Méthodes de partage d'informations visuelles et inertielles pour la localisation et la cartographie simultanées décentralisées multi-robots

Rodolphe Dubois

CENTRALE NANTES

THÈSE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Signal, Image, Vision*

Par

Rodolphe DUBOIS

Méthodes de partage d'informations visuelles et inertielles pour la localisation et la cartographie simultanées décentralisées multi-robots

Thèse présentée et soutenue à l'ONERA Palaiseau, le 12 janvier 2021
Unité de recherche : UMR 6004, Laboratoire des Sciences du Numérique de Nantes (LS2N)

Rapporteurs avant soutenance :

Roland CHAPUIS	Professeur des universités, Université Clermont Auvergne
David FILLIAT	Professeur, ENSTA ParisTech, Palaiseau

Composition du Jury :

Président	Fawzi NASHASHIBI	Directeur de recherche, INRIA, Centre de recherche de Paris
Examinateurs	Alexandre EUDES	Ingénieur de recherche, ONERA, Palaiseau
	Ouiddad LABBANI-IGBIDA	Professeure des universités, ENSIL-ENSCI Limoges
	Patrick RIVES	Directeur de recherche, INRIA, Sophia Antipolis
Dir. de thèse	Vincent FRÉMONT	Professeur des universités, École Centrale Nantes

Invité(s) :

Véronique SERFATY	Direction Générale de l'Armement (DGA), Agence de l'Innovation Défense (AID), Paris

Remerciements

Difficile de réaliser que ces trois années de thèse se sont déjà écoulées ! Je ne pensais pas les voir filer aussi vite, ni même ne soupçonnais à quel point elles seraient enrichissantes techniquement et humainement.

Je remercie tout d'abord Alexandre et Vincent pour m'avoir encadré durant cette thèse. Merci pour la confiance que vous m'avez accordée. Merci pour votre disponibilité et votre engagement qui n'ont jamais fléchi même en cette dernière année de confinement. Merci pour votre enthousiasme, vos encouragements et votre aide extrêmement polyvalente aussi bien pour discuter théorie que débusquer les bourdes dans mes implémentations C++. J'ai énormément appris à vos côtés, merci pour tout !

Je remercie également les membres de mon jury de thèse pour l'intérêt qu'ils ont porté à mes travaux, le temps qu'ils ont consacré à les évaluer, et la discussion que nous avons eue à l'occasion de ma soutenance. Merci à Fawzi Nashashibi d'avoir présidé mon jury de thèse. Merci à Patrick Rives – qui avait déjà participé à mes évaluations annuelles de CSI – et à Ouiddad Labbani pour leur participation en tant qu'examinateurs, ainsi qu'à Véronique Serfaty qui représentait la Direction Générale de l'Armement. Je remercie tout particulièrement Roland Chapuis et David Filliat d'avoir accepté d'être rapporteurs de mon manuscrit de thèse.

Je remercie tous les membres de l'équipe IVA, au sein de laquelle j'ai eu la chance et le plaisir de travailler : Guy, Martial, Julien[2], Anthelme, Élise, Flora, Fabrice[2], Philippe, Frédéric, Pauline, Alexandre[2], Bertrand, Benjamin, Patrick, Stéphane et Adrien. Merci également à Sylvain Bertrand qui m'a encadré avec Alexandre lors de mon premier stage à l'ONERA et a contribué à mon choix de poursuivre en thèse. Merci à mes amis doctorants et jeunes docteurs : Rodrigo, Pierre, Soufiane, Javi, Marcela, Guillaume[2], Rémy, Alexis, Gaston, Louis, Laurane, Maxime[2], William, Benjamin et Camille. Merci pour toutes ces pauses café toujours conviviales (ou encore ces pauses Discord bien réconfortantes pendant les confinements), pour les Tamalous quotidiens, pour les discussions de bureau à parler de tout et de rien, pour les soirées jeux et cocktails sud-américains, etc. Je vous souhaite les plus belles réussites pour les années de thèse qu'il vous reste et/ou pour vos autres projets, et j'ai hâte de vous croiser à nouveau.

Je remercie l'ONERA ainsi que la DGA pour avoir financé ma thèse ainsi que mes participations aux conférences. Merci aux laboratoires Heudiasyc puis LS2N de m'avoir

accueilli lors de mes quelques passages.

Enfin, je remercie ma famille à laquelle mon parcours doit énormément, en particulier à mes parents Frédéric et Véronique, à mon frère Aurélien, à ma grand-mère Monique et mon oncle Philippe.

Sommaire

Acronymes

ARE Absolute Rotation Error (Erreur Absolue en Rotation).

ATE Absolute Translation Error (Erreur Absolue en Translation).

BA Bundle Adjustment (Ajustement de faisceaux).

BoW Bag of Word (vecteur sac-de-mots).

CPU Central Processing Unit (Processeur central).

EKF Extended Kalman Filter (filtre de Kalman étendu).

EKF-SLAM Extended Kalman Filter for SLAM (filtre de Kalman étendu pour le SLAM).

ESDF Euclidean Signed Distance Field (champ de distances signées Euclidiennes).

GNSS Global Navigation Satellite System (géolocalisation et navigation par satellite).

GPS Global Positioning System (Système de Positionnement Global).

GPU Graphical Processing Unit (Carte graphique).

ICP Iterative Closest Point.

ILP Integer Linear Programming (Programmation par entiers).

IMU Inertial Measurement Unit (Centrale Inertielle).

KF Kalman Filter (filtre de Kalman) (Kalman 1960).

KLD Kullback-Leibler Divergence (Kullback et al. 1951).

KLT Lucas-Kanade Tracking (Suivi visuel de Lucas-Kanade).

LiDAR Light Detection And Ranging.

MAP Maximum A Posteriori (estimation par maximisation de l'a posteriori).

MLE Maximum Likelihood Estimation (estimation par maximisation de la vraisemblance).

MSCKF Multi-State Constraint Kalman Filter (Mourikis et al. 2007).

MSE Mean Squared Error (moyenne de l'erreur quadratique).

NTP Network Time Protocol (protocole de temps réseau).

PCL Point Cloud Library (Rusu et al. 2011).

PnP Perspective n-Point.

RANSAC Random Sample Consensus (Fischler et al. 1981).

RGB-D Red-Green-Blue channels + Depth.

RMSE Root Mean Squared Error (racine carrée de la moyenne de l'erreur quadratique).

ROS Robot Operating System (Quigley et al. 2009).

RRE Relative Rotation Error (Erreur Relative en Rotation).

RTE Relative Translation Error (Erreur Relative en Translation).

SAR Search-And-Rescue (opérations de recherche et de sauvetage).

SDF Signed Distance Field (champ de distances signées).

SfM Structure from Motion (reconstruction par le mouvement).

SLAM Simultaneous Localization And Mapping (cartographie et localisation simultanées).

SoNAR Sound Navigation And Ranging.

TSDF Truncated Signed Distance Field (champ de distances signées et tronquées).

VIO Visual-Inertial Odometry (Odométrie Visio-Inertielle).

VO Visual Odometry (Odométrie Visuelle).

NOTATIONS

Ensembles

$\mathcal{A}$	un ensemble
$\mathcal{P}(\mathcal{A})$	ensemble des parties d'un ensemble $\mathcal{A}$
$\lvert\mathcal{A}\rvert$	cardinal de l'ensemble $\mathcal{A}$
$\subset$	inclusion d'un ensemble dans un autre au sens strict
$\subseteq$	inclusion d'un ensemble dans un autre au sens large
$\cap$	intersection entre deux ensembles
$\cup$	union entre deux ensembles
$\uplus$	union disjointe entre deux ensembles
$\mathbb{R}$	ensemble des nombres réels
$\mathbb{H}$	corps des quaternions
$\mathcal{M}_{n,m}(\mathbb{R})$	ensemble des matrices de dimensions $n \times m$ à coefficients réels.
$\mathbb{SO}_n$	groupe spécial orthogonal de dimension n des rotations
$\mathbb{SE}_n$	groupe spécial euclidien de dimension n des transformations Euclidiennes
$\mathbb{Sim}_n$	groupe des similitudes de dimension n
$\mathfrak{so}_n$	algèbre de Lie de $\mathbb{SO}_n$
$\mathfrak{se}_n$	algèbre de Lie de $\mathbb{SE}_n$
$\mathfrak{sim}_n$	algèbre de Lie de $\mathbb{Sim}_n$

Vecteurs et matrices

$\boldsymbol{x}$	un vecteur
$\boldsymbol{M}$	une matrice
$\boldsymbol{M}^\top$	transposée de la matrice $\boldsymbol{M}$
$\boldsymbol{M}^{-1}$	inverse de la matrice $\boldsymbol{M}$
$\sqrt{\boldsymbol{M}}$	racine carrée d'une matrice $\boldsymbol{M}$
$[\boldsymbol{M}]_{AB}$	matrice extraite de la matrice $\boldsymbol{M}$ où A et B désignent respectivement un ensemble de lignes et de colonnes

$\det \boldsymbol{M}$	déterminant d'une matrice $\boldsymbol{M}$
$\text{trace}(\boldsymbol{M})$	trace d'une matrice $\boldsymbol{M}$
$\text{diag}(\boldsymbol{x})$	matrice diagonale dont les coefficients de la diagonale sont ceux de $\boldsymbol{x}$
$\text{diag}(\boldsymbol{M})$	vecteur de la diagonale de la matrice $\boldsymbol{M}$
$\mathbf{1}_n$	vecteur de dimension n dont tous les coefficients valent 1
$\boldsymbol{I}_n$	matrice identité de dimension $n \times n$
$\mathbf{0}_{n,m}$	matrice nulle de dimensions $n \times m$
$\|\boldsymbol{x}\|_2$	norme $\mathcal{L}_2$ d'un vecteur $\boldsymbol{x}$
$\|\boldsymbol{x}\|_{\boldsymbol{A}}$	norme de Mahalanobis d'un vecteur $\boldsymbol{x} \in \mathbb{R}^n$, pondérée par $\boldsymbol{A}$
$\max(\boldsymbol{a}, \boldsymbol{b})$	retourne un vecteur dont la i$^{\text{ème}}$ entrée vaut $\max(a_i, b_i)$
$\boldsymbol{J}_{\boldsymbol{\theta}}^{f(\boldsymbol{\theta})}(\boldsymbol{\theta}_0)$	matrice Jacobienne d'une quantité $f(\boldsymbol{\theta})$ par rapport à $\boldsymbol{\theta}$ évaluée en $\boldsymbol{\theta}_0$
$\nabla f(x)$	vecteur gradient de l'application f évalué en x

Opérateurs en géométrie 3D

$\boldsymbol{a} \times \boldsymbol{b}$	produit vectoriel entre deux vecteurs $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$
$[\boldsymbol{a}]_\times$	matrice de l'application $\forall \boldsymbol{x} \in \mathbb{R}^3 \mapsto \boldsymbol{a} \times \boldsymbol{x} \in \mathbb{R}^3$
$\angle(\boldsymbol{a}, \boldsymbol{b})$	angle entre deux vecteurs $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$

Référentiels

${}^A t_{BC}$	Grandeur physique t attachée à référentiel C par rapport à un référentiel B et exprimée dans un référentiel A. Il s'agit d'une notation générale qui admettra des formes réduites dans lesquelles ces trois référentiels n'apparaîtront pas systématiquement. En particulier, les grandeurs désignant des transformations (e.g. translation, rotation, transformation rigide, similitude etc.) entre deux référentiels B et C seront simplement notées T_{BC}.
W	Référentiel global (*World* en anglais)
B	Référentiel inertiel (*Body* en anglais)
C	Référentiel attaché à une caméra

Groupes de Lie

$(\mathcal{M}, \circ)$	un groupe de Lie de loi de composition interne $\circ$
$\mathfrak{m}$	l'algèbre de Lie associée au groupe de Lie $\mathcal{M}$
$T_x\mathcal{M}$	l'espace (Euclidien) tangent en $x \in \mathcal{M}$ à un groupe de Lie $\mathcal{M}$
$\exp_{\mathcal{M}}$	application exponentielle $\exp_{\mathcal{M}} : \mathfrak{m} \mapsto \mathcal{M}$ liant une algèbre de Lie $\mathfrak{m}$ à son groupe de Lie $\mathcal{M}$
$\log_{\mathcal{M}}$	application logarithme $\log_{\mathcal{M}} : \mathcal{M} \mapsto \mathfrak{m}$, inverse de $\exp_{\mathcal{M}}$
$(\cdot)^\wedge$	isomorphisme canonique $(\cdot)^\wedge : \mathbb{R}^n \mapsto \mathfrak{m}$ entre $\mathbb{R}^n$ et l'algèbre de Lie $\mathfrak{m}$ d'un groupe de Lie de dimension n
$(\cdot)^\vee$	inverse de l'opérateur $(\cdot)^\wedge$
$\boxplus$	opérateur répercurant un incrément euclidien global $\boldsymbol{\delta\tau} \in \mathbb{R}^n$ sur un élément x du groupe de Lie $\mathcal{M}$: $\boldsymbol{\delta\tau} \boxplus x \triangleq \exp_{\mathcal{M}}(\boldsymbol{\delta\tau}^\wedge) \circ x$
$\boxminus$	opérateur renvoyat l'incrément global d'un élément $y \in \mathcal{M}$ par rapport à un autre élément $x \in \mathcal{M}$: $y \boxminus x \triangleq \log_{\mathcal{M}}(y \circ x^{-1})^\vee$
Ad_x	opérateur adjoint en un élément x
$\boldsymbol{Ad}_x$	matrice de l'opérateur adjoint en un élément x

Groupe Spécial Orthogonal et rotations

$\boldsymbol{R}_{AB}$	matrice de rotation entre les bases des référentiels A et B
q_{ab}	quaternion de la rotation entre les bases des référentiels A et B
$\bar{q}$	conjugué du quaternion q
q^{-1}	inverse du quaternion q
$*$	produit des quaternions
$q \cdot \boldsymbol{x}$	action de groupe d'un quaternion q sur un vecteur $\boldsymbol{x} \in \mathbb{R}^3$ transportant la rotation vectorielle

Groupe Spécial Euclidien et transformations affines

T_{AB}	transformation euclidienne entre les référentiels A et B
$\boldsymbol{T}_{AB}$	matrice de transformation homogène entre entre référentiels A et B
$\oplus$	produit sur $\mathbb{SE}_3$
$T \cdot \boldsymbol{x}$	action de groupe d'une transformation $T \in \mathbb{SE}_n$ sur un vecteur $\boldsymbol{x} \in \mathbb{R}^n$ transportant la transformation affine

Bien que $\mathbb{SE}_3$ admette des applications exponentielle et logarithme canoniques telles que définies dans la section A.3, nous paramétriserons $\mathbb{SE}_3$ à l'aide du groupe de Lie composite $\mathbb{SO}_3 \times \mathbb{R}^3$, dont l'algèbre de Lie est $\mathfrak{so}_3 \times \mathbb{R}^3$. Contrairement à la paramétrisation canonique de $\mathbb{SE}_3$, cela permet notamment de découpler les incréments en rotation des incréments de translation. Dans la suite, nous définirons ainsi les opérateurs $\boxplus$ et $\boxminus$ comme suit : étant donné un incrément $\boldsymbol{\delta\tau} = \begin{bmatrix}\boldsymbol{\delta\theta} & \boldsymbol{\delta p}\end{bmatrix}^\top \in \mathbb{R}^6$ où $\boldsymbol{\delta\theta} \in \mathbb{R}^3$ et $\boldsymbol{\delta p} \in \mathbb{R}^3$ désignent respectivement des incréments en rotation et translation, une transformation $T \in \mathbb{SE}_3$ dont la rotation et la translation sont respectivement $\boldsymbol{R} \in \mathbb{SO}_3$ et $\boldsymbol{t} \in \mathbb{R}^3$:

$$\boldsymbol{\delta\tau} \boxplus T \triangleq \begin{bmatrix} \exp_{\mathbb{SO}_3}(\boldsymbol{\delta\theta}^\wedge) \cdot \boldsymbol{R} & \boldsymbol{\delta p} + \boldsymbol{t} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \in \mathbb{SE}_3 \tag{1}$$

Étant donné deux transformations $T_1, T_2 \in \mathbb{SE}_3$, l'opérateur $\boxminus$ est tel que :

$$T_2 \boxminus T_1 \triangleq \begin{bmatrix} \log_{\mathbb{SO}_3}\left(\boldsymbol{R}_2 \cdot \boldsymbol{R}_1^\top\right)^\vee \\ \boldsymbol{t}_2 - \boldsymbol{t}_1 \end{bmatrix} \in \mathbb{R}^6 \tag{2}$$

où $\boldsymbol{R}_1$ et $\boldsymbol{R}_2$ sont les matrices de rotations, et $\boldsymbol{t}_1$ et $\boldsymbol{t}_2$ les vecteurs de translation respectivement associées à T_1 et T_2.

Lois de probabilité

$p(\boldsymbol{x})$	densité de probabilité d'un vecteur aléatoire $\boldsymbol{x}$
$\mathbb{E}_p(\boldsymbol{x})$	espérance du vecteur aléatoire $\boldsymbol{x}$ selon la loi p
$\boldsymbol{\Sigma}_{\boldsymbol{x}}$	covariance du vecteur aléatoire $\boldsymbol{x}$
$\boldsymbol{\Sigma}_{\boldsymbol{x}\|\boldsymbol{y}}$	covariance conditionnelle du vecteur aléatoire $\boldsymbol{x}$ sachant $\boldsymbol{y}$
$\boldsymbol{\Lambda}_{\boldsymbol{x}}$	matrice de précision / d'information du vecteur aléatoire $\boldsymbol{x}$
$\boldsymbol{\Lambda}_{\boldsymbol{x}\|\boldsymbol{y}}$	matrice de précision conditionnelle de $\boldsymbol{x}$ sachant $\boldsymbol{y}$
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	densité Gaussienne d'un vecteur aléatoire $\boldsymbol{x}$, de moyenne $\boldsymbol{\mu}$ et de matrice de covariance $\boldsymbol{\Sigma}$
$\chi^2_{n,\alpha}$	fractile à $\alpha\%$ d'une distribution du χ^2 à n degrés de libertés.

Mesures et estimation

$\hat{\boldsymbol{x}}$	valeur estimée de $\boldsymbol{x}$
$\boldsymbol{x}^m$	valeur mesurée de $\boldsymbol{x}$

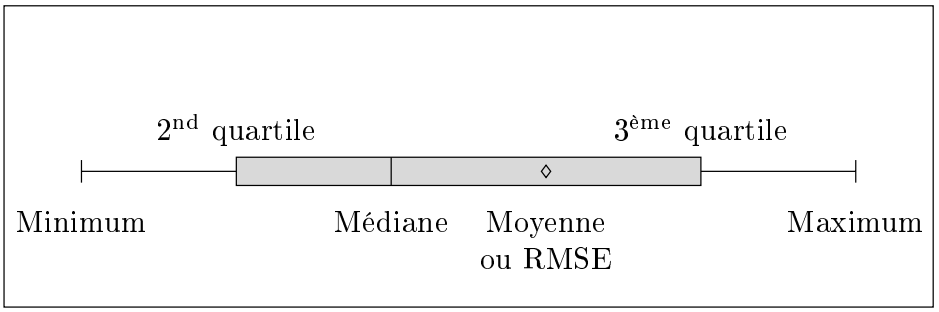
$\boldsymbol{x}^t$	vraie valeur de $\boldsymbol{x}$
$\boldsymbol{\xi}_z(\Theta_z)$	vecteur de résidu quantifiant l'écart entre une observation z et un modèle d'observation dépendant des paramètres Θ_z
$\mathcal{H}_\Theta$	espace des hypothèses pour les variables Θ

Théorie de l'information

$H_p(\boldsymbol{x})$	entropie du vecteur aléatoire $\boldsymbol{x}$ selon la loi p
$I_p(\boldsymbol{x}, \boldsymbol{y})$	information mutuelle entre les vecteurs aléatoires $\boldsymbol{x}$ et $\boldsymbol{y}$ selon la loi p
$\mathcal{D}_{\mathrm{KL}}(p, q)$	divergence de Kullback-Leibler (ou entropie relative) entre les lois de probabilité p et q calculée par rapport à p
$\boldsymbol{\mathcal{I}}_{\mathcal{Z}}^{\Theta}(\Theta_0)$	matrice d'information de Fisher observée sur les variables Θ à partir d'un ensemble d'observations $\mathcal{Z}$ et évaluée en Θ_0

Graphiques

Les distributions sont illustrées par des boîtes de Tukey construites comme suit :



Le symbole en diamant indique la moyenne dans le cas général, et l'erreur RMSE dès lors qu'il s'agit de distributions d'erreurs.

TABLE DES FIGURES

Liste des tableaux

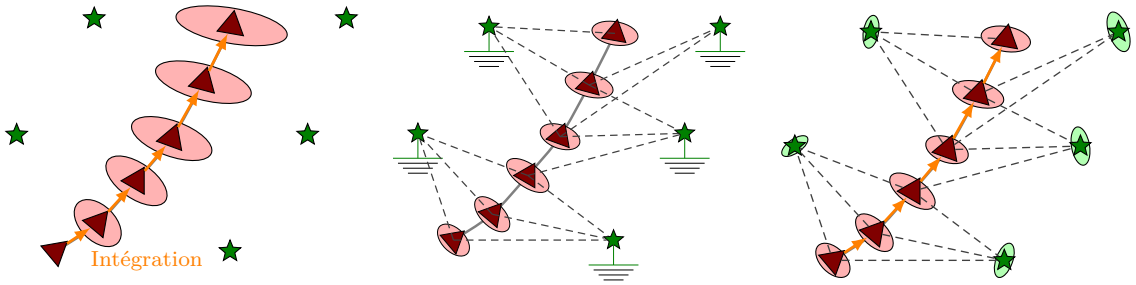
Chapitre 1

Introduction

Les algorithmes de la robotique mobile s'efforcent de percevoir et comprendre l'environnement afin d'interagir avec lui en toute autonomie. Dans cette optique, les méthodes de cartographie et de localisation simultanées constituent une brique algorithmique essentielle pour les applications qui en découlent. On les désigne couramment par leur acronyme anglais, SLAM, dérivé de *Simultaneous Localization And Mapping*.

1.1 La cartographie et la localisation simultanées

La localisation est un problème fondamental de la robotique mobile en ce sens que l'ensemble de ses applications en sont tributaires, qu'il s'agisse d'un simple suivi de trajectoire pour la navigation ou de l'exploration d'environnements inconnus. Elle désigne la capacité d'un robot à estimer en temps réel sa pose (position et orientation) courante, voire sa trajectoire complète. En environnement extérieur, on dispose éventuellement, dans ce but, de systèmes de Géolocalisation et Navigation par un Systèmes de Satellites (GNSS). C'est le cas de la localisation par GPS (*Global Positionning System*). Néanmoins, cette approche comporte trois limitations majeures. Tout d'abord, elle dépend d'une infrastructure extérieure et compromet donc l'autonomie du robot. De plus, les signaux GPS ne sont pas disponibles partout. Au sein des bâtiments, ils sont atténués voire bloqués par les parois et les surfaces. Ce phénomène s'observe également dans d'autres environnements extérieurs tels que les villes (effet canyon), les forêts (en raison de leur canopée) et bien entendu des environnements souterrains et sous-marins. Enfin, la précision de la localisation par GPS s'avère insuffisante pour certaines applications. De l'ordre de quelques mètres, celle-ci convient par exemple lorsqu'il s'agit de localiser un véhicule sur une carte routière (sans pour autant le placer avec précision sur la chaussée). Cependant, elle se révèle inapte pour la navigation de robots en environnements encombrés. Dans ce dernier cas, on requerrait plutôt une précision de l'ordre du décimètre, voire du centimètre.



(a) Navigation à l'estime (b) Navigation à vue (c) Cartographie et Localisation Simultanées (SLAM)

En navigation à l'estime, la trajectoire est estimée en intégrant successivement les mesures proprioceptives. En navigation à vue, les poses ▶ du robot sont estimées à partir de l'observation d'amers ★ dont la position est supposée connue. En SLAM, ces amers sont découverts durant l'exploration et leur position est affinée au fil de leurs observations successives. Les estimées de poses et de position des amers sont caractérisées par des mesures d'incertitude, ici figurées par des ellipsoïdes d'incertitudes.

FIGURE 1.1 – Illustration des trois paradigmes de la localisation

Les limitations des systèmes GNSS incitent à développer des solutions localement accessibles au robot. La première d'entre elles est la navigation à l'estime (c.f. Figure 1.1a), également appelée *Dead-Reckoning* en anglais. Cette méthode consiste à estimer la trajectoire en intégrant successivement les mesures issues des capteurs proprioceptifs du robot. De telles mesures rendent directement compte de son mouvement : il s'agit de mesures d'accélérations, de vitesses angulaires ou encore de nombres de tours de roues par seconde etc. Cependant, ces méthodes intègrent les bruits de mesures. Par conséquent, elles accumulent une dérive qui croît d'autant plus rapidement que la qualité des capteurs est faible. Les capteurs extéroceptifs offrent une solution alternative : la navigation à vue (c.f. Figure 1.1b). Celle-ci consiste à se localiser par rapport à des amers, c'est-à-dire des repères fixes et identifiables dans l'environnement et dont on connait préalablement la position globale. Si la navigation à vue rend l'estimation de la trajectoire plus robuste, elle suppose néanmoins une connaissance, ou du moins un balisage préalable de l'environnement.

Les méthodes de SLAM (c.f. Figure 1.1c) pallient localement à cette limitation en se proposant d'extraire elles-mêmes ces amers de l'environnement au fil de l'exploration, et de se localiser par rapport à eux tout en affinant sans cesse l'estimation de leur position. Ce faisant, elles estiment la trajectoire du robot tout en construisant, en parallèle, une carte de l'environnement par rapport à laquelle le robot se relocalise en permanence et qu'il affine en retour. Il exploite ainsi les contraintes mutuelles qui lient la localisation et la cartographie. Bien qu'à première vue, ces estimations concomittantes puissent paraître complexifier le problème, elles accroissent la robustesse aussi bien de l'estimation de la

carte que de la trajectoire du robot. Ces contraintes mutuelles sont de deux ordres. Ce sont d'abord des contraintes de court-terme lorsque les amers sont repérés et suivis par leurs observations successives. Elles contribuent ainsi à la consistance locale de la carte. Les contraintes globales correspondent quant à elles à la ré-observation de zones précédemment cartographiées. On parle alors de fermetures de boucles. Caractéristiques des méthodes de SLAM, elles permettent de réduire la dérive odométrique accumulée le long de la boucle ainsi fermée et de rendre la carte globalement consistante. La conjonction des contraintes locales et globales permet à des algorithmes d'inférence de dériver des estimations fines, robustes et précises des trajectoires et de la carte de l'environnement observé.

Comme le montrent (Cadena et al. 2016), les méthodes de SLAM jouissent d'une grande diversité. Celle-ci est d'abord de nature contextuelle puisque la littérature a investigué des solutions adaptées à des environnements variés. En particulier, chaque milieu impose des contraintes spécifiques qui affectent les propriétés des signaux d'entrée. C'est par exemple le cas des milieux sous-marins dans lesquels l'obscurité et la turbidité de l'eau dégradent la visibilité (Ferrera et al. 2019b). À la diversité des milieux s'ajoute celle des plateformes robotiques et de leurs bancs de capteurs. A minima, un seul capteur extéroceptif suffit pour faire du SLAM. Parmi ceux-ci figurent les LiDARs (très employés en robotique mobile terrestre) (Deschaud 2018), les Sonars (plus adéquats pour les milieux aquatiques) (Fallon et al. 2013), le RADAR (Holder et al. 2019), ou encore les systèmes mono (Mur-Artal et al. 2015) ou multi-caméras (Brand et al. 2014) (essentiellement destinés aux plateformes terrestres et aériennes). Néanmoins, rien n'empêche de les aggrémenter d'autres capteurs complémentaires tels que des centrales inertielles (Qin et al. 2018b), des encodeurs montés sur les roues, des magnétomètres, des altimètres, des baromètres, etc. Au-delà des capteurs, la nature de la plateforme robotique employée influe également sur la cinématique des trajectoires, qui elle-même impacte les performances des algorithmes de SLAM. Par exemple, des mouvements rapides et aggressifs compliqueront le suivi visuel pour les systèmes de caméras. À la diversité contextuelle répond la diversité algorithmique. Non seulement le choix des capteurs et les contraintes qui leur sont associées imposent la conception d'algorithmes *ad hoc* spécialisés pour traiter leurs données, mais de multiples solutions alternatives ont été proposées pour chacun des modules constitutifs d'un algorithme de SLAM, que ce soit l'odométrie et la cartographie locale, la détection des fermetures de boucles et l'inférence globale. Enfin, la diversité du SLAM est applicative. En effet, se localiser dans l'environnement tout en le cartographiant n'est pas systématiquement une finalité, mais plutôt un maillon algorithmique qui

concourt à des applications plus globales. Ces dernières relèvent aussi bien des domaines militaires, industriels et civils. Les deux premières familles d'applications exploitent prioritairement l'estimation de la trajectoire ou de la carte. En effet, si le SLAM couple la localisation et la cartographie, il peut prioriser l'une par rapport à l'autre. Par exemple, le SLAM orienté localisation se contente d'une représentation minimale de l'environnement, alors réduite à un simple support pour la détection des fermetures de boucles. Ces applications vont de la localisation d'un aspirateur autonome à celle d'un véhicule autonome (Bresson et al. 2017). D'autres applications requièrent une cartographie plus fine de l'environnement. C'est par exemple le cas pour la digitalisation de sites historiques (Leclet-Groux et al. 2018) et archéologiques (Ferrera et al. 2019b), l'exploration extra-planétaire (Maimone et al. 2007 ; Brand et al. 2014), l'inspection de zones inaccessibles ou risquées pour l'être humain (Yamada et al. 2020 ; Kim et al. 2013) ou encore la surveillance de zones (Doitsidis et al. 2011). Notons d'ailleurs que c'est essentiellement avec l'augmentation des puissances de calcul que la cartographie, historiquement support de la localisation, s'est imposée comme un objectif à part entière, et en particulier au travers du SLAM dense. Enfin, la troisième famille d'applications vise l'interaction autonome avec un environnement préalablement inconnu. Elles mobilisent alors toutes les potentialités offertes par un algorithme de SLAM. On y trouvera par exemple des applications de réalité augmentée (Piao et al. 2017 ; Sartipi et al. 2019) qui projettent des modèles 3D dans l'environnement et s'efforcent de les maintenir consistantes avec les déplacements de l'utilisateur. Les opérations de recherche et de sauvetage (*Search & Rescue*) (Queralta et al. 2020) constituent également un contexte d'application particulièrement prisé en raison de l'exhaustivité de leurs problématiques. Fondamentalement, elles consistent à explorer un environnement sinistré pour y identifier, et éventuellement y secourir, d'éventuelles victimes.

Le SLAM est un sujet de recherche actif dans la communauté robotique depuis les années 1980, et continue de l'être aujourd'hui, occupant plusieurs sessions parallèles dans les conférences internationales. Différents articles de revues tels que (Durrant-Whyte et al. 2006 ; Bailey et al. 2006b ; Cadena et al. 2016 ; Stachniss et al. 2016) ont jalonné leur développement et permettent de mesurer l'avancée des connaissances et des performances des systèmes développés.

1.2 Problématiques et contributions de thèse

Portées par de nombreuses publications, les méthodes de SLAM mono-robot ont aujourd'hui atteint un haut degré de maturité. Dans le contexte actuel d'émergence de la robotique collaborative, le SLAM multi-robot, et en particulier le SLAM collaboratif basé vision, s'est imposé comme une problématique de recherche nouvelle. Outre les défis techniques dont il s'accompagne en termes de gestion des interactions entre les robots, le SLAM collaboratif laisse espérer des gains significatifs d'efficacité à l'échelle de la flotte dans l'accomplissement de certaines tâches, par exemple en permettant une couverture de l'espace plus rapide et plus importante. En effet, la coopération entre plusieurs robots peut donner lieu à une meilleure allocation des tâches et des données, via leur parallélisation voire leur distribution, afin de pallier aux restrictions imposées par leurs ressources individuelles limitées de mémoire et de calcul. De plus, l'échange et la fusion d'informations ciblées et complémentaires entre les robots leur permet potentiellement d'accroître la précision et la robustesse des estimations de leur carte et de leur trajectoire.

Au début de cette étude, en 2017, et comme nous le détaillerons davantage dans le chapitre 3, la littérature sur le SLAM collaboratif se structurait essentiellement comme suit. Dès les années 2000, quelques méthodes étendaient des algorithmes de filtrage mono-robot tel que l'EKF-SLAM à la fusion de mesures reçues d'autres robots (Nettleton et al. 2003 ; Thrun et al. 2005 ; Howard 2006 ; Nerurkar et al. 2009), notamment pour le SLAM basé LiDAR. À partir des années 2010, l'intérêt de la communauté se porte davantage sur l'extension au contexte multi-robot des méthodes de lissage inférant sur des modèles probabilistes graphiques (Kim et al. 2010 ; Cunningham et al. 2010). Notons que la littérature sur le SLAM collaboratif se nourrit mais se distingue des publications relatives aux méthodes de localisation ou de cartographie collaboratives. Ces dernières considèrent respectivement que l'environnement ou que la trajectoire sont préalablement connus, et exploitent préférentiellement les observations mutuelles des robots entre eux plutôt que les correspondances entre les observations des uns et les cartes des autres. Enfin, jusqu'à aujourd'hui la recherche s'est essentiellement focalisée sur les méthodes centralisées dans lesquelles les robots échangent exclusivement avec un serveur central (Forster et al. 2013). Elles présentent notamment l'avantage de structurer les échanges inter-robot entre les agents et un serveur central qui exploite sa puissance de calcul accrue pour assumer les opérations inter-robot complexes (e.g. détection des correspondances, estimation globale) et éventuellement certaines tâches qui incombent ordinairement aux

agents (e.g. la détection des fermetures de boucles). De telles méthodes ont notamment été introduites en contexte visuel (Schmuck et al. 2017) et visio-inertiel (Karrer et al. 2018). Néanmoins, les architectures centralisées restreignent l'autonomie des robots, et sont critiquement vulnérable à toute défaillance du serveur central.

Les méthodes décentralisées ont quant à elles été moins investiguées, si ce n'est dans de façon très générique (Cunningham et al. 2010) ou dans des contextes spécifiques (Paull et al. 2015 ; Schuster et al. 2015). Celles-ci introduisent en particulier de nouvelles contraintes techniques. Les premières découlent de l'absence de structure des échanges, ce qui accroît notamment le risque pour chaque robot de fusionner plusieurs fois les mêmes informations dans le cas d'échanges cycliques, et ainsi de compromettre la qualité de ses estimées en termes de précision d'estimation et de caractérisation des incertitudes (i.e. de consistance). De plus, les échanges point-à-point sont davantage susceptibles de saturer la bande-passante disponible entre les robots. Enfin, les contraintes d'embarquabilité se trouvent également exacerbées par la charge de calcul supplémentaire induite par la gestion des interactions multi-robot pour chaque agent. Afin d'y pallier, une approche consiste à mutualiser les ressources des robots à l'aide d'approches distribuées (Lajoie et al. 2020 ; Choudhary et al. 2017 ; Cieslewski et al. 2018). Cependant, si elles contribuent à rendre la flotte plus flexible, ces architectures restreignent également l'autonomie des robots en les rendant tributaires les uns des autres en termes de données et de ressources de calculs. En conséquence, la perte d'un agent peut compromettre le bon fonctionnement de l'ensemble de la flotte (e.g. dans le cas de la détection distribuée de fermetures de boucles (Cieslewski et al. 2018)) ou induire des pertes significatives d'informations (e.g. dans le cas d'inférences distribuées (Cieslewski et al. 2017) et du stockage décentralisé des cartes (Cieslewski et al. 2015)).

