logicielles pour reconstruire des cartes complètes mais lourdes et lentes à manipuler sur des systèmes embarqués. Il est de plus nécessaire de filtrer et de traiter ces cartes pour les rendre utilisables efficacement par un logiciel de planification de trajectoires. La majorité des algorithmes de SLAM RGB-D de la littérature n'est donc pas applicable pour effectuer de la navigation autonome concomitante. L'exécution simultanée de méthodes de SLAM RGB-D et de navigation demande une représentation dense mais légère de l'environnement, avec un processus de mise à jour rapide, et transmissible presque directement à un planificateur de trajectoires. La représentation doit aussi être déformable pour pouvoir être corrigée en temps réel en cas de fermeture de boucle.

La contribution principale de ce chapitre est le couplage de SupersurfelFusion (la méthode de SLAM RGB-D proposée dans le chapitre 4) avec une solution de planification de trajectoires, afin de réaliser des tâches de navigation et de cartographie concomitantes et ouvrir la voie à des applications d'exploration autonome. L'efficacité en terme de mémoire et de vitesse d'exécution de notre SLAM RGB-D en font un système particulièrement adapté à la navigation. En plus de sa capacité à appliquer des fermetures de boucles en ligne, la représentation supersurfel qu'il exploite répond aux critères de densité et de légereté. Elle permet une extraction rapide de la surface navigable qui peut directement servir en entrée de l'algorithme de planification de trajectoires. Dans le cadre d'une application visant la navigation autonome, notre système SupersurfelFusion a été porté sur un robot terrestre se déplaçant grâce à deux roues motrices et adapté pour tirer profit des mesures de leurs encodeurs.

Après un rapide aperçu de la littérature de la fusion de données multi-capteurs, nous

présenterons la solution utilisée pour combiner les données RGB-D de la caméra du SLAM SupersurfelFusion avec les mesures acquises par les encodeurs des roues d'un robot terrestre. Nous parlerons ensuite des algorithmes de SLAM RGB-D qui engendrent des cartes navigables et nous expliquerons notre approche de cartographie, localisation et navigation simultanées. Enfin nous décrirons la plateforme matérielle adoptée et donnerons les résultats des expériences réalisées pour valider l'approche.

5.2 Fusion des données des roues et de la caméra RGB-D

Il s'agit ici de présenter le portage de SupersurfelFusion sur un robot à conduite différentielle équipé de roues encodeuses et l'intégration des mesures des capteurs des roues dans l'algorithme de SLAM. Un robot à conduite différentielle est un robot dont la base mobile est constituée de deux roues motrices alignées sur un même axe, appelé entraxe. Avant de décrire les modifications apportées à SupersurfelFusion pour prendre en compte les données des roues, nous faisons un bref état de l'art de la fusion de données dans les systèmes de SLAM RGB-D dense.

5.2.1 Fusion de données capteurs

L'utilisation d'une caméra RGB-D seule n'est pas suffisante dans de nombreuses situations pour garantir le fonctionnement correct en tout instant d'un algorithme de SLAM. Un capteur RGB-D possède un champ de vision réduit et les systèmes de SLAM visuel purs sont fragiles dans les milieux fortement dynamiques ou pauvres en texture et en géométrie. Il est compliqué de s'assurer qu'une caméra montée sur un robot fasse toujours face à suffisamment d'informations pour permettre une bonne localisation. L'incorporation de données provenant d'autres capteurs est un moyen de surmonter ces problèmes et d'améliorer la précision des algorithmes de SLAM.

Ratter et Sammut [RS15] combinent les images d'une caméra RGB-D aux mesures d'un LiDAR 2D pour tirer profit de son champ de vision étendu. Ils introduisent un terme de régularisation basé sur les données 2D, lors du recalage d'images RGB-D successives, pour guider l'estimation de la pose de la caméra en décourageant la mise à jour du déplacement dans les directions pas assez contraintes géométriquement. Une configuration populaire est l'association de la caméra avec un capteur IMU. La fusion de données capteurs est traditionnellement réalisée à l'aide d'un filtre de Kalman étendu [Bru+15] ; [QAM17]. [Lai+17] est la première approche de SLAM RGB-D-inertiel dense à se baser sur une méthode d'optimisation pour fusionner les données des deux capteurs. Elle améliore la précision d'ElasticFusion [Whe+15] en intégrant un coût inertiel dans le suivi du mouvement de la caméra, mais aussi la conformité globale de la reconstruction en contraignant les déformations de la carte à rester alignées avec la gravité lors d'une fermeture de boucle. Cette approche est néanmoins seulement conçue pour les environnements statiques. Liu, Gao et Hu [LGH19] proposent une méthode pour segmenter et supprimer les éléments dynamiques à l'aide de flot optique éparse et d'une estimation

initiale de l'orientation de la caméra fournie par une centrale inertielle.

Les robots terrestres sont souvent équipés de roues encodeuses dont les données sont facilement accessibles et exploitables. Par rapport à la localisation inertielle, sensible à la température et qui dérive avec le temps, l'odométrie à partir d'encodeurs présente l'avantage de ne dériver qu'en fonction de la distance parcourue. L'intégration des mesures d'encodeurs sur des distances courtes peut donc fournir une estimation préalable relativement fiable du déplacement d'un robot. Cependant, les encodeurs de roues ne peuvent donner que des mesures 2D. La méthode [LJ19] fusionne l'odométrie des roues d'un robot et celle d'un SLAM RGB-D avec un filtre de Kalman. KO-Fusion [HBL19] joint les mesures des roues encodeuses d'un robot avec les données visuelles denses d'une caméra RGB-D dans un cadre d'optimisation commun pour estimer de manière robuste et précise les déplacements. L'odométrie des roues donne une première estimation du mouvement et sert aussi de contrainte lors de l'optimisation pour limiter l'erreur résiduelle issue du recalage d'images. DRE-SLAM [Yan+19] utilise une méthode de fusion similaire, en introduisant les données des roues intégrées dans l'optimisation de la pose de la caméra, mais s'appuie sur une stratégie d'odométrie visuelle éparse basée sur des points d'intérêt. Il implémente en plus une méthode pour supprimer les pixels dynamiques basée sur l'association d'un algorithme de clustering K-means, d'un réseau de neurones détecteur d'objet et de contraintes multi-vues. Notre incorporation des mesures des encodeurs des roues est similaire à celle de DRE-SLAM, mais contrairement à ce système notre SLAM ne limite pas les mouvements du robot au plan 2D et calcule une pose à six degrés de liberté dans l'espace 3D.

5.2.2 Architecture du SLAM adapté

La stratégie de localisation de SupersurfelFusion est adaptée pour prendre en compte les mesures fournies par les encodeurs, en plus des mesures visuelles, dans le but d'améliorer la précision du suivi des déplacements du robot et de pouvoir bénéficier d'une estimation même lorsque les informations apportées par les images sont insuffisantes. L'architecture du système de cartographie et localisation simultanées porté sur le robot est très proche de celle de la version originale de SupersurfelFusion. Elle est montrée sur la figure 5.1.

On peut noter l'ajout d'un bloc réalisant l'intégration des données issues des encodeurs des roues au niveau du module de localisation. Ces données parviennent à bien plus haute fréquence que les images RGB-D et leur intégration sert à initialiser le calcul de la pose du robot $\mathbf{T}_{\mathfrak{r}}^{\mathfrak{w}} \in SE(3)$ dans l'espace monde. $\mathfrak{r}$ désigne le référentiel du robot, situé au centre de sa base mobile. Les étapes de "Suivi Indirect de la Carte Locale" et d'"Affinement Direct de la Pose" permettent de calculer la pose $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}$ de la caméra dans l'espace monde. Pour passer de la pose de la caméra $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}}$ à celle du robot $\mathbf{T}_{\mathfrak{r}}^{\mathfrak{w}}$, il est nécessaire de connaître la transformation $\mathbf{T}_{\mathfrak{r}}^{\mathfrak{c}} \in SE(3)$ du référentiel $\mathfrak{r}$ du robot à celui de la caméra $\mathfrak{r}$. Cette transformation est déterminée à l'avance grâce à la méthode de calibrage de Guo, Mirzaei et Roumeliotis [GMR12]. On a alors :

$$\mathbf{T}_{\mathfrak{r}}^{\mathfrak{w}} = \mathbf{T}_{\mathfrak{c}}^{\mathfrak{w}} \mathbf{T}_{\mathfrak{r}}^{\mathfrak{c}} \tag{5.1}$$

Les différentes transformations et repères impliqués sont représentés sur le schéma 5.2.

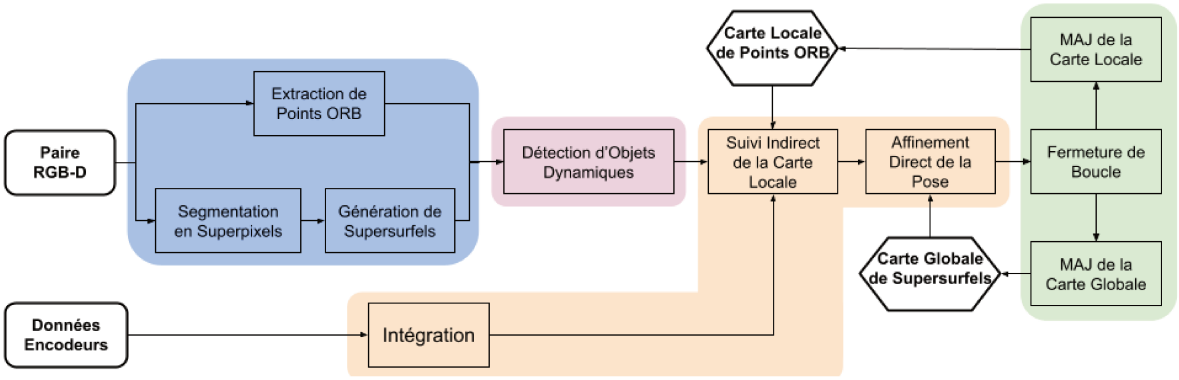

FIGURE 5.1 – L'architecture du portage de SupersurfelFusion sur un robot mobile à roues est toujours composée de quatre modules : l'extraction de primitives (bleu), la détection d'objets dynamiques (violet), la localisation (jaune) et la cartographie (vert). Les données mesurées par les encodeurs des roues du robot ont été incorporées dans le système de base au niveau de la localisation.

A l'initialisation du système de SLAM, les repères $\mathfrak{r}$ du robot et $\mathfrak{w}$ du monde sont confondus. Les mesures des encodeurs des roues et les images de la caméra RGB-D sont ensuite couplées pour localiser le robot de façon robuste et générer une carte globale $\mathcal{M}$ de la partie statique de l'environnement observé avec la représentation supersurfel. La carte $\mathcal{M}$ est dense et particulièrement compacte, ce qui rend son utilisation intéressante pour des tâches de navigation. On rappelle que SupersurfelFusion maintient aussi une carte locale éparse $\mathcal{L}$, constituée d'un nombre limité de points clés ORB détectés dans les dernières images. Elle est indépendante de la carte globale et n'est utilisée que pour la localisation. Les coordonnées géométriques des primitives contenues dans les deux cartes $\mathcal{M}$ et $\mathcal{L}$, sont toutes exprimées dans le repère du monde $\mathfrak{w}$.

Comme la fréquence d'acquisition des encodeurs est beaucoup plus élevée que celle de la caméra, les données des roues sont intégrées entre paires d'images RGB-D successives pour fournir une estimation initiale solide de la nouvelle pose du robot. Similairement à la stratégie d'odométrie visuelle suivie dans le chapitre précédent (section 4.3.4), les positions 3D des points d'intérêt de la carte locale $\mathcal{L}$ sont ensuite appariés aux points ORB détectés dans la vue courante de la caméra. La pose courante est alors calculée en minimisant conjointement les erreurs de reprojection des points d'intérêt et une erreur sur les mesures des encodeurs. Finalement, la pose calculée est affinée en recalant les données 3D de la vue courante avec les supersurfels de la carte globale $\mathcal{M}$ via la méthode de l'ICP point-plan symétrisé (cf. section 4.3.4.2).

5.2.3 Intégration des mesures des encodeurs

Les mesures des encodeurs sont intégrées entre la précédente paire RGB-D $\{C_{prec}, D_{prec}\}$ et la nouvelle paire $\{C, D\}$, en suivant les modèles de [SNS11] et [Yan+19], pour obtenir le déplacement du robot. L'intégration des mesures des encodeurs entre deux acquisitions

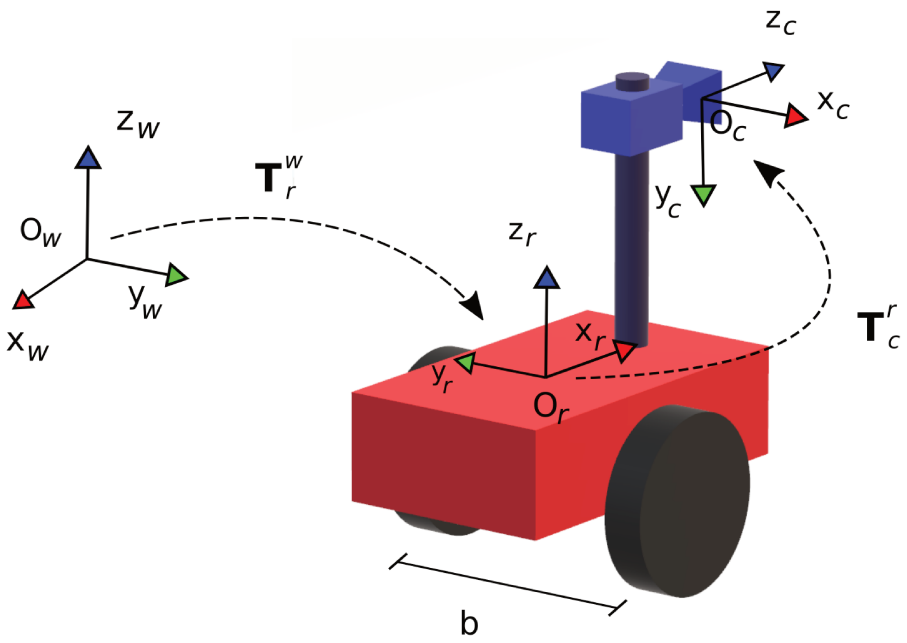

FIGURE 5.2 – Robot à conduite différentielle équipé d'une caméra (en bleu). La base du robot est schématisée en rouge et ses roues en noir. Le robot évolue dans l'espace monde $\mathfrak{w}$ et les transformations $\mathbf{T}_{\mathfrak{r}}^{\mathfrak{w}}$ et $\mathbf{T}_{\mathfrak{c}}^{\mathfrak{r}}$ représentent respectivement la pose du robot par rapport au repère monde $\mathfrak{w}$ et la pose de la caméra par rapport à la base mobile du robot $\mathfrak{r}$. La distance b indique la longueur de l'entraxe du robot.

d'images garantit que la distance parcourue est petite, donc que l'erreur accumulée lors du processus reste faible. A partir des encodeurs, on ne peut qu'extraire des informations 2D sur le déplacement du robot, décrivant son mouvement dans le plan $(O_{\mathfrak{r}}, x_{\mathfrak{r}}, y_{\mathfrak{r}})$ du repère $\mathbf{r}$ de sa base (voir figure 5.2). On exprime la pose du robot dans le plan par un vecteur de trois paramètres $\boldsymbol{\zeta}[x, y, \theta]^\intercal$, où $[x, y]^\intercal \in \mathbb{R}^2$ est la partie translation et $\theta \in [\![-\pi, \pi]\!]$ est l'angle autour de l'axe $(O_{\mathfrak{r}}, z_{\mathfrak{r}})$.

Soit $\boldsymbol{\zeta}_t[x_t, y_t, \theta_t]^\intercal$ la pose du robot dans le plan de sa base à un instant t d'acquisition de mesures d'encodeurs. La pose $\boldsymbol{\zeta}_{t+1}[x_{t+1}, y_{t+1}, \theta_{t+1}]^\intercal$ du robot dans le plan à l'instant $t+1$ suivant est donnée par :

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \Delta s \cdot \cos(\theta_t + 0.5\Delta\theta) \\ \Delta s \cdot \sin(\theta_t + 0.5\Delta\theta) \\ \Delta\theta \end{bmatrix}, \qquad \begin{cases} \Delta s &= \frac{\Delta s_d + \Delta s_g}{2} \\ \Delta\theta &= \frac{\Delta s_d - \Delta s_g}{b} \end{cases} \tag{5.2}$$

avec b l'écart entre les roues (entraxe), Δs la longueur de la translation et $\Delta\theta$ l'angle de rotation. Δs_g et Δs_d sont les déplacements respectifs des roues gauche et droite en mètre, calculés directement à partir du nombre de révolutions Δe_g, Δe_r, de chacune des roues depuis l'instant t :

$$\begin{cases} \Delta s_g &= k_g \cdot \Delta e_g + \delta_g, \quad \delta_g \sim \mathcal{N}(0, \|K \cdot k_g \cdot \Delta e_g\|^2) \\ \Delta s_d &= k_d \cdot \Delta e_d + \delta_d, \quad \delta_d \sim \mathcal{N}(0, \|K \cdot k_d \cdot \Delta e_d\|^2) \end{cases} \tag{5.3}$$

On pose comme hypothèse que Δs_g et Δs_d sont soumis à des bruits gaussiens centrés δ_g et δ_d, modélisant différentes perturbations comme les déformations des roues et les glissements. Ces bruits sont proportionnels au déplacement des roues d'un facteur K.

En considérant que la pose du robot dans le plan suit une loi normale et que la pose à l'instant t est décrite par $\boldsymbol{\zeta}_t \sim \mathcal{N}(\bar{\boldsymbol{\zeta}}_t, \boldsymbol{\Sigma}_{\boldsymbol{\zeta}_t})$, alors la covariance $\boldsymbol{\Sigma}_{\boldsymbol{\zeta}_{t+1}}$ de la pose à l'instant $t+1$ est donnée par :

$$\boldsymbol{\Sigma}_{\boldsymbol{\zeta}_{t+1}} = \mathbf{J}_{\boldsymbol{\zeta}} \boldsymbol{\Sigma}_{\boldsymbol{\zeta}_t} \mathbf{J}_{\boldsymbol{\zeta}}^{\intercal} + \mathbf{J}_S \boldsymbol{\Sigma}_S \mathbf{J}_S^{\intercal} \tag{5.4}$$

où $\mathbf{J}_{\boldsymbol{\zeta}}$ est la matrice jacobienne de l'équation 5.2 par rapport aux paramètres de la pose $\boldsymbol{\zeta}_t$, $\mathbf{J}_S$ est la matrice jacobienne de l'équation 5.2 par rapport au vecteur de paramètres $[\Delta s_g, \Delta s_d]^{\intercal}$ et $\boldsymbol{\Sigma}_S$ sa matrice de covariance, estimée de la manière suivante :

$$\boldsymbol{\Sigma}_S = \begin{bmatrix} \|K \cdot k_g \cdot \Delta e_g\|^2 & 0 \\ 0 & \|K \cdot k_d \cdot \Delta e_d\|^2 \end{bmatrix} \tag{5.5}$$

Les formules 5.2 et 5.5 sont appliquées itérativement entre deux vues RGB-D successives pour calculer le déplacement du robot $\boldsymbol{\zeta}[x, y, \theta]^{\intercal}$ dans le plan de sa base et son incertitude $\boldsymbol{\Sigma}_{\boldsymbol{\zeta}}$ dont la forme est la suivante :

$$\boldsymbol{\Sigma}_{\boldsymbol{\zeta}} = \begin{bmatrix} \sigma_x & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_y & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_\theta \end{bmatrix} \tag{5.6}$$

L'observation 2D du déplacement du robot par les encodeurs $\boldsymbol{\zeta}$ est alors utilisée pour obtenir une estimation initiale préalable $\hat{\mathbf{T}}_{\mathfrak{t}}^{\mathfrak{w}}$ de la pose courante du robot dans l'espace monde. L'initialisation d'une pose 3D directement à partir des mesures 2D des encodeurs est fiable dans notre cas car nous travaillons sur des robots dont la vitesse de déplacement est lente par rapport à la fréquence d'acquisition des images. De plus, ils évoluent dans des habitats humains généralement plats ou avec des pentes légères et continues.

Pour calculer l'estimation préalable $\hat{\mathbf{T}}_{\mathfrak{t}}^{\mathfrak{w}}$, on commence par convertir le déplacement $\boldsymbol{\zeta}[x, y, \theta]^{\intercal}$ en une matrice de transformation $\mathbf{T}_\zeta$ dans l'espace 3D référencé au repère de la base du robot :

$$\mathbf{T}_\zeta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & x \\ \sin(\theta) & \cos(\theta) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.7}$$

De même, la matrice de covariance $\boldsymbol{\Sigma}_{\boldsymbol{\zeta}}$, décrivant l'incertitude sur les paramètres du vecteur $\boldsymbol{\zeta}$ est convertie en une matrice $\boldsymbol{\Sigma}_{\mathbf{T}_\zeta}$ de dimensions 6×6, toujours dans le repère de la base du robot :

$$\boldsymbol{\Sigma}_{\mathbf{T}_\zeta} = \begin{bmatrix} \sigma_x & \sigma_{xy} & 0 & 0 & 0 & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_y & 0 & 0 & 0 & \sigma_{y\theta} \\ 0 & 0 & \epsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & \epsilon & 0 & 0 \\ 0 & 0 & 0 & 0 & \epsilon & 0 \\ \sigma_{\theta x} & \sigma_{\theta y} & 0 & 0 & 0 & \sigma_\theta \end{bmatrix} \tag{5.8}$$

avec ϵ une valeur petite fixée.

Un changement de repère est ensuite effectué sur la transformation relative $\mathbf{T}_\zeta$ et sa covariance $\mathbf{\Sigma}_{\mathbf{T}_\zeta}$, calculées initialement dans le repère $\mathfrak{r}$ de la base du robot, pour les exprimer dans le repère monde $\mathfrak{w}$. Une fois le changement de repère réalisé, l'estimation initiale de la nouvelle pose du robot $\hat{\mathbf{T}}_\mathfrak{r}^\mathfrak{w}$, basée seulement sur les données des roues, peut être calculée à partir de la valeur actuelle de la pose du robot $\mathbf{T}_\mathfrak{r}^\mathfrak{w}$, estimée durant l'acquisition de la paire RGB-D précédente :

$$\hat{\mathbf{T}}_\mathfrak{r}^\mathfrak{w} = \mathbf{T}_\mathfrak{r}^\mathfrak{w}\mathbf{T}_\zeta \tag{5.9}$$

5.2.4 Suivi de la carte locale

La prédiction $\hat{\mathbf{T}}_\mathfrak{r}^\mathfrak{w}$ de la nouvelle pose du robot donnée par l'odométrie des roues est ensuite utilisée pour initialiser la nouvelle valeur de $\mathbf{T}_\mathfrak{r}^\mathfrak{w}$ et calculer une estimation plus sûre à l'aide des données visuelles. Similairement à ce qui a été fait en section 4.3.4.1, les nouveaux points ORB statiques détectés sont utilisés pour aligner la nouvelle image RGB-D $\{C, D\}$ par rapport à la carte 3D locale $\mathcal{L}$ de points caractéristiques, construite à partir des points clés ORB statiques des images précédentes. La mise en correspondance entre les points 3D de la carte locale et les nouveaux points ORB 2D détectés est effectuée comme précédemment, en réalisant tout d'abord une mise en correspondance exhaustive (dite "brute-force") à partir des descripteurs binaires, puis en employant la méthode GMS (Grid Based Motion Statistics) [Bia+17].

Après la mise en correspondance des points d'intérêt ORB, la transformation $\mathbf{T}_\mathfrak{r}^\mathfrak{w}$, qui décrit la pose courante du robot dans le repère du monde, est calculée en minimisant un critère constitué par la somme des erreurs de reprojection E_{proj} avec une erreur E_{enc} basée sur la prédiction $\hat{\mathbf{T}}_\mathfrak{r}^\mathfrak{w}$ donnée par les mesures des encodeurs, comme proposé par DRE-SLAM [Yan+19] :

$${}^*\mathbf{T}_\mathfrak{r}^\mathfrak{w} = \underset{\mathbf{T}_\mathfrak{r}^\mathfrak{w}}{\operatorname{argmin}} \left(E_{enc} + E_{proj}\right) \tag{5.10}$$

L'erreur E_{enc} joue le rôle de terme de régularisation. Elle contraint la transformation optimisée à rester proche de la prédiction $\hat{\mathbf{T}}_\mathfrak{r}^\mathfrak{w}$. Sa formule est donnée ci-dessous :

$$E_{enc} = \hat{\boldsymbol{\xi}}^\intercal \mathbf{\Lambda}\, \boldsymbol{\xi} \tag{5.11}$$

avec $\boldsymbol{\xi}, \hat{\boldsymbol{\xi}} \in se(3)$, $\boldsymbol{\xi} = \log_{SE(3)}(\mathbf{T}_\mathfrak{r}^\mathfrak{w})$ la transformation optimisée et $\hat{\boldsymbol{\xi}} = \log_{SE(3)}(\hat{\mathbf{T}}_\mathfrak{r}^\mathfrak{w})$ la prédiction, toutes deux paramétrisées sous forme de vecteurs de dimension 6 en utilisant l'algèbre de Lie [Bla10]. $\log_{SE(3)}(.)$ est l'application logarithme qui envoie le groupe de Lie $SE(3)$ vers son espace tangent, l'algèbre de Lie $se(3)$, localement Euclidien. $\mathbf{\Lambda}$ est la matrice d'information calculée comme l'inverse de l'incertitude $\mathbf{\Sigma}_{\mathbf{T}_\zeta}$ sur la pose prédite par l'odométrie des roues : $\mathbf{\Lambda} = {\mathbf{\Sigma}_{\mathbf{T}_\zeta}}^{-1}$. L'erreur de reprojection E_{proj} pour l'ensemble des paires 2D-3D $\{\mathbf{u}_j, \mathbf{p}_j\}$ est :

$$E_{proj} = \sum_j \rho(\|\mathbf{u}_j - \pi((\mathbf{T}_\mathfrak{r}^\mathfrak{w}\mathbf{T}_\mathfrak{c}^\mathfrak{r})^{-1}\mathbf{p}_j)\|^2) \tag{5.12}$$

avec $\pi(.)$ le modèle de projection pinhole, $\rho(.)$ le M-estimateur de Huber et $\mathbf{T}_\mathfrak{c}^\mathfrak{r}$ la pose de la caméra par rapport à la base mobile du robot.

Finalement la pose est affinée par recalage direct entre les supersurfels de la carte globale $\mathcal{M}$ et les données de la vue courante en appliquant le même processus que celui décrit en section 4.3.4.2.

5.3 Couplage de SupersurfelFusion avec une solution de navigation

Cette section débute avec une revue des cartes navigables produites par les méthodes de SLAM RGB-D de l'état de l'art. Nous expliquons ensuite comment nous exploitons la méthode de SLAM SupersurfelFusion portée sur le robot, pour réaliser des tâches de navigation autonome dans un environnement intérieur inconnu. SupersurfelFusion produit une localisation robuste et une modélisation 3D dense, compacte et rectifiable en ligne du voisinage du robot, qui peut être transmise presque directement à un algorithme de planification de trajectoire. Seule une petite partie de la surface entourant le robot a besoin d'être cartographiée pour permettre la navigation autonome basée sur le système SupersurfelFusion développé, sans nécessiter l'utilisation d'une carte préconstruite. L'algorithme de SLAM développé est couplé à un système de navigation préexistant qui est modifié de façon à pouvoir utiliser la représentation supersurfel.

5.3.1 Carte navigable basée SLAM RGB-D

Les systèmes de SLAM RGB-D dense de la littérature produisent des modèles 3D qui ne sont généralement pas appropriés pour des tâches de navigation. Les reconstructions surfel et TSDF de [Whe+15] et [Whe+12] sont par exemples trop précises, coûteuses et inefficaces pour servir telles quelles à des opérations de planification. Dans la plupart des cas, des cartes navigables sont générées a posteriori à partir des surfaces 3D reconstruites par des algorithmes de SLAM, puis utilisées dans un second temps par un système de navigation. Une représentation fréquente pour la navigation de véhicules terrestres est la grille d'occupation 2D [LHS14], qui peut être obtenue en projetant les données de profondeur d'une caméra RGB-D sur le sol, ou en ne considérant qu'une tranche d'un nuage de points [Nar+19]. Ces cartes sont efficaces et légères mais ne permettent de naviguer que dans des environnements plats et simples. Elles ne prennent pas en compte les pentes, les surfaces suspendues ou les trous (descente d'escalier par exemple). PUTSLAM [Bel+16] couple un système de SLAM RGB-D éparse, utilisé spécifiquement pour la localisation, à un algorithme de reconstruction 3D dense qui extrait les nuages de points 3D d'images clés et les intègre dans une grille d'élévation (représentation 2.5D). La grille d'élévation permet de naviguer sur des surfaces qui ne sont pas planes mais n'est pas appropriée pour la modélisation des surfaces surplombantes. En effet, elle est amenée à considérer les espaces vides sous-jacents comme occupés.

Les cartes 3D permettent une représentation plus complète des environnements. La plupart des approches de génération de cartes 3D navigables s'appuient sur des représentation volumétriques comme la grille d'occupation 3D du framework Octomap [Hor+13] ou comme le champ

3D de distances euclidiennes signées (3D ESDF pour 3D Euclidean Signed Distance Fiels) de Voxblox [Ole+17]. Octomap utilise une structure d'octree hiérachique pour stocker efficacement des probabilités d'occupation dans des voxels. La grille d'occupation 3D d'Octomap est multi-résolution. Elle peut être mise à jour incrémentalement et étendue en convertissant et en intégrant les nuages de points 3D issus de nouvelles vues RGB-D. L'insertion de nouvelles données et les requêtes requièrent la traversée d'un arbre. Voxblox emploie un table de hachage pour faciliter l'intégration de nouvelles mesures. Il génère une carte navigable à partir d'une reconstruction TSDF. D'autres approches comme [LS17] ; [Sad+14] proposent d'extraire la surface navigable sous forme de mesh (maillage) en triangulant les amers 3D d'une approche de SLAM visuel éparse. Elles supposent toutefois que suffisamment d'amers aient été détectés, ce qui peut s'avérer faux dans les zones pauvres en texture. De plus, la mise à jour incrémentale d'un mesh est une tâche ardue puisqu'elle nécessite la réorganisation de ses arêtes et de ses sommets. Des exemples de cartes navigables basées sur l'utilisation de méthodes de SLAM visuel sont montrés sur la figure 5.3.

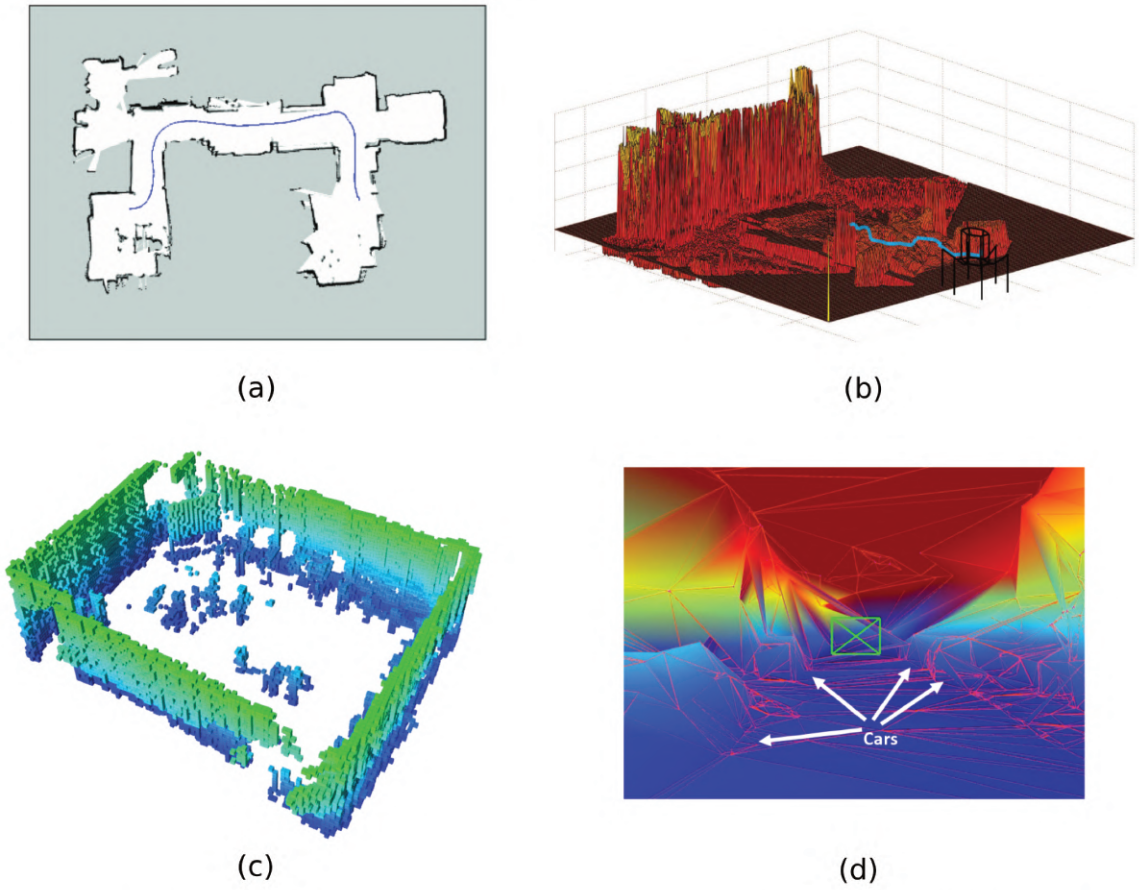

FIGURE 5.3 – Exemples de cartes navigables couramment utilisées : (a) grille d'occupation 2D [Nar+19], (b) carte d'élévation [Bel+16], (c) octomap [Yan+19], (d) mesh [LS17].

RTAB-MAP [LM19] et RGBDSLAMv2 [End+14] intègrent la librairie Octomap pour permettre la construction incrémentale de cartes 3D denses navigables en utilisant les poses et

les nuages de points 3D d'images clés, calculés par une stratégie de localisation éparse. Leur utilisation pour des applications de navigation autonome et de reconstruction simultanée est cependant limitée. En effet, ils ne permettent pas de rectifier la carte navigable reconstruite en cas de fermeture de boucle. DRE-SLAM [Yan+19] décompose la carte représentée sous forme d'octomap en sous-ensembles déformables les uns par rapport aux autres, pour rendre possible l'application à grande échelle des corrections lors de la détection d'une fermeture de boucle. La mise à jour de la carte navigable est néanmoins réalisée à une fréquence bien inférieure à celle de la localisation, car la conversion d'un nuage de points brut extrait d'une image clé en une grille d'occupation 3D implique des opérations de lancer de rayons sur chacun des pixels de l'image. Pour remédier à la nécessité de convertir les nuages de points reconstruits par les méthodes de cartographie LiDAR 3D en cartes navigables basées sur des représentations volumétriques, [Rue+19] ; [Krü+17] ; [PH+20] mettent en avant des systèmes de navigation capables de planifier des trajectoires directement sur des nuages de points. Nous nous appuyons sur ces dernières approches pour permettre la navigation et la cartographie simultanées avec notre représentation supersurfel.