Le SLAM multi-robot introduit trois contraintes majeures : les contraintes d'information, les contraintes de réseau et les contraintes d'embarquabilité ou de ressources. Couplées aux problématiques d'autonomie des agents, ces contraintes interrogent collectivement sur la nature des données à échanger entre les robots et la façon de les intégrer dans les problèmes d'inférence globaux. L'objectif de cette thèse est de concevoir des méthodes de cartographie et de localisation simultanées collaboratives et décentralisées en contexte visio-inertiel et stéréo-visuel qui maximisent l'autonomie des robots et leur permettent d'accroître la qualité de leurs estimations, tout en tenant compte des contraintes d'information, de réseau et d'embarquabilité. Nous souhaitons notamment permettre aux robots de tirer parti du contexte collaboratif pour enrichir leurs connaissances de l'envi-

ronnement et in fine améliorer leur propre navigation et/ou cartographie lorsqu'ils visitent des portions communes de l'environnment. Dans ce but, nous explorerons différents paradigmes d'échange de données en SLAM visio-inertiel et en SLAM stéréo-visuel dense, à partir desquels nous élaborerons des architectures complètes de SLAM collaboratif.

Les contributions de cette thèse sont triples. Tout d'abord, ayant constaté le manque de jeux de données disponibles dans la littérature pour le SLAM collaboratif, nous avons conçu, acquis puis publié le jeu de données *AirMuseum* ; il contient plusieurs scénarios multi-robot hétérogènes destinés à mettre en scène les opportunités et les contraintes associées au SLAM collaboratif. Nous avons investigué trois paradigmes d'échange de données dans un cadre visio-inertiel, respectivement basés sur l'échange de sous-cartes visio-inertielles rigides, l'échange de paquets de données condensés par des techniques de marginalisation et d'éparsification consistante, et enfin l'échange de paquets élagués bâtis à l'aide de méthodes de sélection d'amers visuels. À partir de ces trois paradigmes, nous avons dérivé trois méthodes de SLAM collaboratif décentralisé visio-inertiel basées sur une architecture commune, que nous avons en particulier évaluées sur les jeux de données EuRoC (Burri et al. 2016) et AirMuseum (Dubois et al. 2020a). Une seconde contribution a consisté à appliquer l'architecture proposée dans le contexte du SLAM dense collaboratif. Nous avons pour cela étendu une méthode de cartographie collaborative initialement développée par (Duhautbout et al. 2019) en une méthode de SLAM stéréo-visuel collaboratif et décentralisé basé sur l'échange et la fusion de sous-cartes associées à des représentations de type TSDF.

Au moment de l'écriture de ce manuscrit, ces travaux de thèse ont donné lieu aux trois publications suivantes en conférences internationales :

- Rodolphe Dubois, Alexandre Eudes et Vincent Frémont (2019), « On Data Sharing Strategy for Decentralized Collaborative Visual-Inertial Simultaneous Localization And Mapping », in : *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*
- Rodolphe Dubois, Alexandre Eudes, Julien Moras et Vincent Frémont (2020b), « Dense Decentralized Multi-robot SLAM based on locally consistent TSDF submaps », in : *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*
- Rodolphe Dubois, Alexandre Eudes et Vincent Frémont (2020a), « AirMuseum : a heterogeneous multi-robot dataset for stereo-visual and inertial Simultaneous Lo-

calization And Mapping », in : *2020 IEEE International Conference on Multisensor Fusion and Integration (MFI)*

ainsi qu'à une publication en conférence nationale :

- Rodolphe Dubois, Alexandre Eudes et Vincent Frémont (2019), « Évaluation de deux algorithmes de partage de données en Cartographie et Localisation Simultanées visuelles-inertielles multi-robot et décentralisées », in : *(ORASIS)*

1.3 Plan du manuscrit

Ce manuscrit s'organise comme suit :

Chapitre 2 : ce chapitre présente l'architecture générale d'un algorithme de SLAM mono-robot et détaille le fonctionnement de ses modules constitutifs. Il introduit en particulier les concepts fondamentaux et les outils auxquels nous nous référerons dans les chapitres ultérieurs.

Chapitre 3 : la maturité acquise par les algorithmes de SLAM mono-robot ont ouvert la voie à des architectures collaboratives qui constitueront le sujet de ce mémoire. Ce second chapitre dresse un état de l'art du SLAM multi-robot. En particulier, il présente les trois axes caractéristiques par lesquels les architectures de SLAM collaboratif étendent les algorithmes de SLAM mono-robot, et expose les solutions proposées dans la littératures pour chacun d'entre eux.

Chapitre 4 : comme le décrit le chapitre 2, l'essor des méthodes de SLAM visuel multi-robot s'affirme principalement à partir des années 2010. Or, tout comme leurs analogues mono-robot, le développement d'algorithmes de SLAM collaboratifs requiert d'évaluer leurs performances sur des données réelles adaptées. Cependant, peu de jeux de données actuellement disponibles dans la littérature le permettent. Ce chapitre relate la conception de jeux de données multi-robot pour apporter une réponse à ce manque.

Chapitre 5 : l'étude de la littérature menée dans le chapitre 2 montre que l'essor des méthodes de SLAM collaboratives a commencé par la conception de méthodes centralisées tandis que les architectures décentralisées, plus flexibles mais délicates techniquement, ont été peu investiguées, et en particulier en contexte visio-inertiel. De plus, les solutions apportées en contextes centralisés et décentralisés pour ré-

pondre aux contraintes d'information, de réseau et de ressources ont souvent compromis l'autonomie des robots. Dans ce chapitre, nous proposons et évaluons des méthodes de SLAM collaboratives décentralisées visio-inertielles pour la navigation multi-robot autonome. Si elles reposent sur une architecture commune, les méthodes exposées explorent trois paradgimes différents quant à la nature des informations échangées entre les robots et leur intégration pour l'estimation des trajectoires.

Chapitre 6 : le chapitre précédent s'inscrit dans l'approche historique du SLAM comme solution au problème de localisation. Dans cette dernière, la carte n'est qu'un support à l'odométrie et à la détection de correspondances globales au sein des trajectoires. Cependant, avec l'augmentation des moyens de calcul et le développement de méthodes denses, le volet cartographique des méthodes de SLAM s'est affirmé comme une finalité à part entière. Alors que les méthodes proposées précédemment se focalisaient sur l'estimation des trajectoires, nous proposons dans ce chapitre une méthode de SLAM multi-robot pour la cartographie dense basée sur l'échange et la fusion de sous-cartes TSDF localement consistantes. Nous reprenons pour cela les éléments d'architecture introduits précedemment pour les appliquer en contexte cartographique, et étendons une méthode de cartographie collaborative initialement proposée par (Duhautbout et al. 2019).

Enfin, dans tous les chapitres, nous utiliserons les outils mathématiques de la géométrie 3D et de l'estimation Bayésienne, auxquels les annexes A et B fournissent respectivement une introduction.

Première partie

État de l'art

Chapitre 2

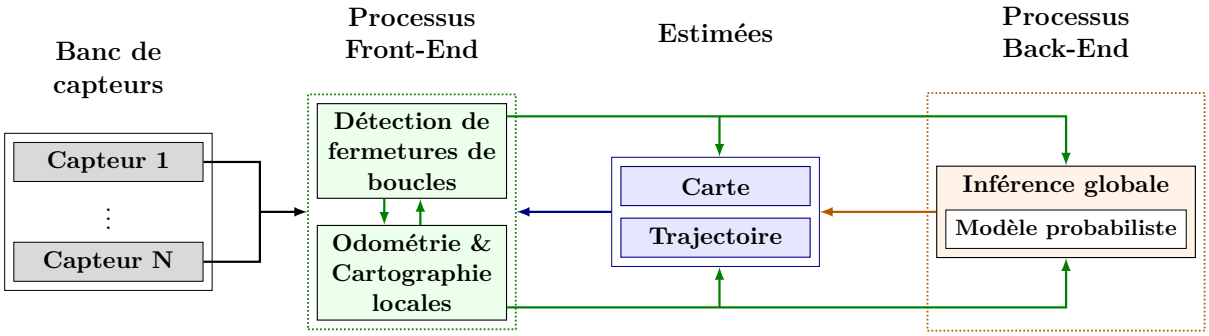
La cartographie et la localisation simultanées mono-robot

Les méthodes de cartographie et la localisation simultanées consistent en l'estimation conjointe de la trajectoire d'un robot et d'une carte de l'environnement qu'il observe. En particulier, elles s'efforcent d'exploiter les contraintes mutuelles qui les lient afin d'en parfaire l'inférence. Dans ce chapitre, nous présenterons l'architecture générale d'un algorithme de SLAM, puis nous introduirons, module par module, les outils et les concepts que nous ré-utiliserons par la suite, en insistant spécifiquement sur les cas visuels et visio-inertiels.

2.1 Architecture générale d'un algorithme de SLAM

Malgré leur grande diversité, les algorithmes de SLAM relèvent d'une architecture commune que détaille la Figure 2.1. Ils reçoivent tout d'abord en entrée les mesures issues d'un banc de capteurs contenant au moins un capteur extéroceptif. Ces mesures brutes sont alors interprétées par un premier module d'odométrie et de cartographie locale. Ce dernier initialise les estimées de carte et de trajectoires, et les affine éventuellement en recourant à des techniques d'inférence locale. Il s'agit nécessairement d'un processus dit de premier-plan (*Front-End* en anglais) en ce sens qu'il opère à la cadence de réception des mesures. Cependant, ses estimées, localement consistantes, pâtissent inéluctablement d'une dérive odométrique. Pour y remédier, un second module recherche et détecte des fermetures de boucles. Il s'agit de correspondances globales entre des observations courantes et une zone précédemment cartographiée. Ces contraintes globales sont ensuite répercutées sur l'estimation à différentes échelles. Cette tâche peut notamment dépendre d'un module d'inférence globale à part entière, lorsqu'on choisit d'affiner l'ensemble de la carte et de la trajectoire. Celui-ci relève alors d'un processus d'arrière-plan (dit *Back-End* en anglais). Pour ce faire, il maintient un modèle probabiliste qui encode l'ensemble des

connaissances accumulées sous la forme de contraintes locales et globales sur les variables de trajectoire et d'environnement estimées, et qui serviront de support à l'inférence.



Classiquement, le module de détection des fermetures de boucles est un processus de premier-plan, mais il peut également relever d'un processus d'arrière-plan pour certains SLAM.

FIGURE 2.1 – Architecture générale d'un algorithme de SLAM.

Le but des sections qui suivent est d'introduire les outils et les concepts auxquels nous nous référerons par la suite, et de les mettre en perspective avec les autres méthodes existantes. La section 2.2 décrit les bancs de capteurs communément utilisés en SLAM. L'interprétation des mesures brutes pour l'estimation initiale du mouvement et l'initialisation des cartes est traitée dans la section 2.3. La section 2.4 expose les méthodes de fermetures de boules. Enfin, les techniques d'inférence locales et globales sont discutées dans la section 2.5.

2.2 Le banc de capteurs

Les capteurs constituent la source primaire d'information du robot sur son environnement et son propre mouvement. Tout algorithme de SLAM, du fait qu'il s'appuie sur l'environnement perçu, nécessite en entrée les données d'au moins un capteur extéroceptif. Ce capteur pourra éventuellement être aggrémenté par d'autres capteurs complémentaires dont il faudra fusionner les mesures.

2.2.1 Taxonomie des bancs de capteurs

Le SLAM laser

Un LiDAR est un capteur télémétrique qui mesure les distances à des surfaces en analysant le délais et les différences de longueur d'onde entre une impulsion laser émise et

réfléchie. Le terme dérive de l'anglais *Light Detection And Ranging*. Ces impulsions laser, qui couvrent de l'infrarouge jusqu'au proche ultraviolet, balayent l'environnement sur 360 degrés et éventuellement sur plusieurs strates. Un LiDAR acquiert ainsi un nuage de points en continu, sans cesse actualisé au fil du balayage. Il présente l'avantage d'être insensible aux conditions d'éclairage ou aux propriétés optiques des surfaces avoisinantes. De plus, on trouve aujourd'hui des capteurs LiDAR suffisamment compacts pour être embarqués sur des robots mobiles, voire des drones (Shen et al. 2013), à l'instar du capteur Zebedee (Bosse et al. 2012) qui intègre également une centrale inertielle. À titre d'exemple, le capteur Velodyne (Moosmann et al. 2011) est fréquemment utilisé sur des plateformes robotiques terrestres, voire sur des véhicules comme ce fut le cas lors de l'acquisition de jeu de données KITTI (Geiger et al. 2013). En revanche, certaines conditions environnementales spécifiques compromettent la perception des capteurs LiDAR. Le brouillard est à ce titre un exemple connu du fait de sa réflexivité particulière. Par ailleurs, nous noterons que les acquisitions LiDAR sont également sujettes à des phénomènes de distorsion (Shen et al. 2013) lorsque les acquisitions sont faites en mouvement. De plus, les données issues d'un LiDAR sont d'autant plus volumineuses à traiter que les nuages de points sont denses et fréquemment rafraîchis.

En ce qui concerne le SLAM, plusieurs méthodes à base de capteurs LiDAR ont été publiées. On mentionnera notamment les travaux de Bosse et Zlot appliqué à la cartographie d'une mine (Zlot et al. 2014). Associant un LiDAR et une centrale inertielle, ils ont développé un algorithme de SLAM complet qui leur a permis de cartographier des tunnels sur plusieurs kilomètres en n'accumulant qu'une dérive de l'ordre de quelques millimètres par mètre. De plus, il s'agissait au moment de sa publication de l'une des rares méthodes basée LiDAR à estimer la trajectoire selon six degrés de liberté. Un autre travail marquant est LOAM-SLAM (Zhang et al. 2014), dont l'acronyme signifie *Lidar Odometry And Mapping*.

Le SLAM acoustique

En milieu sous-marin, l'analogue du capteur LiDAR est le SoNAR (*Sound Navigation And Ranging*). Dans sa version active, il s'agit d'un capteur acoustique qui détecte des obstacles en mesurant la distance écoulée entre l'émission d'un son et la réception de son écho par des antennes directionnelles. Deux exemples d'algorithmes de SLAM sous-marin utilisant un SoNAR sont donnés par (Fallon et al. 2011) et (Teixeira et al. 2016), appliqués à la cartographie du fond marin et à l'inspection de coques de navires.

Le SLAM visuel

Les méthodes de SLAM auxquelles nous nous intéresserons dans la suite utilisent des bancs de caméras. Elles sont d'autant plus populaires que les caméras sont des capteurs bon marché, et dont les images apportent une information extrêment riche d'ordres photométrique, géométrique et sémantique. Les développements rapides des techniques de vision par ordinateur ont progressivement permis d'exploiter ces informations en temps-réel.

Les bancs monoculaires, qui ne comportent qu'une seule caméra, servent de base à nombreux algorithmes de SLAM, ORB-SLAM (Mur-Artal et al. 2015) étant le plus connu d'entre eux. La principale limitation d'une caméra monoculaire est l'absence d'informations de profondeur. Pour cause, la projection d'un environnement 3D sur un plan image est invariante par changement d'échelle. Ainsi, non seulement le facteur d'échelle n'est pas observable, mais les algorithmes d'odométrie qui en découlent doivent veiller à limiter sa dérive le long de la trajectoire. L'information de profondeur peut être apportée par des montages exploitant les propriétés de la géométrie multi-vues (ou épipolaire) tels que les bancs stéréos. Une deuxième méthode consiste à coupler une caméra avec des capteurs de distance. On parle alors de caméras RGB-D à l'instar des caméras Kinect™. Chaque acquisition correspond à la donnée d'une image couleur classique et d'une carte de profondeur, dont chaque pixel contient la distance à la surface rencontrée par le rayon rétro-projeté qui le traverse. La caméra Kinect™ v1 ainsi que le capteur Intel RealSense d435™ dérivent cette carte de profondeur à partir de la déformation d'un motif infrarouge projeté sur les surfaces, tandis que la Kinect v2 se base sur le temps de vol de la lumière émise. L'algorithme de SLAM dense KinectFusion (Newcombe et al. 2011b) est basé sur ce capteur. Dans sa deuxième version, ORB-SLAM2 (Mur-Artal et al. 2017a) s'applique aussi bien aux cas monoculaire, stéréo et RGB-D.

Les images acquises par des caméras peuvent présenter différentes imperfections dont les algorithmes d'odométrie doivent tenir compte. Si les phénomènes de distorsion et les déformations dues aux obturateurs déroulants (*rolling shutter* en anglais) sont explicitement modélisables, d'autres imperfections telles que le flou de mouvement apparaissant lors de mouvements rapides ou encore les défauts d'exposition, compromettent l'exploitabilité des images. Face à ce défaut, des publications récentes (Gallego et al. 2019) militent pour l'utilisation de caméras dites évènementielles. Également appelées caméras neuromorphiques du fait de leur inspiration biologique, elles capturent le mouvement en enregistrant les changements d'intensité lumineuse pixel par pixel et de façon asyn-

chrone. Ces caméras apparaissent ainsi plus à même de traiter les mouvements rapides. Un algorithme d'odométrie visio-inertiel a récemment été proposé pour ce type de caméra (Mueggler et al. 2018).

Dans le cas mono-visuel, l'information apportée par la caméra permet une estimation à 6 degrés de libertés (rotations et translations), mais ne suffit pas pour estimer le facteur d'échelle. Afin d'y remédier, on peut compléter la caméra d'une centrale inertielle (*Inertial Measurement Unit* en anglais, abbrégé IMU). Il s'agit d'un capteur généralement bon marché constitué d'un accéléromètre et d'un gyromètre. L'accéléromètre mesure l'accélération spécifique tandis que le gyromètre mesure la vitesse angulaire. L'IMU rend observable le facteur d'échelle, ainsi que le vecteur de gravité, ce qui permet d'aligner correctement les modèles reconstruits par rapport à la verticale. Des algorithmes de SLAM tels que Vins-Mono (Qin et al. 2018b) ou *Multi-State Constraint Kalman Filter* (MSCKF) (Mourikis et al. 2007) sont deux exemples très connus d'algorithmes de SLAM visio-inertiel. D'autres associations sont possibles en vue de fusionner les mesures de différents capteurs. (Ferrera et al. 2019b) couplent un SLAM visuel-inertiel à un capteur de pression en contexte sous-marin, tandis que (Caruso et al. 2017) utilisent des magnétomètres comme capteurs tachymétriques.

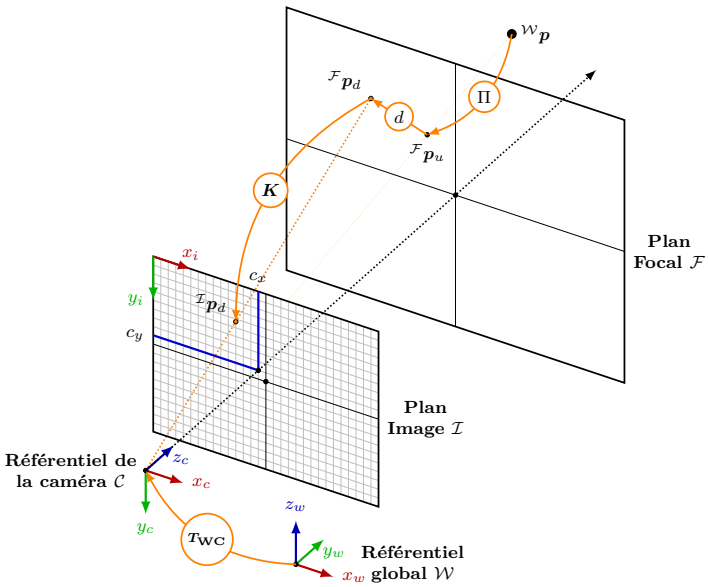
Dans ces travaux, nous nous intéresserons exclusivement au SLAM stéréo-visuel et visio-inertiels. À ce titre, nous détaillons dans les sections suivantes les modèles de mesures que nous utiliserons.

2.2.2 Modélisation d'une caméra

Ci-dessous, nous exposons le modèle de mesure que nous utiliserons décrire le processus de formation des images par la projection d'une structure 3D sur un plan image 2D. Il s'agit du modèle dit *sténopé*. Celui-ci décompose la projection d'un point 3D en trois étapes successives illustrées par la Figure 2.2 : i) la projection sur le plan focal ; ii) la correction due à la distorsion et iii) le passage au plan image.

Considérons un point 3D dont les coordonnées ${}^{\mathcal{C}}\boldsymbol{p} = \begin{bmatrix} {}^{\mathcal{C}}x & {}^{\mathcal{C}}y & {}^{\mathcal{C}}z \end{bmatrix}^\top$ sont exprimées dans le référentiel classique de la caméra tel que représenté sur la Figure 2.2. La première étape consiste à projeter ce point sur le plan focal, également appelé plan image normalisé. On utilise pour cela la fonction perspective [1] :

1. Mentionnons qu'il existe également un modèle de projection stéreographique (Barreto et al. 2001) qui projette d'abord les points 3D sur une sphère unitaire de centre C_1, puis sur le plan image normalisé



La projection se déroule en trois étapes : i) la projection d'un amer sur le plan focal $\mathcal{F}$ par la fonction perspective $\Pi : {}^{\mathcal{C}}\boldsymbol{p} \mapsto {}^{\mathcal{F}}\boldsymbol{p}_u$; ii) l'application du modèle de distorsion $d : {}^{\mathcal{F}}\boldsymbol{p}_u \mapsto {}^{\mathcal{F}}\boldsymbol{p}_d$ et iii) le passage au plan image $\mathcal{I}$: $\boldsymbol{K} : {}^{\mathcal{F}}\boldsymbol{p}_d \mapsto {}^{\mathcal{I}}\boldsymbol{p}_d$.

FIGURE 2.2 – Modèle de caméra sténopé

$$ {}^{\mathcal{F}}\boldsymbol{p}_u = \Pi\left({}^{\mathcal{C}}\boldsymbol{p}\right) = \begin{bmatrix} \frac{{}^{\mathcal{C}}x}{{}^{\mathcal{C}}z} & \frac{{}^{\mathcal{C}}y}{{}^{\mathcal{C}}z} \end{bmatrix}^\top \tag{2.1} $$

Cette fonction modélise l'action des lentilles dans le cadre de l'optique géométrique dérivée des conditions de Gauss. Cependant, ces conditions ne sont valables que dans le cas de rayons paraxiaux de faibles angles d'incidence. En pratique, elles ne sont pas vérifiées, ce qui se traduit par des aberrations géométriques tels que la déformation des droites et que les modèles de distorsion s'efforcent de décrire. La littérature fait état de deux principaux modèles de distorsion : le modèle radial-tangentiel (FRYER et al. 1986) et le modèle équidistant (KANNALA et al. 2006). Ces derniers modélisent les aberrations géométriques à l'aide de polynômes dont il s'agit d'estimer les coefficients. Le modèle radial-tangentiel décompose la distorsion en une composante radiale et une composante tangentielle. Étant donné un point 2D $\boldsymbol{p} = \begin{bmatrix} u & v \end{bmatrix}^\top$, son image $\tilde{\boldsymbol{p}} = \begin{bmatrix} \tilde{u} & \tilde{v} \end{bmatrix}^\top$ par le modèle

grâce un second centre de projection C_2 éventuellement décalé d'un premier du distance ξ selon l'axe z. Ce modèle permet aussi bien de modéliser des projections catadioptriques ($\xi \neq 0$) que de simples projections perspectives ($\xi = 0$ et $C_1 = C_2$).

de distorsion radial-tangentiel est calculée comme suit :

$$\begin{bmatrix}\tilde{u}\\ \tilde{v}\end{bmatrix} = \underbrace{\left(1 + k_1 r^2 + k_2 r^4\right)\begin{bmatrix}u\\ v\end{bmatrix}}_{\text{Composante radiale}} + \underbrace{\begin{bmatrix}2\rho_1 uv + \rho_2(r^2 + 2u^2)\\ 2\rho_2 uv + \rho_1(r^2 + 2v^2)\end{bmatrix}}_{\text{Composante tangentielle}} \tag{2.2}$$

où $r^2 = \|\boldsymbol{p}\|^2$. Ce modèle est adaptée à des distorsions modérées. Dans le cas de fortes distorsion, il est préférable d'utiliser le modèle équidistant ou *fisheye* :

$$\begin{bmatrix}\tilde{u}\\ \tilde{v}\end{bmatrix} = \frac{\theta}{r}\left(k_1 + k_2\theta^2 + k_3\theta^4 + k_4\theta^6\right)\begin{bmatrix}u\\ v\end{bmatrix} \tag{2.3}$$

où $\theta = \text{atan}(r)$ est l'angle d'incidence du rayon. Les modèles radial-tangentiel et équidistant sont donc caractérisés par leur jeu de coefficients respectif : k_1, k_2, ρ_1 et ρ_2 pour le premier, et k_1, k_2, k_3 et k_4 pour le second. Une fois la distortion appliquée, on convertit le point 2D résultant du plan focal dans le plan image. Les paramètres de la transformation entre ces deux plans sont appelés *paramètres intrinsèques*. Classiquement, on note sa matrice de transformation homogène $\boldsymbol{K}$ appelée *matrice de calibration* :

$$\boldsymbol{T}_{\mathcal{IF}} = \boldsymbol{K} = \begin{bmatrix} f_x & & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix} \tag{2.4}$$

où f_x et f_y sont les distances focales selon les axes x et y, et c_x et c_y sont les coordonnées du point principal dans le plan image et s'expriment en pixels. Il s'agit d'un simple changement de repère doublé d'un changement d'échelle (ou d'unité) par le passage de coordonnées métriques aux coordonnées pixelliques.

Dans le cas stéréo-visuel, l'association de deux caméras permet d'obtenir des informations de profondeur. L'analyse des relations géométriques qui lient deux images stéréo, et plus généralement n images observant la même scènes, relève de la géométrie épipolaire (Hartley et al. 2003). Considérons $\mathcal{I}_g$ et $\mathcal{I}_d$ les images gauche et droite d'une même paire stéréo. Lorsqu'on les a préalablement rectifiées – i.e. projetées sur un même plan de telles sortes que leurs lignes épipolaires soient horizontales et correspondent deux à deux –, la profondeur z d'un point est inversement proportionnelle à la disparité $d = |o_g - o_d|$ mesurée entre ses observations gauche $o_g \in \mathcal{I}_g$ et droite $o_d \in \mathcal{I}_d$ sur les images : $z = f_x \cdot b/d$, où f est la distance focale, et b la distance entre les centres optiques des deux caméras.

2.2.3 Modélisation d'une centrale inertielle

Une centrale inertielle associe un accéléromètre et un gyromètre. L'accéléromètre mesure l'accélération spécifique ${}^{\mathrm{B}}\boldsymbol{a}_{\mathrm{WB}}$ du référentiel inertiel B (communément appelé *Body* en anglais) par rapport au référentiel global W, exprimée dans le référentiel inertiel. Cette mesure est corrigée du vecteur de gravité ${}^{\mathrm{W}}\boldsymbol{g}$. De plus, elle est affectée d'un biais dynamique ${}^{\mathrm{B}}\boldsymbol{b}_a$ dont on modélise l'évolution à l'aide d'un processus Gaussien d'écart-type σ_{ba} (appelé *marche aléatoire du biais*). Enfin, elle est entachée d'une erreur de mesure que l'on modélise comme un bruit blanc Gaussien ${}^{\mathrm{B}}\boldsymbol{\eta_a}$ d'écart-type σ_a appelée *densité de bruit*. Il en résulte le modèle de mesure suivant :

$$ {}^{\mathrm{B}}\boldsymbol{a}^m_{\mathrm{WB}} = \boldsymbol{R}^\top_{\mathrm{WB}} \cdot \left({}^{\mathrm{W}}\boldsymbol{a}^t_{\mathrm{WB}} - {}^{\mathrm{W}}\boldsymbol{g} \right) + {}^{\mathrm{B}}\boldsymbol{b_a} + {}^{\mathrm{B}}\boldsymbol{\eta_a} \qquad {}^{\mathrm{B}}\boldsymbol{\eta_a} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma_a^2 \cdot \boldsymbol{I}_3\right) \tag{2.5a} $$

$$ {}^{\mathrm{B}}\dot{\boldsymbol{b}}_{\boldsymbol{a}} = {}^{\mathrm{B}}\boldsymbol{\eta_{ba}} \qquad {}^{\mathrm{B}}\boldsymbol{\eta_{ba}} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma_{ba}^2 \cdot \boldsymbol{I}_3\right) \tag{2.5b} $$

Dans ces équations, le suffixe m désigne la quantité mesurée, et la quantité $\boldsymbol{R}_{\mathrm{WB}}$ est la matrice de rotation entre le référentiel global et le référentiel inertiel.

Le gyromètre mesure la vitesse angulaire $\boldsymbol{\omega}_{\mathrm{WB}}$ entre les référentiels global et inertiel. Son modèle de mesure est relativement semblable à celui de l'accéléromètre. La mesure obtenue est également affectée d'un biais dynamique $\boldsymbol{b_w}$ modélisé comme un processus Gaussien de marche aléatoire σ_{bw}, tandis que l'erreur de mesure est associée à un bruit blanc Gaussien $\boldsymbol{\eta_{bw}}$ de densité de bruit σ_w. Les équations de mesure qui en découlent sont les suivantes :

$$ \boldsymbol{\omega}^m_{\mathrm{WB}} = \boldsymbol{\omega}^t_{\mathrm{WB}} + \boldsymbol{b_w} + \boldsymbol{\eta_w} \qquad \boldsymbol{\eta_w} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma_w^2 \cdot \boldsymbol{I}_3\right) \tag{2.6a} $$

$$ \dot{\boldsymbol{b}}_{\boldsymbol{w}} = \boldsymbol{\eta_{bw}} \qquad \boldsymbol{\eta_{bw}} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma_{bw}^2 \cdot \boldsymbol{I}_3\right) \tag{2.6b} $$

Ainsi, le modèle d'une centrale inertielle est entièrement paramétré par les densités de bruits et les marches aléatoires des biais.

2.3 L'estimation du mouvement et de la carte locale

Le module d'odométrie et de cartographie locale opère directement sur les données fournies par les capteurs. Son but est d'en extraire des contraintes locales pour initialiser les estimées de carte et de trajectoire selon les représentations choisies. C'est à son échelle

un mini algorithme de SLAM qui ne tient compte que des contraintes locales, et recourt éventuellement, à l'échelle locale, aux techniques d'inférences exposées dans la section 2.5. Sa seconde fonction consiste à transmettre aux autres modules des informations dont ils dépendent afin de raffiner ces estimations. Au module de fermeture de boucle, il transmettra par exemple certaines images comme des requêtes pour détecter des fermetures de boucles. Au module d'inférence, il transmettra les contraintes locales qui alimenteront le modèle probabiliste servant de support à l'inférence. La Figure 2.3 donne l'architecture générale d'un module d'odométrie et de cartographie locale dans le cas visio-inertiel. Il comprend deux modules internes aux rôles complémentaires. Schématiquement, le premier estime la trajectoire sachant la carte, tandis que le second met à jour la carte sachant la trajectoire.

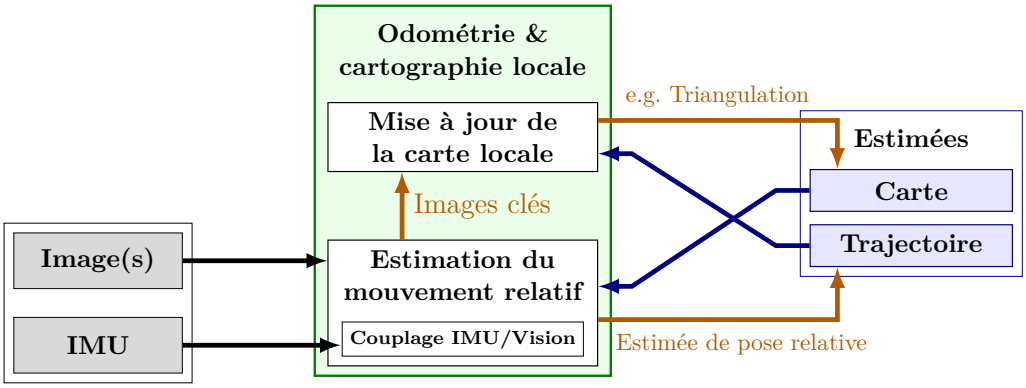


FIGURE 2.3 – Organisation générale d'un algorithme d'odométrie et de cartographie locale dans le cas visio-inertiel

Dans cette section, nous exposons les principes de base de l'odométrie visuelle et visio-inertielle.

2.3.1 Principes de l'odométrie visuelle

Les mesures reçues servent d'abord à l'estimation du mouvement (SCARAMUZZA et al. 2011 ; FRAUNDORFER et al. 2012). Les algorithmes d'odométrie visuel se distinguent selon la nature et la portion de l'information qu'il exploitent dans les images. Les méthodes qui utilisent l'ensemble des pixels sont dites *denses* tandis que celles qui se restreignent à un sous-ensemble de pixels clairsemés sont dites *éparses*. Il existe également des méthodes intermédiaires, dites *semi-denses*, qui opèrent sur des sous-ensemble connexes de pixels.

Les méthodes directes

Les méthodes *directes* exploitent l'information photométriques en opérant directement sur les pixels. Les méthodes directes denses telles que DTAM (Newcombe et al. 2011a) et LSD-SLAM (Engel et al. 2014) construisent un modèle dense de la scène à l'aide d'informations de profondeur rétro-projetées telles que des cartes de profondeur (elles-mêmes estimées (Engel et al. 2013) dans le cas de LSD-SLAM), et utilisent ce modèle pour projeter chaque pixel d'une image acquise au temps t dans l'image acquise au temps $t + 1$. Elles estiment alors la transformation relative entre les deux poses de caméras associées comme étant celle qui minimise l'erreur photométrique de cette reprojection, c'est-à-dire les différences d'intensité pixel par pixel (ou erreur de superposition). Ces méthodes jouissent d'une grande robustesses dans les environnements peu texturés et ne nécessitent pas d'identifier ni de suivre des amers précis. Cependant, elles requièrent une charge de calcul conséquente, éventuellement amoindrie par les techniques de vectorisation (Kerl et al. 2013 ; Forster et al. 2014) permises par la cartes graphiques (*Graphical Processing Units* (GPU) en anglais).

Les méthodes indirectes

Les méthodes indirectes estiment le mouvement à partir des primitives qu'elles extraient des images afin d'en résumer l'information géométrique. Il s'agit majoritairement de points saillants – tels que des coins – suivis dans les images successives, mais d'autres primitives plus complexes telles que des lignes sont utilisées (He et al. 2018). Un grand nombre d'extracteurs ont été proposées dans la littérature tels que les extracteurs de Harris (Harris et al. 1988), FAST (Rosten et al. 2006), STAR (Agrawal et al. 2008) et SURF (Bay et al. 2006) et SIFT (Lowe 2004). La qualité d'un détecteur de primitives découle avant tout de son invariance aux transformations géométriques (translation, rotation, échelle) et photométriques (changement d'intensités, spécularités etc.). La bonne détection des primitives dépend notamment du niveau de contraste dans les images. Celui-ci peut être renforcé à l'aide de divers pré-traitement tels que le moyennage ou encore l'égalisation d'histogrammes (Zuiderveld 1994). Enfin, au-delà du nombre des primitives, l'information qu'elles apportent collectivement pour l'estimation du mouvement dépend de leur distribution au sein des images. C'est pourquoi certains algorithmes, tels que Vins-Mono (Mur-Artal et al. 2015) et OKVIS (Leutenegger et al. 2015), imposent des distances minimales entre les primitives extraites à l'aide de techniques de

masquage afin d'uniformiser leur répartition spatiale.