5.3.2 Présentation du système de navigation

Nous nous appuyons sur le code libre du système de navigation de Pérez-Higueras et al. [PH+20] pour la planification de mouvement et la génération de commandes de contrôle. Il permet de générer des trajectoires de manière autonome à partir de positions d'arrivées données ou encore de réaliser de l'exploration autonome basée sur la détection de frontières. Le but principal de la planification de trajectoires est de calculer dans l'espace de travail du robot un chemin libre d'obstacle entre une position initiale et une position de destination du robot. L'intérêt du système de Pérez-Higueras et al. pour la cartographie et la navigation simultanée est qu'il effectue des calculs de trajectoires directement sur des nuages de points, qui peuvent être déformés pour être corrigés en ligne en cas de fermetures de boucles, au contraire des cartes volumétriques. L'algorithme prend en entrée un nuage de points local, filtré et centré sur le robot, extrait de la carte globale d'un système de SLAM externe. Ce nuage de points est séparé en deux (cf. 5.3.3) : un nuage contenant les obstacles et un autre représentant la surface navigable. Le nuage de points contenant la surface navigable est utilisé comme espace d'échantillonnage d'un planificateur de trajectoires de type Optimal-RRT [KF11](RRT* où RRT est l'acronyme de Rapidly Exploring Random Tree), tandis que celui comportant les obstacles est employé pour la détection de collisions. La surface couverte par le nuage local de points est fixée par rapport à la portée maximale du capteur employé. Le fait de n'utiliser que la surface locale autour du robot, au lieu de tout l'espace 3D dans sa zone d'action, sert à accélérer les calculs en ne cherchant un chemin que sur les points du nuage représentant la surface navigable.

L'approche RRT* est une version optimisée de l'algorithme de planification de trajectoires RRT [Lav98]. Ce dernier est une méthode de planification probabiliste, qui consiste à construire un arbre recouvrant l'espace libre en partant de la position initiale $\mathbf{p}_{start}$ du robot et à explorer les différentes configurations possibles pour trouver un chemin rejoignant le but $\mathbf{p}_{goal}$. La racine de l'arbre est la position initiale $\mathbf{p}_{start}$ du robot et les noeuds sont des positions possibles,

c'est-à-dire sans collision, du robot dans la carte. Les branches de l'arbre sont des commandes à appliquer pour passer d'une configuration à l'autre. L'arbre est étendu itérativement en choisissant au hasard des points dans l'espace navigable avec une méthode de tirage aléatoire uniforme. Pour chaque point $\mathbf{p}_{rand}$ tiré aléatoirement, on vérifie à partir de la carte que le robot n'entre pas en collision avec des obstacles dans cette configuration. Puis on trace un chemin de la position voisine $\mathbf{p}_{near}$ la plus proche dans les noeuds de l'arbre en direction de la position aléatoire $\mathbf{p}_{rand}$, en se déplaçant au maximum d'un pas Δd. En effet, si $\mathbf{p}_{rand}$ est à une distance inférieure on s'arrête avant. L'incrément Δd peut aussi être choisi dynamiquement lors de l'exécution. Par exemple, on peut sélectionner un pas grand lorsque le robot est loin des obstacles et le réduire lorsqu'il en est proche. Si la configuration $\mathbf{p}_{new}$ obtenue à la distance Δd et le chemin qui la relie à $\mathbf{p}_{near}$ ne sont pas obstrués, elle est ajoutée aux noeuds de l'arbre et reliée au noeud parent $\mathbf{p}_{near}$ par une branche. Ce processus est répété jusqu'à ce qu'un noeud suffisamment proche du but soit généré ou qu'un nombre d'itérations maximal ait été dépassé. Pour éviter les cycles, deux éléments de l'arbre ne peuvent pas être identiques. Le principe de base de l'algorithme RRT est donné ci-dessous et illustré sur la figure 5.4 :

Algorithme 2 RRT

Entrées: point de départ $\mathbf{p}_{start}$, destination $\mathbf{p}_{goal}$, pas Δd
Sorties: trajectoire Γ

ajouter le noeud $\mathbf{p}_{start}$ à l'arbre $\mathcal{A}$
$\mathbf{p}_{new} \leftarrow \mathbf{p}_{start}$
tant que DISTANCE($\mathbf{p}_{new}, \mathbf{p}_{goal}$) $> d_{threshold}$ **faire**
  $\mathbf{p}_{rand} \leftarrow$ échantillonnage aléatoire dans l'espace de navigation
  $\mathbf{p}_{near} \leftarrow$ recherche du plus proche voisin dans les noeuds de l'arbre $\mathcal{A}$
  $\mathbf{p}_{new} \leftarrow$ point à la distance Δd de $\mathbf{p}_{near}$ dans la direction de $\mathbf{p}_{rand}$
  si COLLISION($\mathbf{p}_{new}$) == NULL **alors**
    définir $\mathbf{p}_{near}$ comme parent de $\mathbf{p}_{new}$
    ajouter $\mathbf{p}_{new}$ à l'arbre $\mathcal{A}$
  fin si
fin tant que
$\Gamma \leftarrow$ concaténer les points de $\mathbf{p}_{new}$ à $\mathbf{p}_{start}$

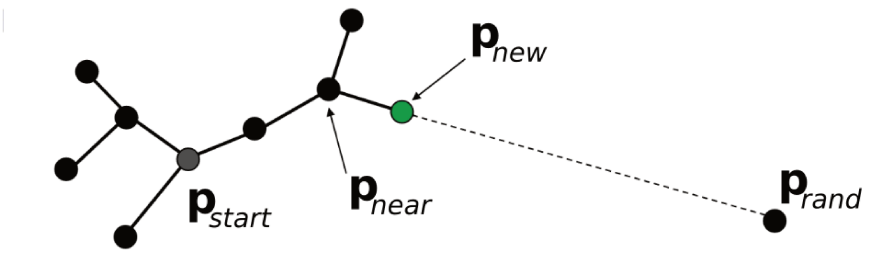

Figure 5.4 – Illustration d'une itération de l'algorithme RRT.

La méthode RRT est simple et efficace mais donne dans la plupart des cas des chemins sous-optimaux. Pour pallier ce problème, l'algorithme RRT* utilise une fonction de coût permettant d'évaluer l'optimalité de la trajectoire. Il garde en mémoire une valeur de coût pour chaque noeud, par exemple basée sur la distance à son parent, et supprime certaines branches de l'arbre si des branches de coût inférieur proches sont découvertes. Après avoir trouvé le noeud de l'arbre le plus proche $\mathbf{p}_{near}$ et placé un nouveau noeud $\mathbf{p}_{new}$, les noeuds compris dans un certain rayon de $\mathbf{p}_{new}$ sont examinés. Si un noeud donnant un coût plus faible que celui donné par $\mathbf{p}_{near}$ est trouvé, alors il remplace $\mathbf{p}_{near}$ comme parent. Un autre processus est aussi mis en place dans le but de modifier le branchage de l'arbre suite à l'insertion d'un noeud. Le coût total de la trajectoire est la somme des coûts de ses noeuds. La méthode RRT* converge vers un optimum lorsque le nombre de noeuds tend vers l'infini, au contraire de l'approche RRT d'origine qui donne des trajectoires plus erratiques (cf. figure 5.5).

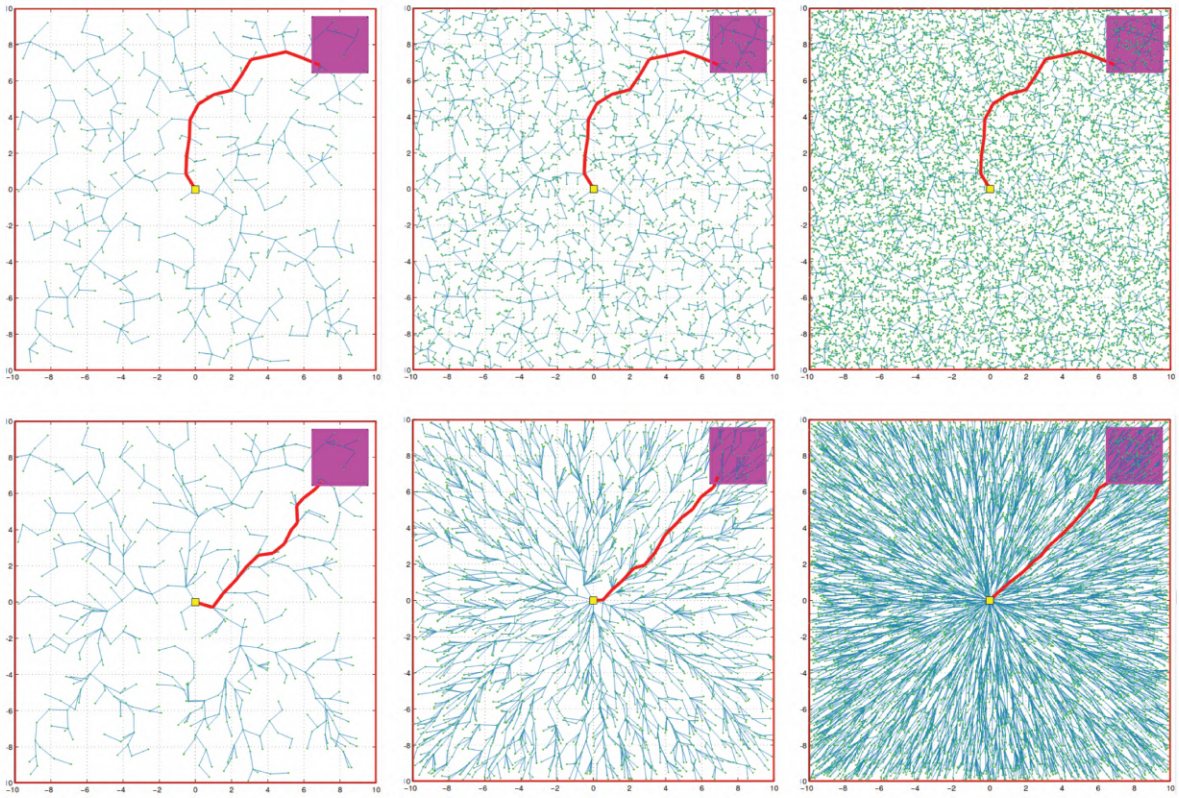

FIGURE 5.5 – Comparaison des trajectoires données par les approches RRT (en haut) et RRT* (en bas) sur un exemple de simulation sans obstacles [KF11], en augmentant le nombre de noeuds de l'arbre d'exploration. Le point jaune est la position de départ et le carré violet la zone d'arrivée.

Dans le cas du système de navigation que nous employons, le coût $c(\mathbf{p})$ associé à un noeud $\mathbf{p}$ est calculé comme la combinaison linéaire de N critères :

$$c(\mathbf{p}) = \sum_{k=1}^{N} w_k f_k(\mathbf{p}) \tag{5.13}$$

où les poids w_k sont normalisés. Les critères f_k prennent en compte les informations suivantes :

- l'inclinaison de la surface en $\mathbf{p}$ par rapport à l'inclinaison de la base du robot et sa rugosité, le but étant de favoriser les zones planes de l'environnement.
- le nombre de primitives du nuage de points 3D représentant la surface navigable contenue dans une boîte englobante de la taille du robot centrée en $\mathbf{p}$. Plus il y a de points et plus la représentation de la surface environnante peut être considérée comme précise, donc sûre.
- la distance entre le point échantillonné $\mathbf{p}$ et la moyenne de l'ensemble de points contenus dans la boîte englobante. Si la valeur calculée est grande cela peut vouloir dire que certaines parties de la surface à l'intérieur de la boîte englobante sont très pauvrement représentées.
- l'écart type de l'ensemble de points contenus dans la boîte englobante, qui permet d'indiquer la dispersion des points dans la boîte englobante. Si les points sont dispersés, la probabilité d'avoir des trous dans la surface considérée est généralement plus faible.
- la distance de $\mathbf{p}$ par rapport au point de destination, afin de réduire la longueur du chemin calculé.

5.3.3 Extraction de la surface navigable

Le système de navigation [PH+20] sur lequel nous nous basons fonctionne usuellement à partir d'un nuage de points 3D, produit par un algorithme de SLAM RGB-D pour représenter l'environnement. En plus de devoir recadrer le nuage global pour n'extraire que les points contenus dans un rayon correspondant à la portée maximale du capteur utilisé, il doit réaliser de nombreuses autres opérations de prétraitement sur la surface locale extraite avant de pouvoir s'en servir pour la planification des mouvements du robot. Des stratégies de filtrage et de sous-échantillonnage basées sur la construction d'une grille de voxels sont appliquées afin d'obtenir une densité de points appropriée pour une planification rapide mais sans danger. Les normales des points sont aussi calculées à partir de la décomposition en valeurs singulières de la matrice de covariance des points contenus dans chaque voxel, pour réaliser l'analyse de la traversabilité. Afin de tirer profit des avantages de la représentation supersurfel, nous avons modifié le système de navigation [PH+20] en supprimant ses modules de prétraitement. En effet, l'utilisation de la carte de supersurfels par le système de navigation est avantageuse car elle est compacte, rapide à manipuler, contient peu de primitives redondantes et encode directement les normales de la surface comme attributs des supersurfels. Par conséquent, nous pouvons nous passer des différentes étapes de prétraitement et analyser directement la traversabilité de la surface locale prélevée. De plus, une densité de reconstruction adéquate pour la navigation peut être implicitement définie par la valeur de la taille initiale des superpixels exploités pour générer les supersurfels (cf. section 3.4.1).

L'algorithme de SLAM développé est couplé au système de navigation [PH+20] selon l'architecture représentée sur la figure 5.6. La carte globale $\mathcal{M}$ de SupersurfelFusion est mise à jour pour chaque nouvelle paire RGB-D reçue puis transmise à un module d'extraction de la surface locale centrée sur la base du robot. Les coordonnées des supersurfels de la surface

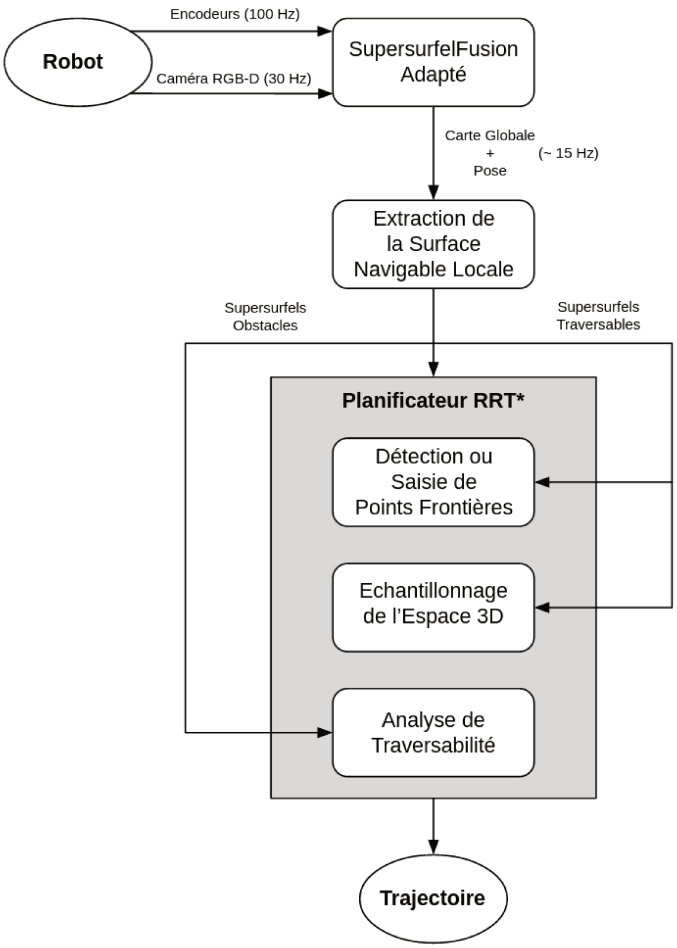

FIGURE 5.6 – Diagramme du système de cartographie, localisation et navigation simultanées. Les fréquences affichées correspondent à celles obtenues sur la plateforme de test présentée dans la section suivante 5.4.

locale extraite initialement dans le repère monde $\mathfrak{w}$ sont exprimées dans le repère $\mathfrak{r}$ de la base du robot, puis les supersurfels sont séparés en deux ensembles $\mathcal{O}$ et $\mathcal{T}$. Les supersurfels traversables sont regroupés dans l'ensemble $\mathcal{T}$ et leurs centres sont utilisés comme espace d'échantillonnage et de navigation pour le planificateur de trajectoires RRT*. L'ensemble $\mathcal{O}$ contient les supersurfels représentant les obstacles, qui sont passés au système de navigation et utilisés pour l'analyse de traversabilité. Pour chaque échantillon tiré aléatoirement dans l'ensemble $\mathcal{T}$ par le planificateur, les supersurfels de l'ensemble $\mathcal{O}$ à l'intérieur d'une boîte englobante aux dimensions du robot sont examinés pour la détection de collisions. La procédure de classification des supersurfels traversables et des obstacles entourant le robot est rapide et directe. Elle repose sur l'analyse de l'angle δ entre la normale d'un supersurfel et l'axe Z du robot, ainsi que sur la hauteur h du supersurfel dans le référentiel du robot $\mathfrak{r}$. Un supersurfel est traversable si $|\delta| \leqslant \delta_{max}$ et $h \leqslant h_{max}$, où les seuils δ_{max} et h_{max} sont fixés par rapport aux capacités motrices de la plateforme robotique mobile employée. Un exemple de navigation est montré sur la figure 5.7.

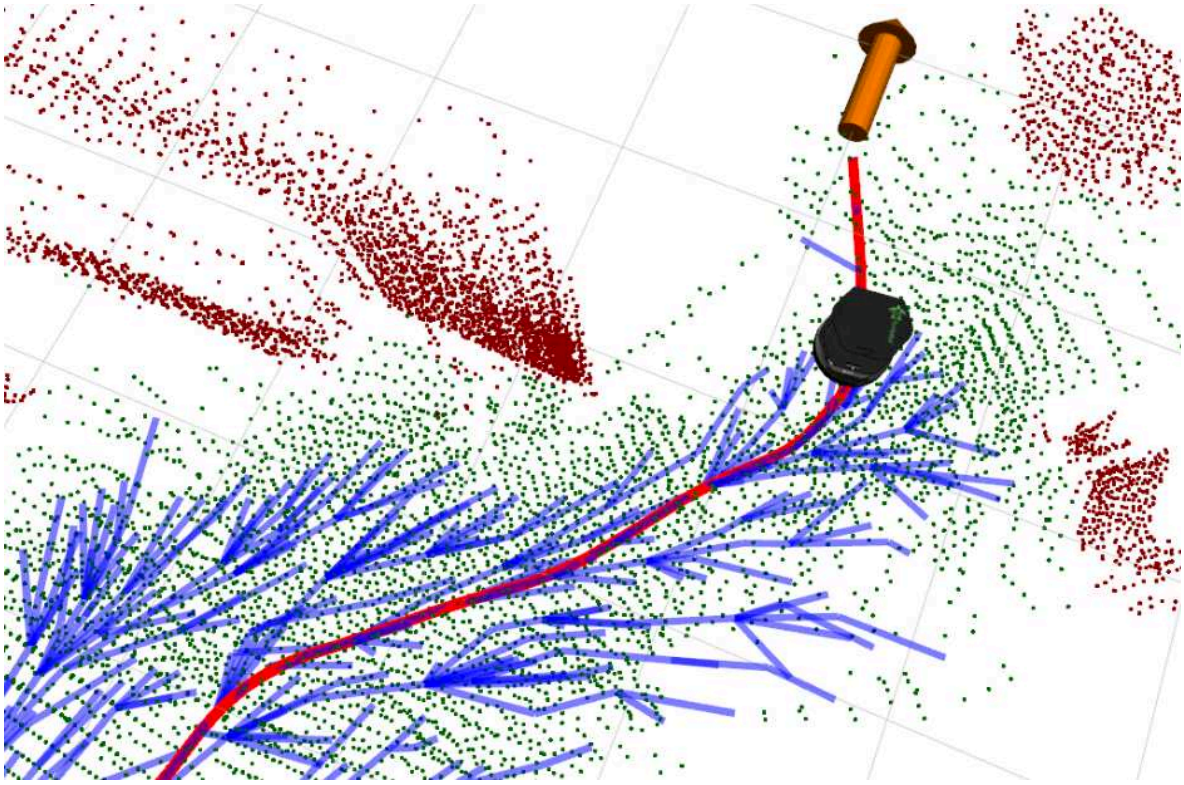

FIGURE 5.7 – Exemple de navigation autonome réalisée en combinant SupersurfelFusion au système de navigation modifié. Les points verts sont les centres des supersurfels traversables, utilisés comme espace d'échantillonnage par le planificateur de trajectoires. Les points rouges sont les obstacles. Les lignes bleues représentent les branches de l'arbre construit par la méthode RRT*, la flèche orange indique la position de destination et le chemin planifié est indiqué en rouge.

5.4 Évaluation

Le système de SLAM SupersurfelFusion porté sur le robot est implémenté en C++ et CUDA sous ROS (Robot Operating System) [Sta18]. Le système de navigation fonctionne aussi sous ROS et est implémenté en C++. ROS est un méta-système d'exploitation pour la robotique de service, dont l'un des avantages principaux est de faciliter l'intégration et l'interaction de différents modules logiciels et matériels pour le développement d'applications robotiques. Les deux systèmes, de navigation et de SLAM, sont exécutés sur la carte embarquée Nvidia Jetson AGX Xavier de notre plateforme de test, qui consiste en un robot terreste à conduite différentielle, équipé d'une caméra RGB-D Intel Realsense D435 et de deux roues encodeuses. Les mesures des encodeurs sont données à une fréquence de 100 Hz et les images de la caméra à 30 Hz avec des dimensions de 640×480. Le robot utilisé est exposé sur la photo 5.8. Il se déplace à une vitesse d'environ 4 m/s.

L'efficacité et la précision de l'algorithme de SLAM ont été évaluées sur le jeu de données de DRE-SLAM [Yan+19] afin de pouvoir comparer les résultats à l'état de l'art. L'évaluation

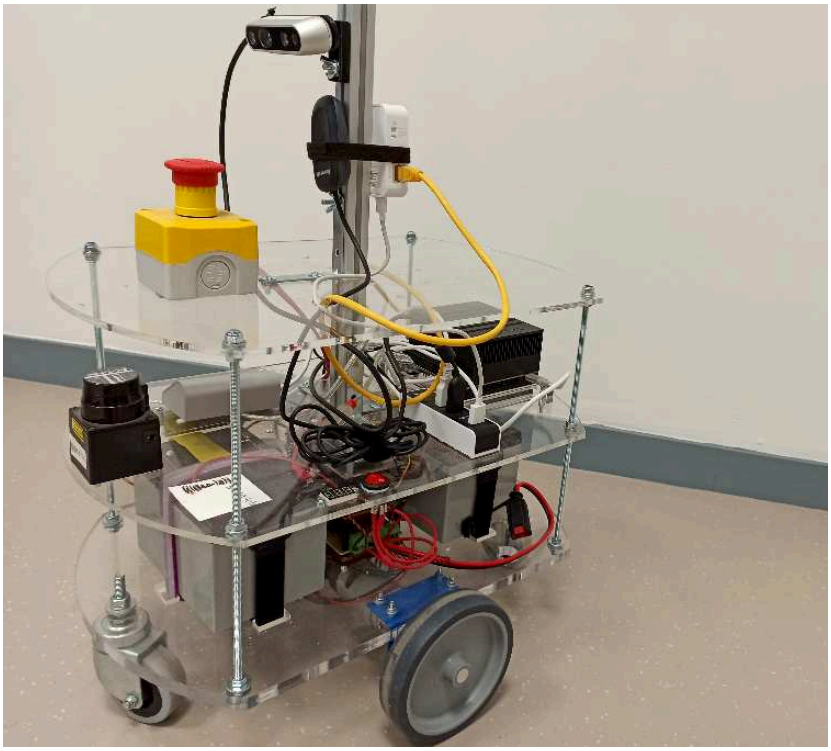

FIGURE 5.8 – La plateforme robotique mobile de test.

de la localisation d'une méthode de SLAM à partir de données réelles est toujours délicate car elle nécessite l'utilisation d'un système de capture de mouvement précis extérieur au robot pour fournir une vérité terrain de ses déplacements. Le jeu de données de DRE-SLAM est l'un des seuls que nous avons trouvé qui soit construit pour l'évaluation d'algorithmes fonctionnant grâce aux images RGB-D et aux mesures d'encodeurs d'une plateforme robotique mobile. Il est composées de séquences vidéos acquises par une caméra RGB-D Kinect V2 montée sur un robot à conduite différentielle, téléopéré à travers différents environnements intérieurs, statiques ou dynamiques. Les trajectoires vérités terrain, les paramètres de calibrage, les images RGB-D et les mesures des roues encodeuses sont données dans chacune des séquences. Trois séquences ont été utilisées : une séquence en milieu statique ST2 et deux séquences HD1, HD2, dans des scènes hautement dynamiques, où trois personnes se déplacent constamment en passant souvent devant la caméra. Les expérimentations menées à partir des séquences du jeu de données de DRE-SLAM ont aussi été exécutées sur la carte embarquée Nvidia. Nous avons aussi réalisé une expérience en ligne à partir des informations fournies par les capteurs de la plateforme robotique de test, dans le but de tester la méthode de cartographie et navigation simultanées proposée. Pour accélérer les calculs au niveau de la visualisation des données 3D, les supersurfels sont affichés sous forme de patchs rectangulaires plutôt qu'elliptiques.

5.4.1 Précision de la localisation

Cette section présente les résultats des expérimentations réalisées dans le but d'évaluer la précision du suivi du déplacement du robot sur les trois séquences ST2, HD1 et HD2 du jeu de données de DRE-SLAM. Nous utilisons la même métrique qu'en section 4.4.2 pour quantifier l'erreur sur chacune des trajectoires estimées : les poses de la trajectoire estimée et celles de la vérité terrain sont associées à partir d'horodatages, puis les deux trajectoires sont alignées en se basant sur les appariements par décomposition en valeurs singulières. La différence entre

Séquence	DRE-SLAM	Co-Fusion	StaticFusion	SupersurfelFusion
ST2	**1.2**	17.3	24.3	5.9
HD1	**1.4**	74.7	476.8	8.4
HD2	**2.8**	77.4	223.5	6.1

TABLE 5.1 – Erreurs quadratiques moyennes de position (cm) des trajectoires estimées.

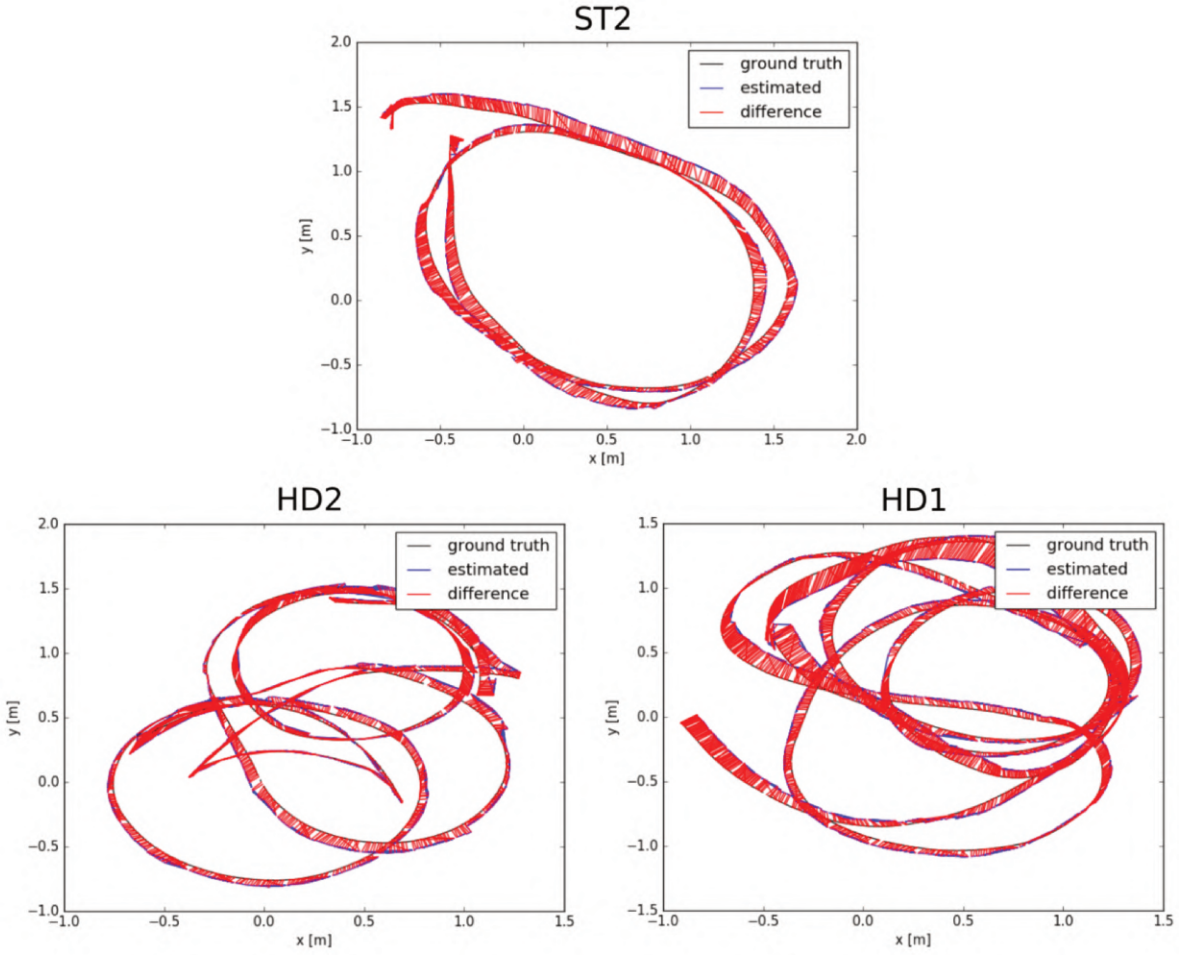

FIGURE 5.9 – Graphiques des trajectoires estimées comparées aux vérités terrains sur les différentes séquences d'évaluation.

chaque paire de poses est alors calculée et on affiche l'erreur quadratique moyenne de ces différences. Les mesures sont répertoriées dans le tableau 5.1 et les trajectoires tracées sur la figure 5.9. Les résultats des méthodes robustes aux scènes dynamiques DRE-SLAM [Yan+19], Co-Fusion [RA17] et StaticFusion [Sco+18] ont aussi été ajoutés pour la comparaison. DRE-SLAM est un système de SLAM développé spécifiquement pour un robot terrestre à conduite différentielle. Il est basé sur une localisation éparse fusionnant les données capteurs d'une caméra RGB-D et d'encodeurs. Co-Fusion et StaticFusion sont deux méthodes de SLAM purement RGB-D, qui s'appuient sur un recalage direct "frame-to-model" pour suivre la pose de la caméra.

Les valeurs affichées dans le tableau 5.1 démontrent la précision de l'approche proposée et sa robustesse dans les milieux statiques et dynamiques. Un exemple de cartographie 3D cohérente obtenue en incorporant les données des roues encodeuses dans SupersurfelFusion et de détection de personnes en mouvement sur la séquence HD2 est montré en figure 5.10. Notre système fonctionne bien mieux que les approches directes Co-Fusion et StaticFusion mais reste inférieur à DRE-SLAM au niveau de la précision du suivi de la pose du robot. Co-Fusion et StaticFusion produisent les moins bonnes performances. En effet, ils perdent parfois

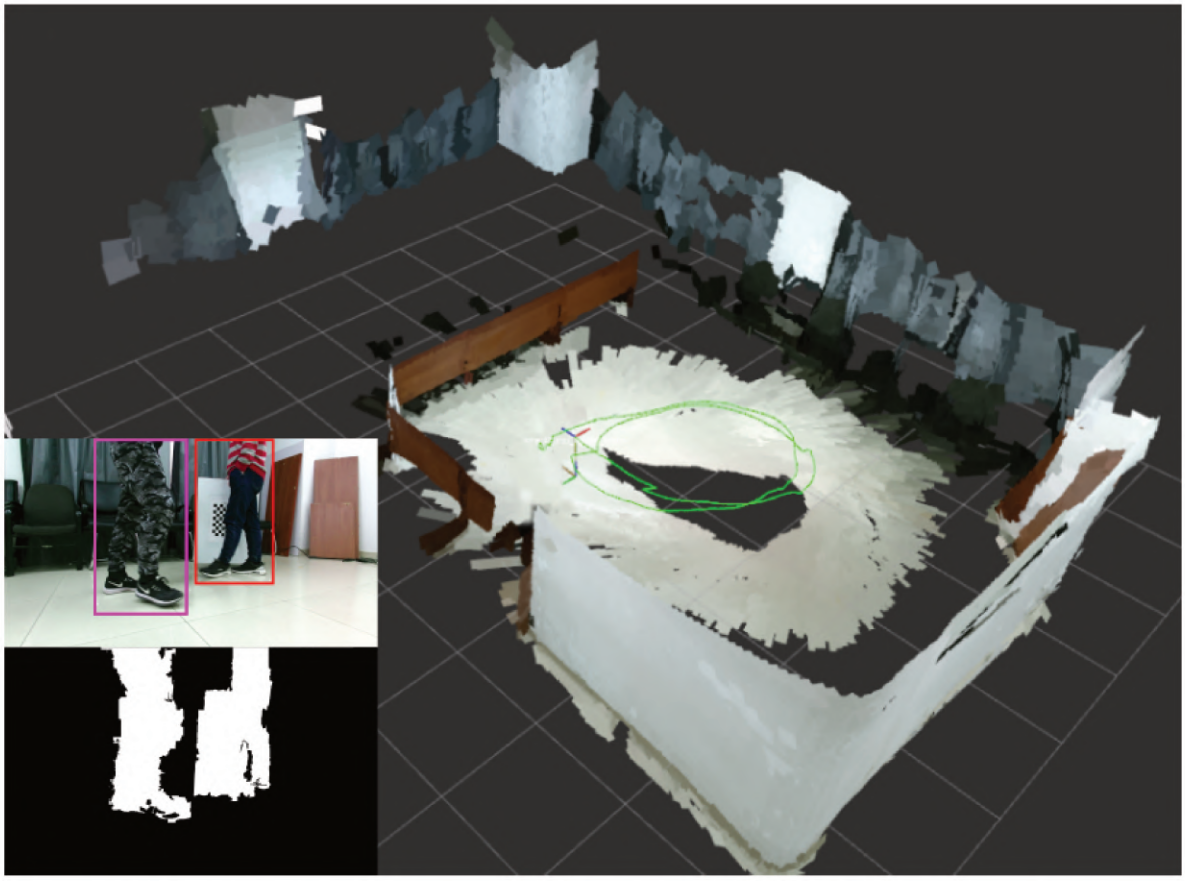

FIGURE 5.10 – Résultat obtenu sur la séquence HD2. La partie statique de l'environnement est reconstruite correctement sans être perturbée par les personnes en mouvement détectées.

complètement la localisation à cause du manque d'informations géométriques et de textures, quand la caméra fait face à un mur blanc par exemple, quand la surface filmée est trop éloignée ou encore quand la scène est occupée par trop d'éléments dynamiques. Au contraire, l'utilisation de l'odométrie des roues du robot permet à DRE-SLAM et à notre solution d'éviter les situations d'échec, en donnant une estimation suffisamment fiable de la pose du robot lorsque l'odométrie visuelle ne fonctionne pas. DRE-SLAM atteint la meilleure précision sur les trois séquences. En dehors de son design remarquable, nous pensons que cela peut aussi être partiellement expliqué par le fait que cet algorithme estime le mouvement du robot en 2D, représentant la pose avec seulement trois degrés de liberté. A la différence de DRE-SLAM, nous paramétrons le mouvement du robot dans l'espace 3D par une transformation à six degrés de liberté, afin de prendre en compte les sols qui ne sont pas complètement plats. Bien que cela puisse entraîner une légère perte de précision, notre méthode est aussi applicable dans des environnements plus complexes. De plus, DRE-SLAM applique aussi en continu des techniques d'optimisation locales et globales, précises mais coûteuses, pour produire une carte éparse cohérente ensuite convertie en octomap. Ces techniques améliorent la précision de la localisation mais ralentissent considérablement l'expansion de l'octomap, compliquant l'application de cette méthode de SLAM pour de la navigation conjointe.