La seconde étape consiste à détecter des correspondances 2D-2D entre les primitives des images consécutives. Deux techniques prévalent dans la littérature. La première est le suivi (ou *tracking* en anglais) direct qui apparient les primitives selon la similarité photométrique de leur voisinage. Une solution commode pour ce faire consiste à calculer le flot optique[2] en chaque primitive pour prédire leur position dans l'image suivante (Baker et al. 2004). C'est l'idée de l'algorithme de *Kanade-Lucas-Tomasi* (KLT) (Tomasi et al. 1991 ; Shi 1994) qui applique cette méthode de suivi aux coins de Harris. Cette méthode de suivi est par exemple utilisée par Vins-Mono (Qin et al. 2018b) et Rovio (Bloesch et al. 2015). À l'opposé, le tracking indirect apparie les primitives à l'aide de descripteurs. Ce sont des vecteurs à valeurs binaires ou numériques, qui caractérisent une primitive par les propriétés de son voisinage local. On trouve des descripteurs numériques basés Histogrammes de Gradients Orientés (HOG) tels que SIFT (Lowe 2004) et SURF (Bay et al. 2006) qui rendent compte de la distribution des orientations des gradients dans le voisinage de la primitive, et des descripteurs binaires tels que BRIEF (Calonder et al. 2010), ORB (Rublee et al. 2011), BRISK (Leutenegger et al. 2011) et FREAK (Alahi et al. 2012) basés sur des comparaisons d'intensités de paires de pixels échantillonnées dans son voisinage. D'une image à la suivante, chaque descripteur est classiquement apparié à son plus proche voisin. Des structures telles que des arbres Kd (Bentley 1975) permettent notamment d'accélérer cette recherche des correspondances. Les correspondances erronées sont éventuellement filtrées par des techniques de correspondances croisées ou le calcul de matrices fondamentales (Hartley et al. 2003) dans une boucle de RANSAC (Fischler et al. 1981).

Enfin, l'ultime étape utilise les correspondances 2D-2D détectées pour estimer la pose relative entre les images successives en minimisant une erreur géométrique associée. Par le calcul (Longuet-Higgins 1981 ; Nistér 2004) puis la décomposition de la matrice fondamentale, les méthodes 2D-2D estiment une transformation relative à un facteur d'échelle près. Pour apporter l'information de profondeur, les méthodes 3D-2D exploitent les correspondances visio-structurelles ainsi dérivées et minimisent l'erreur de re-projection des amers observés dans la première image. Il s'agit d'un problème dit *Perspective-N-Points* (PnP) (Hartley et al. 2003), classiquement résolu dans une boucle de RANSAC pour atténuer l'influence des correspondances aberrantes. Cette méthode est la plus utilisée dans

2. Par analogie avec la mécanique des fluides, le flot optique associe à chaque point d'une image un vecteur vitesse 2D, sous hypothèse de petits déplacements et d'une intensité lumineuse constante.

les algorithmes d'odométrie visuelle en raison de sa robustesse. Enfin, le dernier paradigme d'estimation du mouvement exploite des correspondences 3D-3D. Celui-ci suppose qu'un nuage de points puisse être triangulé à partir de chaque image, comme dans les cas stéréo-visuels et RGB-D. La pose relative estimée est alors celle qui permet de minimiser la distance entre les points 3D mis en correspondance : cela correspond à l'algorithme *Iterative Closest Point* (ICP) (Besl et al. 1992). Néanmoins, cette méthode est moins robuste puisqu'elle intègre également les erreurs d'estimation des positions des points 3D.

2.3.2 Le couplage visio-inertiel

Dans le cas monoculaire, l'information visuelle suffit pour estimer le mouvement et reconstruire l'environnement à un facteur d'échelle près. Un couplage avec des mesures inertielles apporte l'information complémentaire manquante pour rendre ce facteur d'échelle observable. Dans la littérature, on distingue les couplages *lâches* (*loose*) des couplages *serrés* (*tight*). Les premiers estiment d'abord le mouvement de façon disjointe avec les mesures visuelles d'une part, et des mesures inertielles d'autre part, puis fusionne les deux estimations. Dans cet esprit, le framework proposé par (Lynen et al. 2013) expose une méthode générique pour la fusion lâche de mesures issues de différents capteurs, sans se limiter aux IMUs et aux caméras. Les mesures inertielles sont alors utilisées dans l'étape de prédiction du filtre de Kalman, tandis que la pose estimée à partir des images est intégrée dans l'étape de correction. Ce schéma est également repris par (Weiss et al. 2012), ou encore (Grabe et al. 2013) qui corrigent la prédiction inertielle du mouvement par son estimation basée sur un suivi par flot optique. Les publications récentes s'intéressent davantage aux couplages serrés, considérant que le traitement parallèle des mesures visuelles et inertielles est redondant et qu'elles pourraient être fusionnées beaucoup plus précocément. De telles méthodes sont certes plus coûteuses, mais elles permettent une estimation d'emblée plus précise, et fournissent un moyen supplémentaire de filtrer les faux positifs dans les correspondances visuelles. Ainsi, OKVIS (Leutenegger et al. 2015) substitue à la boucle RANSAC classique un test du χ^2 sur les coordonnées des primitives dans les images en utilisant la pose prédite par l'intégration des mesures inertielles. Des algorithmes de fusion serrée ont été proposés aussi bien pour des méthodes basées filtrage telles que MSCKF (Mourikis et al. 2007) ou Rovio (Bloesch et al. 2015) que pour des méthodes basées optimisation telles que ORB-SLAM dans sa version visio-inertielle (Mur-Artal et al. 2017b), Vins-Mono (Qin et al. 2018b) et OKVIS (Leutenegger et al. 2015). Dans ces méthodes, la nouvelle pose est d'abord prédite en intégrant les

mesures inertielles, dont l'incertitude reste suffisamment bornée sur l'horizon temporelle entre deux images consécutives. Elles sont alors utilisées pour guider l'odométrie visuelle, en prédisant les positions des points suivis dans la nouvelle image.

Le couplage des mesures visuelles et inertielles amènent cependant des difficultés qui lui sont spécifiques. Tout d'abord, l'intégration des mesures inertielles est une opération délicate et non-linéaire qui peut être traitée par des méthodes classiques d'intégration numérique (Cartwright et al. 1992) ou des techniques de pré-intégration (Lupton et al. 2011) telles que discutées dans la section 2.5.3. De plus, le couplage visio-inertiel complexifie le problème d'estimation. En effet, il impose d'une part l'estimation supplémentaire des états inertiels qui interviennent explicitement dans les modèles de mesures exposés dans la section 2.2.3, c'est-à-dire les vitesses et les biais inertiels. De plus, l'observabilité de ces biais dépend de la cinématique de la trajectoire car elle requiert une excitation suffisante de l'IMU. D'autre part, un estimateur visio-inertiel nécessite une bonne initialisation (*bootstrapping*) des états inertiels, de la direction du vecteur gravité ainsi que du facteur d'échelle. La méthode communément adoptée procède en deux étapes. Dans un premier temps, un modèle 3D est estimé par des techniques de reconstruction par le mouvement en exploitant exclusivement les mesures visuelles issues des premières images, tandis que les mesures inertielles sont pré-intégrées à biais nuls. Dans la seconde étape, les biais du gyromètre, le vecteur gravité et le facteur d'échelle puis le biais de l'accéléromètre sont successivement initialisés en ajustant mutuellement le modèle reconstruit et les pré-intégrations inertielles. Ce processus est détaillé dans (Qin et al. 2018b) et (Mur-Artal et al. 2017b). La dernière difficulté intrinsèque au couplage visio-inertiel provient de la synchronisation temporelle nécessaire entre les horloges de l'IMU et des caméras, là où seul l'ordonnancement des images importe dans une odométrie purement visuelle.

2.3.3 La cartographie locale

L'estimation du mouvement s'appuie sur la cartographie récente (dite locale). Celle-ci sert de support à la relocalisation locale, que ce soit en projetant les pixels d'une image dans la suivante dans le cas direct, ou pour formuler des problèmes PnP dans le cas indirect. Dans le cas visuel, la carte locale est mise à jour à partir des images-clés sélectionnées le long de la trajectoire de façon à minimiser leur redondance. Initiée dans l'algorithme PTAM (Klein et al. 2007), cette technique s'est ensuite étendue à l'ensemble des méthodes de SLAM visuel. Généralement, une nouvelle image-clé est sélectionnée sur la base

de critères visuels, lorsque trop peu de points de l'image-clé précédente sont encore suivis, ou que la parallaxe moyenne – compensée des mouvements de rotations – suffit pour trianguler les amers observés. En contexte visio-inertiel s'ajoute un critère temporel qui limite l'intervalle de temps entre deux images clés successives afin de borner l'incertitude sur l'intégration des mesures inertielles qui les lient. Les informations visuelles des images intermédiaires sont le plus souvent supprimées, bien que certaines méthodes comme CK-LAM (Nerurkar et al. 2014) proposent de les marginaliser de façon consistante sans accroître la complexité du problème d'inférence. Les algorithmes d'odométrie dense telles que DTAM (Newcombe et al. 2011a) et LSD-SLAM (Engel et al. 2014) reposent sur une cartographie dense mise à jour à partir de cartes de profondeur. Formellement, l'étendue de la carte locale correspond à une fenêtre glissante d'images-clés. ORB-SLAM (Mur-Artal et al. 2015) la matérialise à l'aide d'un graphe de covisibilité qui lie les images partageant suffisamment d'observations communes. Dans cette méthode, la carte locale est gérée et raffinée par des inférences locales dans un processus à part entière appelé à l'ajout de chaque nouvelle image-clé.

2.4 La détection des fermeture de boucles

La trajectoire estimée par un algorithme d'odométrie est inéluctablement sujette à une dérive. Celle-ci résulte de la propagation de petites erreurs d'estimation, et se trouve exacerbée en présence de mouvements agressifs ou de mouvements de rotation rapides qui compromettent la qualité du tracking. Afin de l'amoindrir, un second module détecte et caractérise des fermetures de boucles, qui dérivent de la ré-observation de zones précédemment cartographiées. Celles-ci seront ensuite répercutées sur la trajectoire afin d'atténuer la dérive le long des boucles ainsi fermées. Nous présentons ici les principes généraux de la détection et de la caractérisation des fermetures de boucle en contexte visuel.

En contexte visuel, les méthodes de détection de fermetures de boucles utilisent des techniques de reconnaissance de lieu (Lowry et al. 2015) : l'image courante est comparée à une banque d'images construite au fil des acquisitions afin de repérer les images les plus semblables. Pour ce faire, la méthode classique consiste à quantifier cette similarité en comparant les signatures des images par rapport à un vocabulaire visuel. Initiée par FABMAP (Cummins et al. 2008) pour les descripteurs numériques, puis étendue aux descripteurs binaires par (Galvez-Lopez et al. 2011), cette technique a démontré son habilité à fermer des boucles le long de trajectoires de plusieurs centaines de kilomètres.

Un vocabulaire visuel correspond à une discrétisation de l'espace de description du descripteur utilisé pour le suivi indirect des primitives. Sa construction nécessite une phase d'apprentissage sur une large banque d'images suffisamment large et représentative des scènes observées. Les descripteurs sont alors extraits de chaque image, puis on partitionne ensuite le nuage de points obtenus par k-means. Les centres de Voronoï des cellules ainsi constituées définissent autant de *mots visuels*. En réduisant préalablement la dimension des descripteurs à l'aide d'une matrice projection calculée hors ligne par une analyse en composantes principales de la base d'apprentissage [3], puis en partitionnant l'espace de description en deux sous-espaces quantifiés individuellement pour former un vocabulaire produit, (Lynen et al. 2015) obtiennent des vocabulaires à la fois plus compacts et plus discriminants. Enfin, pour améliorer l'efficacité des recherches, (Nistér et al. 2010) ont proposé de structurer les vocabulaires visuels de façon hiérarchique sous forme d'arbres.

Le vocabulaire visuel sert de support à la détection des images candidates aux fermetures de boucle. On associe à chaque image une signature visuelle qui correspond à l'histogramme des mots visuels qu'elle contient. Cette signature est appelée un vecteur sac-de-mots. Elle consiste en un vecteur épars dont la $i^{\text{ème}}$ entrée indique le nombre d'occurrence du $i^{\text{ème}}$ mot du vocabulaire dans l'image. On quantifie la similarité entre deux images par la distance entre leurs signatures visuelles respectives. Des scores alternatifs tels que TF-IDF (Sivic et al. 2003) permettent de tenir compte de la discriminance des mots visuels dans la base d'apprentissage. Décrire et comparer les images ne suffit pas, il faut trouver rapidement tous les candidats semblables à l'image de requête. Une première technique consiste à utiliser un indexage inversé, c'est-à-dire associer chaque mot visuel à l'ensemble des images qui le contiennent. On peut ainsi restreindre les comparaisons aux seules images qui partagent au moins un mot visuel en commun avec l'image de requête. On retrouve ces techniques dans la majorités des algorithmes de SLAM actuels, tels que ORB-SLAM (Mur-Artal et al. 2015) et Vins-Mono (Qin et al. 2018b). Profitant de la discriminance accrue de ses descripteurs quantifiés, la méthode proposée par (Lynen et al. 2015) adopte une approche granulaire : chaque descripteur de requête est projeté puis quantifié, ses k mots visuels les plus proches sont repérés par un arbre Kd (Bentley 1975), puis enfin, il est comparé en force brute à tous les descripteurs associés à ces mots par un indexage inversé. Notons enfin que des descripteurs globaux peuvent être générés à l'aide de méthodes d'apprentissage automatique, telles que NetVLAD (Arandjelovic et al. 2016).

3. Notons que si le descripteur original est binaire, sa projection est numérique.

Une fois les candidats recensés, l'étape suivante consiste à vérifier qu'ils soient plausibles à l'aide de critères topologiques. L'idée sous-jacente est que dans le cas d'une fermeture de boucle valide, il doit exister une continuité spatiale entre plusieurs candidats. Ainsi, ORB-SLAM impose par exemple qu'au moins trois images clés consécutives figurent parmi les candidats retournés pour envisager une fermeture de boucle valide avec l'une d'entre elles. Dans la méthode proposée par (Lynen et al. 2015), les images clés candidates sont regroupés selon qu'elles partagent des observations communes ; les faux positifs sont filtrés par covisibilité comme proposé par (Sattler et al. 2012). On vérifie alors que l'agglomérat le plus important contiennent suffisamment d'images clés covisibles. Si tel est le cas, l'image la plus représentée en nombre d'observations est retenue comme candidate.

Enfin, l'ultime étape consiste à caractériser géométriquement la fermeture de boucle à partir des correspondances détectées. Descripteur par descripteur, on apparie les primitives des deux images mises en correspondance, puis on estime la pose relative entre les deux images en résolvant un problème PnP dans une boucle de RANSAC. Cette vérification géométrique permet notamment de rejeter la fermeture de boucle si trop peu de correspondances corroborent la pose relative estimée. Si la fermeture de boucle est validée, alors les amers correspondants sont éventuellement fusionnés. En particulier, ORB-SLAM projette la carte dans l'image retenue afin de trouver davantage de correspondances 3D-2D et de préciser la pose relative estimée. Une fermeture de boucle fructueuse ajoute une contrainte globale à la carte. On la répercute sur la trajectoire à l'aide de techniques d'inférences (c.f. section 2.5).

La détection, la caractérisation et la répercussion des fermetures de boucles occupent un pan entier de la littérature sur le SLAM. En contexte visuel, toutes les méthodes développées s'inspirent de la procédure en trois étapes décrite ci-dessus. Il existe également des méthodes dans lesquelles les fermetures de boucles ne sont pas détectées à partir de critères visuels. C'est par exemple le cas dans l'algorithme de SLAM stéréo-visuel développé par (Brand et al. 2014). Ce dernier segmente la carte en une succession de sous-cartes le long de la trajectoire. Chaque sous-carte est associée à un nuage de points dense. On détecte alors les fermetures de boucles en recalant les sous-cartes dont les nuages de points se recouvrent significativement. Dans tous les cas, il est primordial de filtrer autant que possible les faux positifs en amont. En effet, s'il est malheureux de manquer une fermeture de boucle, en intégrer une erronée risque de corrompre durablement la consistance de la carte. Des procédures existent néanmoins pour limiter l'impact de fermetures de boucles douteuses lors des étapes d'inférence ultérieures, à l'instar des *switch constraints* intro-

duites par (Sünderhauf et al. 2013), du modèle *max-mixture* (Olson et al. 2013b), ou encore de l'algorithme *Realizing, Reversing, Recovering* (RRR) de (Latif et al. 2013). Le rôle des fermetures de boucles ne se cantonne pas à l'adjonction de nouvelles contraintes globales ; elles permettent également de relocaliser le robot lorsque celui-ci perd le tracking, ce qu'on appelle également le problème du robot kidnappé. Cette fonctionnalité est par exemple implémentée dans Vins-Mono (Qin et al. 2018b). Classiquement, la recherche des fermetures de boucles est passive, mais lorsque le SLAM est couplé à un algorithme de contrôle qui planifie la trajectoire de façon à fermer des boucles, on parle alors de SLAM actif (Feder et al. 1999).

2.5 L'inférence de la carte et de la trajectoire

2.5.1 Les paradigmes d'inférences

L'inférence est le processus qui consiste à estimer les variables de trajectoire et d'environnement à partir des contraintes locales et globales qui les lient. Elle se recontre à de multiples échelles dans un algorithme de SLAM : l'inférence locale opère sur une portion locale de la carte et de la trajectoire tandis que l'inférence globale s'effectue à l'échelle de la trajectoire entière. La littérature fait état de trois paradigmes d'inférence : les approches basées filtrage, les approches basées optimisation et les approches ensemblistes. Les deux premières approches dérivent d'un cadre Bayésien tandis que la seconde hérite du calcul par intervalles. Dans la suite, nous nous intéresserons exclusivement aux approches basées optimisation que nous présentons dans la section 2.5.2.

Le filtrage est l'estimation récursive d'une distribution de probabilité [4]. L'EKF-SLAM, qui résulte de l'application directe du filtrage de Kalman (Kalman 1960) au SLAM, constitue l'approche historique au problème d'inférence en SLAM. Il consiste à estimer récursivement la pose courante du robot ainsi que les positions des amers qu'il observe. Les filtres d'informations, tels que les *Sparse Extended Kalman Filters* (Thrun et al. 2004), ont apporté une formulation duale populaire du fait des propriétés additives des matrices d'information. Aujourd'hui, divers estimateurs visio-inertiels, utilisés notamment dans le

4. Le filtrage est dit paramétrique lorsque la distribution est décrite par un jeu de paramètres. Le filtrage se réduit alors à l'estimation récursive de ces seuls paramètres, tels que la moyenne et la covariance dans le cas Gaussien linéaire des filtres de Kalman. Dans le cas non-paramétrique, on approxime cette distribution par une méthode de Monte-Carlo doublée d'un mécanisme de ré-échantillonnage d'importance séquentiel. On parle alors de filtres particulaires.

module d'odométrie et de cartographie locale, relèvent d'une approche filtrage, tels que le *Multi-State Constraint Kalman Filter* (MSCKF) (Mourikis et al. 2007) (dont une implémentation est proposée par (Geneva et al. 2019)), Rovio (Bloesch et al. 2015) et Vins-Mono (Qin et al. 2018b). Néanmoins, le filtrage présente trois limitations majeures. Il est très sensible aux mauvaises associations de données, affiche une complexité quadratique avec le nombre d'amers observés, et pâtit de problèmes de consistance (Bailey et al. 2006a) qui se traduisent par une sous-estimation des matrices de covariance, principalement occasionnées par les erreurs de linéarisation successives et le processus de linéarisation lui-même qui altère artificiellement l'observabilité du système (Huang et al. 2010b). Diverses solutions ont été proposées pour y remédier telles que des estimations sur fenêtres glissantes (Mourikis et al. 2007), une meilleure sélection des points de linéarisation (Huang et al. 2010b), des paramétrisations différentes des amers (Civera et al. 2008), des formulations *sans parfums* (Uhlmann 1995) ou invariantes (Barrau et al. 2015). Enfin, les filtres particulaires, tels que FastSLAM (Montemerlo et al. 2003), ont apporté une alternative non-paramétrique à l'estimation d'état, dont l'efficacité rivalise avec les approches basées EKF (Calonder 2006).

Un second paradigme d'inférence est le SLAM ensembliste : encore marginal à ce jour, il s'écarte du cadre Bayésien et emprunte au calcul d'intervalles. Le degré de connaissance sur une variable est alors modélisé par un intervalle auquel sa vraie valeur est supposée appartenir de façon certaine. Ainsi, les vecteurs sont caractérisés par des produits cartésiens d'intervalles, tandis que la trajectoire d'un robot est modélisée par une *tube*, qui à chaque instant associe l'intervalle correspondant pour les variables de trajectoires. Les opérations classiques d'intégrations du modèle dynamique du robot sont alors transposées à l'intégration d'intervalles, tandis que les observations des amers sont répercutées en élimant le tube des trajectoires rendues infaisables par les nouvelles contraintes. Cette approche est notamment appliquée dans le cadre du SLAM sous-marin dans les travaux de (Rohou et al. 2019). Compliquée à mettre en place, elle jouit néanmoins d'une très bonne consistance, et ne requiert aucune technique de linéarisation.

2.5.2 Le SLAM basé optimisation

Ce paradigme d'inférence relève de l'estimation par maximum de vraisemblance. Dans cette approche, on modélise le degré de connaissance fourni par un ensemble d'observations $\mathcal{Z}$ sur les variables Θ de la carte et de la trajectoire par une distribution $p(\Theta|\mathcal{Z})$, dite

distribution a posteriori. Le problème d'estimation revient alors à choisir Θ de façon à maximiser cette distribution (estimation par Maximisation A Posteriori (MAP)) :

$$\hat{\Theta} = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\min}\ p(\Theta|\mathcal{Z}) = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\min}\ p(\mathcal{Z}|\Theta) \cdot p(\Theta) \tag{2.7}$$

Par conséquence du théorème de Bayes, cela revient également à maximiser la vraisemblance des mesures $p(\mathcal{Z}|\Theta)$ en présence d'un a priori $p(\Theta)$. En l'absence d'a priori, lorsque $p(\Theta)$ équivaut à une distribution uniforme [5], le problème se réduit à une estimation du maximum de vraisemblance. Son application au SLAM est introduite par (Lu et al. 1997) puis (Gutmann et al. 1999).

On construit le modèle $p(\Theta|\mathcal{Z})$ incrémentalement au fil des observations. Sous l'hypothèse communément admise d'indépendance des mesures, cette distribution se factorise par rapport aux mesures, permettant ainsi d'isoler la contribution de chaque observation :

$$\hat{\Theta} = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\max}\ p(\Theta) \prod_{z \in \mathcal{Z}} p(z|\Theta) = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\min} \{-\log p(\Theta)\} + \sum_{z \in \mathcal{Z}} \{-\log p(z|\Theta)\} \tag{2.8}$$

Dans le cas Gaussien, $p(z|\Theta) = \mathcal{N}\left(\boldsymbol{\xi}_z(\Theta); \mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\xi}_z}\right)$ où $\boldsymbol{\xi}_z(\Theta)$ désigne le résidu associé à la mesure $z \in \mathcal{Z}$ et quantifie une erreur entre une observation et un modèle, et $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z}$ est sa matrice de covariance associé au résidu. Le terme a priori vaut $p(\Theta) = \mathcal{N}\left(\boldsymbol{\xi}_{\tilde{\Theta}}(\Theta); \mathbf{0}; \boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\tilde{\Theta}}}\right)$. On en dérive un problème d'optimisation au sens des moindres carrés :

$$\hat{\Theta} = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\min}\ \left\{ \left\| \boldsymbol{\xi}_{\tilde{\Theta}}(\Theta) \right\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\tilde{\Theta}}}} + \sum_{z \in \mathcal{Z}} \left\| \boldsymbol{\xi}_z(\Theta) \right\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z}} \right\} \tag{2.9}$$

La norme de Mahalanobis permet ici de normaliser les résidus en fonction des incertitudes estimées dans les différentes directions. Pour accroître la résilience aux mesures aberrantes (i.e. aux mauvaises associations de données), la norme $\mathcal{L}^2$ peut être remplacée par des pénalisations robustes telles que la fonction d'Huber (Huber 1992). On parle alors de M-estimateur. Dans la suite, on confondra les observations et les facteurs qu'elles engendrent.

Le problème formulé par l'équation (2.8) peut s'interpréter en termes d'inférence sur un graphe de facteurs tel qu'introduit par (Kschischang et al. 2001), et dont un exemple est donné par la Figure 2.4 dans le cas visio-inertiel. Il s'agit d'un graphe bipartite dans lequel chaque nœud représente une variable $\theta \in \Theta$ à estimer, tandis que chaque observation $z \in \mathcal{Z}$ est associée à "facteur", qui intervient explicitement dans la factorisation (2.8). Le

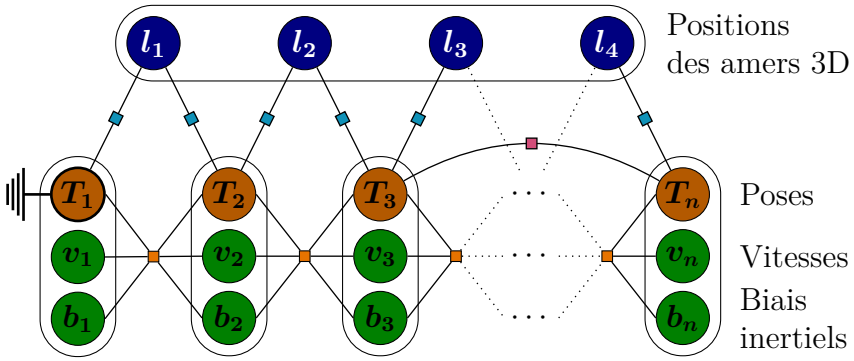
5. Dans le cas d'un espace d'hypothèses $\mathcal{H}_\Theta$ non borné, l'a priori uniforme sera impropre.

facteur qui en dérive est relié à l'ensemble Θ_z des variables sur lesquelles il porte. Cette représentation permet non seulement de visualiser la structure du problème d'inférence de façon intuitive, mais elle offre également un cadre flexible et générique pour englober des contraintes hétérogènes. Elle invite également à découvrir et exploiter les liens entre les propriétés topologiques du graphe et le problème d'inférence qui en découle : (Khosoussi et al. 2014) étudient par exemple l'impact du nombre d'arbres couvrants dans le graphe ou encore du degré moyen des nœuds sur la qualité de l'estimée obtenue.

On résout généralement le problème (2.9) à l'aide des algorithmes de Gauss-Newton ou de Levenberg-Marquardt (voir l'Annexe B). Ce sont des méthodes itératives qui procèdent en minimisant des approximations quadratiques successives de la fonction de coût. Chaque itération donne lieu à la résolution d'un système linéaire appelé *équations normales*, par exemple à l'aide de méthodes de gradients conjugués accélérées par préconditionnement (Byröd et al. 2010). Des implémentations efficaces de tels solveurs ont été proposées telles que Ceres (Agarwal et al. 2012), `g2o` (Kümmerle et al. 2011) et `GTSAM` (Dellaert 2012), et une introduction à l'optimisation graphique sur variété différentielle est proposée par (Grisetti et al. 2010a). Il est essentiel de constater que la structure de la matrice, dite Hessienne, qui intervient dans les équations normales, reflète celle du graphe de facteurs sous-jacent. Elle présente par conséquent une structure éparse, ce qui implique que l'intégration de nouvelles mesures dans l'optimisation n'impacte souvent qu'une faible portion des variables du problème. Ce constat motive l'utilisation de solveurs incrémentaux qui exploitent les résultats des optimisations précédentes afin de circonscrire les mises à jour. C'est le cas d'iSAM (Kaess et al. 2008) qui résout les équations normales par décomposition QR, et actualise directement cette décomposition seulement vis-à-vis des variables nécessitant une relinéarisation. D'autres approches permettent d'accroître l'efficacité des inférences, telles que l'optimisation hiérarchique proposée par (Grisetti et al. 2010b) et (Suger et al. 2014). Formuler le problème d'estimation sous la forme d'un problème d'optimisation permet une meilleure gestion des non-linéarités, et in fine une estimation plus précise que dans le cas du filtrage de Kalman.

2.5.3 L'inférence visio-inertielle

Dans cette section, et dans la continuité de la précédente, nous présentons le cas spécifique des graphes de facteurs visio-inertiels, et détaillons les termes de résidus que nous utiliserons par la suite.



Les facteurs inertiels ■, visuels ■ et de fermetures de boucles ■ encodent des contraintes stochastiques entre les variables de pose ●, inertielles ● et les positions de amers 3D ●. On fixe les degrés de liberté de jauge en figeant une variable de pose. Lorsque seuls les facteurs visuels sont considérés dans l'inférence, on parle d'ajustement de faisceaux (Triggs et al. 1999) tel qu'utilisé en reconstruction par le mouvement.

Figure 2.4 – Structure d'un graphe de facteurs visio-inertiel.

La trajectoire est classiquement discrétisée en une séquence d'image-clés. Chaque image-clé $\mathcal{I}_k$, où k désigne l'indexage temporel, est décrite par sa pose T_{WB_k}, son vecteur vitesse ${}^{\mathrm{W}}\boldsymbol{v}_{\mathrm{WB}_k}$ et son vecteur ${}^{\mathrm{B}_k}\boldsymbol{b}_k$ des biais de l'accéléromètre et du gyroscope de la centrale inertielle. Dans les notations précédentes, W désigne le référentiel global tandis que B désigne le référentiel inertiel. L'environnement est représenté par un nuage éparse d'amers 3D dont les positions sont estimées relativement au référentiel inertiel de la première image-clé depuis laquelle ils ont été observés. Comme décrit par la Figure 2.4, toutes ces variables sont mutuellement contraintes par des facteurs Gaussiens inertiels, visuels et de fermeture de boucle, que nous explicitons ci-dessous.

Les facteurs inertiels

Une mesure inertielle $\boldsymbol{u}_k$ consiste en la donnée d'une vecteur d'accélération spécifique ${}^{\mathrm{W}}\boldsymbol{a}_{\mathrm{WB}}$ et de vitesse angulaire $\boldsymbol{\omega}_{\mathrm{WB}}$. L'intégration des mesures inertielles dépend du modèle cinématique du robot régit par les trois équations suivantes :

$$\dot{\boldsymbol{R}}_{\mathrm{WB}} = \boldsymbol{R}_{\mathrm{WB}}^{\top} \cdot [\boldsymbol{\omega}_{\mathrm{WB}}]_{\times} \qquad {}^{\mathrm{W}}\dot{\boldsymbol{t}}_{\mathrm{WB}} = {}^{\mathrm{W}}\boldsymbol{v}_{\mathrm{WB}} \qquad {}^{\mathrm{W}}\dot{\boldsymbol{v}}_{\mathrm{WB}} = {}^{\mathrm{W}}\boldsymbol{a}_{\mathrm{WB}} \tag{2.10}$$

où $\boldsymbol{R}$, $\boldsymbol{t}$ et $\boldsymbol{v}$ désignent respectivement une matrice de rotation, un vecteur de translation et un vecteur vitesse entre les référentiels indiqués. On note que les grandeurs d'accélération et de vitesse angulaire interviennent directement dans ces équations. Cependant, les signaux de l'IMU sont bruités et biaisés comme en témoigne son modèle de mesure que

nous reproduisons ci-dessous conformément aux équations (2.5) et (2.6) :

$$\begin{aligned} \text{Accéléromètre :} \quad & {}^{\mathrm{B}}\boldsymbol{a}^{m}_{\mathrm{WB}} = \boldsymbol{R}^{\top}_{\mathrm{WB}} \cdot \left({}^{\mathrm{W}}\boldsymbol{a}^{t}_{\mathrm{WB}} - {}^{\mathrm{W}}\boldsymbol{g} \right) + {}^{\mathrm{B}}\boldsymbol{b}_{\boldsymbol{a}} + {}^{\mathrm{B}}\boldsymbol{\eta}_{\boldsymbol{a}} \\ \text{Gyromètre :} \quad & \boldsymbol{\omega}^{m}_{\mathrm{WB}} = \boldsymbol{\omega}^{t}_{\mathrm{WB}} + \boldsymbol{b}_{\boldsymbol{w}} + \boldsymbol{\eta}_{\boldsymbol{w}} \\ \text{Biais inertiels :} \quad & \dot{\boldsymbol{b}}_{\boldsymbol{w}} = \boldsymbol{\eta}_{\boldsymbol{bw}} \quad \text{et} \quad {}^{\mathrm{B}}\dot{\boldsymbol{b}}_{\boldsymbol{a}} = {}^{\mathrm{B}}\boldsymbol{\eta}_{\boldsymbol{ba}} \end{aligned} \tag{2.11}$$

où les vecteurs de biais et de bruit sont respectivement notés $\boldsymbol{b}$ et $\boldsymbol{\eta}$.

Une première formulation des résidus inertiels consiste à comparer la pose et les états inertiels au temps $k+1$ à leur prédiction obtenue à l'aide de méthode de Runge-Kutta (Cartwright et al. 1992) à partir des états au temps k. Cela suppose de propager les matrices de covariances à travers l'intégration afin d'associer une matrice de covariance $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\boldsymbol{u}_k}}$ à la prédiction. Le résidu inertiel vaut alors :

$$\boldsymbol{\xi}_{\boldsymbol{u}_k}(\Theta) = \begin{bmatrix} T_{\mathrm{WB}_{k+1}} \boxminus \hat{T}^{-1}_{\mathrm{WB}_{k+1|k}} \\ {}^{\mathrm{W}}\boldsymbol{v}_{\mathrm{B}_{k+1}} - {}^{\mathrm{W}}\hat{\boldsymbol{v}}_{\mathrm{B}_{k+1|k}} \\ {}^{\mathrm{B}_{k+1}}\boldsymbol{b}_{k+1} - {}^{\mathrm{B}_k}\boldsymbol{b}_k \end{bmatrix} \tag{2.12}$$

où $\hat{T}_{\mathrm{WB}_{k+1|k}}$ et ${}^{\mathrm{W}}\hat{\boldsymbol{v}}_{\mathrm{B}_{k+1|k}}$ sont la pose et la vitesse prédites au temps $k+1$. Notons que la valeur des biais est supposée constante entre t_k et t_{k+1}. Néanmoins, l'intégration doit idéalement être refaite lorsque la valeur des états inertiels au temps k changent.