5.4.2 Performances de calculs

Le coût calculatoire de l'algorithme de SLAM développé sur la carte embarquée Nvidia Jetson AGX Xavier de la plateforme robotique a été évalué via la séquence HD2 du jeu de données contenant plusieurs fermetures de boucles et des éléments dynamiques. Les temps moyens d'exécution des parties assurant la cartographie et la localisation (incluant la détection d'objets en mouvement) ont été mesurés, de même que la durée moyenne d'une fermeture de boucle (de la détection à l'application de la correction sur la carte globale). Nous avons aussi quantifié la vitesse moyenne d'extraction de la surface navigable passée au planificateur RRT* décrit dans la section 5.3.2, ainsi que l'empreinte mémoire de la carte reconstruite. Les différentes mesures sont reportées dans le tableau 5.2.

Processus	**Temps (moyenne ± écart type ms)**
Localisation & Cartographie	67.833 ± 7.024
Fermeture de Boucle	85.807
Extraction de la Surface Nav.	1.932 ± 0.495
Taille Max de la Carte Globale	13293 (1.318 Mo)
Nb Fermetures de Boucle	2

Table 5.2 – Statistiques computationnelles sur HD2.

L'algorithme de SLAM est rapide, capable de fonctionner à presque 15 Hz sur la carte embarquée. Deux fermetures de boucles ont été détectées et appliquées en une durée moyenne de 85 ms, que nous considérons suffisante dans le cadre d'une application en ligne, puisqu'il s'agit de phénomènes rares et ponctuels. Par ailleurs, pendant la durée de l'application d'une

fermeture de boucle, le déplacement du robot est toujours pris en compte par les encodeurs. Plus de fermetures de boucles auraient pu être détectées sur la séquence HD2, mais les éléments dynamiques ont perturbé la reconnaissance de lieu. Le temps nécessaire à l'extraction de la carte navigable des supersurfels traversables et intraversables est négligeable, variant de 1.5 à 2 ms, ce qui valide la pertinence de l'emploi de notre approche pour la planification de trajectoire pendant la cartographie. En comparaison, DRE-SLAM prend environ 50 ms pour extraire une octomap à partir de 5 images clés (couvrant donc une surface réduite), avec une résolution de voxels de 0.1 m. La carte globale de supersurfels reconstruite (visible en figure 5.10) occupe seulement 1.3 Mo, ce qui est moins que les 7 Mo requis pour stocker le nuage de points colorés d'une seule paire RGB-D de résolution 640×480.

5.4.3 Expérience de navigation autonome

Nous avons testé notre méthode de SLAM dans le cadre d'une application de navigation autonome, visant à déplacer le robot dans les couloirs de notre laboratoire, en la couplant avec le système de navigation présenté en 5.3, inspiré de Pérez-Higueras et al. [PH+20]. L'algorithme de SLAM SupersurfelFusion et le système de navigation ont tous les deux été portés et exécutés simultanément sur la carte embarquée du robot. L'expérience a été menée dans un environnement complètement statique car l'approche de navigation de Pérez-Higueras et al. n'a pas été conçue pour gérer les obstacles dynamiques et l'améliorer est hors du champ de ce travail. Néanmoins leur planificateur local comporte l'option de souscription aux mesures immédiates d'un capteur de distance, qui pourrait servir à l'évitement d'obstacles en mouvement. La taille initiale du rayon des superpixels a été fixée à 20 pixels pour permettre un bon compromis entre la précision de la reconstruction et son efficacité (cf. section 3.4.1). Avec notre robot, cette configuration produit des supersurfels répartis initialement au minimum à une distance d'environ 5 cm entre leurs centres. La carte dense reconstruite lors de la navigation autonome est présentée en figure 5.12.

A l'initialisation, le robot a été placé dans un bureau et téléopéré sur un demi-tour et une avancée d'un mètre, afin de reconstruire suffisamment de supersurfels traversables sous ses roues pour permettre la navigation autonome. Le robot a ensuite navigué de manière autonome pour sortir du bureau et traverser les couloirs en suivant des points de passage. La carte reconstruite est étendue au fur et à mesure du déplacement, ce qui permet d'envoyer un nouveau point de passage, défini à la frontière de la surface reconstruite, à chaque fois qu'un précédent but est atteint. Le robot s'est déplacé sur des distances de 3 à 5 mètres entre chaque point de passage, selon les occultations et la portée maximale de la caméra. Le graphique 5.11 indique l'évolution du temps d'extraction de la surface navigable utilisée par le planificateur de trajectoires, par rapport à la taille de la carte globale reconstruite par le système de SLAM pendant l'expérience. On peut observer que le temps d'extraction augmente linéairement mais lentement et que l'extraction et l'analyse de traversabilité nécessite seulement quelques millisecondes, permettant au planificateur d'être rapidement au courant des mises à jour possibles de la carte globale. Par la suite, il serait intéressant de considérer les aptitudes du système de navigation utilisé pour l'exploration complètement autonome.

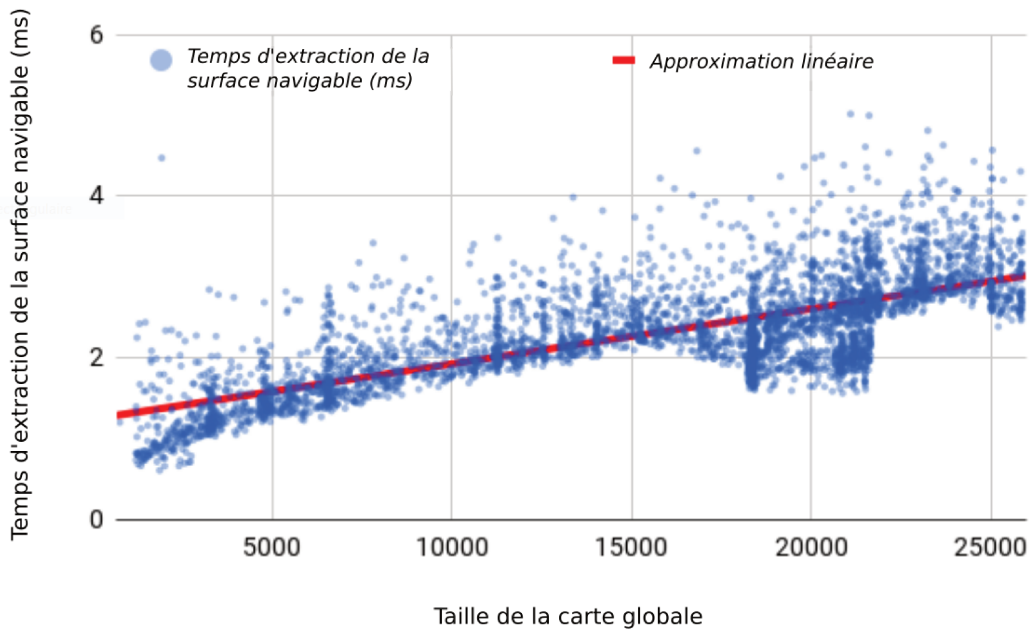

FIGURE 5.11 – Graphique affichant le temps nécessaire pour extraire la surface navigable de la carte globale du système de SLAM, par rapport au nombre de supersurfels dans la carte.

5.5 Bilan

Dans cette partie nous avons porté l'algorithme de SLAM RGB-D SupersurfelFusion, présenté au chapitre 4, sur un robot terrestre à conduite différentielle, équipé de roues encodeuses. Les mesures apportées par les encodeurs ont été fusionnées à celles de la caméra RGB-D pendant la localisation dans le but d'accroître la robustesse et la précision du système. Les résultats d'évaluation sur la précision de la localisation ont montré la supériorité des méthodes de SLAM combinant capteur de profondeur et encodeurs sur celles qui utilisent seulement une caméra RGB-D. Nous avons aussi proposé une stratégie permettant de tirer profit de la légèreté de la reconstruction supersurfel pour réaliser des tâches de cartographie et de navigation autonome simultanément, en rattachant un algorithme de planification de trajectoires à SupersurfelFusion. La solution complète de cartographie, localisation et navigation concomitantes a été déployée avec succès sur une plateforme mobile robotique de test. Son efficacité a été validée lors d'une expérience de navigation autonome réelle menée dans les locaux de notre laboratoire de recherche.

Si nos travaux ont permis de souligner l'utilité de SupersurfelFusion pour la navigation, le couplage réalisé entre le planificateur de trajectoires et l'algorithme de SLAM reste élémentaire. La localisation et la cartographie ne prennent pas en compte la dynamique des commandes de contrôle envoyées au robot par le système de navigation. De même, les trajectoires générées ne tiennent pas compte des images de la caméra. Pour améliorer les performances et la robustesse de la méthode, il serait intéressant de mettre en place une boucle de rétroaction entre les modules de perception et de navigation, par exemple en contraignant les déplace-

ments du robot de manière à maximiser l'information visuelle vue par la caméra. Dans le cadre de l'évitement d'obstacles en milieu dynamique, il pourrait aussi être intéressant de réaliser le suivi indépendant de chaque objet en mouvement perçu dans la scène et de prédire ses déplacements afin d'en tenir compte lors de la planification de la trajectoire.

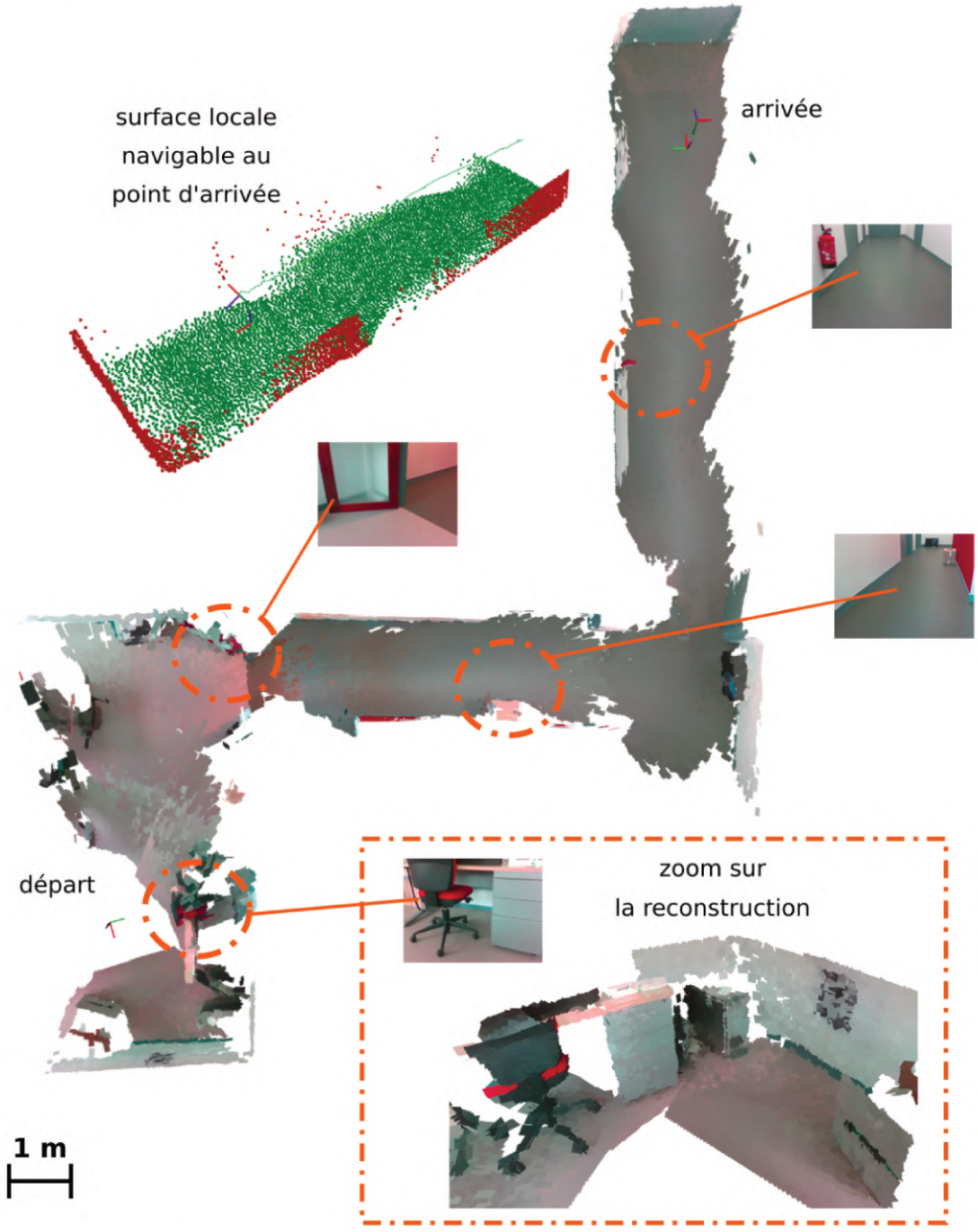

FIGURE 5.12 – Vue de dessus d'une scène cartographiée par l'algorithme de SLAM proposé, en faisant de la navigation par points de cheminement à l'aide du planificateur de trajectoires de PÉREZ-HIGUERAS et al. [PH+20]. Les centres des supersurfels traversables (en vert) et obstacles (en rouge) dans le périmètre local du robot sont affichés dans le coin supérieur gauche de la figure.

Chapitre 6

Conclusion et perspectives

6.1 Bilan général

Cette thèse a été réalisée au sein d'une équipe de recherche qui développe des méthodes pour la perception d'un robot compagnon terrestre en milieu intérieur, ayant pour missions la vérification de l'état de santé de personnes fragiles, la prévention de risques et le lancement d'alerte en cas d'accidents. Afin de pouvoir naviguer et interagir au mieux avec le monde qui l'entoure, un tel robot doit avoir accès à une carte de son environnement, lui permettant de planifier ses mouvements et de se déplacer tout en évitant les obstacles. Il doit aussi savoir se repérer dans sa carte.

La motivation principale derrière les travaux de cette thèse était de concevoir une solution de localisation et cartographie RGB-D adaptée à la navigation du robot, c'est-à-dire répondant aux critères suivants :

— la méthode doit être assez efficace pour fonctionner en temps réel sur la carte embarquée du robot, de puissance de calcul et d'espace mémoire limités ;

— elle doit aussi être robuste pour ne pas être perturbée par la présence de personnes en mouvement ou par le manque d'information pertinente dans les images ;

— il faut qu'elle soit fiable sur le long terme ;

— elle doit être capable de produire une carte 3D dense, suffisamment précise pour garantir une navigation sans risque dans différents types d'environnements, mais compacte et légère pour permettre une planification de trajectoires rapide.

Dans ce but, nous avons dans un premier temps proposé une nouvelle forme de représentation, pour la cartographie 3D basse résolution mais cohérente d'environnements intérieurs avec une caméra RGB-D. Au contraire des systèmes de SLAM RGB-D dense conventionnels, qui se focalisent sur la précision de la reconstruction 3D et emploient pour cela des formes de représentation lourdes à stocker et lentes à traiter, notre méthode de reconstruction favorise l'efficacité. La modélisation 3D de l'environnement observé consiste en un ensemble de primitives appelées *supersurfels*. Les supersurfels sont des patchs elliptiques générés à partir de la segmentation en superpixels d'images RGB-D, qui encodent la géométrie et la couleur locale d'une surface. Les expériences menées ont montré que la représentation supersurfels réduit de manière importante la charge mémoire et les temps de calculs nécessaires à la reconstruction 3D par rapport aux formes de représentation conventionnelles, tout en maintenant un bon niveau de précision. L'utilisation de la représentation mise en place est particulièrement

intéressante pour les systèmes embarqués de puissance de calcul et d'espace mémoire limités, n'ayant pas besoin d'une reconstruction très détaillée de leur milieu, mais plutôt d'une cartographie flexible, légère et rapide.