Les facteurs inertiels pré-intégrés offrent une alternative. Initialement introduits par (Lupton et al. 2011) avec un paramétrage des rotations à l'aide des angles d'Euler, ils sont adaptés aux quaternions par (Eckenhoff et al. 2016b) et formulés sur $\mathbb{SE}_3$ par (Forster et al. 2016). La pré-intégration consiste à intégrer les mesures inertielles entre les images-clés aux temps k et $k+1$ à la valeur de biais ${}^{\mathrm{B}}\bar{\boldsymbol{b}}_k$ estimée au temps k afin de prédire les valeurs $\Delta\boldsymbol{R}_{k,k+1}({}^{\mathrm{B}}\bar{\boldsymbol{b}}_k) \in \mathbb{SO}_3$, $\Delta\boldsymbol{t}_{k,k+1}({}^{\mathrm{B}}\bar{\boldsymbol{b}}_k) \in \mathbb{R}^3$ et $\Delta\boldsymbol{v}_{k,k+1}({}^{\mathrm{B}}\bar{\boldsymbol{b}}_k) \in \mathbb{R}^3$ d'orientation, de translation et de vitesse relative. Ces valeurs, ainsi que les matrices de covariance associées, sont d'ailleurs calculables de façon incrémentale au fur et à mesure que les mesures inertielles sont acquises. L'astuce consiste alors à éviter les ré-intégrations en corrigeant directement ces valeurs relatives à l'ordre 1 :

$$\Delta\boldsymbol{R}_{k,k+1}({}^{\mathrm{B}_k}\boldsymbol{b}_k) = \exp_{\mathbb{SO}_3}\left(\left\{ \left[\boldsymbol{J}^{\boldsymbol{\Delta R}}_{\boldsymbol{b}}({}^{\mathrm{B}_k}\bar{\boldsymbol{b}}_k) \right] \cdot \left({}^{\mathrm{B}_k}\boldsymbol{b}_k - {}^{\mathrm{B}_k}\bar{\boldsymbol{b}}_k \right) \right\}^{\wedge} \right) \cdot \Delta\boldsymbol{R}_{k,k+1}({}^{\mathrm{B}_k}\bar{\boldsymbol{b}}_k) \tag{2.13a}$$

$$\Delta\boldsymbol{t}_{k,k+1}({}^{\mathrm{B}_k}\boldsymbol{b}_k) = \Delta\boldsymbol{t}_{k,k+1}({}^{\mathrm{B}_k}\bar{\boldsymbol{b}}_k) + \left[\boldsymbol{J}^{\boldsymbol{\Delta t}}_{\boldsymbol{b}}({}^{\mathrm{B}_k}\bar{\boldsymbol{b}}_k) \right] \cdot \left({}^{\mathrm{B}_k}\boldsymbol{b}_k - {}^{\mathrm{B}_k}\bar{\boldsymbol{b}}_k \right) \tag{2.13b}$$

$$\Delta\boldsymbol{v}_{k,k+1}({}^{\mathrm{B}_k}\boldsymbol{b}_k) = \Delta\boldsymbol{v}_{k,k+1}({}^{\mathrm{B}_k}\bar{\boldsymbol{b}}_k) + \left[\boldsymbol{J}^{\boldsymbol{\Delta v}}_{\boldsymbol{b}}({}^{\mathrm{B}_k}\bar{\boldsymbol{b}}_k) \right] \cdot \left({}^{\mathrm{B}_k}\boldsymbol{b}_k - {}^{\mathrm{B}_k}\bar{\boldsymbol{b}}_k \right) \tag{2.13c}$$

où $\boldsymbol{J}_b^{\boldsymbol{\Delta R}}$, $\boldsymbol{J}_b^{\boldsymbol{\Delta t}}$ et $\boldsymbol{J}_b^{\boldsymbol{\Delta v}}$ sont les matrices Jacobiennes par rapport aux biais. Dans ce cas, le résidu vaut inertiel vaut :

$$\boldsymbol{\xi}_{\boldsymbol{u}_k}(\Theta) = \begin{bmatrix} \log_{\mathbb{SO}_3}\left(\boldsymbol{R}_{\mathrm{WB}_k}^\top \cdot \boldsymbol{R}_{\mathrm{WB}_{k+1}} \cdot \boldsymbol{\Delta R}_{k,k+1}^\top({}^{\mathrm{B}_k}\boldsymbol{b}_k)\right)^\vee \\ {}^{\mathrm{W}}\boldsymbol{t}_{\mathrm{WB}_{k+1}} - {}^{\mathrm{W}}\boldsymbol{t}_{\mathrm{WB}_k} - \boldsymbol{\Delta t}_{k,k+1}({}^{\mathrm{B}_k}\boldsymbol{b}_k) \\ {}^{\mathrm{W}}\boldsymbol{v}_{\mathrm{WB}_{k+1}} - {}^{\mathrm{W}}\boldsymbol{v}_{\mathrm{WB}_k} - \boldsymbol{\Delta v}_{k,k+1}({}^{\mathrm{B}_k}\boldsymbol{b}_k) \end{bmatrix} \tag{2.14}$$

et sa matrice de covariance $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{u_k}}$ est calculée incrémentalement.

Les facteurs visuels

L'observation $\boldsymbol{z}_{\mathrm{ij}}$ d'un amer 3D $\boldsymbol{\ell}_j$ depuis une image-clé $\mathcal{K}_i$ engendre un facteur dont le résidu quantifie une erreur de re-projection de l'amer dans ladite image :

$$\boldsymbol{\xi}_{\boldsymbol{z}_{\mathrm{ij}}}(\Theta) = \boldsymbol{z}_{\mathrm{ij}} - \pi_{\mathrm{B}_i}\left(T_{\mathrm{WB}_i} \cdot {}^{\mathrm{B}_i}\boldsymbol{\ell}_j\right) \tag{2.15}$$

Dans l'équation précédente, π_{B_i} désigne le modèle de caméra qui permet la projection sur le plan image associé d'un point 3D exprimé dans le référentiel inertiel B_i :

$$\pi_{\mathrm{B}_i}\left({}^{\mathrm{B}_i}\boldsymbol{p}\right) = \boldsymbol{K} \cdot d\left(\Pi\left(T_{\mathrm{C}_i\mathrm{B}_i} \cdot {}^{\mathrm{B}_i}\boldsymbol{p}\right)\right) \tag{2.16}$$

où $T_{\mathrm{B}_i\mathrm{C}_i}$ est la transformation extrinsèque entre les référentiels IMU et caméra, Π désigne la fonction perspective, d est le modèle de distorsion et K permet la conversion du plan focal vers le plan image à l'aide des paramètres intrinsèques. Pour davantage de détails sur les modèles de caméra, nous invitons le lecteur à se référer à la section 2.2.2. Notons qu'en pratique, la position de l'amer pourra être exprimée dans le référentiel B_k d'une autre image-clé $k \neq i$. Dans ce cas, les coordonnées de l'amer devront préalablement être exprimée dans le référentiel B_i et la pose T_{WB_k} interviendra explicitement dans le facteur précédent. On associera classiquement une matrice de covariance sphérique à ce type de résidu i.e. $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z} = \sigma_{\mathrm{vis}} \cdot \boldsymbol{I}_2$, où σ_{vis} est l'incertitude de mesure visuelle exprimée en pixels.

Les facteurs de pose absolue

Les facteurs de pose absolue encodent un a priori $z = \bar{T} \in \mathbb{SE}_3$ sur une variable de pose $T \in \mathbb{SE}_3$ à l'aide du résidu suivant :

$$\boldsymbol{\xi}_z(\Theta) = \log_{\{\mathbb{SO}_3 \times \mathbb{R}^3\}}\left(T \oplus \bar{T}^{-1}\right) \tag{2.17}$$

Les facteurs de pose relative

Les facteurs de pose relative se dérivent notamment des correspondances intra et inter-robot. Étant donné une contrainte $z = \hat{T}_{\mathrm{B}_i\mathrm{B}_j}$ entre les référentiels inertiels de deux images-clés $\mathcal{K}_i$ et $\mathcal{K}_j$, le résidu associé est la forme :

$$\boldsymbol{\xi}_z(\Theta) = \log_{\{\mathbb{SO}_3 \times \mathbb{R}^3\}} \left(T_{\mathrm{WB}_i}^{-1} \oplus T_{\mathrm{WB}_j} \oplus \hat{T}_{\mathrm{B}_i\mathrm{B}_j}^{-1} \right)^{\vee} \tag{2.18}$$

Notons également que les facteurs de pose relative peuvent contraintes deux variables exprimées dans des référentiels distincts (e.g. dans le cas multi-robots). Dans ce cas, étant donné deux référentiels $\mathcal{R}_i$ et $\mathcal{R}_j$, le résidu aura la forme :

$$\boldsymbol{\xi}_z(\Theta) = \log_{\{\mathbb{SO}_3 \times \mathbb{R}^3\}} \left(T_{\mathrm{R}_i\mathrm{B}_i}^{-1} \oplus T_{\mathrm{R}_i\mathrm{R}_j} \oplus T_{\mathrm{R}_j\mathrm{B}_j} \oplus \hat{T}_{\mathrm{B}_i\mathrm{B}_j}^{-1} \right)^{\vee} \tag{2.19}$$

2.6 Conclusion

Dans ce chapitre, nous avons présenté l'architecture générale d'un algorithme de SLAM, puis détaillé les techniques introduites dans la littérature pour chacun de ses modules, avec une emphase particulière sur les cas visuels et visio-inertiels, afin de présenter les notions et les outils que nous utiliserons plus spécifiquement dans la suite de ce manuscrit. Dans le chapitre suivant, nous verrons comment cette architecture mono-robot est étendue en contexte collaboratif.

Chapitre 3

L'émergence des méthodes de SLAM multi-robots

Comme le soulignent (Cadena et al. 2016), les méthodes de SLAM mono-robot ont aujourd'hui atteint un haut degré de maturité et ouvert de nouvelles perspectives de recherche. Parmi celles-ci figure le SLAM multi-robots, porté par l'essor continu des puissances de calcul ainsi que des capacités d'interconnexion des robots en réseau. De par les opportunités qu'il apporte et les contraintes qu'il impose, le SLAM multi-robots est davantage qu'une simple extension de son analogue mono-robot. Certes, il exploite les techniques introduites pour le SLAM mono-robot, mais la gestion des interactions entre les robots requiert d'adapter les modules et l'architecture classiques du SLAM.

3.1 Définition du SLAM multi-robots

On parle de SLAM multi-robots lorsque plusieurs robots concourent simultanément aux objectifs de cartographie et de localisation en échangeant des informations. Bien que proche du SLAM multi-session avec lequel il partage un certain nombre de techniques (Lynen et al. 2015), il s'en distingue en imposant la simultanéité des acquisitions et des estimations, là où le SLAM multi-session se contente d'acquisitions différées fusionnées hors-ligne.

Dans la suite, nous proposons de parcourir la littérature multi-robots selon une grille de lecture en trois axes, correspondant chacun à une dimension constitutive d'un algorithme de SLAM multi-robots. Le premier ingrédient d'une méthode multi-robots est son schéma d'allocation des tâches et des données. Dans la section précédente, nous avons vu que tout algorithme de SLAM comprend au moins un module d'odométrie et de cartographie locale, un module pour la détection des fermetures de boucle et un module d'inférence. De plus, il opère sur différents types de données telles que des mesures brutes, un modèle

probabiliste qu'il maintient s'il recourt à des inférences globales, et des estimées de carte et de trajectoire. En contexte multi-robots, chaque module peut être centralisé, décentralisé ou distribué, et il en va de même pour les données quant à leur stockage. Une tâche est dite centralisée lorsqu'elle est effectuée par un seul agent. Dans ce cas, celui-ci procède généralement en aggrégeant les données collectées par les autres agents. Il leur rend éventuellement compte des résultats de ses traitements une fois ceux-ci réalisés. On dira au contraire d'une tâche qu'elle est décentralisée lorsqu'elle est effectuée par plusieurs agents en parallèle, qu'ils agissent ou non pour leur propre compte. Enfin, une tâche est distribuée dès lors que plusieurs agents se partagent les calculs qu'elle requiert. La seconde dimension d'une méthode de SLAM multi-robots est sa politique de communication. Elle recouvre le choix de la topologie des échanges (i.e. quels robots peuvent communiquer), la nature des informations échangées, ainsi que la planification spatiale (i.e. choix des informations à communiquer) et temporelle des échanges entre les robots. Enfin, la troisième et dernière dimension tient à la stratégie employée par chaque robot pour associer et fusionner les données multi-robots. Elle consiste à détecter les correspondances entre les informations reçues des autres robots et ses propres connaissances. Ces correspondances permettront d'ancrer les référentiels d'odométrie des robots les uns par rapport aux autres (i.e. estimer les transformations relatives qui les lient) ainsi qu'à accroître l'information mutuelle entre les trajectoires des robots pour propager ces informations entre les trajectoires et les cartes individuelles via des procédures d'inférence multi-robots, et ainsi construire une carte globale. Elle nécessite une extension des modules de détection des fermetures de boucles et d'inférence. On pourra également se référer à la revue sur le SLAM multi-robots publiée par (Saeedi et al. 2016).

Le SLAM multi-robots se distingue des méthodes de localisation multi-robots et de cartographie multi-robots seules. La localisation collaborative vise à estimer les poses relatives entre les robots où à les localiser dans une carte établie à l'avance, sans pour autant fusionner leurs cartes individuelles. Par exemple, dans les travaux de (Oleynikova et al. 2015), un robot terrestre et un drone se relocalisent par rapport à une carte commune. (Luft et al. 2016) proposent une approche décentralisée basée filtrage pour la localisation collaborative et qui n'utilise pas de carte. En cartographie collaborative, les robots contribuent à construire une carte commune en détectant les correspondances entre leurs observations, mais sans raffiner l'estimation de leur trajectoire en retour. C'est par exemple le cas des travaux de (Vidal-Calleja et al. 2011) dans lesquels des robots terrestres et aériens construisent et partagent des cartes locales qu'ils ancrent à l'aide

d'observations mutuelles directes et de mesures GPS. Ces méthodes constituent des pans à part entière de la littérature et font l'objet de nombreuses publications. Contrairement aux deux familles précédentes, les méthodes de SLAM présentent la particularité de coupler l'odométrie et la cartographie. Néanmoins, la différence est parfois mince ou sujette à diverses interprétations. Selon (Schmuck et al. 2019a), il est par exemple discutable que certaines méthodes centralisées à l'instar de (Forster et al. 2013) correspondent à la définition du SLAM. En effet, dans cette dernière, les agents délèguent la cartographie au serveur mais ne bénéficient pas de ses estimations en retour pour affiner leur propre estimation de trajectoire. Ainsi, s'il s'agit algorithmiquement d'une méthode de SLAM à l'échelle de la flotte, en ce sens que tous les modules constitutifs d'un algorithme de SLAM sont présents et que les contraintes mutuelles entre la carte et les trajectoires sont exploitées au niveau du serveur, ce n'est pas le cas à l'échelle des agents car ils ne se relocalisent pas dans la carte construite en parallèle. Les estimées de carte et de trajectoire calculées sur le serveur sont alors uniquement destinées à un opérateur extérieur. Néanmoins, les méthodes de SLAM s'appuient sur les techniques développées pour la localisation et la cartographie multi-robots.

3.1.1 Opportunités et contraintes associées

La robotique collaborative offre de nouvelles opportunités pour accroître les performances des algorithmes de SLAM. Celles-ci se manifestent aussi bien à l'échelle de la flotte qu'à celle des agents. Tout d'abord, le passage au contexte multi-robots permet de gagner en efficacité quant à l'objectif poursuivi par la flotte grâce à la parallélisation des opérations. Par exemple, dans le cas de scénarios d'exploration, la surface inconnue peut être balayée plus rapidement et permettre un gain de temps significatif. La deuxième avantage est la subsidiarité : non seulement les robots s'assistent les uns les autres dans leurs opérations de localisation et de cartographie en s'échangeant des informations complémentaires, mais ils peuvent également se substituer les uns aux autres en cas de neutralisation d'un agent afin de ne pas compromettre la mission. Cette résilience est particulièrement adéquate en contexte militaire. Enfin, la troisième opportunité se situe à l'échelle des agents : l'échange et la fusion des informations inter-robot accroît potentiellement la précision et la robustesse des estimations de cartes et de trajectoires individuelles en renforçant l'information mutuelles entre les estimées, tandis que la parallélisation et l'éventuelle distribution des tâches entre les agents de la flotte suppléent possiblement aux limitations de leurs ressources de calcul pour gérer les interactions multi-robots.

Le SLAM multi-robots constitue néanmoins un défi technique et apporte de nouvelles contraintes que nous pouvons classer en trois catégories. On rencontre d'abord des contraintes réseaux. Il s'agit par exemple de limitations de bande passante, c'est-à-dire du débit de données qui peut voyager à travers le réseau sans le saturer. Dans de nombreux travaux, ces limitations sont soit traitées explicitement sous forme de contraintes sur la taille des paquets à envoyer, ou de façon plus souples en minimisant la quantité de données échangées sans pour autant s'astreindre à des limites fixes. Parmi les contraintes réseaux figurent également les limitations de portée de communication. En effet, les robots peuvent occasionnellement perdre le contact les uns avec les autres. Un réseau wifi, qui permet des communications jusqu'à plusieurs dizaines de mètres, peut notamment ne pas suffire lors de l'exploration de larges zones. Une autre problématique dérivée de l'aspect réseau est celle de la sécurité, qui est explicitement traitée dans certains travaux comme (Choudhary et al. 2017). En effet, en contexte militaire, la flotte devrait être protégée contre des tentative de corruption ou d'interception des données par un robot ennemi. Enfin, la dé-synchronisation temporelle des horloges des robots peut s'avérer problématique pour la fusion de données.

La deuxième famille rassemble les contraintes qui dérivent de la limitation des ressources de calcul et de mémoire des agents face à la complexité spatiale et temporelle additionnelle engendrée par SLAM collaboratif. Déjà prégnantes dans le cas mono-robot, elles se retrouvent potentiellement exacerbées en contexte multi-robots. Pour cause, chaque robot doit à la fois gérer son propre SLAM, mais également les interactions avec les autres robots. Il doit notamment calculer les paquets de données qu'il leur envoie, et surtout intégrer ceux qu'ils reçoit, avec ce que cela suppose en termes de détection des correspondances inter-robot et d'inférence. En conséquence, on doit s'assurer que ces opérations n'empiètent pas sur les ressources nécessaires pour faire tourner le SLAM ainsi que les autres processus.

Enfin, la troisième famille recouvre des contraintes posées par la fusion des données multi-robots. Le premier soucis consiste à intégrer les données reçues sans compromettre ses propres estimations. En effet, le problème du SLAM étant non-linéaire, l'initialisation des estimées conditionne notamment la bonne convergence des algorithmes itératifs d'inférence globale. Ainsi, intégrer sans précautions des estimées erronées reçues d'un autre robot pourrait dégrader l'ensemble de la carte. Par ailleurs, l'hétérogénéité des données échangées peut compliquer leur intégration. Lorsque des robots terrestres coopèrent avec des drones volant à basse altitude, leurs différences de points de vue peut en effet compro-

mettre la détection des correspondances inter-robot. Cette difficulté s'avère encore plus corriace lorsqu'il s'agit de fusionner des représentations de natures différentes comme dans les travaux de (Käslin et al. 2016). Le problème principal posé par la fusion de données multi-robots est celui de la fusion consistante. Les inconsistances surviennent lorsque des informations corrélées – on parle alors de données *consanguines* – sont intégrées sans pleinement tenir compte de ces corrélations, voire comme si elles étaient indépendantes. Ce problème, appelé comptage multiple, est expliqué dans (Indelman 2015) dans les termes suivants : supposons que le robot A communique au robot B la distribution $p(\Theta|\mathcal{Z}_A)$ avec Θ un ensemble de variables communes, et $\mathcal{Z}_A$ les mesures acquises par A. Le robot B calcule $p(\Theta|\mathcal{Z}_A, \mathcal{Z}_{,B})$ et l'envoie au robot C, qui calcule $p(\Theta|\mathcal{Z}_A, \mathcal{Z}_B, \mathcal{Z}_C)$. Si A reçoit cette distribution et la considère comme indépendante de sa distribution $p(\Theta|\mathcal{Z}_A)$, alors l'information est comptée deux fois. Comme l'illustre cet exemple, c'est essentiellement dans le cas d'échanges cycliques entre les robots que ce problème est susceptible de survenir. Outre qu'elle corrompt l'estimation des incertitudes, l'information artefactuelle qui en résulte risque de biaiser l'estimation des variables.

3.1.2 Les applications du SLAM multi-robots

Les applications du SLAM multi-robots étendent celle du SLAM mono-robot. Rappelons néanmoins que le SLAM n'est souvent qu'une brique algorithmique intermédiaire en support d'autres objectifs ayant trait à l'interaction avec l'environnement. Les applications les plus courantes relèvent de l'exploration ou de l'inspection collaborative. Dans les travaux de (Olson et al. 2013a), une flotte comprenant 14 robots, chacun équipé d'une caméra RGB et d'un télémètre laser, ont exploré et cartographié un large environnement urbain dans le cadre de la compétition MAGIC (*Multi Autonomous Ground robots International Challenge*) en 2010. Des opérateurs humains interagissaient de façon limitée avec les robots pour corriger rétroactivement les erreurs de cartographie dues à des correspondances erronées. Des opérations similaires ont été menées dans des environnements variés, que ce soit pour de l'exploration sous-marine (Elibol et al. 2014), souterraine (Rogers III et al. 2017), ou encore dans la perspective d'explorations extra-planétaires (Schuster et al. 2015). Ces capacités peuvent notamment être mises à profit dans le cadre d'opérations de recherche et de sauvetage (*Search And Rescue* (SAR)) (Queralta et al. 2020) qui nécessitent une collecte rapide d'informations sur la zone sinistrée (et potentiellement risquée pour un humain) en vue d'interventions ultérieures, opérées possiblement par les robots eux-mêmes. Dans cette optique, (Michael et al. 2012) utilisent une flotte compo-

sée de deux robots terrestres et d'un drone afin de cartographier un bâtiment endommagé par un séisme, sans intervention humaine autre que le contrôle des robots. De la même façon, (Nagatani et al. 2011) font collaborer plusieurs robots terrestres pour cartographier un environnement tout en planifiant leur trajectoires de façon coordonnée pour en minimiser le temps d'exploration. De façon générale, toutes les applications du SLAM mono-robot peuvent être étendues en contexte multi-robots, des tâches de nettoyage par une flotte d'aspirateurs à des opérations de surveillance en contexte militaire.

3.2 Anatomie d'un algorithme de SLAM multi-robots

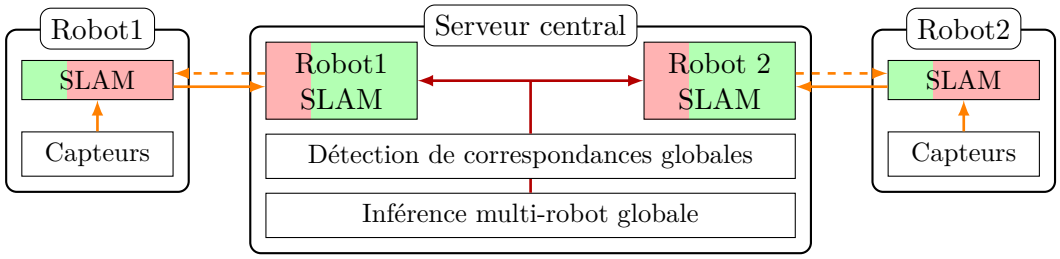
3.2.1 Schéma d'allocation des tâches et des données

Le schéma d'allocation des tâches et des données est la première dimension d'une méthode de SLAM multi-robots. On distingue généralement les méthodes centralisées des méthodes décentralisées. Dans cette section, nous passons en revue les différentes allocations multi-robots proposées dans la littérature.

Les allocations centralisées

Les méthodes de SLAM centralisées (c.f. Figure 3.1) impliquent généralement deux types d'agents de natures différentes. D'une part, des robots mobiles acquièrent des observations et leur appliquent différents traitements, tandis qu'une unité ou serveur central aggrège ces données pour effectuer dessus des opérations plus lourdes et plus complexes. Dans une telle architecture, le serveur central inclut souvent une interface pour un opérateur extérieur, et toute l'ingéniosité du schéma d'allocation consiste à exploiter les disparités de puissances de calcul entre les agents et le serveur. On peut classer les méthodes centralisées selon le compromis qu'elles adoptent entre d'une part l'autonomie qu'elle accorde aux agents mobiles et d'autre part le degré d'intégration des tâches au serveur. À ce titre, on distingue trois approches possibles.

La première stratégie consiste à centraliser toutes les tâches sur le serveur. Elle réduit alors les agents à de simples capteurs mobiles. C'est notamment le cas de CoSLAM (Zou et al. 2012). Largement inspirée par les techniques de reconstruction par le mouvement, CoSLAM aggrège directement les images acquises par toutes les caméras. Il construit la carte sur la base d'un groupement dynamique de ces caméras selon les recouvrements de leur champ de vue. Cette méthode de SLAM est particulière puisque les agents ne sont



Les allocations centralisées se classifient selon le degré d'intégration des agents au serveur central, ici symbolisé par la jauge SLAM dans les modules de SLAM. Les tâches qui ne sont pas assurées par les agents le sont par le serveur, qui détecte les correspondances globales et fusionne les cartes à travers des inférences multi-robots. Les échanges sont bidirectionnels si le serveur renvoie des informations aux agents.

FIGURE 3.1 – Allocation des tâches dans une méthode centralisée

pas des robots mais de simple capteurs, ce qui signifie que tous les traitements classiques d'un algorithme de SLAM sont effectués sur le serveur pour chacun d'entre eux. Ainsi, l'estimation du mouvement de chaque caméra par extraction et suivi des points d'intérêt est réalisée par le serveur lui-même. Il en va de même pour la sélection des images-clés, la triangulation des amers et l'estimation de leur position. Une spécificité de CoSLAM tient à sa gestion des points statiques et dynamiques, ainsi qu'à sa technique de détection des recouvrements entre les champs de vue des caméras pour les opérations inter-caméras telles que la cartographie ou l'estimation de poses relatives. Néanmoins, aucun module de fermeture de boucle n'est implémenté, et le recours imposé à des cartes graphiques pour les modules de cartographie complique le déploiement de cette méthode sur des plateformes robotiques.

La seconde approche s'inscrit en opposition à la première. Elle maximise l'autonomie des agents par rapport au serveur en les dotant chacun d'un algorithme de SLAM complet. Au serveur, elle ne délègue que les opérations inter-robots, c'est-à-dire la fusion de leurs cartes en une carte globale ainsi que l'inférence sur cette carte. C'est l'approche adoptée par (CHEBROLU et al. 2015) pour du SLAM collaboratif monoculaire, ainsi que par MOARSLAM (MORRISON et al. 2016). Cette dernière méthode adopte une architecture client-serveur inspirée du *Cloud Computing* qui avait été appliquée au SLAM pour la première fois par C2TAM (RIAZUELO et al. 2014). Le client de chaque agent comprend trois processus parallèles. Le premier et le second s'occupent respectivement de l'odométrie et de la cartographie locale. Ils reprennent les travaux de (KEIVAN et al. 2016) et envoient les mises à jour de la carte au serveur à chaque nouvelle image clé. Le troisième

processus recherche les fermetures de boucles à l'aide du détecteur de (Gálvez-López et al. 2012). Ce faisant, il adresse également une requête au module de reconnaissance de place du serveur pour chercher les correspondances avec les cartes des autres robots. Si des correspondances sont trouvées, alors le client télécharge les portions de la carte correspondantes depuis le serveur pour les fusionner avec la sienne. Le serveur ne sert ainsi que d'espace de stockage auquel les clients peuvent formuler des requêtes de façon à comparer les cartes des robots entre eux et n'en télécharger que les portions qui les intéressent. Adoptant une démarche similaire à MOARSLAM, CORB-SLAM (Li et al. 2017) étend ORB-SLAM (Mur-Artal et al. 2015) en une méthode collaborative centralisée. Chaque client implémente les trois processus constitutifs d'ORB-SLAM que sont le suivi visuel (*tracking*), la cartographie locale et la détection de fermeture de boucle (qui intègre l'inférence globale, appelée pour chaque fermeture de boucle détectée). Le serveur récupère les cartes construites par les clients et s'efforce de les fusionner au sein d'une carte globale en détectant les correspondances inter-robot. Dès qu'il parvient à fusionner deux cartes, il les raffine à l'aide d'un ajustement de faisceaux global, puis communique cette mise à jour de la carte globale à tous les robots. (Qin et al. 2018a) ont proposé une extension collaborative centralisée de l'algorithme Vins-Mono (Qin et al. 2018b) dans laquelle les modules de fermetures de boucles et d'optimisation de graphe de poses originaux sont délocalisés sur un serveur central ; ce dernier aggrège les données des robots, détectent les correspondances, ré-estime les trajectoires et communique les mises à jour aux agents. Les travaux de (Deutsch et al. 2016) généralisent cette logique et proposent un framework appelé *Team SLAM* (TSLAM). Celui-ci peut potentiellement faire collaborer plusieurs agents faisant tourner des algorithmes de SLAM différents. Il requiert uniquement que chaque agent fournisse au serveur un graphe de poses à l'échelle ainsi que les images associées à ses noeuds. Ces informations sont alors récupérées par un processus tournant sur chaque agent et qui pré-traite les images avant de les envoyer au serveur central. Celui-ci dispose d'un module de reconnaissance de lieux qui détecte les correspondances entre les graphes de poses des différents robots, ainsi que d'un module d'inférence qui les ré-optimise périodiquement ou lorsque de nouvelles fermetures de boucles inter-robot sont ajoutées. Le serveur renvoie à chaque robot les corrections concernant son propre graphe, et un module se charge de les répercuter sur ses estimées. Enfin, la méthode proposée par (Egodagamage et al. 2017) adopte également ce type d'approche centralisée, mais laisse la porte ouverte à une architecture intermédiaire multi-centrique, où plusieurs serveurs pourraient simultanément assurer des opérations de fusion.

Enfin, la troisième approche adopte une posture intermédiaire, et tend à s'imposer dans la littérature actuelle. Recherchant un compromis entre autonomie des agents et intégration au serveur, elle distribue les modules des algorithmes de SLAM entre les agents et le serveur. Si elle laisse aux agents les modules nécessaires à leur navigation, elle cherche à exploiter toutes les potentialités du serveur au service des SLAMs individuels. Cette stratégie est d'abord adoptée dans l'algorithme *Collaborative Structure from Motion* (CSfM) proposé par (Forster et al. 2013). Chaque agent estime alors son propre mouvement à l'aide d'un algorithme d'odométrie visuelle qui s'occupe de sélectionner les images-clés et d'en extraire les primitives et les descripteurs. Chaque image-clé est alors communiquée au serveur central avec les coordonnées de ses points d'intérêts et leurs descripteurs, ainsi que la pose relative estimée avec l'image-clé précédente. Au niveau du serveur, un processus séparé tourne pour chaque agent et implémente les autres modules caractéristiques d'un algorithme de SLAM. Dans chaque processus, un module d'odométrie visuelle et de cartographie affine l'estimation envoyée par l'agent en se relocalisant par rapport à la carte qu'il construit en parallèle. Cette carte est raffinée à l'aide d'inférences locales par ajustement de faisceaux. Ceci permet notamment d'estimer le facteur d'échelle ainsi que sa dérive. La détection des fermetures de boucles est assurée par l'appel à un module de reconnaissance de place extérieur. Ce module permet non seulement de détecter les correspondances au sein de la carte associée à l'agent, mais également les correspondances avec les autres cartes des autres agents. Ces correspondances inter-robot sont alors exploitées pour fusionner les cartes des robots. Ces derniers opèrent alors sur la même carte.

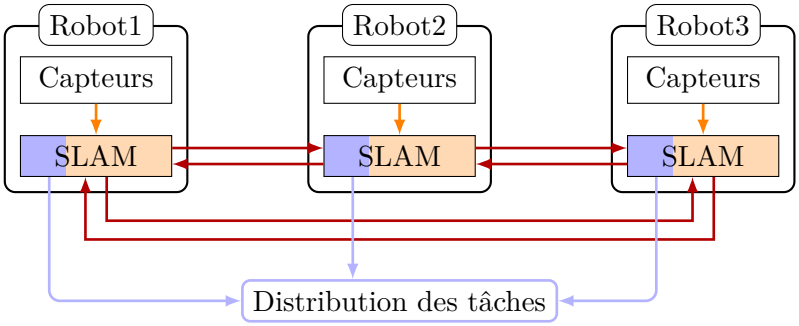
Dans la méthode précédente, le serveur aggrège les données des agents, mais les échanges sont unidirectionnels. Les agents ne bénéficient donc pas des estimations générées par le serveur pour améliorer leur propre estimation de trajectoire. Selon certaines interprétations (Schmuck et al. 2019a), l'appellation SLAM s'en trouve alors discutable. D'autres architectures autorisent ces échanges. C'est l'approche aujourd'hui adoptée par la majorité des méthodes de SLAM centralisées, à l'instar de C2TAM (Riazuelo et al. 2014). Cette méthode se base sur l'algorithme PTAM (Klein et al. 2007) qu'il étend en s'inspirant des techniques de *Cloud Computing* : les agents deviennent des clients auxquels le serveur fournit des services exempts des contraintes de temps-réel, tels que les opérations d'inférence et de détection de fermeture de boucles intra et inter-robot. De façon à assurer l'autonomie des agents, chacun possède son propre module d'odométrie et de cartographie, ainsi qu'un mini-module de fermeture de boucles uniquement dédié à sa relocalisation en cas de perte du suivi visuel. Chaque nouvelle image-clé sélectionnée

est envoyée au serveur qui construit la carte en parallèle, l'optimise périodiquement et la renvoie alors à l'agent. Le serveur recherche également les fermetures de boucles associées à chaque nouvelle image-clé dans la carte de l'agent mais également dans les cartes des autres robots pour détecter des correspondances inter-robot. Plus précisément, c'est uniquement la reconnaissance de lieu qui lui incombe dans ce cas. Si le serveur détecte des fermetures de boucles intra-robot potentielles, alors il renvoie à l'agent les images-clés impliquées afin que celui-ci la caractérise géométriquement. On peut ainsi considérer que le module de fermeture de boucle est en quelques sortes distribué entre les agents et le serveur. Lorsque le serveur détecte des correspondances inter-robot entre deux cartes séparées, il fusionne ces cartes, puis les renvoie aux agents concernés. Enfin, en cas de perte prolongée du suivi, les agents peuvent solliciter le serveur et son module de fermeture de boucles pour une relocalisation globale. Le point faible de cette méthode est qu'elle impose une charge réseau importante car les estimées de cartes et de trajectoires sont régulièrement renvoyées aux agents après mise à jour. Ceci peut s'avérer probitif lorsque les robots explorent de larges zones.

Récemment, (Schmuck et al. 2017) ont proposé CCM-SLAM, une nouvelle architecture centralisée pour le SLAM mono-visuel, ensuite étendue par CVI-SLAM (Karrer et al. 2018) au cas visio-inertiel. Celle-ci reprend l'idée d'échanges bidirectionnels entre les agents et le serveur, mais sans concéder aux agents l'entière responsabilité de leur SLAM. Pour ce faire, ils renforcent l'intégration des agents aux serveur en leur laissant les modules de SLAM cruciaux pour leur navigation et en délégant le reste au serveur. Ce dernier aggrège les données des agents, et leur renvoie régulièrement des mises à jour ainsi que des portions covisibles des cartes des autres robots selon les fermetures de boucles détectées. Les agents intègrent alors ces informations dans leur cartographie locale afin d'améliorer leur estimation de trajectoire. Côté agent, chacun dispose d'un module complet d'odométrie et de cartographie locale. Au serveur central, il communique chaque image-clé (primitives et descripteurs inclus), les amers qu'il a triangulés, ainsi que les mesures inertielles qui lient les images-clés entre elles. Comme dans (Forster et al. 2013), un processus séparé est exécuté pour chaque agent sur le serveur pour détecter les fermetures de boucles intra et inter-robot, ainsi que pour pour fusionnner les cartes individuelles. Le serveur se charge également d'éliminer les images-clés redondantes pour réduire la complexité du problème d'inférence.

Les allocations décentralisées

Les méthodes décentralisées privilégient la flexibilité de la flotte et l'autonomie des agents. Contrairement aux méthodes centralisées, les agents ne peuvent plus recourir à un serveur extérieur pour lui déléguer les opérations les plus complexes et la fusion des cartes. Ils échangent directement entre eux. Cependant, cette architecture se prête davantage aux algorithmes distribués qui peuvent pallier l'absence d'un serveur central. En partageant les calculs des tâches conjointes, les agents augmentent ainsi virtuellement leur puissance de calcul effective. Si l'on peut classer les méthodes centralisées selon le degré de centralisation des tâches des agents vers le serveur, on peut comparer les méthodes décentralisées selon le degré de distribution des tâches entre les robots.



Les allocations décentralisées se classifient selon le degré de distribution des modules de SLAM des agents, ici symbolisé par la jauge SLAM dans les modules de SLAM. La distribution porte généralement sur la détection des correspondances intra ou inter-robot ainsi que les opérations d'inférence multi-robots. Comme illustré ici, l'allocation décentralisée engendre potentiellement des échanges cycliques qui imposent de prévenir le double comptage des données consanguines.

FIGURE 3.2 – Allocation des tâches dans une méthode décentralisée

Classiquement, les méthodes décentralisées (c.f. Figure 3.2) cherchent à maximiser l'autonomie des agents en les dotant de modules de SLAM complets, et en leur confiant la responsabilité d'intégrer les données qu'ils reçoivent des autres robots. Ainsi, c'est à chaque agent de détecter lui-même les correspondances inter-robot pour fusionner les données qu'ils reçoit avec les siennes, et de mener les inférences multi-robots qui s'imposent. C'est le schéma d'allocation adopté par diverses méthodes telles que DDF-SAM (CUNNINGHAM et al. 2010), AUV-CSLAM (PAULL et al. 2015) pour du SLAM multi-robots sous-marin acoustique, le SLAM stéréo-visuel de (SCHUSTER et al. 2015), la méthode de SLAM monoculaire multi-robots de (ZHANG et al. 2018a), (LAZARO et al. 2013) et (SARTIPI et al. 2019). Elles recouvrent une grande variété de politiques de communication et de stratégies d'intégration qui seront traitées dans les sections suivantes.

Une stratégie complémentaire consiste à mutualiser les ressources de calculs ou les connaissances des différents robots sans pour autant les partager au préalable. On distribue alors les calculs propres à certains modules entre les robots. Cela concerne le plus souvent des modules de fermeture de boucles et des modules d'inférence. Les auteurs de (Cieslewski et al. 2017) proposent à ce titre un algorithme distribué de détection de fermetures de boucles dans lequel une même requête est traitée successivement par plusieurs robots. Dans le domaine de l'inférence, (Nerurkar et al. 2009) et (Choudhary et al. 2017) présentent tous deux des algorithmes d'inférence distribuée dans lesquels l'optimisation est menée sur des données réparties entre plusieurs robots avec un échange minimal de données. Ces approches sont combinées dans des architectures décentralisées complètes telles que celles proposées par (Cieslewski et al. 2018) ou encore DOOR-SLAM (Lajoie et al. 2020). Notons enfin que la distribution peut égalemement porter sur le stockage des cartes. À cet égard, (Cieslewski et al. 2015) et (Quraishi et al. 2016) proposent un stockage distribué doublé de mécanismes de contrôle de version pour gérer les éventuels conflits. Chaque agent possède physiquement un morceau de la carte globale, mais peut accéder aux portions stockées par les autres robots pour les mettre à jour.

Résumé

Le schéma d'allocation des tâches et des données est un choix architectural déterminant d'une méthode de SLAM collaboratif. La littérature s'est dans un premier temps intéressé aux méthodes centralisées qui présentent l'avantage de structurer les échanges entre les robots et d'exploiter les ressources de calcul et de stockage accrue du serveur central. La recherche privilégie aujourd'hui des architectures centralisées qui préservent au mieux l'autonomie des robots par une intégration raisonnée des agents au serveur. Le développement de méthodes décentralisées est plus récent, du fait des contraintes techniques et algorithmiques qu'elles imposent quant aux charges de calculs induites pour chaque agents, et le contrôle des informations transmises pour gérer la consanguinité des données et prévenir le comptage multiple.

3.2.2 Politiques de communication

Dans une méthode de SLAM multi-robots, la politique de communication définit pour chaque robot : quelles données communiquer, sous quelle forme, à quel(s) robot(s) et à quels instants. Elle recouvre donc plusieurs aspects que sont la topologie des échanges, la

nature des informations communiquées, et enfin la planification des échanges (i.e. décider quand communiquer, avec qui, et sur quelle portion des informations).

La topologie des échanges

La topologie des échanges définit pour chaque agent l'ensemble des robots avec lesquels il a d'une part l'autorisation de communiquer, et d'autre part la possibilité physique de le faire. C'est une conséquence directe du schéma d'allocation retenu pour les tâches et les données. Néanmoins, même dans le cas de méthodes centralisées, une marge de liberté demeure. En effet, si les échanges se font uniquement entre les agents et le serveur, deux politiques sont possibles. Lorsque le serveur aggrège les informations des robots sans leur faire de retours, comme dans les cas de CoSLAM (Zou et al. 2012) ou CSfM (Forster et al. 2013), les échanges sont unidirectionnels. À l'inverse, d'autres méthodes autorisent les échanges bidirectionnels. C'est le cas de CCM-SLAM (Schmuck et al. 2017), CVI-SLAM (Karrer et al. 2018) et coVins (Qin et al. 2018a). La topologie des échanges perd toute structuration dans le cas décentralisé. Ceci impose de prévenir le problème de consanguinité des données, qui survient notamment dans le cas de communications cycliques entre les robots. Par conséquent, des méthodes telles que (Nettleton et al. 2003) les interdisent. Une difficulté supplémentaire est posée lorsque les portées de communications des robots sont limitées. La topologie des échanges évoluent alors dynamiquement selon que les robots restent à portée les uns des autres.

La nature des informations échangées

Les algorithmes de SLAM manipulent au moins trois types de données. Ce sont d'abord les mesures brutes acquises en sortie des capteurs. Celles-ci subissent éventuellement un pré-traitement, comme les images dans les méthodes indirectes dont on extrait préalablement les primitives. Ce sont ensuite les contraintes probabilistes dérivées des observations via leur modèle de mesure ainsi que des fermetures de boucles : elles constituent le modèle probabiliste sous-jacent à l'inférence, et en particulier l'inférence globale. Enfin, ce sont les estimées de carte et de trajectoires déduites de l'inférence. Chaque type de données joue un rôle indispensable en SLAM multi-robots : dans le cas visuel, les images pré-traitées permettent de détecter des fermetures de boucles inter-robot, et le modèle probabiliste permet au récepteur de ré-estimer les cartes et les trajectoires dans leur globalité après des opérations de fusion. Pour ce faire, il nécessite également les valeurs des estimées

pour initialiser l'intégration des nouvelles portions de cartes et de trajectoires. Deux stratégies majeures coexistent : la première consiste à communiquer des données brutes dans leur exhaustivité, tandis que la seconde communique des représentations condensées ou élaguées.

Une première stratégie consiste à communiquer des données brutes, c'est-à-dire des données originales non altérées, qu'il s'agisse de mesures capteurs, de portions du modèle probabiliste ou d'un sous-ensemble des estimées. C'est l'approche communément adoptée dans les méthodes centralisées. Dans la plupart d'entre elles, les agents communiquent au serveur central les informations associées à leurs images-clés. Dans CVI-SLAM (Karrer et al. 2018), ceci englobe en particulier les mesures inertielles acquises entre les images-clés successives, la pose relative estimée avec l'image-clé précédente, les positions des amers triangulés ainsi que la liste de leurs observations. En effet, le serveur doit aggréger l'exhaustivité des informations collectées et traitées par les agens de façon à mener les opérations d'inférence et de fermeture de boucles en croisant leurs données. Cette stratégie n'est néanmoins pas l'apanage des méthodes centralisées. Dans l'architecture décentralisée proposée par (Zhang et al. 2018a), les robots échangent leurs nouvelles images-clés deux-à-deux. La communication des descripteurs associés est également requise pour l'algorithme développé par (Tardioli et al. 2015) qui détecte en temps réel les observations communes entre deux robots lorsqu'ils se rencontrent. De façon générale, la détection des fermetures de boucles inter-robot sur critères visuels nécessitera toujours l'échange des coordonnées des pritimives extraites dans les images ainsi que leur descripteur associé, qu'il s'agisse du descripteur entier ou bien de son mot visuel correspondant comme dans (Tardioli et al. 2015). Au lieu d'échanger les informations image-clé par image-clé, il est possile de les échanger sous-carte par sous-carte. Une sous-carte délimite une portion de la carte caractérisée par une continuité temporelle et spatiale. On lui associe généralement son propre référentiel. Dans la méthode décentralisée de SLAM stéréo-visuel proposée par (Schuster et al. 2015), les sous-cartes constituent l'unité élémentaire d'échange entre les robots : elles comprennent les facteurs séquentiels de poses relatives qui restituent les informations de trajectoire, ainsi que les nuages de points stéréo-visuels denses. Les observations visuelles, qui représentent une information de taille conséquente, ne sont en revanche pas communiquées. (Nettleton et al. 2003) communiquent également des sous-cartes qu'ils utilisent comme un moyen de borner la taille des messages envoyés. Dans un cadre visio-inertiel, (Sartipi et al. 2019) échangent également des sous-cartes correspondant aux observations courantes qui incluent notamment les descripteurs asso-

ciés aux amers triangulés, associés aux matrices de covariances qui les caractérisent. Il n'en reste que communiquer de telles données peut rapidement saturer la bande-passante disponible. C'est d'autant plus le cas dans les méthodes décentralisées où tous les robots communiquent potentiellement deux-à-deux.

La seconde stratégie consiste à communiquer des représentations condensées de portions locales des cartes. Elle est majoritairement adoptée par des méthodes décentralisées qui sont davantage sujettes aux contraintes de bande-passante que les méthodes centralisées. Non seulement cette approche vise à réduire la quantité de données échangées, mais elle permet aussi de mieux cibler l'information envoyée. Par exemple, dans DDF-SAM (Cunningham et al. 2010), les robots échangent des distributions a posteriori marginalisées sur des séparateurs, c'est-à-dire des variables communément observées telles que amers globaux identifiables de façon unique. Pour préserver la consistance des cartes, les informations reçues ne sont cependant pas intégrées dans la carte locale du robot. Celui-ci maintient une carte de voisinage dans laquelle il met à jour les contraintes reçues entre les amers. Dans la seconde version de DDF-SAM (Cunningham et al. 2013), les auteurs fusionnent la carte locale et la carte de voisinage, mais veillent à retrancher préalablement l'information apportée par les autres robots à l'aide d'*anti-facteurs* lorsqu'ils calculent la distribution marginale à envoyer. Ceci permet d'éviter le comptage multiple des informations. (Lazaro et al. 2013) proposent une méthode similaire basée sur l'échange de mesures condensées (Grisetti et al. 2012). Lorsque les robots sont à portée de communications, ils échangent leur carte locale constituée des poses les plus récentes. Chacun recherche alors les nœuds avec lesquels il parvient à calculer les correspondances puis en informe l'autre. Les mesures condensées sont alors calculées entre les nœuds mis en correspondance. (Paull et al. 2015) appliquent cette idée en contexte sous-marin acoustique où les contraintes de bande-passante sont extrêmement fortes. Les robots échangent alors périodiquement des paquets résumant leurs cartes locales successives. Ils marginalisent la distribution locale sur les poses séparant les sous-cartes et les amers identifiés, puis l'approximent par un modèle simplifié par éparsification (Mazuran et al. 2014) tout en veillant explicitement à sa consistance. Face aux représentations exhaustives, les paquets condensés présentent l'avantage d'être moins coûteux à intégrer pour le récepteur. Une autre technique, complémentaire ou auto-suffisante, consiste à sélectionner un sous-ensemble d'informations à transmettre. C'est par exemple le cas de l'algorithme CTC proposé par (Nettleton et al. 2003) : il sélectionne les amers les plus informatifs dans les sous-cartes échangées.

La planification des échanges

Enfin, la planification des échange consiste à déterminer quelles informations envoyer à quels robots à un instant donné. Deux stratégies s'opposent. La première ne tient compte que des connaissances actuelles du robot émetteur pour décider quelles données envoyer. Selon cette approche, le robot communique de façon uniforme avec les autres robots de la flotte ou le serveur central. C'est la méthode adoptée par l'ensemble des méthodes centralisées dans lesquels le serveur doit aggréger les informations collectées par les agents sans que ceux-ci n'aient à se soucier de l'état des connaissances du serveur. La plupart des méthodes décentralisées relèvent également de cette stratégie par laquelle les robots communiquent périodiquement leurs nouvelles informations, que ce soit image-clé par image-clé (Lajoie et al. 2020 ; Zhang et al. 2018a), sous-carte par sous-carte (Schuster et al. 2015 ; Paull et al. 2015), ou bien mise-à-jour globale par mise-à-jour globale (Cunningham et al. 2010 ; Cunningham et al. 2013 ; Sartipi et al. 2019). Dans les derniers cas, les échanges sont la plupart du temps déclenchés selon des critères temporels pour assurer une certaine périodicité. (Sartipi et al. 2019) communiquent la sous-carte courante dès lors que le temps d'inférence qu'elle requiert excède un certain seuil. Notons que (Paull et al. 2015) maintiennent un historique de communication dont il se servent pour circonscrire les portions de la carte que doivent restituer les paquets envoyés.

La seconde stratégie est plus complexe : elle consiste à adapter les paquets transmis aux connaissances supposées des destinataires, voire à planifier les échanges de la flotte entière à un instant donné selon les connaissances de ses agents. C'est notamment l'approche adoptée dans les travaux de (Giamou et al. 2018) pour la fermeture de boucle distribuée, puis étendus par (Tian et al. 2018). Les robots s'échangent préalablement des métadonnées qui leur permettent de repérer les fermetures de boucles inter-robot potentielles sans pour autant les caractériser géométriquement. Ce sont des données légères telles que par exemple des vecteurs BoW associés aux images-clés. Vérifier ces fermetures de boucle candidates impose d'échanger dans un second temps des informations géométriques plus conséquentes telles que les primitives et les descripteurs. Ces échanges sont alors planifiés selon un problème d'optimisation graphique de couverture par sommets. La politique optimale minimise ainsi conjointement la charge réseau induite tout en répartissant le plus équitablement possible la charge de travail engendrée pour vérifier les fermetures de boucles.

Résumé

La politique de communication est l'ingrédient central d'une méthode de SLAM collaborative. Elle comporte trois volets que sont le choix de la topologie des échanges, la nature des données échangées et la planification des échanges. Si la topologie des échanges reflète essentiellement l'allocation retenue pour les tâches et les données, elle dispose malgré tout de plusieurs degrés de liberté qu'illustrent la diversité des méthodes centralisées. La nature des données échangées conditionne fortement leur intégration ultérieure par les robots récepteurs : les paradigmes adoptés oscillent entre la communication de données brutes et de données condensées afin de restituer aussi bien des informations de localisation pour ré-estimer les trajectoires que des informations visio-structurelles nécessaires à la détection des correspondances inter-robot. Enfin, la planification des échanges rythme les communications entre les robots et détermine les zones portions d'intérêts à communiquer, à travers une approche qui tient éventuellement compte des connaissances des autres robots de la flotte.

3.2.3 Stratégie d'association et de fusion des données

Le troisième volet d'une méthode multi-robots concerne l'intégration des données échangées. Celle-ci comprend deux étapes. La première consiste en une étape d'association de données multi-robots. Chaque robot recherche alors des correspondances entre ses propres connaissances et celles qu'il a reçues des autres robots. Sur la base de ces correspondances, la seconde étape consiste à fusioner ces cartes en une carte globale dans un référentiel commun. Lorsque les référentiels d'odométrie des robots sont inconnus initialement, l'estimation des transformations relatives (on parle d'ancrage) qui les lient est un préalable à l'inférence globale. On fusionne en particulier leurs modèles probabilistes sous-jacents pour permettre de raffiner l'ensemble par des inférences globales.

L'association de données multi-robots

Cette première étape repose sur la détection de correspondances entre les données échangées par les robots. Il s'agit d'une extension du module de fermeture de boucles qui vise à détecter de nouvelles contraintes inter-robot et en particulier apparier les variables dupliquées telles que des positions d'amers communément observés. On distingue les correspondances directes des correspondances indirectes. Les premières renvoient aux observations directes entre les robots, grâce auxquelles ils parviennent éventuellement à

estimer leur pose relative à un instant donné lorsqu'ils se rencontrent. Pour ce faire, une solution consiste à équipper les robots de marqueurs visuels détectables par les autres robots, tels que ARTag (Fiala 2005), AprilTag (Olson 2011), et ReacTIVision (Bencina et al. 2005). Initialement introduits pour des applications de réalité augmentée, leur usage a été popularisé pour la robotique. Les marqueurs AprilTag sont des codes QR ou matriciels desquels on extrait une signature binaire. Connaissant leur taille et les paramètres intrinsèques de la caméra, la simple détection d'un tel marqueur dans une image suffit non seulement pour estimer sa pose relative dans le référentiel de la caméra via l'estimation d'une homographie, mais également à l'identifier de façon unique. Les sources d'incertitudes (respectivement de l'ordre du centimètre et du degré) et leur ampleur sont analysées empiriquement par (Schuster 2019) et analytiquement par (Schweighofer et al. 2006). Les marqueurs AprilTags sont utilisés par de nombreux systèmes multi-robots, à l'instar de TagSLAM (Pfrommer et al. 2019), que ce soit pour identifier les robots eux-mêmes ou constituer des amers dans l'environnement. Notons également que de tels marqueurs peuvent être utilisés en début de session pour initialiser un référentiel d'odométrie commun entre les robots. Néanmoins, ils se montent difficilement sur des drones, à moins d'en réduire considérablement la taille. Pour les systèmes de suivi multi-caméras (*Motion Capture* ou MoCap en anglais) tels que VICON, on préferera d'ailleurs utiliser de petits marqueurs sphériques uniformes, moins susceptibles d'altérer la dynamique du drone et moins sensibles aux problèmes d'illumination. Dans le cas des drones, d'autres types de marqueurs sont possibles tels que des constellations de marqueurs infrarouges ou ultraviolets (Walter et al. 2018) ou constitués de LED (Teixeira et al. 2018). Il existe également des méthodes basées apparence (Madhavan et al. 2004) qui utilisent des techniques de reconnaissance d'objet, afin de se dispenser d'équipper le robot de marqueurs. Connaissant le modèle ou l'apparence des autres robots, l'idée est de les détecter directement dans les observations et de l'utiliser pour caractériser la correspondance ainsi obtenue (e.g. calcul de poses relatives).

Les correspondances indirectes sont de deux natures : simultanées et différées. Dans le cas simultané, elle constituent une alternative aux détections directes lors des rencontres entre plusieurs robots. Elles consistent alors à comparer des acquisitions concomittantes pour en extraire des observations communes, à partir desquelles des poses relatives sont estimées. Dans le système proposé par (Tardioli et al. 2015), deux robots se synchronisent pour échanger les points d'intérêts extraits de leurs observations courantes ainsi que les mots visuels associés à leurs descripteurs. Les correspondances trouvées permettent

d'estimer des homographies relatives ou des poses relatives à un facteurs d'échelle près. (Achtelik et al. 2011) proposent de lever l'ambigüité sur ce facteur d'échelle à l'aide des mesures inertielles au sein d'un EKF qui assemble les caméras des deux robots en un banc stéreo-visuel dynamique virtuel. Dans le cas de robots hétérogènes telles que des robots terrestres et des drones, la détection des correspondances doit tenir compte des différences de point de vue. (Morel et al. 2009) proposent à ce titre une méthode d'appariement visuel invariante aux transformations affines mais dont les temps d'exécution sont incompatibles avec une application temps réel. D'autres méthodes estiment les poses relatives grâce à des recalages entre les cartographies du drone et du robot terrestre. C'est le cas des travaux de (Käslin et al. 2016), (Shim et al. 2017) et (Oleynikova et al. 2015). Il s'agit de méthodes de localisation collaborative dans lesquelles l'estimation des poses relatives coïncide souvent avec le recalage complet des cartes par des algorithmes coûteux tels que l'ICP. Enfin, dans le cas visio-inertiel, (Sartipi et al. 2019) ont récemment proposé de détecter les correspondances inter-robots à partir de l'échange et la mise à jour de sous-cartes associées aux observations courantes.

Les correspondances indirectes différées sont des fermetures de boucles inter-robot. Dans le cas d'architectures centralisées, leur détection incombe au serveur central. Ce dernier met à profit sa puissance de calcul pour rechercher les correspondances entre une image de requête et les images collectées auprès de tous les robots à l'aide d'algorithmes classiques de fermeture de boucle. La grande majorité des systèmes centralisés mentionnés précédemment, de CoSLAM (Zou et al. 2012) à CVI-SLAM (Karrer et al. 2018), s'en remettent aux seules fermetures de boucles inter-robot en vue de fusionner les cartes ultérieurement. Dans le cas des méthodes décentralisées, on impose souvent que les robots s'échangent leur images-clés, et chaque robot est seul responsable pour détecter les fermetures de boucles inter-robot. Se posent alors des problématiques d'échelles lorsque le nombre de robots augmente. La détection des fermetures de boucles intra et inter-robot peut néanmoins tirer parti de structures distribuées, comme proposé par (Cieslewski et al. 2017). Dans cette approche, les auteurs utilisent le descripteur global NetVlad (Arandjelovic et al. 2016), et partitionnent l'espace de description associé. À chaque robot est attribuée la responsabilité d'une partition. Les robots calculent le descripteur NetVlad pour chaque nouvelle image-clé, et l'envoie au robot correspondant. Celui-ci lui renvoie alors l'identifiant des images-clés les plus proches. Le premier robot peut alors s'adresser aux robots associés à ces correspondances pour leur demander les images-clés retournées. Ce système décentralisé permet d'envoyer les requêtes à un seul robot, mais il

exige une connexion constante entre tous les robots. La méthode introduite par (Giamou et al. 2018) permet également de vérifier des fermetures potentielles dans un cadre décentralisé. DOOR-SLAM (Lajoie et al. 2020) fait préalablement s'échanger des descripteurs globaux NetVLAD entre les robots, et ne communique points caractéristiques et descripteur que dans le cas de correspondances potentielles. Une fois que les cartes sont recalées, ou bien sous hypothèse de poses initiales connues dans un référentiel commun, la détection des recouvrements permet de caractériser de nouvelles correspondances. Cette techniques est utilisée par (Schuster et al. 2015) pour estimer les poses relatives entre des sous-cartes dès lors qu'elles présente un recouvrement initial suffisant. Leurs nuages de points stéréo-visuels associés sont alors recalés par ICP. Enfin, certaines méthodes reposent sur l'identification d'amers communément observés, telles que DDF-SAM (Cunningham et al. 2013) ou encore (Paull et al. 2015). Pour cette dernière, on suppose que les amers sont distribués de manière suffisamment éparse pour être associés par plus proche voisin.

La détection des correspondances entre les observations des robots est une première étape, la seconde consiste à les propager entre les robots lorsque la politique de communication retenue le prévoit. En contexte centralisé, ce rôle est assuré par le serveur lorsqu'il renvoie des informations aux agents. C'est le cas dans CVI-SLAM (Karrer et al. 2018) : le serveur communique aux agents les portions des cartes des autres robots avec lesquels il a détecté des fermetures de boucles inter-robot. Ainsi, l'agent peut utiliser ces informations de covisibilité pour accroître sa carte locale et améliorer la précision de son odométrie. En contexte décentralisé, (Montijano et al. 2013) proposent une approche par consensus afin de propager les correspondances locales détectées entre robots communiquant entre eux, dans un contexte limité en communications, et en déduire des correspondances globales. Cette propagation se double parfois de mécanismes de filtrage des correspondances douteuses, basées sur l'analyse de cycles de correspondances (Estrada et al. 2009). (Lajoie et al. 2020) étendent la méthode PCM proposée par (Mangelson et al. 2018) qui recherche le plus grand ensemble de correspondances inter-robot mutuellement cohérentes. Cette vérification est effectuée de façon distribuée sur les cycles de correspondances inter-robot. Les correspondances peuvent également être vérifiées durant des phases d'inférence comme expliqué ci-dessous.

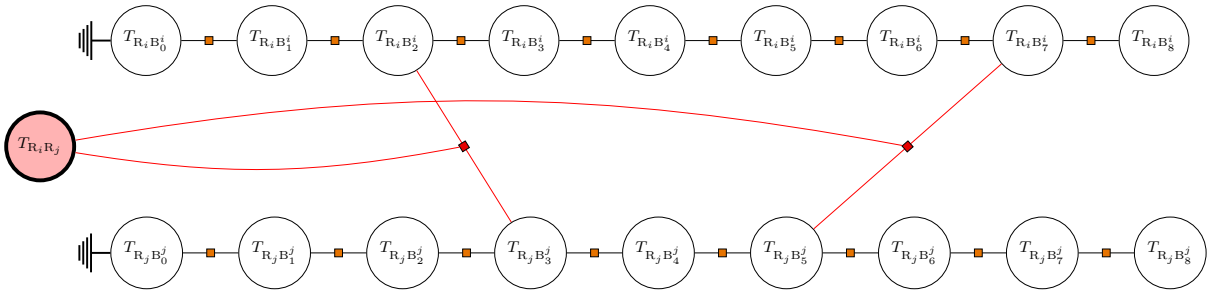
La fusion de données multi-robots

La fusion de données multi-robots étend le module d'inférence du SLAM mono-robot. Elle consiste à estimer une carte globale à partir des cartes individuelles des robots et des

correspondances qui les contraignent mutuellement. Bien qu'elle reprenne les paradigmes de l'inférence mono-robot que nous avons présenté dans la section 2.5, l'inférence multi-robots doit estimer conjointement les variables associées à plusieurs robots, inclure les correspondances inter-robot, ancrer les référentiels d'odométrie puis raffiner leur pose, et fusionner les variables dupliquées. C'est à ce niveau que se pose le problème de la consanguinité des données, auquel les méthodes de filtrages sont particulièrement sensibles. Il survient si les corrélations entre certaines mesures ou connaissances sont ignorées durant la fusion. Alors que l'inférence multi-robots est assurée par le serveur dans les architectures centralisées, elle incombe à chaque robot dans le cas décentralisé. Cependant, certaines approches proposent des inférences distribuées : les robots se répartissent alors les calculs de l'inférence globale sans pour autant partager préalablement leurs données. On pourra également se référer à l'article publié par (Indelman 2015) qui dresse un panorama des méthodes d'inférence multi-robots.

Les premières solutions d'inférence proposées pour le SLAM multi-robots relèvent du filtrage et étendent principalement des approches d'abord développées pour la localisation collaborative. (Roumeliotis et al. 2000) abordent ce problème à l'aide d'un filtre de Kalman étendu conçu dans un cadre centralisé où les robots partagent un référentiel commun, et en dérivent une version décentralisée, dans laquelle les robots ne doivent échanger des informations que lorsqu'une nouvelle mesure de pose relative entre deux robots est disponible. (Carrillo-Arce et al. 2013) reprennent cette approche qu'ils couplent avec la méthode d'intersection des covariances (Julier et al. 2017). De cette façon, chaque robot maintient un EKF pour son seul état, et fusionne les mesures de pose relative de façon conservative. D-SLAM (Nettleton et al. 2003) étend la méthode précédente pour le SLAM décentralisé. Les robots estiment leur carte à l'aide d'un filtre de Kalman étendu, mais communiquent des sous-cartes sous forme informative. Du fait de ses propriétés additives, cette forme sied particulièrement aux problèmes d'échange d'informations. Chaque robot implémente un filtre d'information, appelé *channel filter*, comme interface avec chaque interlocuteur. Ce filtre maintient le vecteur et la matrice d'information associée à tous les amers communiqués. Ce faisant, il peut, dans une approche proche de DDF-SAM (Cunningham et al. 2010), retrancher l'information déjà comptabilisée des sous-cartes qu'il reçoit. Pour cela, il utilise lui aussi la méthode d'intersection des covariances qui produit une estimée certes consistante mais très conservative. (Thrun et al. 2005) appliquent les filtres d'information étendus (*Sparse Extended Information Filters* (SEIF) au contexte multi-robots. (Zhou et al. 2006) s'intéressent à l'estimation d'un référentiel commun à

partir des observations directes des robots entre eux. Enfin, (Howard 2006) adaptent le filtre particulaire de FastSLAM (Hahnel et al. 2003) à l'inférence multi-robots.



Dans ce graphe de facteurs, les R_i et R_j désignent respectivement les référentiels d'odométrie des robots i et j, tandis que B_k^i désigne le référentiel inertiel attaché au robot i au temps k. Ces variables sont liées par des facteurs odométriques ■ de pose relative et des correspondances inter-robot ■. Le noeud d'ancrage ◯ désigne l'offset entre les référentiels d'odométrie R_i et R_j, et intervient explicitement dans les facteurs des correspondances. Le symbole ⫴— indique que la variable associée est fixée.

Figure 3.3 – Structure de graphe de poses multi-robots proposée par (Kim et al. 2010)

La flexibilité des méthodes graphiques a permis de les étendre naturellement au cas multi-robots. C-SLAM (Andersson et al. 2008) transpose ainsi directement le formalisme des graphe de facteurs à la fusion de sous-cartes échangées entre les robots. Les correspondances inter-robot se traduisent par l'ajout de nouveaux facteurs entre les poses de plusieurs robots ou par la fusion des noeuds d'amers dupliqués. (Kim et al. 2010) proposent une extension multi-robots de iSAM (Kaess et al. 2008). De plus, ils incluent explicitement l'estimation des transformations entre le référentiel global et les référentiels d'odométrie des robots sous forme de *nœuds d'ancrage* dans le graphe de facteurs global, représenté par la Figure 3.3. La formulation du problème d'estimation sous la forme d'un problème d'optimisation permet d'adapter des mécanismes d'optimisation distribuée au contexte multi-robots. C'est notamment l'approche proposée par (Choudhary et al. 2017), puis reprise par (Cieslewski et al. 2018) et (Lajoie et al. 2020). Les robots échangent les estimées des seules poses impliquées dans des correspondances inter-robot jusqu'à ce qu'ils atteignent un consensus. Le problème global est optimisé par une méthode de Gauss-Newton dont chaque itération est résolue par une approche de Gauss-Siedel distribuée. L'estimation distribuée peut être couplée au problème d'association de données lors de l'estimation d'un référentiel commun pour des robots dont les positions initiales sont inconnues. (Indelman et al. 2016) proposent une optimisation par maximisation de l'espérance (Moon 1996). Dans ce cas, on associe à chaque correspondance une va-

riable latente binaire indiquant son status d'*inlier* ou d'*outlier*. L'inférence alterne alors entre marginalisation des variables latentes, et optimisation MAP sur la vraisemblance résultante.

Résumé

La politique d'association et de fusion de données multi-robots comprend trois volets. Le premier est la détection des correspondances inter-robot directes via l'observation de marqueurs montés sur les robots ou bien par apparence, et indirectes en comparant les observations d'un robot à la carte d'un autre. Dès que suffisamment de correspondances sont identifiées, les référentiels d'odométrie des robots peuvent être ancrés les uns par rapport aux autres, et des inférences globales peuvent être pratiquées afin d'exploiter les informations mutuelles entre les observations des robots.

3.3 Conclusion

Dans ce chapitre, nous avons dressé un état de l'art du SLAM collaboratif en contexte visuel et présenté les trois aspects essentiels de toute architecture de SLAM multi-robots. Comme nous l'avons indiqué précédemment, il en ressort que jusqu'à aujourd'hui, et en particulier jusqu'au début de cette thèse, la recherche a essentiellement porté sur les méthodes centralisées pour lesquelles des solutions robustes ont été proposées en contextes mono-visuel et visio-inertiels. Néanmoins, la recherche sur les méthodes décentralisée a délaissé certains champs. En contexte stéréo-visuel, (Schuster et al. 2015) ont développé une méthode particulièrement aboutie basée sur la fusion de sous-cartes denses. Dans le chapitre 6, nous investiguerons une méthode stéréo-visuelle décentralisée appliquée à une représentation dense plus flexible de type TSDF, à des fins de reconstruction de l'environnement. Enfin, dans le chapitre 5, nous investiguerons différents paradigmes d'échanges de données en contexte visio-inertiel, à partir desquels nous proposerons trois méthodes de SLAM collaboratif décentralisé pour l'estimation des trajectoires.

Deuxième partie

Contributions

Chapitre 4

Construction d'un jeu de données multi-robots

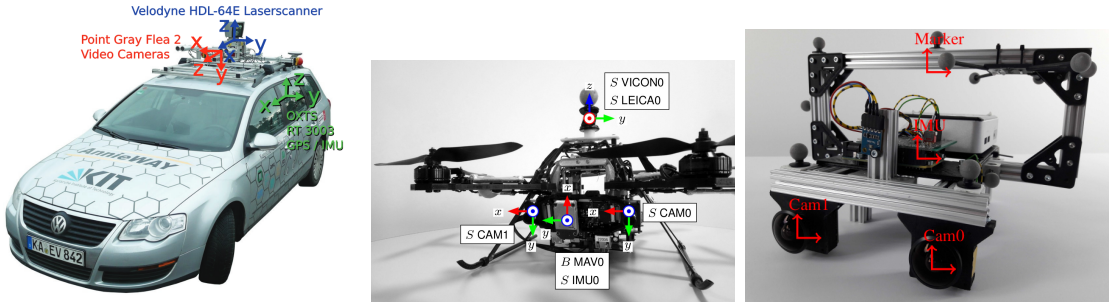
4.1 Introduction

La conception d'algorithmes de localisation et de cartographie impose de les tester au plus proche des conditions réelles. C'est pourquoi on procède généralement en rejouant des séquences d'acquisitions réelles issues de jeux de données publics, qui fournissent une référence de comparaison commune. La littérature abonde de tels jeux de données, dont certains ont profondément impacté la recherche en SLAM. C'est par exemple le cas de KITTI (Geiger et al. 2013) et d'EuRoC (Burri et al. 2016), respectivement pour le SLAM visuel et visio-inertiel. Cependant, si la littérature regorge de jeux de données pour évaluer les algorithmes mono-robot, peu se consacrent explicitement au SLAM multi-robots. En effet, nous verrons que si l'émergence des jeux de données multi-session offre une alternative acceptable, ceux-ci n'en restent pas moins limités par différents écueils.

Constatant ce manque, nous avons mené deux campagnes d'acquisition de jeux de données spécifiquement dédiés à l'évaluation des algorithmes de SLAM multi-robots, et ainsi tester les méthodes développées en parallèle. Nous entamons ce chapitre par une revue de la littérature existante quant aux jeux de données publiés pour évaluer les méthodes de SLAM. Dans un second temps, nous présentons les principes directeurs que nous avons suivis afin de concevoir les scénarios multi-robots, puis nous présentons la démarche adoptée pour acquérir ces scénarios, les post-traiter et les évaluer.

4.2 Travaux associés

La conception de jeux de données a jalonné le développement des algorithmes de SLAM. Dans cette section, nous retraçons brièvement les tendances qui ont orienté la



(a) Voiture instrumentée utilisée dans KITTI (Geiger et al. 2013)

(b) Drone utilisé dans EuRoC (Burri et al. 2016)

(c) Banc de capteurs utilisé dans TUM-VI (Schubert et al. 2018)

Figure 4.1 – Exemples de bancs de plateformes robotiques

conception de ces jeux de données et leur utilisation en contexte multi-robots. On peut classer les jeux de données disponibles d'une part selon les plateformes robotiques et les bancs de capteurs qu'ils emploient, selon les propriétés de l'environnement qu'ils mettent en scène et des trajectoires parcourues par les robots, puis enfin selon les applications auxquelles ils se destinent.