Nous avons ensuite développé *SupersurfelFusion*, un système de SLAM RGB-D complet construit autour de la représentation supersurfel. Il associe des stratégies de localisation éparse indirecte et de recalage dense pour une meilleure robustesse face aux mouvements rapides de la caméra et dans les environnements difficiles (faiblement texturés ou dont la structure est principalement plane). Afin d'éviter que la présence de personnes ou d'objets en mouvement dans le champ de vision de la caméra ne viennent corrompre la localisation et déteriorer la cartographie, nous avons élaboré et intégré au système une solution de détection et de filtrage des éléments dynamiques. Elle combine réseau de neurones, compensation de mouvement, différence de profondeurs et flot optique dense, pour identifier les superpixels des images associés à des éléments dynamiques. Afin de garantir la cohérence globale de la cartographie, une méthode de fermeture de boucle applicable en ligne a été adaptée au cas des supersurfels. Elle repose sur la déformation d'un graphe embarqué dans la surface du modèle 3D reconstruit. L'algorithme de SLAM proposé présente des résultats compétitifs comparés à d'autres plus complexes et un fonctionnement en temps réel.

Finalement, nous avons porté le système de SLAM réalisé sur une plateforme robotique mobile à conduite différentielle, pour l'appliquer au cas de la navigation autonome. Les données des roues encodeuses du robot ont été intégrées au module de localisation de SupersurfelFusion dans le but d'améliorer ses performances, puis un système de navigation a été choisi et modifié pour planifier les déplacements du robot à partir de la carte globale de supersurfels reconstruite. La faisabilité de l'utilisation de SupersurfelFusion pour des tâches de navigation a été validée lors d'une expérience de localisation, cartographie et planification concomitantes, menée sur le robot dans les locaux de notre laboratoire de recherche.

6.2 Perspectives

Les études et expérimentations réalisées nous ont également permis de mettre en évidence de nombreuses perspectives de recherche qui n'ont malheureusement pas pu être abordées ou explorées de manière aboutie durant cette thèse. Il s'agit ici d'exposer les plus significatives.

Du point de vue de la cartographie 3D, comme le robot évolue dans des environnements intérieurs structurés, il serait intéressant de regrouper les supersurfels appartenant à un même plan et de les fusionner sous la forme d'une unique primitive pour diminuer encore plus la taille du modèle reconstruit et améliorer l'estimation des normales de la surface. L'exploitation des structures régulières de la scène, telles que les lignes et les plans, pourrait aussi servir à réduire l'accumulation d'erreurs dans le suivi de la pose du robot.

Par ailleurs, le système de SLAM SupersurfelFusion a été élaboré pour la cartographie d'environnement de l'échelle d'une pièce d'un bâtiment et évalué dans des scénarios de fonctionnement relativement courts, de l'ordre de plusieurs minutes. Pour utiliser SupersurfelFu-

sion dans des environnements beaucoup plus étendus (un bâtiment complet par exemple) et sur des durées beaucoup plus longues, il faudrait réfléchir à un partitionnement de la carte de supersurfels en sous-ensembles et introduire une méthode pour optimiser ces sous-ensembles à un niveau global. De plus, pour ne pas saturer la mémoire disponible sur la carte embarquée du robot et maintenir un temps d'exécution relativement constant, il serait nécessaire d'oublier les parties de la carte les plus éloignées ou de les transférer au fur et à mesure sur un dispositif externe.

Une autre amélioration possible de notre algorithme de SLAM consisterait à reconstruire le fond des images clés obstruées par des objets en mouvement avec des techniques d'"inpainting". En effet, nous avons pu observer que la présence d'éléments dynamiques dans les images pouvait faire échouer la méthode de reconnaissance de lieu permettant la détection de fermeture de boucle, et donc empêcher la correction de la pose estimée du robot et de la carte générée.

Une boucle de rétroaction pourrait aussi être mise en place entre le système de SLAM et le système de navigation. Pour l'instant les processus de cartographie et de planification de trajectoires sont réalisés de façon indépendante, mais le rapprochement des deux permettrait par exemple de maintenir une incertitude faible sur la localisation ou encore de s'assurer que le robot navigue toujours face à des zones saillantes de l'environnement.

Enfin, dans le cadre de l'évitement d'obstacles en milieu dynamique, on peut envisager d'effectuer le suivi et la modélisation de chaque objet ou personne en mouvement, en plus de reconstruire la partie statique de l'environnement. Cela n'a pas été fait dans le cadre de la thèse pour ne pas accroître l'empreinte mémoire et le temps d'exécution de la méthode, mais pourrait permettre d'améliorer la compréhension qu'a le robot de la scène et l'aider dans ses décisions. Il faudrait alors en plus étendre la représentation supersurfel à la modélisation d'éléments non rigides. Une idée de départ pour cela serait d'utiliser une structure de graphe en connectant les supersurfels d'un même objet déformable.

Annexe A

Filtrage de la profondeur avec superpixels

Soit d_i la disparité du centre $[\bar{x_i}, \bar{y_i}]^\intercal$ de chaque superpixel, obtenu par moyennage des positions des pixels qui le composent, et $\frac{\partial d_i}{\partial x}$, $\frac{\partial d_i}{\partial y}$ les dérivées de la disparité par rapport à la position. Pour lisser la carte de pronfondeur, on cherche les nouveaux états $\mathbf{X}_i$ des superpixels qui minimisent la distance aux observations $\mathbf{Z}_i[d_i, \frac{\partial d_i}{\partial x}, \frac{\partial d_i}{\partial y}]^\intercal$ ainsi que les variations entre les superpixels voisins :

$$E_{smo} = \sum_i \left(\alpha \|\mathbf{Z_i} - \mathbf{X}_i\|^2 + \beta \sum_{j \in \mathcal{N}_i} \|\mathbf{X}_i - \mathbf{H}_{ij}\mathbf{X}_j\|^2 \right) \tag{A.1}$$

avec $\mathbf{H}_{ij} = \begin{bmatrix} 1 & \bar{x_i} - \bar{x_j} & \bar{y_i} - \bar{y_j} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ et $\mathcal{N}_i$ les superpixels voisins du superpixel d'indice i. On peut réécrire E_{smo} sous forme matricielle :

$$E_{smo} = \alpha(\mathbf{Z} - \mathbf{X})^\intercal(\mathbf{Z} - \mathbf{X}) + \beta(\mathbf{HX})^\intercal(\mathbf{HX}) \tag{A.2}$$

avec $\mathbf{H}$ matrice de taille $3R \times 3N$, où N est le nombre de noeuds (superpixels) et R la somme des nombres de voisins par noeud, qui prend la forme suivante :

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & -\mathbf{H}_{12} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \mathbf{I} & 0 & \cdots & 0 & -\mathbf{H}_{1j} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -\mathbf{H}_{i1} & \cdots & \mathbf{I} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \mathbf{I} & 0 & \cdots & -\mathbf{H}_{ij} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

Minimiser E_{smo} revient à résoudre $\frac{\partial E_{smo}}{\partial \mathbf{X}} = 0$

$$\Leftrightarrow -2\alpha(\mathbf{Z} - \mathbf{X}) + 2\beta\mathbf{H}^\intercal\mathbf{HX} = 0 \tag{A.3}$$

$$\Leftrightarrow (\alpha\mathbf{I} + \beta\mathbf{H}^\intercal\mathbf{H})\mathbf{X} = \alpha\mathbf{Z} \tag{A.4}$$

On retrouve la forme d'une système du type $\mathbf{AX} = \mathbf{B}$, avec $\mathbf{A} = \alpha\mathbf{I} + \beta\mathbf{H}^\intercal\mathbf{H}$ et $\mathbf{B} = \alpha\mathbf{Z}$. Les éléments diagonaux de la matrice $\mathbf{A}$ sont de la forme :

$$\mathbf{A}_{ii} = \alpha\mathbf{I} + \beta \sum_{j \in \mathcal{N}_i} (\mathbf{I} + \mathbf{H}_{ji}^\intercal\mathbf{H}_{ji})$$

et les autres éléments de la forme :

$$\mathbf{A}_{ij} = -\beta(\mathbf{H}_{ij} + \mathbf{H}_{ji}^{\intercal})$$

La matrice A est à diagonale strictement dominante sur les lignes. Le problème peut donc être résolu de manière itérative avec la méthode de Block-Jacobi [Saa03] :

$$\mathbf{X}^{(k+1)} = \mathbf{D}^{-1}(\alpha \cdot \mathbf{Z} - \mathbf{R}\mathbf{X}^{(k)}) \tag{A.5}$$

où $\mathbf{D}$ est la partie diagonale de la matrice $\mathbf{A}$ et $\mathbf{R} = \mathbf{A} - \mathbf{D}$ la même matrice sans la partie diagonale. Le calcul à appliquer de manière itérative au niveau de chaque noeud i est donc :

$$\mathbf{X}_i^{(k+1)} = \mathbf{A}_{ii}^{-1}(\alpha\mathbf{Z} + \beta \sum_{j \in \mathcal{N}_i} (\mathbf{H}_{ij} + \mathbf{H}_{ji}^{\intercal})\mathbf{X}_j^{(k)}) \tag{A.6}$$

Annexe B

Algorithme RANSAC

L'algorithme RANSAC (RANdom SAmple Consensus) est une méthode itérative proposée par Fischler et Bolles [FB81] pour estimer de manière robuste un modèle paramétrique $\mathbf{\Theta}$ à partir d'un ensemble d'observations $\mathcal{O}$ qui contient des données aberrantes. L'algorithme générique est le suivant :

Algorithme 3 RANSAC

Entrées: observations $\mathcal{O}$, nombre maximal d'itérations N

Sorties: paramètres du modèle $\mathbf{\Theta}$, sous ensemble d'observations valides $\mathcal{I} \subset \mathcal{O}$ (inliers) d'après le modèle estimé

1: $iteration \leftarrow 0$, $inliersMax \leftarrow 0$
2: **répéter**
3: $\quad\mathcal{S}_i \leftarrow$ sélectionner aléatoirement un sous-ensemble minimal d'observation
4: $\quad\mathbf{\Theta}_i \leftarrow$ calculer une hypothèse du modèle à partir du sous-ensemble $\mathcal{S}_i$
5: $\quad\mathcal{I}_i \leftarrow$ calculer l'ensemble d'observations vérifiant l'hypothèse $\mathbf{\Theta}_i$ du modèle avec un seuil de tolérance prédéfini
6: $\quad$**si** CARD$(\mathcal{I}_i) > inliersMax$ **alors**
7: $\quad\quad\mathcal{I} \leftarrow \mathcal{I}_i$
8: $\quad\quad inliersMax \leftarrow$ CARD$(\mathcal{I}_i)$
9: $\quad$**fin si**
10: $\quad iteration \leftarrow iteration + 1$
11: **jusqu'à** $iteration == N$
12: $\mathbf{\Theta} \leftarrow$ estimer le modèle final à partir de toutes les observation du plus grand ensemble d'inliers $\mathcal{I}$

Le nombre d'itérations N est choisi suffisamment grand pour garantir la probabilité p qu'au moins un des sous-ensemble de données sélectionnées aléatoirement ne contienne pas d'aberrations. Soit m le nombre minimal d'observations nécessaires pour calculer le modèle et a la probabilité que les m observations sélectionnées soit valides. La valeur de N peut être fixée en fonction de la probabilité p désirée avec la formule suivante :

$$N = \frac{\log(1-p)}{\log(1-a^m)} \tag{B.1}$$

Annexe C

Jacobien de l'erreur de reprojection

On détail ici le calcul visant à minimiser l'erreur de reprojection E_{proj}. Dans sa forme la plus simple, elle est définie par la formule suivante :

$$E_{proj} = \sum_j (\mathbf{r}_j)^2 = \sum_j \|\mathbf{u}_j - \pi(\mathbf{T}\mathbf{p}_j)\|^2 \tag{C.1}$$

où les $\mathbf{p}_j$ sont des positions 3D d'une scène observée par une caméra et les $\mathbf{u}_j$ leurs observations 2D correspondantes dans l'image. $\pi(.)$ est l'opérateur de projection, décrit par les paramètres intrinsèques de la caméra, qui permet de calculer la position pixel d'un point 3D connu dans le champ de vision de la caméra. $\mathbf{T} \in SE(3)$ est la transformation à six degrés de liberté que l'on souhaite déterminer. Elle permet de convertir un point du référentiel de la scène à celui de la caméra. Pour estimer la transformation $\mathbf{T}$ en minimisant le critère E_{proj} sur l'ensembles des mesures et des observations, il faut construire la matrice Jacobienne du système, qui regroupe les dérivées des résidus $\mathbf{r}_j$ par rapport aux paramètres de $\mathbf{T}$.

En utilisant l'algèbre de Lie, on peut se ramener à un problème d'optimisation non contraint sur la variété $SE(3)$ et facilement résoluble par la méthode de Gauss-Newton. On considère alors la dérivation des résidus par rapport à une petite perturbation $\Delta\boldsymbol{\xi} \in se(3)$, appliquée à la valeur courante de la transformation $\mathbf{T}$:

$$\frac{\partial \mathbf{r}}{\partial \Delta\boldsymbol{\xi}} = \frac{\partial(\mathbf{u} - \pi(\exp_{SE(3)}(\Delta\boldsymbol{\xi})\mathbf{T}\mathbf{p}))}{\partial \Delta\boldsymbol{\xi}} \tag{C.2}$$

En applicant la règle de dérivation en chaîne, on obtient :

$$\frac{\partial \mathbf{r}}{\partial \Delta\boldsymbol{\xi}} = -\left.\frac{\partial \mathbf{u}'}{\partial \mathbf{p}'}\right|_{\mathbf{p}'=\mathbf{T}\mathbf{p}} \cdot \left.\frac{\partial \mathbf{p}'}{\partial \Delta\boldsymbol{\xi}}\right|_{\Delta\boldsymbol{\xi}=0} \tag{C.3}$$

où $\mathbf{p}'[x', y', z']^\intercal$ est obtenu en transformant un point 3D $\mathbf{p}[x, y, z]^\intercal$ du repère de la scène à celui de la caméra avec l'estimation courante de la transformation $\mathbf{T}$:

$$\mathbf{p}' \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{T}\mathbf{p} = \mathbf{R} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \mathbf{t} \tag{C.4}$$

et $\mathbf{u}'[u', v']^\intercal$ est la position pixel de la projection du point 3D $\mathbf{p}'$, calculée en applicant la projection $\pi(.)$:

$$\mathbf{u}' \begin{bmatrix} u' \\ v' \end{bmatrix} = \pi(\mathbf{p}') = \begin{bmatrix} \frac{x' f_x}{z'} + c_x \\ \frac{y' f_y}{z'} + c_y \end{bmatrix} \tag{C.5}$$

avec f_x, f_y, c_x, c_y, les paramètres intrinsèques de la caméra.

Le premier terme de l'équation C.3 est la dérivée de la projection par rapport au point transformé. Elle est donnée par :

$$\frac{\partial \mathbf{u}'}{\partial \mathbf{p}'} = \begin{bmatrix} \frac{f_x}{z'} & 0 & -\frac{x' f_x}{z'^2} \\ 0 & \frac{f_y}{z'} & -\frac{y' f_y}{z'^2} \end{bmatrix} \tag{C.6}$$

Le deuxième est la dérivée du point transformé par rapport à l'algèbre de Lie :

$$\frac{\partial \mathbf{p}'}{\partial \Delta \boldsymbol{\xi}} = [\mathbf{I} \quad -(\mathbf{p}')^{\wedge}] = \begin{bmatrix} 1 & 0 & 0 & 0 & -z' & y' \\ 0 & 1 & 0 & z' & 0 & -x' \\ 0 & 0 & 1 & -y' & x' & 0 \end{bmatrix} \tag{C.7}$$

La démonstration de cette formule est présentée dans [GZ21] ; [Bla10]. Finalement, la multiplication de ces deux dérivées donne la matrice jacobienne de chaque résidu :

$$\mathbf{J} = -\begin{bmatrix} \frac{f_x}{z'} & 0 & -\frac{f_x x'}{z'^2} & -\frac{f_x x' y'}{z'^2} & f_x + \frac{f_x x'^2}{z'^2} & -\frac{f_x y'}{z'} \\ 0 & \frac{f_y}{z'} & -\frac{f_y y'}{z'^2} & -f_y - \frac{f_y y'^2}{z'^2} & \frac{f_y x' y'}{z'^2} & \frac{f_y x'}{z'} \end{bmatrix} \tag{C.8}$$

Annexe D

Algorithme ICP

L'*Iterative Closest Point* (ICP), introduit pour la première fois par [BM92], est un algorithme qui vise à trouver la transformation $\mathbf{T}$ entre deux surfaces, souvent présentées sour la forme d'un nuage de points $\mathcal{S} = \{\mathbf{p}_i\}$ décalé (source) et d'un nuage de points référence $\mathcal{D} = \{\mathbf{p}_j\}$ (destination). A partir d'une estimation initiale de la transformation $\mathcal{T}$, il minimise itérativement la distance entre les points des deux ensembles en répétant les étapes principales suivante :

1. la transformation des points de l'ensemble source $\mathcal{S}$ en un ensemble $\mathcal{S}' = \mathbf{p}'_i$ par application de la valeur courante de la transformation $\mathbf{T}$;
2. la création d'une liste d'associations $\mathcal{A}_{i,j}$, appariant à chaque point $\mathbf{p}'_i$ son plus proche voisin dans l'ensemble de référence $\mathcal{D}$;
3. l'estimation de la transformation incrémentale $\Delta\mathbf{T}$ par minimisation de la somme des carrés des distances point-point :

$$\sum_{\mathcal{A}_{i,j}} \|\mathbf{p}_j - \Delta\mathbf{T}\mathbf{p}'_i\|^2 \tag{D.1}$$

4. la mise à jour de la transformation $\mathbf{T}$ à partir de l'estimation incrémentale : $\mathbf{T} \leftarrow \Delta\mathbf{T}\mathbf{T}$.