4.2.1 Plateformes robotiques et les capteurs utilités

Les voitures instrumentées

La première famille de jeux de données couvre les séquences acquises par les capteurs embarqués par des véhicules. Les plus notables sont notamment les jeux de données MALAGA (Blanco-Claraco et al. 2014), Oxford RobotCar (Maddern et al. 2017), ou encore le celèbre KITTI (Geiger et al. 2013) (c.f. Figure 4.1a). Ce dernier s'est imposé comme un étalon algorithmique dans plusieurs domaines allant de l'odométrie à la segmentation sémantique. Les véhicules disposent de l'envergure suffisante pour embarquer de nombreux capteurs, dont en particulier des capteurs volumineux tels que des systèmes multi-caméras, des capteurs LiDARs ou des système de navigation inertiels aidés par GPS (GNSS), plus précis que leurs analogues compacts utilisés en robotique mobile. Par exemple, le véhicule de MALAGA embarque 5 capteurs LiDAR, et celui de MIT DARPA (Huang et al. 2010a) en comporte 13. Généralement, on profite de cette latitude pour combiner au moins un capteur LiDAR, un banc de caméras stéreo et un module GNSS. Le récent jeu de données MVSEC (Zhu et al. 2018) y ajoute des caméras évenementielles pour gérer des séquences rapides acquises depuis un motocycle. Enfin, la disponibilité

des signaux GPS facilite le calcul des vérités terrain. Elle est déterminée à une précision centimétrique dans la première version de MALAGA (Blanco et al. 2009) en fusionnant les signaux GPS acquis par plusieurs récepteurs répartis sur le véhicule.

Les plateformes robotiques autonomes

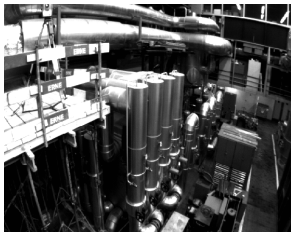
Une seconde famille englobe les jeux de données acquis à l'aide de plateformes robotiques classiques, autonomes en ce sens qu'elles n'embarquent aucun opérateur humain, et telles que des drones ou des robots terrestres. Si ces derniers sont plus contraints que les véhicules en termes de charges utiles, ils peuvent néanmoins évoluer dans des environnements plus variés et suivre des trajectoires plus malléables. Historiquement, les premiers jeux de données publiés s'appliquent à l'estimation 2D par LiDAR embarqué par une plateforme terrestre. C'est le cas de Victoria Park (Guivant et al. 2002) et de Intel Research Lab (Carlone et al. 2014). Rawseeds (Ceriani et al. 2009) ouvre la voie à l'estimation 3D et complète les capteurs LiDAR par des caméras. Cette tendance se confirme avec les jeux de données New College (Smith et al. 2009) et Ford Campus (Pandey et al. 2011). L'émergence des jeux de données aériens est plus tardive, car concomittante à l'essor des capteurs et des algorithmes basés vision pour ces plateformes. Les jeux de données EuRoC (Burri et al. 2016) (c.f. Figure 4.1b) et Zurich Urban Micro Aerial Dataset (Majdik et al. 2017) en sont deux exemples récents. EuRoC est d'ailleurs devenu un étalon de référence pour évaluer les algorithmes de SLAM visuels et visio-inertiels.

Les capteurs portatifs

On parle de SLAM portatif lorsque le banc de capteurs est directement manipulé par un opérateur humain. Bien que les trajectoires qui en résultent restituent les mouvements induits par la marche, ce SLAM s'adapte particulièrement aux environnements d'intérieur, et notamment à l'ascension d'escaliers. Quelques exemples de jeux de données de SLAM portatifs sont TUM-MonoVO (Engel et al. 2016) dans le cas monoculaire, TUM-RGB-D (Sturm et al. 2012) dans le cas RGB-D pour d'odométrie directe, ainsi que les récents PennCosyvio (Pfrommer et al. 2017) et TUM-VI (Schubert et al. 2018) pour le SLAM stéréo-inertiel (c.f. Figure 4.1c).

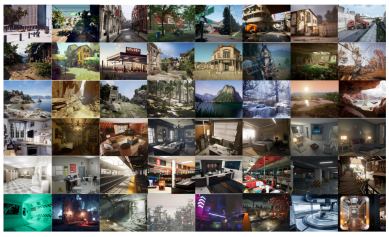
Les plateformes synthétiques

Très récemment, des jeux de données entièrement synthétiques ont été publiés. En effet, les simulateurs dédiés à l'expérimentation robotique tels que Gazebo (Koenig et al. 2004) ont accompli d'énormes progrès en termes de réalisme, que ce soit pour générer des trajectoires (à l'instar de Rotors (Furrer et al. 2016) dans Gazebo) simuler les acquisitions des capteurs visuels. Portés par les moteurs graphiques tels que Unreal-Engine 4 (UE4) et Unity3D, développés par l'industrie du jeu vidéo, les environnements de simulation produisent aujourd'hui des visuels de haute qualité, exploitables pour des algorithmes basés vision. À titre d'exemples, les simulateurs CARLA (Dosovitskiy et al. 2017), Virtual KITTI (Gaidon et al. 2016), AirSim (Shah et al. 2018) et TartanAir (Wang et al. 2020) exploitent tous trois UE4 pour simuler les déplacements et les acquisitions de véhicules autonomes dans une grande variété d'environnements et de conditions météorologiques. La virtualisation permet de simuler un grand nombre de capteurs. TartanAir simule non seulement les images acquises par les caméras monoculaires, stéréos et RGB-D, mais également les scans LiDAR. AirSim simule également les baromètres, les centrales inertielles, les magnétomètres, les signaux GPS. Tout ceci repose sur une modélisation fine de la physique, de la dynamique des robots et de leurs capteurs. Par exemple, dans le jeu de données BlackBird (Antonini et al. 2018), les auteurs simulent jusqu'à la dérive temporelle des horloges des centrales inertielles. Ce dernier utilise d'ailleurs le module FlightGoggles (Sayre-McCord et al. 2018), développé pour la navigation visio-inertielle. La synthèse des jeux de données permet de plus de s'abstraire des diverses contraintes qui ponctuent généralement les campagnes d'acquisition. Ainsi, les calibrations des capteurs virtuels sont parfaitement connues là où leur estimation restent entâchée d'incertitudes dans le cas réel. Les acquisitions ne souffrent d'aucune avarie technique telle que des erreurs de trajectoires, des décalibrations fortuites et autres problèmes matériels. Au contraire, les acquisitions deviennent précises, optimales et automatisées. Dans TartanAir (Wang et al. 2020), les trajectoires des caméras virtuelles sont générées dans les environnements simulés selon les contraintes dynamiques des robots, mais également de façon à leur conférer des propriétés globales (couverture de l'espace, mouvements aggressifs etc.). Ces avantages se concrétisent également dans le post-traitement, et en particulier la détermination des vérités terrain, étape pourtant délicate dans le cas réel. Les trajectoires simulées sont parfaitement connues, de même que les cartes de disparités ou les flots optiques observés. L'inconvénient principal des jeux de données synthétiques pour l'évaluation d'algorithmes est l'incertitude qui persiste quant au passage au réel.



(a) Visuel extrait de la séquence Machine Hall 1 d'EuRoC (Burri et al. 2016)

(b) Visuel extrait de Robex (Vayugundla et al. 2018)



(c) Mosaïque de scènes simulées par TartanAir (Wang et al. 2020)

Figure 4.2 – Exemples de visuels des environnements explorés ou simulés par des jeux de données

En effet, les distributions des bruits de mesures peuvent différer, certaines perturbations peuvent ne pas avoir été prises en compte, etc.

4.2.2 Propriétés environnementales et cinématiques

Les environnements explorés

Les propriétés des environnements mis en scène impactent les algorithmes de SLAM de deux façons. Tout d'abord, elles impactent la qualité des estimations du module de localisation et de cartographie locale, et en particulier lorsqu'il repose sur un suivi visuel. D'autre part, elle influe sur la détection des correspondances indirectes entre les robots. Une grande diversité d'environnements a été balayée par les jeux de données publiés. Quand ils se destinent à la conduite autonome, ces jeux de données, tels que KITTI (Geiger et al. 2013) et Malaga (Blanco-Claraco et al. 2014), présentent des environnements routiers et urbains très structurés. Certains d'entre eux prêtent particulièrement attention aux changements saisonniers. C'est le cas du jeu de données Oxford RobotCar (Maddern et al. 2017) qui cumule plus de 1000 kilomètres de trajectoires acquises sur plusieurs saisons. Les jeux de données impliquant des plateformes robotiques autonomes fournissent une plus ample palette d'environnements. Outre les milieux urbains explorés dans le jeu de données Zurich OldTown (Schneider et al. 2018), on trouvera également les environnements intérieurs (Sturm et al. 2012), industriels avec EuRoC (Burri et al. 2016) (c.f. Figure 4.2a), des milieux sous-marins avec AquaLoc (Ferrera et al. 2019a), sous-terrains avec Chilean Underground (Leung et al. 2017) ou extérieurs dans UZH-FPV Drone Racing (Delmerico et al. 2019). Aujourd'hui, les problématiques associées

à l'exploration extra-planétaire motivent la publication de jeux de données impliquant des plateformes terrestres évoluant sur des terrains accidentés. Ainsi, les jeux de données ROBEX (Vayugundla et al. 2018) (c.f. Figure 4.2b), Canadian Planetary Dataset (Lamarre et al. 2020) et Katwijk Beach dataset (Hewitt et al. 2018) émulent des environnements lunaires et martiens. Enfin, dans le cas des jeux de données synthétiques, la virtualisation rend accessible une grande variété d'environnement et de mouvements. Par exemple, TartanAir (Wang et al. 2020) couvre des scènes urbaines, d'extérieur, d'intérieur, des paysages naturels et même des environnements issus de la science fiction, comme en témoigne la Figure 4.2c. De plus, certains moteurs graphiques gèrent les interactions avec des objets mobiles et permettent un réglage précis de paramètres environnementaux tels que la luminosité, la météo, etc.

Les propriétés des trajectoires

Les propriétés des trajectoires qui impactent les performances des algorithmes de SLAM sont leurs propriétés cinématiques. En effet, celles-ci affectent fortement les performances d'estimation des robots, que ce soit dans le cas LiDAR où la vitesse peut induire des effets de distorsion sur les scans, ou dans le cas visuel où les mouvements de rotations brutaux peuvent perturber le suivi visuels des amers. Les jeux de données acquis depuis des voitures instrumentées rivalisent avant tout quant à la durée de leurs séquences et aux distances parcourues. Si MALAGA (Blanco-Claraco et al. 2014) consiste en une séquence de 93 minutes, KITTI cumule par exemple plus de 6 heures d'acquisitions. La trajectoire cumulée de Oxford RobotCar excède quant à elle le millier de kilomètres. Les voitures constituent des supports stables soumis à de faibles accélérations. Au contraire, les séquences acquises à l'aide de drones permettent de générer des trajectoires aux degrés d'agressivité variés. Le jeu de données EuRoC (Burri et al. 2016) propose ainsi cinq séquences de difficulté croissante, tandis que UHZ-FVP Drone Racing (Delmerico et al. 2019) conçoit des trajectoires volontairement très agressives. Les jeux de données synthétiques peuvent quant à eux générer des trajectoires aux propriétés cinématiques globales et locales très précises comme c'est le cas dans TartanAir (Wang et al. 2020).

4.2.3 Applications visées des jeux de données

Enfin, les jeux de données disponibles se classent selon les applications auxquelles ils peuvent prétendre. S'ils sont bien entendu tous adaptés à l'évaluation d'algorithmes de

SLAM mono-robot, nous distinguons deux types d'applications multi-robots que sont le SLAM multi-session et le SLAM collaboratif.

Le SLAM multi-session

Le SLAM multi-session exploite les correspondances inter-robot qu'il identifie entre des observations et des trajectoires acquises et estimées de façon différée, éventuellement par plusieurs robots. Même s'ils n'ont pas été initialement conçus dans une optique multi-robots, cetains de ces jeux de données peuvent s'y prêter. C'est le cas de KITTI dont les trajectoires de certaines de ses séquences se recouvrent en partie, et dont certaines séquences individuelles sont suffisamment longues pour être segmentées en plusieurs sous-trajectoires. Ces approches sont utilisées par certains travaux multi-robots, tels que (Giamou et al. 2018) et (Cieslewski et al. 2017). Le fait que plusieurs séquences aient été acquises aux mêmes endroits rend EuRoC viable pour le SLAM multi-session, et par extension, le SLAM multi-robots. À ce titre, les auteurs de CVI-SLAM (Karrer et al. 2018) resynchronisent les séquences Machine Hall pour simuler des acquisitions simultanées. Les deux jeux de données introduits avec Maplab (Schneider et al. 2018), à savoir Zurich Old Town et CLA, relèvent également de cette catégorie. Le premier comprend 45 séquences acquises dans les rues de Zurich et cumule près de 17 kilomètres de trajectoires.

Le SLAM collaboratif

Contrairement au SLAM multi-session, le SLAM collaboratif implique la simultanéité des acquisitions et des interactions entre les robots. Certes, les jeux de donnée adaptés au SLAM multi-session peuvent servir à évaluer des algorithmes de SLAM collaboratif à condition de préalablement synchroniser leurs séquences. Cependant, utiliser de tels jeux de données pour évaluer des algorithmes de SLAM multi-robots présente deux écueils majeurs. Tout d'abord, on néglige les interactions entre les robots telles que leurs observations directes. Enfin et surtout, les trajectoires de ces jeux de données sont souvent conçues individuellement pour constituer des scénarios auto-sufisants. Ainsi, les trajectoires ne sont pas élaborées dans une approche globale qui s'intéresserait à la distribution spatio-temporelle des correspondances intra et inter-robot le long des trajectoires. La simple resynchronisation de séquences différées ne permet pas de contrôler les propriétés des scénarios qui en résultent, ni d'en garantir l'intérêt. Par exemple, un algorithme de SLAM multi-robots aura peu à apporter si les deux trajectoires mises en regard jouissent déjà d'un grand nombre de fermetures de boucles en leur sein, et que les fermetures de

boucles inter-robot n'incluent que de faibles portions des trajectoires de part et d'autre. À notre connaissance, UTIAS (Leung et al. 2011), publié en 2011, est le seul jeux de données spécifiquement consacré au SLAM collaboratif, en ce sens qu'il contient les acquisitions simultanées des robots et leurs observations directes. Il implique 5 robots équipés d'une centrale inertielle, d'une caméra monoculaire et d'un capteur LiDAR 2D. La surface d'acquisition mesure 15 mètres × 8 mètres. Elle est balisée par un maximum de 15 amers cylindriques identifiables de façon unique à l'aide de leur code barre. On identifie également les robots de façon similaire. Le jeu de données consiste en 9 scénarios dans lesquels le placement des amers et les trajectoires des robots diffèrent. Mis à ce part ce jeu de données 2D, aucun autre ne propose des scénarios simultanés impliquant plusieurs robots, et spécifiquement conçu pour tester des algorithmes de SLAM multi-robots. Encore récents, les jeux de données synthétiques ne couvrent pas encore le cas des scénarios multi-robots. Ils en constitueraient néanmoins une extension naturelle qui permettrait de contourner les difficultés techniques inhérentes aux acquisitions simultanées. Face à ce constat, nous avons mené deux campagnes d'acquisition durant cette thèse afin de constituer des jeux de données spécifiquement dédiés à l'évaluation d'algorithme de SLAM multi-robots. La première a eu lieu durant une journée à Sotteville-lès-Rouen en 2018 et la seconde, plus conséquente, sur trois jours à l'ONERA Meudon en 2019.

4.3 Acquisition des jeux de données

4.3.1 Plateformes robotiques et calibration

À Sotteville et à Meudon, nous avons utilisé des robots terrestres ainsi que des drones représentés par la Figure 4.3. Les robots terrestres sont des Wifibots™ Lab V4 (voir la Figure 4.3a). À Sotteville, nous avons utilisé deux robots terrestres respectivement dénommés A et B, ainsi qu'un drone de modèle AscTec Pelican (c.f. Figure 4.3b). À Meudon, les acquisitions ont impliqué trois robots Wifibots A, B et C, ainsi qu'un drone de modèle DJI-M100 (Figure 4.3c). Chaque robot, qu'il soit terrestre ou aérien, était équipé a minima d'un banc stéréo-visuel et d'une centrale inertielle. Chaque banc stéréo comprenait deux caméras monochromes IDS UL124xLE de focale 4mm, séparées d'une distance de base de 26cm. Leur résolution était de 640 × 512 pixels, tandis que leur fréquence d'acquisition était de 20 Hz. Durant les acquisitions à Meudon, les robots possédaient également un banc visio-inertiel supplémentaire de type d435™. Sur les Wifibots, nous avons utilisé

une centrale inertielle de modèle MPU-9250™ fonctionnant à 100 Hz. Sur les drones, nous avons exploité les centrales inertielles associées au pilote automatique. Les robots étaient connectés entre eux à l'aide d'un routeur Wifi TP-Link Archer-C7 à 5Ghz. Nous avons collecté les acquisitions des robots directement sur une station centrale à travers le réseau sous forme de fichiers `.bag` en utilisant le middleware ROS (Quigley et al. 2009). En particulier, un protocole de temps réseau (*Network Time Protocol* (NTP)) commun assurait la synchronisation entre les horloges des robots.

(a) Robots Wifibot (b) Drone AscTec Pelican (c) Drone DJI-M100

Certains Wifibots sont dotés de marqueurs AprilTags (Olson 2011) , afin de permettre d'estimer des poses relatives entre les robots lors d'observations directes.

Figure 4.3 – Plateformes robotiques utilisées dans les jeux de données acquis à Sotteville et Meudon

Nous avons équippé deux robots terrestres de deux marqueurs AprilTags portatifs (Olson 2011) comme représenté dans la Figure (4.3a). La détection de ces marqueurs permet d'estimer des poses relatives entre les robots lorsque ceux-ci s'observent directement. Le premier marqueur pointe vers l'arrière du robot tandis que le second est orienté vers le haut, de sorte à être observable par le drone.

Pour que les séquences acquises soient utilisables par des algorithmes d'odométrie et de SLAM visio-inertiels, on doit fournir une calibration fiable des caméras et des centrales inertielles afin de paramétrer leurs modèles de mesure précédemment présentés dans la section 2.2.3. Calibrer une caméra consiste à estimer les paramètres intrinsèques (coordonnées du point principal et distances focales) ainsi que les coefficients des modèles de distorsion choisis (équidistant ou radial-tangentiel). De plus, dans le cadre d'un banc visio-inertiel, on doit également estimer des paramètres relatifs entre les caméras et la centrale inertielle. Il s'agit d'une part de la transformation dite *extrinsèque* entre l'IMU et les caméras, et d'autre part, du délai entre les horloges des caméras et de la centrale inertielle. On parle de calibration spatiale pour l'estimation des paramètres géométriques, et de calibration temporelle quant à l'estimation de ce délai. Pour ce faire, nous avons

utilisé le logiciel Kalibr (Furgale et al. 2014). Celui-ci estime les calibrations à partir de l'observation de mires fixes aux motifs aisément repérables tels que des marqueurs AprilTag (Olson 2011). En contexte visio-inertiel, il convient alors d'exciter suffisamment l'IMU avec des mouvements de rotation et de translation circulaire selon les trois axes. On doit également s'assurer que l'observation des mires balaie l'exhaustivité du plan image, et en particulier ses coins, afin de parfaire l'estimation des coefficients de distorsion. Les résultats des calibrations sont fournis avec les jeux de données. Un dernier aspect des calibrations consiste à estimer les transformations spatiales entre les repères des marqueurs AprilTags portatifs et les référentiels des caméras. Dans ce but, on acquiert des séquences dans lesquelles la mire est simultanément observée par une caméra du robot et une autre caméra mobile calibrée qui observe également les marqueurs portatifs. Quant aux paramètres des IMU, ils sont estimés en calculant la variance d'Allan (El-Sheimy et al. 2007) à partir des mesures acquises pendant plusieurs heures tout en maintenant l'IMU immobile.

4.3.2 Lieux d'acquisition

Entrepôt SNCF de Sotteville-lès-Rouen

La zone d'acquisition à Sotteville-lès-Rouen est un entrepôt mis à disposition par la Société Nationale des Chemins de Fer (SNCF). Le plan en est donné sur la Figure (4.6a), tandis que la Figure (4.4) fournit un aperçu global. Ce visuel ne date cependant pas du jour de l'acquisition, ce qui explique que l'arrangement décrit sur le plan en diffère quelque peu, notamment quant à la présence d'un îlot dans la grande zone vacante sur la droite de l'image. Le terrain mesure 36 mètres sur 48 mètres et comprend plusieurs zones distinctes séparées les unes des autres par de grandes caisses et autres éléments ferroviaires stockés en rangées et en îlots.

Ancien musée de l'air à l'ONERA Meudon

La zone d'acquisition est un grand hangar, qui faisait autrefois partie du Musée de l'Air de Chalais-Meudon, de 1921 jusqu'en 1977, avant que sa collection ne soit transférée au Musée de l'Air et de l'Espace du Bourget. Aujourd'hui, c'est un vaste entrepôt mesurant 80 mètres par 40 mètres. La zone d'acquisition comporte deux parties distinctes. La partie Nord est majoritairement vide tandis que la zone Sud comporte davantage d'obstacles. La plupart des séquences ont été acquises dans cette dernière zone. Le plan du terrain est

FIGURE 4.4 – Visuel de l'aire d'acquisition à Sotteville-lès-Rouen

donnée par la Figure (4.6b).

(a) Zone Nord (b) Zone Sud

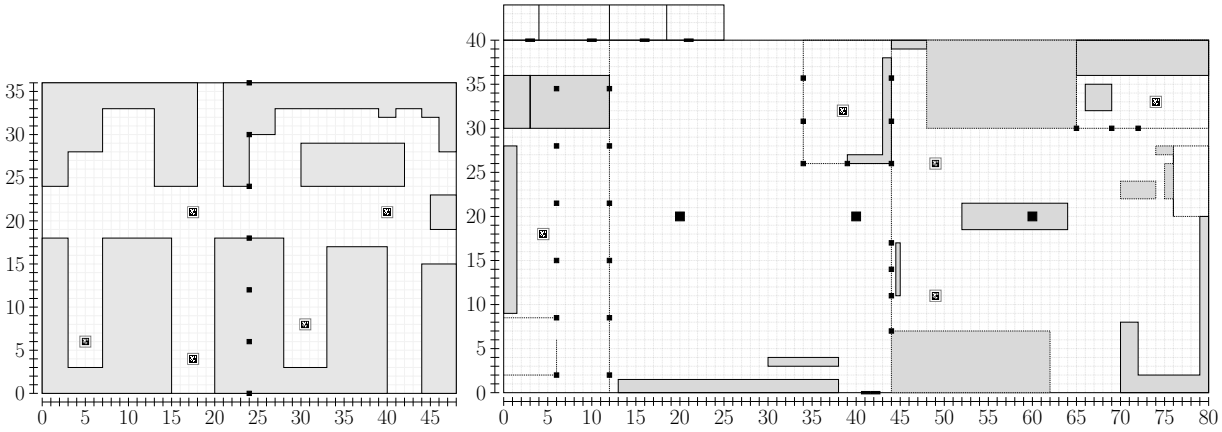
FIGURE 4.5 – Visuels du site de Meudon

4.3.3 Scénarios multi-robots

Dans cette section, nous présentons les principes que nous avons suivis afin de concevoir les acquisitions multi-robots, puis nous détaillons les scénarios acquis à Sotteville et à Meudon.

Principes directeurs et modes d'acquisition

Comme mentionné dans le chapitre 3, le SLAM multi-robots offre de nouvelles opportunités pour accroître la précision des estimations des cartes et des trajectoires. Cependant, il confronte également les robots à de nouvelles contraintes. Il s'agit de contraintes réseau,



(a) Plan Sotteville-lès-Rouen (b) Plan Meudon

FIGURE 4.6 – Plan des aires d'acquisitions à Sotteville-lès-Rouen et à Meudon

de contraintes calculatoires et des contraintes dérivées de la fusion des données de plusieurs robot. La finalité d'un jeu de données multi-robots est d'évaluer les performances d'un algorithme de SLAM quant à saisir ces opportunités et à surmonter ces contraintes. C'est pourquoi il doit idéalement scénariser ces contraintes et ces opportunités. Nous avons vu que la littérature manque cependant de jeux de données multi-robots, et qu'à notre connaissance, UTIAS (LEUNG et al. 2011) est le seul jeu de données expressément conçu pour tester des algorithmes de SLAM multi-robots. Néanmoins, ce dernier couvre une faible surface et se prête davantage à du SLAM 2D basé LiDAR qu'à du SLAM 3D basé vision. Comme nous l'avons mentionné, les jeux de données multi-session tels qu'EuRoC (BURRI et al. 2016) offrent une alternative intéressante, mais l'autonomie de leurs trajectoires, conçues indépendamment les unes des autres, en restreint l'intérêt dans une optique collaborative simultanée.

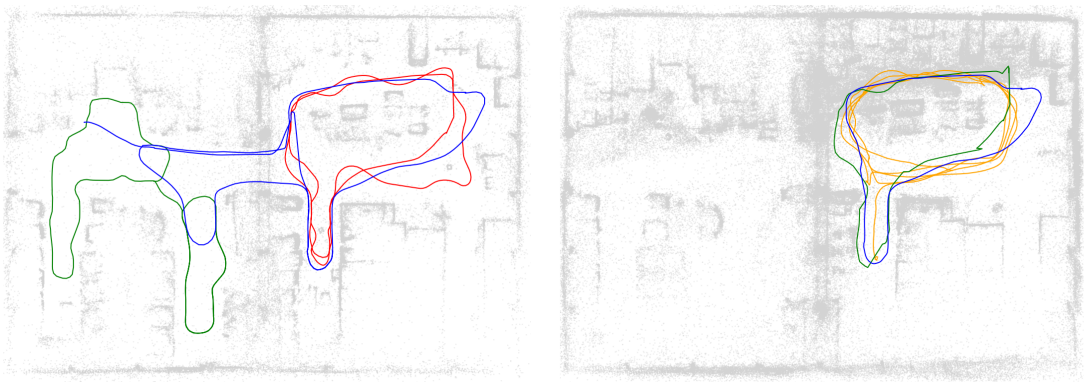
Nous exposons ici les principes directeurs que nous avons suivis afin de concevoir des scénarios multi-robots. Nous avons prêté attention à deux propriétés essentielles pour chaque chaque scénario : i) l'accumulation des incertitudes le long des trajectoires individuelles, et ii) la propagation des informations entre les trajectoires. Vis-à-vis de la première propriété, il s'agit de laisser une marge d'amélioration significative quant à la précision d'estimation des cartes et des trajectoires d'une partie au moins des robots. Dans un cadre visuel, cette incertitude dépend de trois facteurs. Le premier est la cinématique des trajectoires. En effet, des mouvements brusques et aggressifs, voire simplement les mouvements de rotations, compliquent le suivi des points d'intérêts et risquent d'entâcher

les images d'un flou de mouvement. Le second critère tient à la nature des scènes observées : l'odométrie visuelle est d'autant plus fiable qu'elle s'appuie sur une grande quantité d'amers visuels bien contraints (nombre d'observateurs et parallaxe d'observation suffisants). Enfin, le dernier facteur influençant l'accumulation des incertitudes le long de trajectoires est la distribution des fermetures de boucles. Naturellement, plus les boucles fermées sont grandes, plus elles contraignent de larges portions de la trajectoire, et plus la dérive odométrique s'en trouve amoindrie. La seconde propriété essentielle d'un scénario multi-robots est sa propension à la propagation des informations entre les trajectoires des robots. Celle-ci dépend de deux facteurs que sont la distribution des correspondances inter-robot ainsi que de leurs observations directes. Leur intérêt est double. Elles permettent d'une part d'estimer les transformations entre leur référentiels d'odométrie, et d'autre part de fermer des boucles dites inter-robot dès qu'elles induisent des cycles de contraintes impliquant les segments de trajectoires appartenant à plusieurs robots distincts. Notons qu'une fermeture de boucle inter-robot apporte une information dissymétrique. En effet, les contraintes odométriques induites par une trajectoire sur l'autre ne sont pas équivalentes. Le principe général que nous avons suivi a consisté à envisager ces propriétés de façon globale afin de maximiser la complémentarité ou la disparité des informations entre les robots. Concrètement, nous avons privilégié des trajectoires amples et sinueuses promptes à accumuler une incertitude à laquelle la distribution des correspondances inter-robot doit permettre de remédier. Nous avons en particulier construit des scénarios dans lesquels des trajectoires complémentaires et d'incertitudes comparables induisent des contraintes fortes l'une sur l'autre, et des cas disparates où une trajectoire fiable induit de fortes contraintes sur une trajectoire plus incertaine. Les deux jeux de données sont hétérogènes car ils font collaborer des robots terrestres et aériens. Dans le cas visio-inertiel, les séquences terrestres induisent des difficultés supplémentaires puisque les excitations inertielles sont plus faibles, le rapport signal sur bruit est plus faible d'autant plus que la cinématique des trajectoires terrestre est plus aggressive avec des mouvements par à-coups, et le champ de vue des robots est réduit par rapport à celui des drones et donc visuellement moins riche.

Jeu de données acquis à Sotteville-lès-Rouen

Le jeu de données de Sotteville consiste en deux scénarios dont les vérités terrain sont représentées par la Figure 4.7. Le scénario 1 met en jeu trois trajectoires terrestres balayant toute la surface de l'entrepôt. En particulier, on note que les trajectoires des

robots A et B ne se croisent jamais. Ceci implique qu'il n'existe aucune correspondance entre ces robots, qu'elle soit directe ou indirecte. C'est le robot C qui assure le relais entre ces deux robots. De même, leurs trajectoires n'ont pas la même propension aux fermetures de boucles. Le robot A fait deux fois le tour de sa zone et cumule donc un grand nombre de fermetures de boucles très informatives. Au contraire, le robot B ne ferme pas de boucle qui implique la première portion de sa trajectoire ; celle-ci se termine en revanche par une boucle qui ne contraint que la dernière portion de sa trajectoire. Enfin, le robot C est celui dont la trajectoire est la plus ample et alterne entre les zones explorées par les robots A et B. Il peut potentiellement compter sur un grand nombre de correspondances avec le robot A, ainsi qu'avec le robot B mais dans une moindre mesure. En revanche, les fermetures de boucles inter-robot qui découlent de ces correspondances sont potentiellement très informatives car les boucles fermées incluent de longues portions des trajectoires des robots. C'est par exemple le cas pour la boucle entre les robots B et C dérivée des correspondances au début et à la fin de la trajectoire de B.



(a) Trajectoires du scénario 1 (b) Trajectoires du scénario 2

FIGURE 4.7 – Trajectoires de vérité-terrain des robots A (rouge), B (vert), C (bleu) et du drone (orange) dans chaque scénarios.

Le second scénario met en jeu deux robots terrestre et un drone. Les deux robots terrestres parcourent chacun une boucle autour d'un îlot central, mais dans des sens opposés. Au-dessus, le drone effectue également plusieurs boucles autour de l'îlot et alterne les observations directes avec les deux robots terrestres qui eux ne se croisent qu'une seule fois au milieu de leur boucle. L'intérêt de ce scénario est d'utiliser le drone comme relais d'information entre les deux robots terrestres afin d'aligner leur référentiel d'odométrie et d'en dériver des fermetures de boucles inter-robot. Même s'ils passent aux mêmes

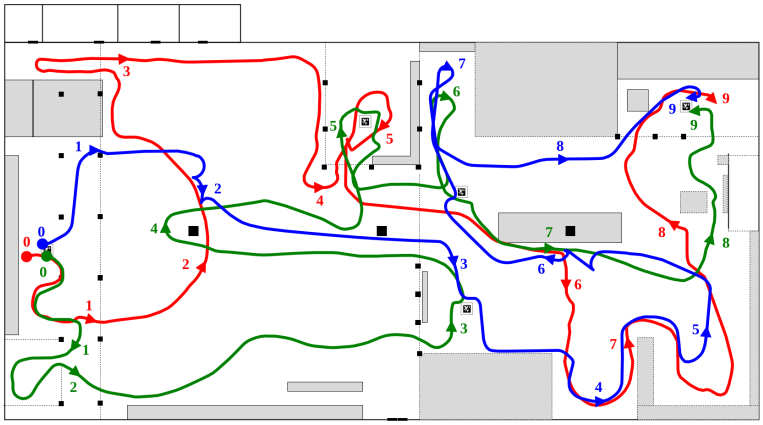
endroits, les deux robots détecteront cependant peu de correspondances indirectes car ils parcourent leur boucle en sens opposés, si bien que leurs observations ne coïncident pas, ou très occasionnellement.

Jeu de données acquis à Meudon

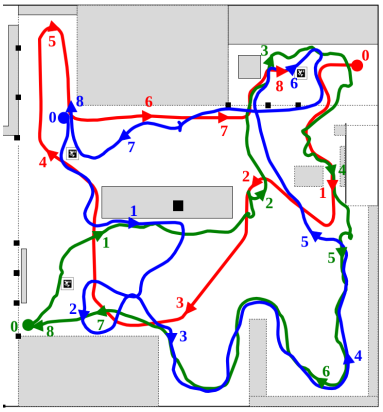
Le jeu de données acquis à Meudon comprend cinq scénarios dont les trajectoires sont représentées sur les Figures 4.8. Le premier scénario (c.f. Figure 4.8a) est un scénario d'exploration. C'est le seul dans lequel les robots couvrent la totalité du hangar. Ceux-ci commencent au même endroit à une extrémité de la zone Nord et la traversent pour rejoindre la zone Sud. La zone Nord, dépourvue d'obstacles sur la majorité de sa surface, impose aux robots de s'appuyer sur des amers lointains à faible parallaxe. Cette traversée s'avère plus difficile pour les robots B et C qui la traversent en son milieu, et ne peuvent alors se reposer que sur des points caractéristiques extraits sur le sol proche et fréquemment renouvelés. De plus, les conditions d'éclairage, couplées à la réflexivité du sol, peuvent compliquer le suivi, notamment pour le robot C. Tout au long de leur trajectoire, les robots se rencontrent régulièrement et peuvent ainsi fermer des boucles de grande ampleur et ainsi amoindrir la dérive accumulée lors de la première phase. Notons cependant qu'aucun robot ne ferme de boucle le long de sa trajectoire, hormis d'éventuelles boucles locales. Ils ne peuvent donc s'appuyer que sur des correspondances inter-robot, et notamment les correspondances indirectes entre les trajectoires. Enfin, les robots terminent au même endroit en s'observant les uns les autres, ce qui permet de fermer de grandes boucles inter-robot incluant l'ensemble de leur trajectoire.

Le scénario 2, représenté sur la Figure 4.8b n'implique également que les robots terrestres. Ceux-ci admettent cependant des points de départ distincts et leurs trajectoires se cantonnent à la zone Sud, comme ce sera le cas pour les scénarios suivants. Mis à part ces différences, ce scénario repose sur les mêmes principes que le premier scénario : de longues trajectoires amples dépourvues de fermetures de boucles intra-robots, mais donnant lieu à de multiples correspondances directes et indirectes entre les robots de façon à en dériver de larges fermetures de boucles contraignant de longues portions de leurs trajectoires. Les robots terminent à leur point de départ initial, sans pour autant fermer de boucle à cette occasion.