Ces étapes sont répétées un nombre fixé d'itérations ou bien jusqu'à convergence de la solution.

Il existe de nombreuses variantes de l'ICP [CM91] ; [BTP13] ; [Rus19] ; [SHT09], qui diffèrent par exemple dans la façon de rechercher les plus proches voisins ou dans la métrique qu'elles utilisent à la place de la distance point-point pour estimer la transformation.

Annexe E

Jacobien de la métrique point-plan symétrisée

Soit $\mathcal{S} = \{\mathbf{p}_i\}$ un ensemble de points source à recaler sur l'ensemble de points destination $\mathcal{D} = \{\mathbf{p}_j\}$ (ou de référence) à partir de l'algorithme ICP et $\mathcal{A}_{i,j}$ le jeu d'associations entre ces deux ensembles. On détail ici le calcul de la transformation optimale $\{\mathbf{R}|\mathbf{t}\}$ entre les deux ensembles par minimisation de la métrique point-plan symétrisée. Cette variante de l'ICP se base sur une scission en deux parties symétriques de la transformation à estimer. La scission résulte en une transformation scindée $\{\mathbf{R}_{sym}|\mathbf{t}_{sym}\}$, telle que $\mathbf{R} = \mathbf{R}_{sym}\mathbf{R}_{sym}$ et $\mathbf{t} = \mathbf{R}_{sym}\mathbf{t}_{sym}$. Pour améliorer la convergence de l'ICP, Rusinkiewicz [Rus19] propose d'appliquer la transformation scindée sur l'ensemble source $\mathcal{S}$ et son inverse sur l'ensemble de destination $\mathcal{D}$:

$$E_{sym} = \sum_{\mathcal{A}_{i,j}} \left[(\mathbf{R}_{sym}\mathbf{p}_i - \mathbf{R}_{sym}^{-1}\mathbf{p}_j + \mathbf{t}_{sym}) \cdot (\mathbf{n}_i + \mathbf{n}_j) \right]^2 \tag{E.1}$$

où $\mathbf{n}_i$ et $\mathbf{n}_j$ sont les normales des points $\mathbf{p}_i$ et $\mathbf{p}_j$. La méthode [Rus19] se ramène à un système de moindres carrés linéaires de manière à simplifier l'optimisation, en linéarisant la rotation $\mathbf{R}_{sym}$ par la formule de Rodrigues [MSZ94]. Cette dernière exprime l'effet d'une matrice rotation $\mathbf{R}_{sym}$ sur un vecteur $\mathbf{v}$ par la forme suivante :

$$\mathbf{R}_{sym}\mathbf{v} = \mathbf{v}\cos\theta + (\mathbf{a} \times \mathbf{v})\sin\theta + \mathbf{a}(\mathbf{a} \cdot \mathbf{v})(1 - \cos\theta) \tag{E.2}$$

où θ et $\mathbf{a}$ sont l'angle et l'axe de rotation. Le dernier terme de l'équation E.2 étant quadratique, il est négligé pour retrouver une forme linéaire :

$$\mathbf{R}_{sym}\mathbf{v} \approx \mathbf{v}\cos\theta + (\mathbf{a} \times \mathbf{v})\sin\theta = \cos\theta(\mathbf{v} + (\tilde{\mathbf{a}} \times \mathbf{v})) \tag{E.3}$$

avec $\tilde{\mathbf{a}} = \mathbf{a}\tan\theta$. En appliquant l'approximation linéaire précédente à E_{sym} on obtient :

$$\begin{aligned} E_{sym} &\approx \sum_{\mathcal{A}_{i,j}} \left[\cos\theta(\mathbf{p}_i - \mathbf{p}_j) \cdot \mathbf{n}_{i,j} + \cos\theta(\tilde{\mathbf{a}} \times (\mathbf{p}_i + \mathbf{p}_j)) + \mathbf{t}_{sym} \cdot \mathbf{n}_{i,j} \right]^2 \\ &= \sum_{\mathcal{A}_{i,j}} \cos^2\theta \left[(\mathbf{p}_i - \mathbf{p}_j) \cdot \mathbf{n}_{i,j} + ((\mathbf{p}_i + \mathbf{p}_j) \times \mathbf{n}_{i,j}) \cdot \tilde{\mathbf{a}} + \mathbf{n}_{i,j} \cdot \tilde{\mathbf{t}} \right]^2 \end{aligned} \tag{E.4}$$

avec $\mathbf{n}_{i,j} = \mathbf{n}_i + \mathbf{n}_j$ et $\tilde{\mathbf{t}} = \mathbf{t}_{sym}/\cos\theta$. L'approximation supplémentaire de remplacer le facteur $\cos^2\theta$ par 1 est faites. Elle est valide dans le cas des petits angles ($\cos\theta \approx 1$, $\sin\theta \approx \theta$ pour un angle θ petit) et donne la forme finale :

$$\sum_{\mathcal{A}_{i,j}} \left[(\mathbf{p}_i - \mathbf{p}_j) \cdot \mathbf{n}_{i,j} + ((\mathbf{p}_i + \mathbf{p}_j) \times \mathbf{n}_{i,j}) \cdot \tilde{\mathbf{a}} + \mathbf{n}_{i,j} \cdot \tilde{\mathbf{t}} \right]^2 \tag{E.5}$$

Il s'agit d'un système linéaire en $\tilde{\mathbf{a}}$ et $\tilde{\mathbf{t}}$, facile à minimiser. La matrice Jacobienne du système est obtenue en dérivant la fonction objective E.5 par rapport à $\tilde{\mathbf{a}}$ et $\tilde{\mathbf{t}}$:

$$\mathbf{J}_{(i,j)\in\mathcal{A}_{i,j}} = \begin{bmatrix}(\mathbf{p}_i + \mathbf{p}_j) \times \mathbf{n}_{i,j} & \mathbf{n}_{i,j}\end{bmatrix} \tag{E.6}$$

Les rotation $\mathbf{R}$ et translation $\mathbf{t}$ de la transformation optimale de $\mathcal{S}$ vers $\mathcal{D}$ sont :

$$\mathbf{R} = \mathrm{rot}\left(\theta, \frac{\tilde{\mathbf{a}}}{\|\tilde{\mathbf{a}}\|}\right) \mathrm{rot}\left(\theta, \frac{\tilde{\mathbf{a}}}{\|\tilde{\mathbf{a}}\|}\right) \quad \text{et} \quad \mathbf{t} = \mathrm{rot}\left(\theta, \frac{\tilde{\mathbf{a}}}{\|\tilde{\mathbf{a}}\|}\right) \tilde{\mathbf{t}} \cos\theta \tag{E.7}$$

Bibliographie

[AR18] M. Aladem et S. Rawashdeh. « Lightweight Visual Odometry for Autonomous Mobile Robots ». In : *Sensors* 18.9 (2018), p. 2837 (cf. p. 12, 62).

[Bay+08] Herbert Bay et al. « Speeded-Up Robust Features (SURF) ». In : *Computer Vision and Image Understanding* 110.3 (2008), 346–359 (cf. p. 20).

[Bel+16] D. Belter et al. « RGB-D terrain perception and dense mapping for legged robots ». In : *International Journal of Applied Mathematics and Computer Science* 26 (2016), p. 81-97 (cf. p. 88, 89).

[Bes+18] B. Bescos et al. « DynaSLAM : Tracking, Mapping, and Inpainting in Dynamic Scenes ». In : *IEEE Robotics and Automation Letters* 3.4 (2018), p. 4076-4083 (cf. p. 74).

[Bia+17] J. Bian et al. « GMS : Grid-based Motion Statistics for Fast, Ultra-robust Feature Correspondence ». In : *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 (cf. p. 62, 67, 87).

[BL95] G. Blais et M.D. Levine. « Registering multiview range data to create 3D computer objects ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8 (1995), p. 820-824 (cf. p. 23).

[Bla10] Jose Luis Blanco. « A tutorial on SE(3) transformation parameterizations and on-manifold optimization ». In : (2010) (cf. p. 17, 87, 114).

[BM92] P.J. Besl et N. D. McKay. « A method for registration of 3-D shapes ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), p. 239-256 (cf. p. 30, 63, 115).

[Bru+15] N. Brunetto et al. « Fusion of Inertial and Visual Measurements for RGB-D SLAM on Mobile Devices ». In : *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. 2015, p. 148-156 (cf. p. 82).

[BTP13] S. Bouaziz, A. Tagliasacchi et M. Pauly. « Sparse Iterative Closest Point ». In : *Computer Graphics Forum (Symposium on Geometry Processing)* 32.5 (2013), p. 1-11 (cf. p. 30, 115).

[BWC18] P. Bergmann, R. Wang et D. Cremers. « Online Photometric Calibration of Auto Exposure Video for Realtime Visual Odometry and SLAM ». In : *IEEE Robotics and Automation Letters* 3 (2018), p. 627-634 (cf. p. 23).

[BWL20] A. Bochkovskiy, C. Y. Wang et H. Y. M. Liao. *YOLOv4 : Optimal Speed and Accuracy of Object Detection*. 2020. arXiv : `2004.10934 [cs.CV]` (cf. p. 58).

[Byl+13] E. Bylow et al. « Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions ». In : *Robotics : Science and Systems*. 2013 (cf. p. 52).

[Cad+16] C. Cadena et al. « Past, Present, and Future of Simultaneous Localization and Mapping : Toward the Robust-Perception Age ». In : *IEEE Transactions on Robotics* 32.6 (2016), p. 1309-1332 (cf. p. 3).

[CL96] B. Curless et M. Levoy. « A Volumetric Method for Building Complex Models from Range Images ». In : *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. 1996, 303–312 (cf. p. 29).

[CM91] Y. Chen et G. Medioni. « Object modeling by registration of multiple range images ». In : *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. 1991, 2724-2729 vol.3 (cf. p. 23, 51, 63, 115).

[Dai+17] A. Dai et al. « BundleFusion : Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration ». In : *ACM Transactions on Graphics 2017 (TOG)* (2017) (cf. p. 53).

[Don+21] X. Dong et al. « FSD-SLAM : a fast semi-direct SLAM algorithm ». In : *Complex Intelligent Systems* (mar. 2021) (cf. p. 20).

[EKC18] J. Engel, V. Koltun et D. Cremers. « Direct Sparse Odometry ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018) (cf. p. 12).

[End+14] F. Endres et al. « 3-D Mapping With an RGB-D Camera ». In : *IEEE Transactions on Robotics* 30.1 (2014), p. 177-187 (cf. p. 30, 53, 89).

[Eng16] King Mongkut's University of Technology North Bangkok Faculty of Engineering. *Invigorating Robot Activity Project*. 2016. url : `https://www.iraprobot.com` (cf. p. 1).

[ESC15] Jakob Engel, Jörg Stückler et Daniel Cremers. « Large-scale direct SLAM with stereo cameras ». In : *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, p. 1935-1942 (cf. p. 22).

[FB81] M. A. Fischler et R. C. Bolles. « Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography ». In : 24.6 (1981), 381–395 (cf. p. 22, 60, 67, 111).

[FCT20] A. Fontán, J. Civera et R. Triebel. « Information-Driven Direct RGB-D Odometry ». In : *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, p. 4928-4936 (cf. p. 38).

[Fre+10] B. Freedman et al. « Depth Mapping Using Projected Patterns ». Patent Application US 2010/0118123 A1 (United States). 13 mai 2010 (cf. p. 12).

[Gal+21] L. Gallagher et al. « A Hybrid Sparse-Dense Monocular SLAM System For Autonomous Driving ». In : *European Conference on Mobile Robotics, ECMR*. Bonn, Germany, 2021 (cf. p. 20).

[GGMCG16] D. Gutierrez-Gomez, W. Mayol-Cuevas et J.J. Guerrero. « Dense RGB-D visual odometry using inverse depth ». In : *Robotics and Autonomous Systems* 75 (2016), p. 571-583 (cf. p. 50).

[Glo+15] B. Glocker et al. « Real-Time RGB-D Camera Relocalization via Randomized Ferns for Keyframe Encoding ». In : *IEEE Transactions on Visualization and Computer Graphics* 21.5 (2015), p. 571-583 (cf. p. 66).

[GMR12] C. X. Guo, F. M. Mirzaei et S. I. Roumeliotis. « An analytical least-squares solution to the odometer-camera extrinsic calibration problem ». In : *2012 IEEE International Conference on Robotics and Automation*. 2012, p. 3962-3968 (cf. p. 83).

[GVR18] S. Giancola, M. Valenti et S. Remo. *A Survey on 3D Cameras : Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopy Technologies*. 1st. Springer International Publishing, 2018 (cf. p. 12).

[GZ21] X.g Gao et T. Zhang. *Introduction to Visual SLAM : From Theory to Practice*. Publishing House of Electronics Industry, 2021 (cf. p. 17, 25, 114).

[Han+14] A. Handa et al. « A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM ». In : *IEEE Intl. Conf. on Robotics and Automation, ICRA*. Hong Kong, China, 2014 (cf. p. 40, 71).

[HBL19] C. Houseago, M. Bloesch et S. Leutenegger. « KO-Fusion : Dense Visual SLAM with Tightly-Coupled Kinematic and Odometric Tracking ». In : *2019 International Conference on Robotics and Automation (ICRA)*. 2019, p. 4054-4060 (cf. p. 83).

[HDG17] G. He K.and Gkioxari, P. Dollár et R. Girshick. « Mask R-CNN ». In : *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, p. 2980-2988 (cf. p. 54, 55).

[Hen+12] P. Henry et al. « RGB-D mapping : Using Kinect-style depth cameras for dense 3D modeling of indoor environments ». In : *The International Journal of Robotics Research* 31 (2012), p. 647 -663 (cf. p. 31, 53).

[HN07] K. L. Ho et P. Newman. « Detecting Loop Closure with Scene Sequences ». In : *International Journal of Computer Vision* 74 (2007), p. 261-286 (cf. p. 25).

[Hop+92] H. Hoppe et al. « Surface Reconstruction from Unorganized Points ». In : *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '92. Association for Computing Machinery, 1992, 71–78 (cf. p. 36).

[Hor+13] A. Hornung et al. « OctoMap : An Efficient Probabilistic 3D Mapping Framework Based on Octrees ». In : *Autonomous Robots* (2013) (cf. p. 29, 53, 88).

[Hua+18] J. Huang et al. « Optical Flow Based Real-time Moving Object Detection in Unconstrained Scenes ». In : *ArXiv* abs/1807.04890 (2018) (cf. p. 61).

[Hub11] P. J. Huber. « Robust Statistics ». In : *International Encyclopedia of Statistical Science*. Sous la dir. de Miodrag Lovric. Springer Berlin Heidelberg, 2011, p. 1248-1251 (cf. p. 16).

[Hyd15] Hydronalix. *Emergency Integrated Lifesaving Lanyard*. 2015. url : `https://www.emilyrobot.com.au` (cf. p. 1, 2).

[HZ03] R. Hartley et A. Zisserman. *Multiple View Geometry in Computer Vision*. 2e éd. Cambridge University Press, 2003 (cf. p. 9).

[Iza+11] S. Izadi et al. « KinectFusion : Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera ». In : *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST '11. 2011, 559–568 (cf. p. 23, 29, 30, 51, 52).

[JGJ15] M. Jaimez et J. Gonzalez-Jimenez. « Fast Visual Odometry for 3-D Range Sensors ». In : *IEEE Transactions on Robotics* 31.4 (2015), p. 809-822 (cf. p. 12, 50).

[JU07] S. J. Julier et J. K. Uhlmann. « Using covariance intersection for SLAM ». In : *Robotics and Autonomous Systems* 55.1 (2007). Simultaneous Localisation and Map Building, p. 3-20 (cf. p. 39).

[KA08] K. Konolige et M. Agrawal. « FrameSLAM : From Bundle Adjustment to Real-Time Visual Mapping ». In : *IEEE Transactions on Robotics* 24.5 (2008), p. 1066-1077 (cf. p. 52).

[KE12] K. Khoshelham et S. O. Elberink. « Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications ». In : *Sensors* 12.2 (2012), p. 1437-1454 (cf. p. 13).

[Kel+13] M. Keller et al. « Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion ». In : *2013 International Conference on 3D Vision - 3DV 2013*. 2013, p. 1-8 (cf. p. 31, 32, 38, 52).

[KF11] S. Karaman et E. Frazzoli. « Sampling-based algorithms for optimal motion planning ». In : *The International Journal of Robotics Research* 30.7 (2011), p. 846-894 (cf. p. 90, 92).

[KM07] G. Klein et D. Murray. « Parallel Tracking and Mapping for Small AR Workspaces ». In : *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. 2007, p. 225-234 (cf. p. 18, 21).

[Kro+16] T. Kroeger et al. « Fast Optical Flow using Dense Inverse Search ». In : *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016 (cf. p. 60).

[Krü+17] P. Krüsi et al. « Driving on Point Clouds : Motion Planning, Trajectory Optimization, and Terrain Assessment in Generic onplanar Environments ». In : *Journal of Field Robotics* 34.5 (2017), p. 940-984 (cf. p. 90).

[KSC13a] C. Kerl, J. Sturm et D. Cremers. « Robust odometry estimation for RGB-D cameras ». In : *2013 IEEE International Conference on Robotics and Automation*. 2013, p. 3748-3754 (cf. p. 22, 50).

[KSC13b] C. Kerl, J. Sturm et D. Cremers. « Robust odometry estimation for RGB-D cameras ». In : *2013 IEEE International Conference on Robotics and Automation*. 2013, p. 3748-3754 (cf. p. 38).

[Lai+11] K. Lai et al. « A large-scale hierarchical multi-view RGB-D object dataset ». In : *2011 IEEE International Conference on Robotics and Automation*. 2011, p. 1817-1824 (cf. p. 35).

[Lai+17] T. Laidlow et al. « Dense RGB-D-inertial SLAM with map deformations ». In : *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, p. 6741-6748 (cf. p. 82).

[Lav98] S. M. Lavalle. *Rapidly-Exploring Random Trees : A New Tool for Path Planning*. Rapp. tech. Computer Science Dept., Iowa State University, 1998 (cf. p. 90).

[Leu+13] S. Leutenegger et al. « Keyframe-based Visual-Inertial SLAM using Nonlinear Optimization ». In : *Proceedings of Robotis Science and Systems (RSS) 2013*. 2013 (cf. p. 21).

[LGH19] Y. Liu, W. Gao et Z. Hu. « 3D Scanning of High Dynamic Scenes Using an RGB-D Sensor and an IMU on a Mobile Device ». In : *IEEE Access* 7 (2019), p. 24057-24070 (cf. p. 82).

[LHS14] D.V. Lu, D. Hershberger et W. D. Smart. « Layered costmaps for context-sensitive navigation ». In : *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, p. 709-715 (cf. p. 88).

[Li+20] Y. Li et al. « SplitFusion : Simultaneous Tracking and Mapping for Non-Rigid Scenes ». In : *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, p. 5128-5134 (cf. p. 55).

[Liu+19] K. Liu et al. « Semi-direct Tracking and Mapping with RGB-D Camera ». In : *Intelligent Robotics and Applications*. Springer International Publishing, 2019, p. 461-472 (cf. p. 20).

[LJ19] A. Ligocki et A. Jelínek. « Fusing the RGBD SLAM with Wheel Odometry ». In : *IFAC-PapersOnLine* 52.27 (2019). 16th IFAC Conference on Programmable Devices and Embedded Systems PDES 2019, p. 7-12 (cf. p. 83).

[LM19] M. Labbé et F. Michaud. « RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation ». In : *Journal of Field Robotics* 36.2 (2019), p. 416-446 (cf. p. 89).

[LM97] F. Lu et E. Milios. « Globally Consistent Range Scan Alignment for Environment Mapping ». In : *Autonomous Robots* 4.4 (1997), 333–349 (cf. p. 23).

[LMNF09] V. Lepetit, F. Moreno-Noguer et P. Fua. « EPnP : An Accurate O(n) Solution to the PnP Problem ». In : 81.2 (2009), 155–166 (cf. p. 21).

[Low04] D. G. Lowe. « Distinctive Image Features from Scale-Invariant Keypoints ». In : *International Journal of Computer Vision* 60.2 (2004), 91–110 (cf. p. 20, 53).

[LS17] Y. Ling et S. Shen. « Building maps for autonomous navigation using sparse visual SLAM features ». In : *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, p. 1374-1381 (cf. p. 89).

[Mac67] J. B. MacQueen. « Some Methods for Classification and Analysis of Multi-Variate Observations ». In : *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. T. 1. University of California Press, 1967, p. 281-297 (cf. p. 54).

[Mag09] M. Magnusson. « The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection ». Thèse de doct. 2009 (cf. p. 30).

[MAMT15] R. Mur-Artal, J. M. M. Montiel et J. D. Tardós. « ORB-SLAM : A Versatile and Accurate Monocular SLAM System ». In : *IEEE Transactions on Robotics* 31.5 (2015), p. 1147-1163 (cf. p. 21).

[MAR20] NASA Science MARS. *2020 Mission Perseverance Rover*. 2020. URL : `https://mars.nasa.gov/mars2020` (cf. p. 1, 2).

[MAT17] R. Mur-Artal et J. D. Tardós. « ORB-SLAM2 : an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras ». In : *IEEE Transactions on Robotics* 33.5 (2017), p. 1255-1262 (cf. p. 57).

[McC+17] J. McCormac et al. « SemanticFusion : Dense 3D semantic mapping with convolutional neural networks ». In : *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, p. 4628-4635 (cf. p. 22).

[MMY06] R.A. Maronna, D.R. Martin et V.J. Yohai. *Robust Statistics : Theory and Methods*. Wiley Series in Probability and Statistics. Wiley, 2006 (cf. p. 16).

[MNT04] K. Madsen, H. B. Nielsen et O. Tingleff. *Methods for Non-Linear Least Squares Problems (2nd ed.)* 2004 (cf. p. 13).

[MSZ94] R. M. Murray, S. S. Sastry et L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. 1st. CRC Press, Inc., 1994, p. 26-28 (cf. p. 117).

[Nar+19] F. Nardi et al. « Generation of Laser-Quality 2D Navigation Maps from RGB-D Sensors ». In : *RoboCup 2018 : Robot World Cup XXII*. Springer International Publishing, 2019, p. 238-250 (cf. p. 88, 89).

[Nie+13] M. Niessner et al. « Real-Time 3D Reconstruction at Scale Using Voxel Hashing ». In : *ACM Transactions on Graphics (TOG)* 32.6 (2013) (cf. p. 29).

[NLD11] R. A. Newcombe, S. J. Lovegrove et A. J. Davison. « DTAM : Dense tracking and mapping in real-time ». In : *2011 International Conference on Computer Vision*. 2011, p. 2320-2327 (cf. p. 22).

[Ole+17] H. Oleynikova et al. « Voxblox : Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning ». In : *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017 (cf. p. 89).

[Pal+19] E. Palazzolo et al. « ReFusion : 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals ». In : *arXiv* (2019) (cf. p. 54, 71).

[Pfi+00] H. Pfister et al. « Surfels-Surface Elements as Rendering Primitives ». In : *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*. 2000, p. 335-342 (cf. p. 30).

[PH+20] N. Pérez-Higueras et al. « 3D Exploration and Navigation with Optimal-RRT Planners for Ground Robots in Indoor Incidents ». In : *Sensors* 20.1 (2020) (cf. p. 90, 93, 100, 103).

[Pri+17] V. A. Prisacariu et al. « InfiniTAM v3 : A Framework for Large-Scale 3D Reconstruction with Loop Closure ». In : *arXiv pre-print arXiv :1708.00783v1* (2017) (cf. p. 52).

[QAM17] U. Qayyum, Q. Ahsan et Z. Mahmood. « IMU aided RGB-D SLAM ». In : *2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. 2017, p. 337-341 (cf. p. 82).

[QLS18] T. Qin, P. Li et S. Shen. « VINS-Mono : A Robust and Versatile Monocular Visual-Inertial State Estimator ». In : *IEEE Transactions on Robotics* 34.4 (2018), p. 1004-1020 (cf. p. 21).

[RA17] M. Rünz et L. Agapito. « Co-Fusion : Real-time Segmentation, Tracking and Fusion of Multiple Objects ». In : *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, p. 4471-4478 (cf. p. 55, 71, 98).

[RBA18] M. Runz, M. Buffier et L. Agapito. « MaskFusion : Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects ». In : *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2018, p. 10-20 (cf. p. 55).

[RM03] X. Ren et J. Malik. « Learning a Classification Model for Segmentation ». In : *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*. ICCV '03. IEEE Computer Society, 2003 (cf. p. 32).

[RS15] A. Ratter et C. Sammut. « Fused 2D/3D position tracking for robust SLAM on mobile robots ». In : *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, p. 1962-1969 (cf. p. 82).

[Rub+11] E. Rublee et al. « ORB : An efficient alternative to SIFT or SURF ». In : *2011 International Conference on Computer Vision*. 2011, p. 2564-2571 (cf. p. 20, 55, 57).

[Rue+19] F. Ruetz et al. « OVPC Mesh : 3D Free-space Representation for Local Ground Vehicle Navigation ». In : *2019 International Conference on Robotics and Automation (ICRA)* (2019), p. 8648-8654 (cf. p. 90).

[Rus19] S. Rusinkiewicz. « A Symmetric Objective Function for ICP ». In : *ACM Transactions on Graphics (Proc. SIGGRAPH)* 38.4 (2019) (cf. p. 30, 63, 67, 115, 117).

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. 2nd. Society for Industrial et Applied Mathematics, 2003 (cf. p. 35, 110).

[Sad+14] S. A. Sadat et al. « Feature-rich path planning for robust navigation of MAVs with Mono-SLAM ». In : *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, p. 3870-3875 (cf. p. 89).

[SC86] R. C. Smith et P. Cheeseman. « On the Representation and Estimation of Spatial Uncertainty ». In : *The International Journal of Robotics Research* 5.4 (1986), p. 56-68 (cf. p. 3).

[Sco+18] R. Scona et al. « StaticFusion : Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments ». In : *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, p. 3849-3856 (cf. p. 54, 71, 98).

[SHT09] Aleksandr Segal, Dirk Hähnel et Sebastian Thrun. « Generalized-ICP. » In : *Robotics : Science and Systems*. The MIT Press, 2009 (cf. p. 115).

[SLK15] H. Sarbolandi, D. Lefloch et A. Kolb. « Kinect range sensing : Structured-light versus Time-of-Flight Kinect ». In : *Computer Vision and Image Understanding* 139 (2015), p. 1-20 (cf. p. 12).

[SNS11] R. Siegwart, I. R. Nourbakhsh et D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. 2nd. The MIT Press, 2011 (cf. p. 84).

[SO+14] L. Seong-Oh et al. « RGB-D fusion : Real-time robust tracking and dense mapping with RGB-D data fusion ». In : *IEEE International Conference on Intelligent Robots and Systems* (2014), p. 2749-2754 (cf. p. 52).

[Sol13] J Solomon. *Mathematical Methods for Computer Vision, Robotics, and Graphics*. Department of Computer Science Stanford University, 2013 (cf. p. 14).

[SSC14] F. Steinbrücker, J. Sturm et D. Cremers. « Volumetric 3D mapping in real-time on a CPU ». In : *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, p. 2021-2028 (cf. p. 29).

[SSP07] R. W. Sumner, J. Schmid et M. Pauly. « Embedded Deformation for Shape Manipulation ». In : *ACM Trans. Graph.* 26.3 (2007), 80–es (cf. p. 67, 68).

[Sta18] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. 23 mai 2018 (cf. p. 95).

[Stu+12] J. Sturm et al. « A Benchmark for the Evaluation of RGB-D SLAM Systems ». In : *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. 2012 (cf. p. 40, 71).

[Tri+00] B. Triggs et al. « Bundle Adjustment — A Modern Synthesis ». In : *Vision Algorithms : Theory and Practice*. Springer Berlin Heidelberg, 2000, p. 298-372 (cf. p. 21, 23).

[Wan+19] Z. Wang et al. « A Computationally Efficient Semantic SLAM Solution for Dynamic Scenes ». In : *Remote Sensing* 11.11 (2019) (cf. p. 54).

[Whe+12] T. Whelan et al. « Kintinuous : Spatially Extended KinectFusion ». In : *Proceedings of Robotics : Science and Systems (RSS '12) Workshop on RGB-D : Advanced Reasoning with Depth Cameras*. 2012 (cf. p. 29, 52, 88).

[Whe+13] T. Whelan et al. « Robust real-time visual odometry for dense RGB-D mapping ». In : *2013 IEEE International Conference on Robotics and Automation*. 2013, p. 5724-5731 (cf. p. 23, 38).

[Whe+15] T. Whelan et al. « ElasticFusion : Dense SLAM Without A Pose Graph ». In : *Proceedings of Robotics : Science and Systems*. Rome, Italy, 2015 (cf. p. 23, 31, 32, 40, 51, 52, 55, 64, 71, 82, 88).

[Won+21] Y. S. Wong et al. « RigidFusion : RGB-D Scene Reconstruction with Rigidly-moving Objects ». In : *Computer Graphics Forum* 40.2 (2021) (cf. p. 55).

[Xie+15] J. Xie et al. « Fine registration of 3D point clouds fusing structural and photometric information using an RGB-D camera ». In : *Journal of Visual Communication and Image Representation* 32 (2015), p. 194-204 (cf. p. 64).

[Xin+18] F. Xingyin et al. « Real-Time Large-Scale Dense Mapping with Surfels ». In : *Sensors* 18.5 (2018) (cf. p. 53).

[Xu+19] B. Xu et al. « MID-Fusion : Octree-based Object-Level Multi-Instance Dynamic SLAM ». In : *2019 International Conference on Robotics and Automation (ICRA)*. 2019, p. 5231-5237 (cf. p. 55).

[Yan+19] D. Yang et al. « DRE-SLAM : Dynamic RGB-D Encoder SLAM for a Differential-Drive Robot ». In : *Remote Sensing* 11.4 (2019) (cf. p. 83, 84, 87, 89, 90, 95, 98).

[Yan+20] S. Yang et al. « SGC-VSLAM : A Semantic and Geometric Constraints VSLAM for Dynamic Indoor Environments ». In : *Sensors* 20.8 (2020) (cf. p. 55).

[YMU14] K. Yamaguchi, D. McAllester et R. Urtasun. « Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation ». In : *Computer Vision – ECCV 2014*. 2014, p. 756-771 (cf. p. 33, 35).

[Zha00] Z. Zhang. « A flexible new technique for camera calibration ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), p. 1330-1334 (cf. p. 10).

[Zha+20] T. Zhang et al. « FlowFusion : Dynamic Dense RGB-D SLAM Based on Optical Flow ». In : *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, p. 7322-7328 (cf. p. 54).

Résumé —

Les capacités de perception et de localisation sont des enjeux majeurs de la robotique mobile, nécessaires dans la réalisation de missions impliquant des processus décisionnels autonomes. Elles s'appuient sur les mesures d'un ou plusieurs capteurs embarqués sur la plateforme robotique mobile en question. Dans cette thèse, on s'intéresse à des techniques de perception visuelle basées sur une caméra RGB-D pour permettre la navigation d'un robot compagnon dans un milieu intérieur inconnu. Afin de pouvoir se déplacer de façon autonome, ce robot doit avoir accès à une carte représentative de la structure de son environnement et être capable de s'y repérer. Bien que de nombreuses méthodes de localisation et cartographie simultanées (SLAM - Simultaneous Localization And Mapping) RGB-D capables de résultats impressionnants aient été développées, elles sont bien souvent trop coûteuses en termes de ressources informatiques pour être exécutées sur les cartes embarquées de robots mobiles d'intérieur. Elles font aussi la plupart du temps l'hypothèse que la scène observée par la caméra est statique, ce qui limite leur utilisation dans de nombreuses situations où des personnes sont présentes. De plus, les cartes qu'elles produisent sont souvent inadéquates pour la planification de trajectoires et ne peuvent pas être utilisées directement pour de tâches de navigation. Dans le but de répondre à ces problèmes, nous proposons une nouvelle forme de représentation 3D pour la reconstruction basse résolution, mais compacte, rapide et légère d'environnements, au contraire des approches conventionnelles qui se focalisent sur la production de modèles 3D complexes avec un haut niveau détails. Un système de SLAM RGB-D dense complet, robuste aux éléments dynamiques, est conçu autour de cette représentation, puis porté sur une plateforme robotique mobile à conduite différentielle. En outre, une stratégie de navigation efficace est proposée en couplant l'algorithme de SLAM développé à un planificateur de trajectoires. Les différentes solutions proposées sont évaluées et comparées avec les méthodes de l'état de l'art, pour les valider et montrer leurs forces et faiblesses.

Mots clés : localisation et cartographie simultanées, caméra RGB-D, robotique mobile, navigation, reconstruction 3D.

Abstract — Perception and localization abilities are major challenges in mobile robotics. They are required for the achievement of mobile robotics missions that involves autonomous decision-making process. They rely on the measurements coming from one or multiple sensors embedded on the relevant mobile robot. In this thesis, we focus on visual perception techniques based on the use of a RGB-D camera to enable a companion robot to navigate in an unknown indoor environment. To be able to move autonomously through its nearby space, this robot needs to get a map representing the structure of its environment. It also needs to be able to locate itself within the environment. Although the actual RGB-D simultaneous localization and mapping (SLAM) solutions achieve impressive results, most of them are too heavy to run live on the embedded devices of the mobile robots. They also assume that the scene that is observed by the camera is static, which limits their use for many applications where humans are involved. In addition, the maps they produce are often unsuitable for path planning and cannot be used directly for navigation purpose. Aiming at mitigating these issues, we propose a novel 3D representation for low resolution, but fast, compact, and lightweight scenes reconstruction. A complete dense RGB-D SLAM algorithm, featuring robustness to moving objects, is developed based on the introduced representation and implemented on a differential drive mobile robot. Further, a navigation strategy is proposed by combining our SLAM algorithm with a path planner. The differents methods proposed are evaluated and compared with the state of the art for validation and to show their benefits and limits.

Keywords : simultaneous localization and mapping, RGB-D camera, mobile robotics, navigation, 3D reconstruction.

GIPSA-lab, 11 rue des Mathématiques
38400 Saint-Martin d'Hères, france