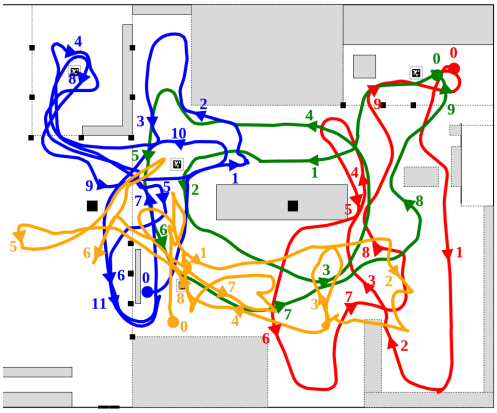
Contrairement aux deux premiers scénarios, les trois suivants incluent également le drone. Dans ces premiers scénarios, il n'existait pas une disparité flagrante d'information



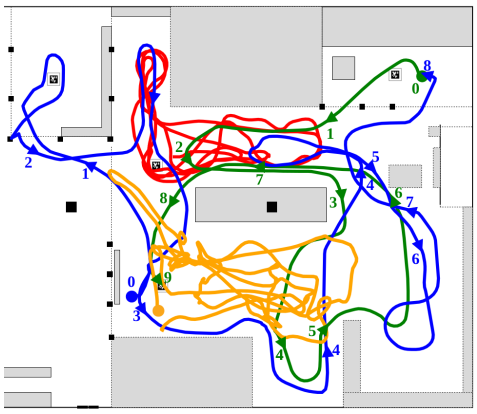
(a) Trajectoires du scénario 1



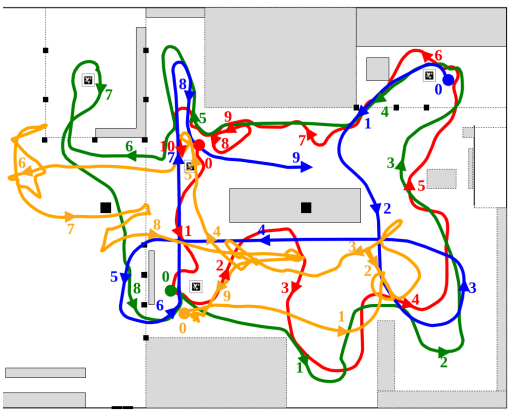
(b) Trajectoires du scénario 2



(c) Trajectoires du scénario 3



(d) Trajectoires du scénario 4



(e) Trajectoires du scénario 5

FIGURE 4.8 – Trajectoires vérité-terrain des robots A (rouge), B (vert), C (bleu) et du drone (orange) dans chaque scénarios.

entre les robots : leurs trajectoires étaient semblables et les correspondances directes et indirectes entre les robots étaient distribuées relativement équitablement entre elles. Les scénarios 3 et 4, respectivement décrits par les Figures 4.8c et 4.8d, explorent au contraire le cas d'une disparité d'information entre les robots. Dans le scénario 3, les robots A et C explorent des zones distinctes et ne se rencontrent jamais, à l'instar du premier scénario de Sotteville. Le robot B ainsi que le drone assurent le relais entre les deux robots en alternant les correspondances directes et indirectes avec l'un et l'autre. La trajectoire du robot C présente jouit de davantage de fermetures de boucles intra-robot que celle du robot A. L'idée de ce scénario est donc d'évaluer s'il est possible de propager l'information de la trajectoire du robot C à celle du robot A par l'intermédiaire de celles du drone et du robot B. Le scénario 4 est conçu de façon similaire. Dans ce cas, ce sont les trajectoires du drone et du robot A – là encore complètement dissociées l'une de l'autre – qui admettent une forte densité de correspondances intra-robot. Le robot A explore cependant une zone restreinte. Les robots B et C explorent eux l'ensemble de la zone Sud. Ils traversent régulièrement la zone du robot A et multiplient les correspondances directes et indirectes avec lui. De plus, les deux robots se rencontrent ocassionnellement en dehors de la zone du robot A, B et C peuvent également bénéficier des correspondances avec le drone afin d'en déduire de nouvelles contraintes inter-robots entre leurs trajectoires.

Dans le dernier scénario (c.f. Figure 4.8e), les robots terrestres n'admettent plus aucune observation directe l'un de l'autre. Ils doivent alors exploiter les seules correspondances indirectes afin de fermer des boucles inter-robot, d'autant plus que leurs trajectoires ne contiennent pas de boucles.

4.4 Traitement des jeux de données

4.4.1 Construction des vérités terrain

Le calcul de la vérité terrain est une étape délicate de la construction d'un jeu de données. Fondamentalement, c'est un problème d'estimation de trajectoires à partir des données capteurs, et dont les estimées sont entachées d'incertitudes. Cependant, à la différence des algorithmes d'estimation en ligne, qui sont les cas d'utilisation du jeu de données, on dispose pour ce faire de ressources complémentaires et additionnelles. Il peut s'agir d'exploiter des données supplémentaires spécifiquement dédiées au calcul de la vérité terrain telles que des signaux GPS, de systèmes de suivi optique, ou encore des détections

de marqueurs visuels ou d'amers balisant l'aire d'acquisition. De plus, on peut recourir à des algorithmes hors-lignes, tels que des algorithmes de reconstruction par le mouvement, capables de fusionner une grande quantité des données en s'exonérant de toute contrainte temporelle stricte. Fournir une vérité terrain fiable est une exigence cruciale, car l'exploitabilité du jeu de données en est tributaire.

Dans le cas de séquences acquises en extérieur, on peut exploiter les signaux GPS afin de reconstituer la trajectoire, comme c'est par exemple le cas pour KITTI (Geiger et al. 2013) et Malaga (Blanco-Claraco et al. 2014). En revanche, les signaux GPS ne sont pas exploitables dans les environnements intérieurs car les parois et les surfaces les atténuent ou les bloquent. Cependant, lors des acquisitions à Meudon, le toit de l'entrepôt s'apparentait à une large verrière relativement perméable aux signaux GPS. Nous avons donc installé une base GPS fixe dans l'espoir que les signaux captés permettent de reconstruire une vérité terrain avec une précision suffisante. Néanmoins, ces signaux se sont malgré tout avérés trop bruités pour y concourir.

En l'absence de signaux GPS exploitables, d'autres stratégies peuvent s'y substituer. La première solution consiste à utiliser des systèmes de suivi visuels. C'est le cas du jeu de données EuRoC (Burri et al. 2016). Dans les séquences Industrial Machine Hall, il utilise un *laser tracker* Leica Nova MS50 [1] qui mesure la position d'un prisme monté sur chaque drone avec une précision millimétrique. Dans les séquences Vicon Room, c'est un système de capture de mouvements Vicon [2] qui estime sa pose absolue à l'aide de marqueurs réflexifs attachés au drone. Ces estimations sont ensuite fusionnées avec les mesures inertielles acquises par les drones afin d'une part d'estimer la transformation relative entre le référentiel attaché au système de suivi et le référentiel inertiel du drone (on parle d'alignement spatial), et d'autre part d'estimer la synchronisation temporelle entre les vérités terrain et l'horloge de l'IMU (on parle d'alignement temporel). Ce problème est formulé à l'aide d'un estimateur de maximum de vraisemblance qui permet également d'estimer une vérité terrain pour les vitesses et les biais inertiels. Ceci permet également de compléter l'estimation de la pose lorsque seule la position est mesurée comme dans le cas du *laser tracker* Leica. En dépit d'une très bonne précision, les systèmes de suivi visuels présentent plusieurs limitations. Outre qu'il s'agit de systèmes généralement onéreux, ils sont d'autant plus efficaces qu'ils observent les robots en permanence. Or rien ne le garantit lorsque la surface d'acquisition est grande et comporte de multiples occlusions. C'était

1. https ://leica-geosystems.com/products/total-stations/multistation/leica- nova-ms60
2. https ://www.vicon.com/hardware/cameras/

le cas à Sotteville et à Meudon. Même dans ces situations, ils apporteraient certes des informations pertinentes pour reconstruire la vérité terrain, bien celle-ci serait parcellaire. Notons à ce titre que dans le jeu de données PennCosyvio (Pfrommer et al. 2017), les auteurs se contentent, en plus des marqueurs AprilTags qui jalonnent la trajectoire, que cette dernière commence et termine dans une salle équipée d'un système de suivi optique pour ensuite estimer la vérité terrain sur la base de la boucle ainsi fermée.

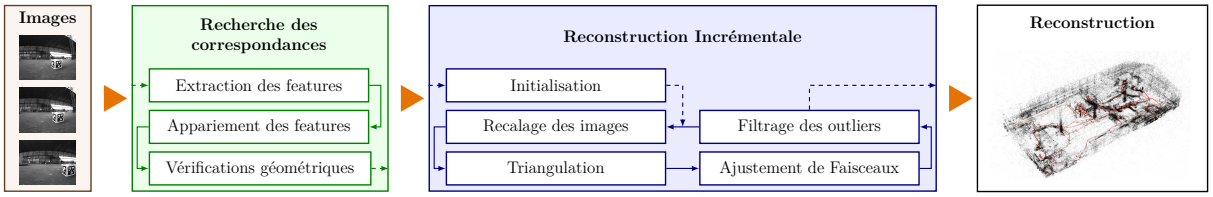


Figure 4.9 – Diagramme de fonctionnement de Colmap (Schonberger et al. 2016)

Pour estimer la vérité terrain des trajectoires à Sotteville et à Meudon, nous avons opté pour une reconstruction en deux étapes. La première phase consiste à reconstruire la trajectoire ainsi que l'environnement observé à un facteur d'échelle près à l'aide d'algorithmes de reconstruction par le mouvement. Dans chaque scénario, nous avons sélectionné une image sur 15, ce qui équivaut à un peu plus d'une image par seconde étant donné que les caméras fonctionnent à 20Hz. Nous avons ensuite utilisé le logiciel Colmap (Schonberger et al. 2016) dont le diagramme de fonctionnement est adapté de l'article en question par la Figure 4.9.

Tout d'abord, les primitives et descripteurs SIFT (Lowe 2004) sont calculés sur toutes les images extraites des trajectoires. Dans un second temps, on recherche des appariements entre toutes les paires d'images (on parle d'appariements exhaustifs). Cette étape permet notamment de détecter un grand nombre de fermetures de boucles au sein des trajectoires mais également entre les trajectoires. Les appariements détectés sont vérifiés à l'aide des outils de la géométrie épipolaire en calculant les matrices fondamentales entre les couples d'images appariées dans une boucle de RANSAC, puis en vérifiant que suffisamment de correspondances point à point corroborent le modèle estimé. Une fois toutes les correspondances détectées, le modèle et la trajectoire sont estimés incrémentalement. Elle est initialisée à l'aide d'une paire d'images extraites d'une zone à fort recouvrement entre les caméras. Puis à chaque itération, de nouvelles images sont recalées en résolvant un problème de perspective à N points, puis de nouveaux amers sont triangulés. On raffine périodiquement l'ensemble en résolvant un problème d'ajustement de faisceaux. Celui-ci

s'exprime ainsi :

$$\{\boldsymbol{T}^*_{C_iW}\}_{i\in\mathcal{C}}, \{{}^W\boldsymbol{l}^*_j\}_{j\in\mathcal{L}}, \boldsymbol{\theta}^* = \arg\min \sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{L}} \rho\left(\left\|\Pi_{\boldsymbol{\theta}}(\boldsymbol{T}_{C_iW}\cdot {}^W\boldsymbol{l}_j) - \boldsymbol{z}_{ij}\right\|_2^2\right) \tag{4.1}$$

où $\mathcal{C}$ et $\mathcal{L}$ désignent respectivement l'ensemble des caméras et des amers, $\mathcal{C}_i$ est le référentiel de la caméra i, $\boldsymbol{z}_{ij}$ est l'observation de l'amer j dans l'image i, et $\Pi_{\boldsymbol{\theta}}$ est le modèle de caméra, paramétrée par les coefficients intrinsèques et de distorsion $\boldsymbol{\theta}$ tel que décrit dans la section (2.2.2). Cette dernière étape permet également de filtrer les amers et les caméras manifestement mal contraintes, et dont l'influence dans le problème précédent est amoindrie par l'utilisation de fonctions robustes ρ telles que la fonction d'Huber (Huber 1992). À la fin de cette première phase, on dispose d'une estimée des trajectoires à l'échelle près, ainsi que d'une reconstruction 3D de l'environnement sous la forme d'un nuage de points. La Figure 4.10 en présente quelques exemples. Les Tableaux 4.1 exposent diverses statistiques sur les reconstructions Colmap pour les scénarios retenus. Elles permettent d'en évaluer la qualité. En particulier, on remarque que l'erreur de reprojection moyenne des amers dans les images des caméras avoisine un demi-pixel, ce qui suggère que les reconstructions obtenues sont fiables.

Scenario	#1	#2
Nombre d'images	1279	2386
Nombre d'amers 3D	181768	379127
Longueur de suivi moyenne	10.87	17.70
Erreur de reprojection moyenne (pixels)	0.531	0.626

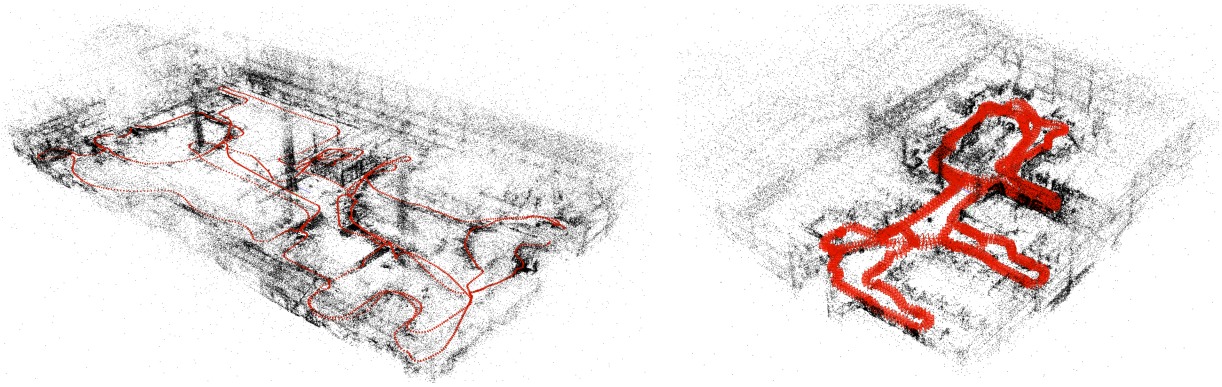
(a) Scénarios de Sotteville

Scenario	#1	#2	#3	#4	#5
Nombre d'images	2478	2330	2442	2534	2448
Nombre d'amers 3D	295284	474593	323766	314173	309865
Longueur de suivi moyenne	8.830	9.706	10.555	12.217	11.528
Erreur de reprojection moyenne (pixels)	0.573	0.520	0.549	0.604	0.577

(b) Scénarios de Meudon

Table 4.1 – Statistiques sur les reconstructions Colmap

La seconde étape vise à mettre la reconstruction à l'échelle. Pour cela, plusieurs méthodes sont possibles. La première, inspirée des procédures d'initialisation des algorithmes



(a) Meudon – Scénario 1 (b) Sotteville – Scénario 1

FIGURE 4.10 – Exemple de reconstructions Colmap

de SLAM visio-inertiels (voir section 2.3.2), consisterait à aligner la reconstruction sur les mesures de la centrale inertielle. Ces méthodes procèdent en plusieurs étapes, estimant successivemenent les biais du gyromètre, puis le vecteur gravité et le facteur d'échelle, puis raffinent enfin l'ensemble par l'estimation du biais de l'accéléromètre. La méthode que nous utilisons ne dépend pas des calibrations extrinsèques IMU-caméras ni des mesures inertielles. Elle repose sur l'observation des marqueurs AprilTags (OLSON 2011) montés en cubes balisant la zone d'acquisition. L'observation $\boldsymbol{z}_{\mathrm{CT}}$ d'un marqueur T par une caméra C permet d'une part d'identifier le marqueur observé, et d'autre part d'estimer une pose relative $\hat{\boldsymbol{T}}_{CT} \in \mathbb{SE}_3$ entre le référentiel de la caméra et le référentiel attaché au marqueur. Ainsi, le problème de mise à l'échelle peut se formuler comme un problème d'optimisation dont les variables Θ incluent le facteur d'échelle s, ainsi que la pose globale $\boldsymbol{T}_{WT}$ de chaque marqueur T observé, la morphologie des trajectoire étant fixée :

$$\Theta^* = \arg\min_{\Theta} \mathcal{C}_1(\Theta) = \arg\min_{\Theta} \left\{ \sum_{z \in \mathcal{Z}} \rho\left(\|\boldsymbol{\xi}_R^z(\Theta)\|^2 + \|\boldsymbol{\xi}_t^z(\Theta)\|^2 \right) \right\} \tag{4.2}$$

$$\text{avec} \begin{cases} \boldsymbol{\xi}_R^z \triangleq \log_{\mathbb{SO}_3}\left(\boldsymbol{R}_{\mathrm{WC}_z}^\top \cdot \boldsymbol{R}_{\mathrm{WT}_z} \cdot \hat{\boldsymbol{R}}_{\mathrm{C}_z\mathrm{T}_z}^\top \right)^\vee \\ \boldsymbol{\xi}_t^z \triangleq s \cdot \boldsymbol{R}_{\mathrm{WC}_z}^\top \left({}^W\boldsymbol{t}_{\mathrm{WT}_z} - {}^{\mathrm{W}}\boldsymbol{t}_{\mathrm{WC}_z} \right) - {}^{\mathrm{C}_z}\hat{\boldsymbol{t}}_{\mathrm{C}_z\mathrm{T}_z} \end{cases}$$

De plus, on impose les contraintes relatives entre les marqueurs appartenant à un même cube à l'aide de termes d'erreurs de pose relative associés à de fortes pondérations :

$$\mathcal{C}_2(\Theta) = \sum_{\mathrm{Cube}_k} \sum_{T_i, T_j \in \mathrm{Cube}_k} \left\| \left(\boldsymbol{T}_{WT_i}^{-1} \oplus \boldsymbol{T}_{WT_j} \right) \boxminus \hat{\boldsymbol{T}}_{T_iT_j} \right\|^2 \tag{4.3}$$

Ainsi, le problème d'optimisation final est de la forme :

$$\Theta^* = \arg\min_{\Theta} \ [\mathcal{C}_1(\Theta) + \omega \cdot \mathcal{C}_2(\Theta)] \tag{4.4}$$

où ω est un coefficient de pondération auquel on assigne une valeur très élevée afin de contraindre les poses des marqueurs appartenant à un même cube. Une fois le facteur d'échelle s estimé, toutes les distances estimées par Colmap sont mises à l'échelle. Afin de limiter la complexité du problème de reconstruction, toutes les images n'ont pas été intégrées dans l'estimation de Colmap. Afin de récupérer des valeurs de vérité terrain à tous les instants, on procède par interpolation à l'aide de splines cubiques pour les positions et d'interpolations linéaires sphériques pour les quaternions (Shoemake 1985). Enfin, les trajectoires obtenues portant sur les poses des caméras, on utilise alors l'estimation de la transformation extrinsèque fournie par les calibrations pour en dériver la vérité terrain dans le référentiel inertiel.

4.4.2 Comparatif de performances d'algorithmes de SLAM

La dernière étape a consisté à tester différents algorithmes de SLAM sur les séquences afin d'une part de s'assurer de leur exploitabilité, et d'autre part de fournir des performances mono-robot de référence pour évaluer les apports des algorithmes multi-robots sur l'estimation des trajectoires. Nous avons testé trois algorithmes de SLAM issus de paradigmes différents. Le premier d'entre eux est ORB-SLAM (Mur-Artal et al. 2017a) dans sa version purement monoculaire. Pour le SLAM visio-inertiel, nous avons testé l'algorithme Vins-Mono (Qin et al. 2018b). Enfin, nous avons utilisé l'algorithme Vins-Fusion (Qin et al. 2019) pour le SLAM stéréo-visuel.

Les Tableaux 4.2 et 4.3 fournissent, pour chaque robot et chaque scénario, les métriques absolues et relatives des erreurs en translation. Pour les métriques relatives, on considère les sous-trajectoires de longueurs 5 et 10 mètres. Dans ces tableaux, e_{traj} est l'erreur absolue en translation (ATE) calculée sur toute la trajectoire, tandis que e_{xm} est l'erreur de translation relative (RTE) moyennée sur toutes les sous-trajectoires de longueur x mètres. L'erreur d'échelle est fournie dans les cas visio-inertiel et stéréo-visuel pour lesquels il est observable.

Comme attendu, l'algorithme de SLAM stéréo-visuel démontre de meilleures performances d'estimation, suivi par le SLAM visio-inertiel, et enfin le SLAM monoculaire. Dans

	Erreur en translation [m]				Erreur d'échelle			
Robot	A	B	C	D	A	B	C	D
Scénario 1								
ORB-SLAM	1.425	1.393	2.616	—	—	—	—	—
Vins-Mono	0.240	0.320	0.620	—	0.028	0.066	0.007	—
Vins-Fusion	0.157	0.204	0.437	—	0.004	0.010	0.006	—
Scénario 2								
ORB-SLAM	1.270	0.341	—	0.591	—	—	—	—
Vins-Mono	0.990	0.290	—	0.130	0.072	004	—	0.092
Vins-Fusion	0.907	0.221	—	0.067	0.066	0.036	—	0.036

(a) Erreurs Absolues en Translation

	Erreur relative 5 m				Erreur relative 10 m			
Robot	A	B	C	D	A	B	C	D
Scénario 1								
ORB-SLAM	2.097	1.16	1.513	—	2.273	1.555	2.261	—
Vins-Mono	0.292	0.478	0.464	—	0.473	0.865	0.695	—
Vins-Fusion	0.168	0.239	0.143	—	0.391	0.465	0.313	—
Scénario 2								
ORB-SLAM	0.489	0.338	—	1.695	1.025	0.432	—	2.021
Vins-Mono	0.946	0.276	—	0.116	1.466	0.487	—	0.222
Vins-Fusion	0.924	0.221	—	0.097	1.388	0.366	—	0.153

(b) Erreurs Relatives en Translation

TABLE 4.2 – Benchmark des algorithmes Vins-Mono, ORB-SLAM en version monoculaire et Vins-Fusion en version stéréo sur les scénarios du jeu de données de Sotteville.

	Erreur en translation [m]				Erreur d'échelle			
Robot	A	B	C	D	A	B	C	D
Scénario 1								
ORB-SLAM	1.043	1.512	2.135	—	—	—	—	—
Vins-Mono	2.617	2.382	2.314	—	0.091	0.050	0.086	—
Vins-Fusion	0.970	1.244	0.704	—	0.095	0.071	0.067	—
Scénario 2								
ORB-SLAM	0.760	1.851	0.516	—	—	—	—	—
Vins-Mono	1.510	1.998	2.305	—	0.215	0.065	0.059	—
Vins-Fusion	0.354	0.867	0.299	—	0.016	0.033	0.014	—
Scénario 3								
ORB-SLAM	0.682	0.096	0.417	0.163	—	—	—	—
Vins-Mono	3.743	1.306	0.969	0.439	0.059	0.055	0.059	0.015
Vins-Fusion	0.945	0.508	0.144	0.259	0.001	0.011	0.004	0.001
Scénario 4								
ORB-SLAM	0.320	0.967	1.323	0.051	—	—	—	—
Vins-Mono	0.761	1.476	2.482	0.147	0.126	0.209	0.024	0.017
Vins-Fusion	0.164	0.260	0.472	0.122	0.001	0.004	0.014	0.004
Scénario 5								
ORB-SLAM	0.086	0.093	2.405	0.062	—	—	—	—
Vins-Mono	3.338	1.440	1.422	0.254	0.065	0.083	0.132	0.008
Vins-Fusion	0.175	0.360	0.124	0.559	0.002	0.019	0.005	0.026

(a) Erreurs Absolues en Translation

	Erreur relative 5 m				Erreur relative 10 m			
Robot	A	B	C	D	A	B	C	D
Scénario 1								
ORB-SLAM	1.630	2.872	2.537	—	1.458	2.245	2.668	—
Vins-Mono	0.851	0.881	1.278	—	1.680	1.422	2.119	—
Vins-Fusion	0.151	0.162	0.147	—	0.245	0.342	0.718	—
Scénario 2								
ORB-SLAM	0.839	1.748	0.634	—	1.239	1.381	0.446	—
Vins-Mono	0.907	0.934	0.731	—	1.862	1.614	1.712	—
Vins-Fusion	0.146	0.206	0.165	—	0.246	0.419	0.266	—
Scénario 3								
ORB-SLAM	1.266	0.303	0.700	3.183	1.108	0.313	0.709	3.401
Vins-Mono	0.686	0.609	0.791	0.149	1.389	1.056	1.108	0.227
Vins-Fusion	0.292	0.181	0.153	0.127	0.512	0.323	0.254	0.183
Scénario 4								
ORB-SLAM	0.510	1.034	1.778	2.539	0.630	0.831	1.575	3.360
Vins-Mono	0.831	0.778	1.049	0.126	1.363	1.059	1.584	0.173
Vins-Fusion	0.185	0.107	0.169	0.125	0.268	0.195	0.295	0.173
Scénario 5								
ORB-SLAM	0.708	0.593	1.903	4.210	0.335	0.515	1.966	2.966
Vins-Mono	1.065	0.765	0.474	0.134	1.703	1.241	0.794	0.207
Vins-Fusion	0.133	0.145	0.085	0.158	0.231	0.270	0.139	0.242

(b) Erreurs Relatives en Translation

TABLE 4.3 – Comparatif des algorithmes Vins-Mono, ORB-SLAM en version monoculaire et Vins-Fusion en version stéréo-visuel sur les scénarios du jeu de données AirMuseum.

le cas monoculaire, les erreurs les plus importantes surviennent lors des mouvements de rotations, propices aux pertes de suivi visuel que les mesures inertielles compensent dans le cas visio-inertiel. En particulier, en l'absence de fermetures de boucles, on observe des dérives d'échelles avec ORB-SLAM, notamment à la suite de mouvements de rotation qui ont détérioré le suivi visuel. Un second point concerne l'écart entre les performances des drones et celles robots terrestres, les premiers surpassant les seconds. Celle-ci s'explique par plusieurs facteurs. Tout d'abord, les drones ferment davantage de boucles que les robots terrestres. En effet, profitant d'une dynamique plus véloce, les drones balayent une plus grande surface que les robots terrestres, bénéficient de la richesse visuelle offerte par leur point de vue pour fermer davantage de boucles. D'autre part, dans le cas visio-

inertiel, les drones bénéficient de plus amples excitations de leur centrale inertielle, tandis que les robots terrestres, sont confrontés à un rapport signal sur bruit plus défavorable. Enfin, un désavantage majeur des robots terrestres par rapport aux drones est leur point de vue restreint. En particulier, sur les séquences acquises à Sotteville et à Meudon, on remarque que le suivi visuel des robots terrestres porte avant tout sur des features extraits du sol proche (et donc suivis sur de faibles distances) lorsque les robots naviguent loin des obstacles et des parois.

Les performances des algorithmes testés suggèrent qu'il existe une marge d'amélioration significative pour des algorithmes mono-robot, en particulier dans les scénarios acquis à Meudon. Dans les scénarios 1 et 2, les robots montrent des erreurs relativement uniformes et élevées. Elles témoignent notamment qu'il s'agit de scénarios inviduellement difficiles pour l'estimation visio-inertielle. Celles-ci sont dues à la cinématique des trajectoires et à l'absence de fermetures de boucles en leur long, ainsi qu'à certaines difficultés liées à des artefacts lumineux. Les disparités entre les robots apparaissent manifestement dans les scénarios 3 et 4. On remarquera notamment l'écart de précision entre les robots A et C dans ces deux scénarios, et en particulier dans le scénario 4 où le robot A explore une zone restreinte et ferme une très grande quantité de boucles. Les marges d'amélioration sont en revanche plus limitées sur les séquences de Sotteville.

4.5 Conclusion

Ce chapitre a restitué la démarche suivie pour concevoir des jeux de données multi-robots explicitement dédiés à l'évaluation d'algorithme de SLAM collaboratifs. En effet, nous avons constaté un déficit de tels jeux de données dans la littérature en dépit de sa richesse quant aux données mono-robot et l'émergence récente des jeux de données multi-session. Suivant des principes de conception que nous avons exposés, nous avons élaboré des scénarios multi-robots que nous avons acquis à travers deux campagnes d'acquisition. Ces scénarios impliquent des plateformes robotiques hétérogènes avec des robots terrestres et un drone. En particulier, les scénarios acquis à Meudon ont été rendus publics (Dubois et al. 2020a). En guise de perspectives, nous pourrions envisager de diversifier les environnements d'acquisition des scénarios multi-robots en conservant les mêmes principes directeurs : scènes d'extérieurs dans des environnements moins structurés et présentant une richesse visuelle accrue.

Chapitre 5

SLAM visio-inertiel décentralisé pour la navigation multi-robots

5.1 Introduction

Les approches collaboratives permettent aux robots de partager et de fusionner les informations qu'ils acquièrent afin de collectivement gagner en efficacité dans l'accomplissement des tâches qui leur sont assignées. Nous avons vu dans le chapitre 3 que le degré d'intégration entre les robots est un élément structurant des architectures collaboratives : ils délèguent les tâches les plus complexes au serveur dans les méthodes centralisées (Forster et al. 2013 ; Schmuck et al. 2017 ; Karrer et al. 2018), tandis qu'ils suppléent éventuellement aux limitations de leurs ressources en les mutualisant à l'aide d'algorithmes distribués pour le stockage des cartes (Cieslewski et al. 2015 ; Quraishi et al. 2016), la détection des correspondances (Giamou et al. 2018 ; Cieslewski et al. 2017) ou l'inférence multi-robots (Choudhary et al. 2017 ; Lajoie et al. 2020) dans le cas décentralisé.

Un haut degré d'intégration compromet néanmoins l'autonomie des agents, et par conséquent la flexibilité de la flotte. Ceci impacte en particulier sa résilience à la perte d'un agent, et aux défaillances éventuelles du serveur central dans le cas centralisé. Une approche intermédiaire consiste à laisser à chaque robot la responsabilité de ses modules de SLAM pour estimer sa trajectoire et la carte de l'environnement, tout en tirant parti des informations reçues des autres robots, même dans le cas où ils doivent temporairement opérer seul ou sans accès à un serveur central. Se pose alors la problématique suivante : quelles informations échanger entre les robots pour apporter suffisamment d'informations sur l'environnement afin d'une part de détecter des correspondances inter-robot entre les trajectoires et d'autre part de ré-estimer les cartes et les trajectoires, sans compromettre leur autonomie ni la qualité de leurs estimées ? Notons que la qualité des estimées découle

de la précision d'estimation et de la consistance de l'estimation i.e. la caractérisation correcte de l'incertitude. Ainsi, nous souhaitons que les informations transmises permettent aux robots qui les reçoivent d'obtenir des cartes de bonne qualité et en particulier d'améliorer la précision sur leur trajectoire dès lors qu'ils visitent des zones communes.

Afin de maximiser l'autonomie des robots, nous devons donc privilégier des architectures décentralisées, et qui restreignent la distribution éventuelle des tâches à des opérations opportunistes et non critiques. Les architectures purement décentralisées engendrent cependant trois problématiques. Tout d'abord, elles sont davantage sujettes aux contraintes réseau car, reposant sur des échanges point-à-point, elles consomment plus de bande-passante. C'est pourquoi nous devons veiller à minimiser la quantité d'informations transmises entre les robots. La seconde problématique dérive des contraintes d'embarquabilité : étant complètement responsable de l'intégration des données reçues des autres agents, chaque robot voit sa charge de calcul s'accroître. Afin de ne pas rogner sur les ressources de calcul dévolues aux autres processus et respecter les contraintes de temps-réel, nous devons limiter la complexité de l'intégration des informations transmises. Enfin, la troisième problématique est celle de la consistance : non seulement les architectures décentralisées y sont davantage sensibles du fait de communications cycliques, mais la nature des informations transmises joue également un rôle dans la préservation de la consistance dès lors qu'on échange des représentations autres que des mesures brutes. La prise en compte de ces contraintes complète la problématique énoncée précédemment : comment restituer les observations et les estimations de la façon la plus exhaustive, la plus compacte, la plus consistante et la moins complexe possible en vue de leur intégration ?

Ces problématiques ont été partiellement abordées pour certains types de SLAM dans des contextes spécifiques. Comme nous l'avons vu dans la section 3.2.2, deux approches principales ont émergé dans la littérature. La première consiste à transmettre des représentations exhaustives. C'est notamment le cas de l'approche proposée par (Schuster et al. 2015) pour le SLAM stéréo-visuel dense décentralisé, dans une optique axée sur la reconstruction de l'environnement via l'échange de sous-cartes stéréo-visuelles. L'approche opposée consiste à condenser l'information. C'est notamment le cas pour des méthodes de SLAM LiDAR (Lazaro et al. 2013), étant donné que chaque scan de l'environnement est une information lourde et complexe à intégrer, du SLAM visuel (Cunningham et al. 2013), ou encore pour du SLAM sous-marin acoustique (Paull et al. 2015), soumis à des restrictions particulièrement fortes en bande-passante. Dans la suite, nous nous intéresserons au SLAM visio-inertiel pour lequel peu de méthodes collaboratives ont été

proposées, malgré sa popularité pour la navigation des drones et l'estimation des trajectoires à 6 degrés de liberté. On compte en particulier des publications à destination du SLAM visio-inertiel multi-session (Lynen et al. 2015) qui expose des techniques d'élagage et de compression de données sans pour autant les inscrire dans une architecture de SLAM collaboratif. De même, les méthodes de condensation que nous évoquions ci-dessus ne sont pas adaptées aux spécificités des méthodes visio-inertielles qui représentent l'environnement à l'aide de nuages éparses d'amers. En effet, (Schuster et al. 2015) exploite la forte densité des nuages de points stéréo-visuels pour établir des correspondances inter-robots, tandis que (Cunningham et al. 2013), (Lazaro et al. 2013) et (Paull et al. 2015) s'appuient sur l'identification de rares amers globaux communément observés. Néanmoins, ces méthodes introduisent différentes approches de partage de données ; dans ce chapitre, nous proposons d'adapter ces approches au cas du SLAM visio-inertiel et de construire à partir d'elles des méthodes de SLAM visio-inertiel collaboratives et décentralisées.

5.2 Objectifs du chapitre

Dans ce chapitre, nous comparons trois approches de partage de données que nous adaptons au cas du SLAM visio-inertiel, et à partir desquelles nous construisons des méthodes de SLAM visio-inertielles collaboratives décentralisées. L'objectif primordial de ces échanges est de permettre aux robots d'exploiter les apports du cadre collaboratif tout en maximisant leur autonomie, et en préservant la qualité de leurs estimations tout en leur donnant l'opportunité de l'accroître. Nous investiguerons tout d'abord une méthode basée sur l'échange de sous-cartes visio-inertielles rigides, inspirée de la méthode proposée par (Schuster et al. 2015), une méthode exploitant des techniques de condensation par marginalisation et éparsification consistantes et inspirée des travaux de (Paull et al. 2015), et enfin, des méthodes basées sur l'échange de paquets visio-inertiels élagués empruntant à des techniques de sélection d'amers visuels.

Dans ce chapitre, nous supposerons que les robots disposent de modules *Front-End* qui produisent un graphe de facteurs visio-inertiel servant de support à l'inférence globale. Les méthodes proposées seront donc agnostiques à l'estimateur visio-inertiel sous-jacent. Nous supposerons également que les transformations entre les référentiels d'odométrie des robots sont initialement inconnues et doivent être estimées.

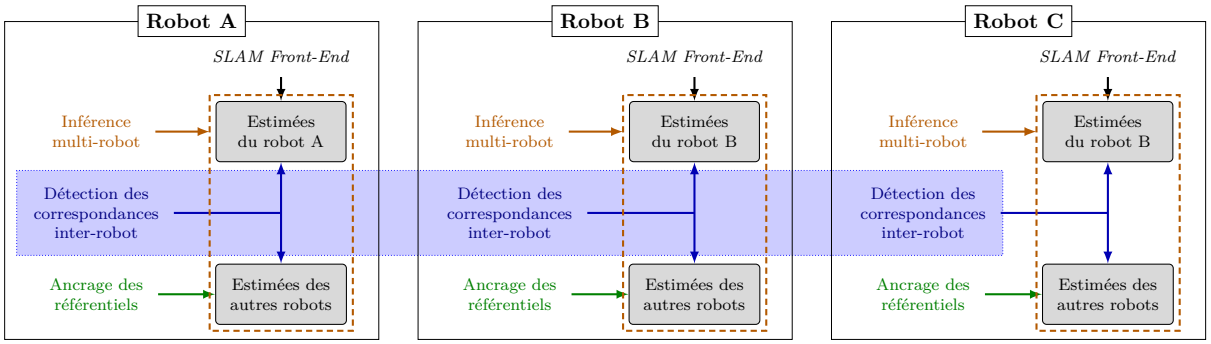
Le chapitre s'organise comme suit. Tout d'abord, nous présenterons l'architecture dé-

centralisée commune aux méthodes proposées dans la section 5.3. Les sections 5.4, 5.5 et 5.6 détailleront les différentes méthodes d'échange de données envisagées. Enfin, une évaluation expérimentale de ces méthodes est présentée dans la section 5.7 sur des scénarios multi-robots construits à l'aide des jeu de données EuRoC (Burri et al. 2016) et AirMuseum (Dubois et al. 2020a).

5.3 Architecture générale des méthodes proposées

Dans cette section, nous détaillons et motivons l'architecture commune que nous avons retenue afin d'organiser le calcul, l'échange et l'intégration des paquets de données entre les robots. Nous présentons ici le schéma d'allocation des tâches et des données basé sur une architecture décentralisée ; la politique de communication bi-régimes gérant les communications régulières à portée de communication et la régularisation des connaissances lors de recouvrement de contacts ; et enfin la stratégie d'association et de fusion de données inter-robots, dont les inférences multi-robots exploitent des modèles probabilistes spécifiques liés aux représentations échangées.

5.3.1 Schéma d'allocation des tâches et des données



Chaque robot dispose de son propre module *Front-End* pour construire sa carte et détecter les fermetures de boucles au sein de sa trajectoire. Il gère également les opérations d'inférence multi-robots et d'ancrage des référentiels. En revanche, il ne détecte que les correspondances inter-robot entre les observations associées aux trajectoires qu'il reçoit et les siennes (qu'il partagera ensuite avec les autres robots).

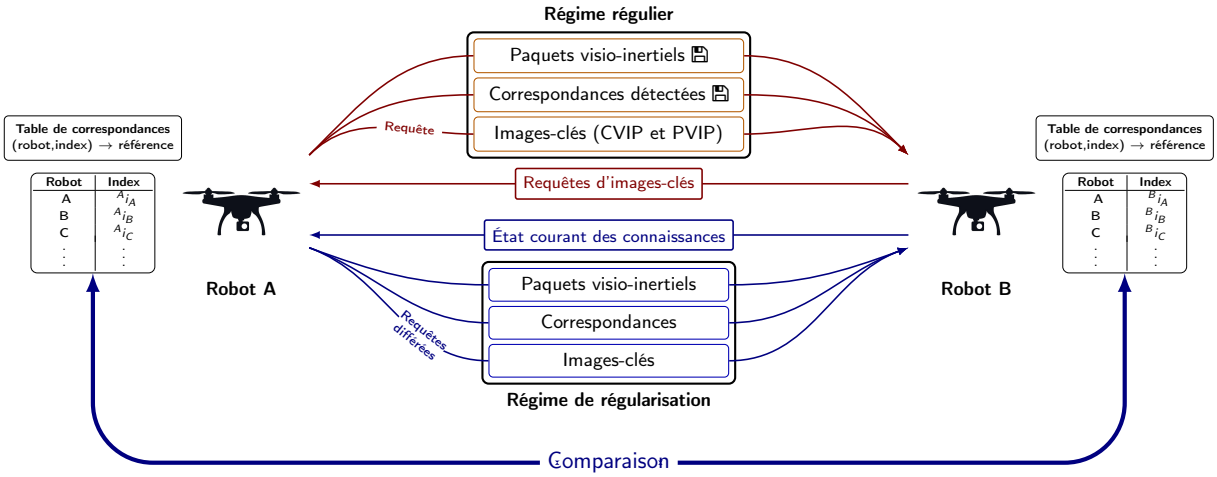
Figure 5.1 – Schéma d'allocation des tâches et des données

Dans la suite, nous nous restreignons à des schémas d'allocation complètement décentralisés, illustrés par la Figure 5.1. Chaque robot est responsable de son propre algorithme

de SLAM et construit sa propre instance de carte dans laquelle il intègre, associe et fusionne les données qu'il reçoit des autres robots. En particulier, il dispose de ses propres modules afin de détecter les correspondances inter-robot et de procéder aux inférences sur l'ensemble des connaissances reçues. La détection des correspondances inter-robot est implicitement distribuée entre les robots : ceux-ci ne recherchent que les correspondances entre les trajectoires qu'ils reçoivent et leur propre trajectoires, et que les correspondances détectées sont partagées aux autres robots.

5.3.2 Politique de communication commune

La politique de communication recouvre la topologie des échanges, la nature des informations échangées, ainsi que le déroulement processus d'échange lui-même. Conformément à l'allocation adoptée ci-dessus, nous considérons une topologie décentralisée où les échanges paire-à-paire prévalent. Concernant le processus d'échange, nous imposons qu'il puisse s'accomoder des pertes et recouvrements éventuels de contact entre les robots, ce qui contribue à renforcer leur autonomie en relaxant la contrainte qui les oblige à demeurer à portée de communication les uns des autres. C'est pourquoi nous proposons



La politique de communication comprend un régime régulier dans lequel le robot A envoie des paquets visio-inertiels résumant des portions de sa carte (envoi déclenché selon un critère de distance parcourue), les correspondances qu'il détecte et des informations visuelles sur les images-clés requises. Lorsque deux robots recouvrent le contact, la régularisation est orientée au préalable par l'échange et l'analyse comparée de l'*état courant de leurs connaissances*.

FIGURE 5.2 – Vue d'ensemble de la politique de communication proposée

une politique de communication scindée en deux régimes : un régime régulier qui gère

les échanges lorsque les deux robots évoluent à portée de communication, et un régime de régularisation, basée sur le maintien d'un historique de communication, qui prend le relais lorsque deux robots recouvrent le contact après l'avoir perdu.Ces deux régimes sont décrits dans la section 5.3.2. Le dernier volet de la politique de communication concerne la nature des données échangées. Les politiques de communications proposées sont basées sur l'échange de trois types de données. Tout d'abord, les robots échangent régulièrement des paquets de données visio-inertielles résumant des portions locales successives de leurs cartes. C'est sur ce point que diffèrent radicalement les méthodes proposées dans ce chapitre. En complément, comme précisé dans le paragraphe précédent, les robots communiquent également les fermetures de boucles qu'ils détectent au sein de leur propre trajectoire, ainsi que les correspondances inter-robot qu'ils décèlent entre les trajectoires des autres agents et la leur. L'échange de ces correspondances est crucial car ce sont elles qui engendrent l'information mutuelle entre les trajectoires, susceptible d'améliorer les précisions d'estimations de celles-ci. Enfin, lorsque le paradigme s'y prête, nous implémentons une architecture client-serveur afin de permettre aux robots de requérir davantage d'informations visuelles. Le but est d'étoffer localement les informations visuelles communiquées de façon standard lorsque des correspondances sont détectées à partir d'elles.

Politique de communication régulière

La politique de communication régulière supervise les échanges de données de trois natures différentes : i) les paquets calculés le long des trajectoires, ii) les correspondances intra et inter-robot détectées et iii) les informations visuelles additionnelles sur les images. Elle est schématisée dans son ensemble par la Figure 5.2.

La synthèse et l'échange de paquets de données visio-inertielles est la principale manière de communiquer des informations entre les robots. Afin de prévenir les problèmes de consanguinité des données (c.f. section 3.1.1), nous imposons que les robots échangent des paquets calculés sur des portions successives et indépendantes de leur trajectoires, calculés à partir des seules informations locales à la portion considérée. Pour chaque robot, nous déclenchons le calcul d'un paquet dès lorsqu'il estime avoir parcouru une distance $d \geq d_{\min}$ depuis qu'il a calculé le paquet précédent, où $d_{\min}$ est un paramètre de distance fixé par l'utilisateur. Ce simple critère permet de circonscrire des paquets de tailles relativement uniformes et induit donc une complexité de calcul approximativement constante. On extrait et copie alors le sous-graphe de facteurs correspondant à la sous-trajectoire ainsi délimitée, ainsi que les informations visuelles associées. C'est sur la base de ce sous-

graphe de facteurs et des informations visuelles associées que le robot calcule le paquet qu'il diffuse aux autres robots, selon des modalités spécifiques à chacune des méthodes et que nous présenterons ultérieurement. Une constante dans la conception des paquets est de découpler les informations dédiées à la localisation et les informations visuelles dédiées à la détection des correspondances inter-robot.

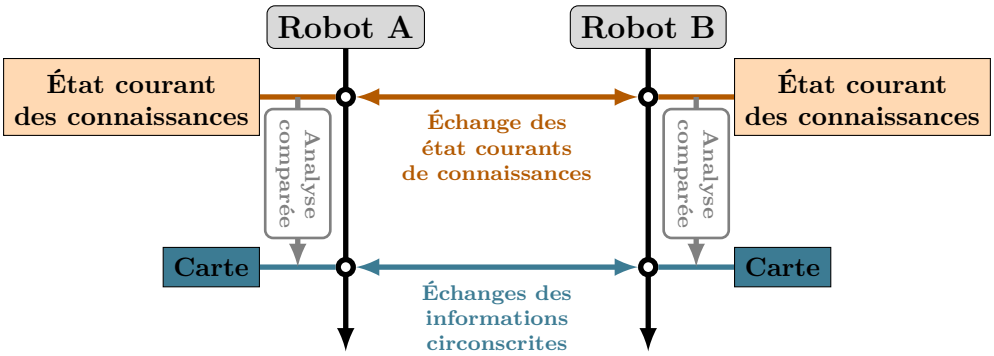
Le second volet de la politique de communication régulière est la communication des correspondances détectées. Dès qu'une fermeture de boucle est identifiée au sein de la trajectoire ou qu'une correspondance inter-robot est repérée avec une trajectoire reçue d'un autre robot, celle-ci est aussitôt diffusée aux autres robots à portée de communication. Chaque correspondance est caractérisée par la date de sa détection, les identifiants des deux robots impliqués, les dates des images-clés mises en correspondance de part et d'autre, puis enfin par la pose relative ainsi que la matrice de covariance ainsi dérivées. Les correspondances inter-robot détectées sont de deux natures : les correspondances directes dérivent de l'observation directe d'un robot telle que décrite dans la section 5.3.3, tandis que les correspondances indirectes dérivent de l'observation d'une même scène par les deux robots. L'intégration correcte d'une correspondance directe requiert une bonne synchronisation temporelle entre les horloges des robots. Nous le supposerons, bien qu'en pratique, les horloges des robots dérivent potentiellement les unes par rapport aux autres.

Enfin, le troisième volet concerne les requêtes d'informations visuelles complémentaires afin d'étoffer localement la détection de correspondances inter-robot. En effet, lorsqu'un robot détecte une correspondance inter-robot à partir d'informations visuelles communiquées de façon standard (i.e. dans un paquet visio-inertiel) sur une image-clé, il est très probable qu'il puisse détecter d'autres correspondances inter-robot à partir des images-clés voisines. Nous proposons dès lors le processus suivant. Lorsqu'une correspondance inter-robot est détectée à partir d'une image-clé reçue de façon standard, le robot vérifie s'il possède les informations visuelles associées au n_{requete} images-clés voisines antérieures et postérieures, et inclut dans sa requêtes les dates des images-clés pour lesquelles ce n'est pas le cas, n_{requete} étant fixé préalablement. Afin d'éviter les requêtes redondantes, le robot mémorise ses requêtes courantes. Recevant cette requête, le robot interlocuteur renvoie pour chaque image-clé demandée un sous-ensemble de ses points caractéristiques et de leurs descripteurs sélectionnés selon la méthode décrite dans la section 5.3.4. [1]

1. Cette architecture client-serveur ne concernera que les méthodes basées sur l'échange de paquets condensés et élagués, et non la méthode reposant sur l'échange de sous-cartes visio-inertielles rigides car celles-ci communiquent l'information visuelle de façon radicalement différente aux deux premières

Politique de régularisation lors des recouvrement de contact

Afin de favoriser la flexibilité de la flotte, il convient de s'accomoder des pertes de contact potentielles, et de prévoir des procédures de régularisation des connaissances lorsque deux robots recouvrent le contact. C'est l'objet du second régime de la politique de communication que nous proposons. Il repose sur le maintien, l'échange et l'analyse d'un vecteur représentant l'état courant des connaissances de chaque robot.



Lorsque deux robots recouvrent le contacts, ils échangent leurs état courant de connaissances. Après analyse comparée, ils régularisent leurs connaissances en s'échangeant les informations complémentaires.

FIGURE 5.3 – Politique de communication retenue (régime de régularisation)

Chaque robot i consigne l'état courant de ses connaissances comme suit. Pour chaque robot $j \neq i$, il mémorise l'index de l'élément reçu le plus récemment. Son fonctionnement repose sur l'hypothèse que les données sont transmises dans l'ordre chronologique, sans quoi sa consistance s'en trouverait compromise. Il est donc nécessaire que les paquets de données visio-inertielles ainsi que les correspondances intra et inter-robot détectées puis diffusées par chaque robot soient correctement indexés, de façon à refléter la chronologie de leur synthèse et de leur détection. Cette indexage permet notamment à chaque robot de vérifier qu'il n'a pas manqué de message. Le cas échéant, l'intégration du message reçu est mise en attente, le temps de requérir les données manquantes auprès du robot correspondant. Mais son utilité principale s'exprime lorsque deux robots recouvrent le contact après l'avoir perdu. Ils entament un processus de recouvrement mutuel qui débute par l'échange du vecteur représentant l'état courant de leurs connaissances (ainsi que des requêtes en suspens destinées au robot interlocuteur). Ceux-ci servent alors à orienter l'échange de régularisation qui va suivre. Chaque robot parcourt chaque entrée du vecteur reçu et vérifie s'il possède des informations postérieures à celles connues par

méthodes.

l'autre robot. Si c'est le cas, il les lui transmet. Cela suppose que chaque robot maintienne en parallèle une table de correspondance ordonnée qui associe chaque index à l'élement diffusé. Concernant les paquets de données visio-inertielles transmis, retenir leur étendue spatiale suffira pour les reconstruire en récupérant les facteurs qu'ils recouvrent. En effet, les résultats de leur synthèse (nouveaux facteurs, sélection d'amers etc.) sont systématiquement consignés dans la carte. Quant aux correspondances, il suffira d'associer l'index de communication à l'identifiant qui les caractérise. La procédure de recouvrement se traduit logiquement par un pic de communication en termes de charge réseau. Durant son déroulement, la diffusion et l'intégration des messages reçus est différée afin de ne pas briser la cohérence du mécanisme. Ce processus est schématisé par la Figure 5.3.

5.3.3 Stratégie commune d'association et de fusion de données

La stratégie d'association et de fusion des données recouvre l'intégration des informations reçues (paquets, correspondances, etc.). En particulier, l'intégration d'un paquet reçu comprend 5 aspects distincts : sa désérialisation dans la carte hôte, la détection des correspondances inter-robot rétrospectives, la détection ultérieure des correspondances inter-robot immédiates, l'ancrage des référentiels d'odométrie et l'inférence multi-robots.

Intégration différée des correspondances reçues

Une fois que le paquet reçu a été désérialisé et que ses informations ont été reconstruites et intégrées dans la carte, la première étape consiste à identifier, parmi les correspondances reçues des autres robots (observations inter-robot directes et correspondances indirectes), lesquelles peuvent désormais être intégrées dans la carte. Cette recherche est facilitée en la bornant aux correspondances dont la date est incluse dans l'intervalle de temps couvert par le paquet reçu. L'intégration éventuelle de nouvelles contraintes est alors notifiée au module d'inférence multi-robots.

Détection des correspondances inter-robot

La détection des correspondances inter-robot est un préalable crucial à l'estimation conjointe des trajectoires des robots car ce sont elles qui induisent de l'information mutuelle inter-robot et permettent de propager l'information d'une trajectoire à l'autre. Nous imposons à chaque robot de rechercher les correspondances entre sa propre trajectoire et les trajectoires reçues des autres robots puis de les leur diffuser afin de partitionner le

travail de détection des correspondances entre les robots en fonction des informations visuelles disponibles. L'idée retenue consiste à exploiter les informations visuelles transmises avec les paquets visio-inertiels échangés afin de repérer les correspondances. Celles-ci sont détectées sur la base de correspondances visio-structurelles dites 2D-3D entre les primitives de l'image de requête et des amers triangulés. Nous avons alors besoin d'une part des points caractéristiques et des descripteurs des images-clés pour formuler les requêtes, et d'autre part des images-clés associées à des informations structurelles. On utilise la méthode introduite par (Lynen et al. 2015) (c.f. section 2.4).

Nous devons considérer deux cas de figure. Lorsque le robot récepteur ajoute une nouvelle image-clé à sa trajectoire, alors il recherche les fermetures de boucles au sein de celle-ci, puis les corrrespondances inter-robot rétrospectives entre cette nouelle image et les observations qu'il a reçues des autres robots. Lorsqu'il intègre un paquet reçu d'un autre robot, alors il doit détecter, là encore de façon rétrospective, les correspondances inter-robot entre ces nouvelles observations et celles qu'il a accumulées le long de sa propre trajectoire. Dans les méthodes exposées ci-dessous, les techniques de détection des correspondances inter-robot rétrospectives et courantes changent selon que les appariements 2D-3D se font de la trajectoire hôte vers les trajectoires reçues ou inversement selon que les robots échangent des informations visuelles ou visio-inertielles. La détection de correspondances inter-robot est systématiquement notifiée d'une part au module d'alignement, décrit dans la section 5.3.3 ainsi qu'au module d'inférence globale.

Ancrage des référentiels d'odométrie

Dans la suite, nous supposons que les transformations relatives entre les référentiels d'odométrie des robots sont inconnues initialement. Étant donné que les problèmes d'optimisation associés à l'inférence globale sont des problèmes non-linéaire et non-convexes, l'inférence jointe sur les trajectoires des robots en requiert une initialisation correcte ; on parle d'ancrage des référentiels reçus par rapport au référentiel hôte. Cette tâche incombe à un module dédié qui procède comme suit. On regroupe les trajectoires ancrées les unes par rapport aux autres. Toute correspondance inter-robot détectée ou reçue d'un autre robot sont notifiées au module d'ancrage qui comptabilise les nombres de correspondances entre les différents groupes de trajectoires mutuellement ancrées. Dès lors que le nombre de correspondances liant deux groupes excède un certain seuil, on tente d'ancrer les deux groupes en question l'un par rapport à l'autre. Chaque correspondance induit une hypothèse d'ancrage entre les deux groupes. On choisit alors l'hypothèse la plus consensuelle

et corroborée par le plus de correspondances [2]. Les deux groupes de trajectoires sont alors fusionnés et les décomptes de correspondances entre les groupes restants sont mis à jour. Ce processus peut alors occasionner des alignements en cascade entre les différents groupes au fil des fusions. L'ancrage fournit des estimées initiales des poses des référentiels des trajectoires. Or, ces poses apparaissent explicitement dans les facteurs de pose relative dérivés des correspondances inter-robot et peuvent dès lors être affinées durant l'inférence multi-robots, en adoptant l'architecture de graphe proposée par (Kim et al. 2010).

Inférence globale multi-robots

Les opérations d'inférence globale sont déclenchées dès lors que des fermetures de boucles intra-robot ou correspondances inter-robot sont détectées, ou bien que de nouvelles trajectoires sont ancrées avec la trajectoire hôte. En effet, ces événements sont systématiquement notifiés au module d'inférence globale. Un compteur est enclenché à la première notification reçue de telles sortes que l'inférence globale est déclenchée si aucune autre notification n'est reçue dans un certain intervalle de temps de l'ordre de quelques secondes. À chaque nouvelle notification reçue, le compteur est réinitialisé, mais l'inférence est tout de même déclenchée dès lors que le nombre de correspondances en attente excède un certain seuil. Cette procédure permet d'impacter plusieurs correspondances nouvelles au cours d'une seule inférence.

En contexte visio-inertiel, la méthode d'inférence globale la plus exhaustive est l'ajustement de faisceaux visio-inertiel (Triggs et al. 1999), qui optimise les poses et les états inertiels le long des trajectoires ainsi que les positions des amers observés en prenant en compte l'intégralité (ou partie) des facteurs visio-inertiels. C'est ce type d'optimisation que nous utilisons éventuellement lors de la synthèse des paquets de données visio-inertiels afin d'extraire l'information locale au sous-graphe de facteurs extrait et que nous décrivons dans la section 2.5.3. Néanmoins, ce type d'optimisation est lent et complexe, ce qui la rend inadéquate dans un contexte orienté vers la navigation multi-robots où les correspondances inter-robot doivent être rapidement impactées sur l'estimation des trajectoires, d'autant plus qu'elle ne permet pas d'extraire rapidement les matrices de covariances sur les trajectoires qui peuvent être utiles en aval du processus de SLAM.

Dans les méthodes proposées ci-dessous, l'inférence globale exploite les modèles pro-

2. Une autre technique consisterait à déduire la transformation relative en formulant un problème d'inférence à partir des correspondances inter-robot considérées.

babilistes calculés lors de la synthèse des paquets visio-inertiels communiqués aux autres robots ainsi que de l'intégration des paquets reçus. En effet, les facteurs ou les sélections d'amers résultant des calculs desdits paquets sont sauvegardés, d'une part de façon à pouvoir être reconstitués et renvoyés dans le cas de procédures de régularisation, mais également pour servir de support à l'inférence globale. Cela garantit une optimisation uniforme et moins coûteuse sur toutes les trajectoires. Les problèmes d'inférence globale résultants seront détaillés au cas par cas pour chaque méthode proposée.

5.3.4 Sélection d'informations visuelles

Sélection d'images-clés le long des trajectoires

Dans cette section, nous présentons un mécanisme de sélection d'images-clés commun à l'ensemble des méthodes exposées ci-après. Cette technique de sélection sera notamment utilisée lors de la synthèse des paquets visio-inertiels pour adjoindre l'information visuelle, et afin de limiter la complexité de la recherche des correspondances inter-robot rétrospectives. Il vise à sélectionner un sous-ensemble d'images-clés le long de la trajectoires de façon à couvrir l'ensemble de l'environnement observé. Pour autant, le processus de sélection n'est pas formulé comme un problème de couverture ; nous sélectionnons les images-clés sur la base de critères de covisibilité, plus laxes que ceux généralement retenus par les algorithmes d'odométrie visuelle classiques. En effet, alors que ces derniers conservent l'information nécessaire à l'estimation du mouvement, nous ne visons ici qu'à préserver l'information visuelle suffisante pour des tâches de reconnaissance de lieux. La procédure est décrite par l'Algorithme 1.

Sélection de points d'intérêt sur les images

Plusieurs des méthodes présentées ci-après recourent à des mécanismes de sélection de points d'intérêts dans les images. Une image comprend généralement plusieurs centaines de points d'intérêts, chacun étant associé à un descripteur. Cependant, on peut généralement se contenter d'un sous-ensemble de ces points dans une optique de détection de correspondances visuelles. Lorsqu'il s'agit de communiquer des informations visuelles sur une image en particulier en contexte multi-robots, nous ne transmettons ainsi qu'une partie de ses points d'intérêts et descripteurs, que nous sélectionnons comme suit. Nous sélectionnons les $n_{\text{keypoints}}$ points-clés associés aux amers les plus observés.

Algorithme 1 – Sélection des images-clés sur une sous-trajectoire

Paramètres

$n_{\min}$: nombre seuil d'amers en commun;

$q_{\min}$: quotient seuil d'amers en commun minimum;

α : coefficient compris entre 0 et 1

Entrée : $I_{1:N} = \{I_0, ..., I_N\}$ l'ensemble des images-clés successives extraites d'une sous-trajectoire

Sortie : $\mathcal{S}$ l'ensemble des images-clés sélectionnées.

`// Initialisation`

$\mathcal{S} \leftarrow \{I_0, I_N\}$ `// Pré-sélection des images-clés extrémales`

$r \leftarrow 0$ `// Index de l'image-clé de référence`

pour $i \in \{1, ..., N-1\}$ **faire**

  `// Sélectionner l'image-clé comme candidate`

  $n_{ri} \leftarrow |\mathcal{L}_r \cap \mathcal{L}_i|$ `//` $\mathcal{L}_j$ `est l'ensemble des amers observés par l'image` I_j

  $q_{ri} \leftarrow \dfrac{|\mathcal{L}_r \cap \mathcal{L}_i|}{|\mathcal{L}_r|}$;

  si $n_{iN} \leq n_{\min}$ ***ou*** $q_{iN} \leq q_{\min}$ **alors**

    `// Vérifier que l'image candidate n'est pas trop proche de` I_N

    $n_{iN} \leftarrow |\mathcal{L}_i \cap \mathcal{L}_N|$;

    $q_{iN} \leftarrow \dfrac{|\mathcal{L}_i \cap \mathcal{L}_N|}{|\mathcal{L}_N|}$;

    si $n_{ri} \leq \alpha \cdot n_{\min}$ ***ou*** $q_{ri} \leq \alpha \cdot q_{\min}$ **alors**

      `// Sélectionner l'image courante`

      $\mathcal{S} \leftarrow \mathcal{S} \cup \{I_i\}$;

      $r \leftarrow i$;

Cette heuristique simple s'appuie sur le fait que le nombre d'observations d'un même amer fournit une indication significative sur sa stabilité et sa saillance – qui elle-même résulte conjointement de sa distingabilité, de sa répétitivité et de sa détectabilité selon (Buoncompagni et al. 2015). Par ailleurs, notons qu'en utilisant les descripteurs quantifiés introduits par (Lynen et al. 2015), si les robots partagent le même vocabulaire produit, alors il suffit de communiquer les indexes des mots visuels.

5.4 Échange de sous-cartes visio-inertielles rigides

La première méthode investiguée repose sur l'échange de sous-cartes visio-inertielles rigides entre les robots. Elle s'inscrit dans la philosophie générale qui sous-tend les approches basées sous-cartes : les sous-cartes localement consistantes sont calculées, tandis que la consistance globale est recherchée en recalant ces sous-cartes les unes par rapport aux autres. En entérinant les optimisations locales, ce découplage local/global réduit ainsi la complexité de l'inférence globale.

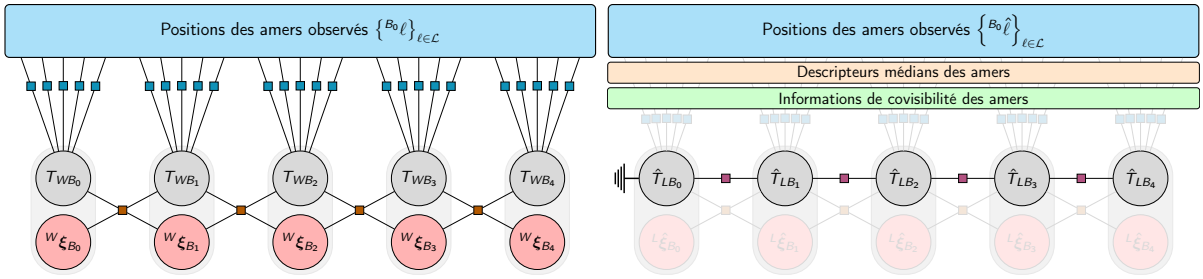
5.4.1 Travaux associés

Faire des sous-cartes l'unité élémentaire d'échange entre les robots sied particulièrement à la segmentation des échanges en une succession de paquets de données résumant des informations locales. Cette approche constraste avec les mécanismes d'échange image-clé par image-clé qu'implémentent la majorité des méthodes de SLAM collaboratif, qu'elles soient centralisées ou décentralisées. Dans le cas centralisé, l'algorithme CSfM (Forster et al. 2013) impose à chaque agent de transmettre toutes les informations relative à son image-clé courante au serveur central. Les informations communiquées incluent l'ensemble des points d'intérêts extraits des images, leurs descripteurs, ainsi que la pose relative estimée par rapport à l'image-clé précédente. Les algorithmes de SLAM multi-robots mono-visuel CCM-SLAM (Schmuck et al. 2019a) ainsi que son extension visio-inertielle CVI-SLAM (Karrer et al. 2018) prolongent cette logique : en plus des informations visuelles et odométriques standards, les robots communiquent également au serveur les amers dont ils ont triangulé ou mis à jour la position dans l'image courante. La communication instantanée des images-clés, bien qu'elle induise une charge réseau permanente, n'est pas l'apanage des méthodes centralisées. Par exemple, dans l'algorithme de SLAM collaboratif décentralisé DOOR-SLAM (Lajoie et al. 2020), chaque robot diffuse aux autres robot les informations visuelles extraites de son image-clé courante.

Si le concept de sous-carte n'est pas récent, son utilisation en contexte collaboratif est restée relativement limitée. Une contribution notable dans le cas décentralisé est la méthode développée par (Schuster et al. 2015) en SLAM stéréo-inertiel. Celle-ci exploite l'échange de sous-cartes stéréo-visuelles rigides associées à un graphe de facteurs local, lui-même construit à partir des estimées de pose et de covariance de filtres de Kalman à référence locale, ré-initialisés pour chaque nouvelle sous-carte. Les correspondances intra

et inter-robot entre les sous-cartes sont détectées en recalant les nuages de points denses des sous-cartes à l'aide de l'algorithme *Iterative Closest Point* (ICP) et d'une étape de pré-alignement exploitant l'appariement de descripteurs 3D de type C-SHOT (Tombari et al. 2011). Toujours en contexte stéréo-visuel, la méthode proposée par (Duhautbout et al. 2019), repose également sur l'échange de sous-cartes mais substitue aux nuages de points stéréo une représentation implicite de type TSDF. Enfin, en contexte visio-inertiel, nous pouvons citer la méthode publiée par (Sartipi et al. 2019). Il s'agit d'une méthode décentralisée, employée à des fins de localisation collaborative pour assurer la cohérence des projections de modèles 3D pour les robots dans le cadre d'applications de réalité augmentée. Les robots diffusent régulièrement des sous-cartes visio-inertielles de leur environnement immédiat, incluant des mises à jour explicites des sous-cartes précédemment communiquées. Celles-ci résultent d'un ajustement de faisceaux fenêtré pratiqué périodiquement. Les robots utilisent alors les sous-cartes qu'ils reçoivent pour détecter des correspondances inter-robot avec leur propre sous-carte courante et estimer leur pose relative avec les autres robots. Cette méthode ne vise donc qu'une relocalisation instantanée, mais n'utilise pas les sous-cartes reçues afin de détecter des correspondances rétrospectives ni opérer des inférences multi-robots globales.

5.4.2 Vue d'ensemble de la méthode proposée



(a) Sous-graphe visio-inertiel initial (b) Paquet SVIP calculé

Les paquets SVIP sont calculés à partir d'un sous-graphe de facteurs contenant des facteurs visuels ■ et inertiels ■ sur des variables de pose ○ et d'états inertiels ○. Les paquets incluent des facteurs de pose relative ■ marginalisés qui résument l'information de localisation, ainsi qu'un nuage d'amers rigide qui résulte d'une optimisation locale. On leur adjoint leur descripteurs médians ainsi qu'une table de covisibilité des amers qui sont requises afin de détecter les correspondances inter-robot.

Figure 5.4 – Schéma des paquets communiqués par SVIP

Dans cette section, nous proposons une méthode basée sur l'échange de sous-cartes visio-inertielles rigides. Les paquets échangés entre les robots incluent des informations

odométriques et visio-structurelles tel que représenté par la Figure 5.4. Il s'agit tout d'abord d'un nuage d'amers rigide triangulés sur la base d'une inférence locale, et accompagnés de leur descripteur visuel médian. On y adjoint une table de covisibilité des amers afin de rendre la détection des correspondances entre les sous-cartes plus robuste selon le procédé établi par (Lynen et al. 2015). Concernant la trajectoire, l'information de localisation est condensée sur des facteurs de pose relative entre les image-clés successives. Ces facteurs résultent d'une marginalisation directe du résultat de l'optimisation locale de la sous-carte. Ces informations permettent de détecter les correspondances entre les robots, tandis que les fermetures de boucles au sein des trajectoires sont détectées à l'aide des méthodes classiques en comparant les images-clés. Conformément à la politique de communication décrite dans la section 5.3.2, celles-ci sont ensuite diffusées aux autres robots. Notons cependant que l'exhaustivité des informations visuelles communiquées rend superflu le mécanisme de requête d'informations visuelles supplémentaires qui n'est donc pas utilisé ici. La détection des correspondances déclenchent enfin des processus d'inférence globale s'appuyant sur les graphes de facteurs communiqués. Nous prénommons cette méthode SVIP pour *Submap Visual-Inertial Packets*.

5.4.3 Calcul d'un paquet

Le calcul d'un paquet comprend trois étapes successives. D'abord, la sous-carte est extraite et optimisée localement. À partir des résultats de l'inférence locale, on construit ensuite le graphe de facteurs qui récapitule les informations de localisation. Enfin, on adjoint les informations visio-structurelles qui permettront ultérieurement de détecter les correspondances inter-robot lors de l'intégration des paquets échangés.

Inférence visio-inertielle locale

Conformément au procédé décrit dans la section 5.3.2, on extrait le sous-graphe sous-jacent à la sous-carte courante. On réalise alors une inférence locale :

$$\hat{\Theta} = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\min} \sum_{z \in \mathcal{Z}_{\text{VI}}} \|\boldsymbol{\xi}_z(\Theta)\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z}} \tag{5.1}$$

où $\mathcal{Z}_{\text{VI}}$ est l'ensemble des facteurs visuels et inertiels contenus dans le sous-graphe extrait, $\mathcal{H}_\Theta$ est l'espace des hypothèses, $\boldsymbol{\xi}_z$ est le résidu associé à l'observation $z \in \mathcal{Z}_{\text{VI}}$, $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z}$ est la matrice de covariance associée, et Θ est l'ensemble des variables optimisées. Θ contient

l'ensemble $\mathcal{T}$ des variables de pose des images-clés successives, l'ensemble $\mathcal{E}$ des états inertiels associés et l'ensemble $\mathcal{L}$ des positions des amers observés dans la sous-carte. La pose de la première image-clé est fixée, et l'inférence se déroule dans un référentiel local L quelconque. On ne considère donc que les informations internes à la sous-carte. L'optimisation se termine lorsqu'elle converge ou bien qu'elle atteint le nombre maximal d'itérations autorisé. On filtre alors les amers mal contraints : on considère qu'un amer est mal contraint s'il est observé par moins de 5 images-clés et si sa parallaxe (ou disparité angulaire) n'excède pas 5 degrés.

Ajout de l'information de localisation

La seconde étape consiste à restituer les contraintes sur la trajectoire sous la forme d'un graphe de facteurs liant les images-clés incluses dans la sous-carte. On utilise un procédé sommaire de marginalisation et d'éparsification afin de calculer des facteurs de pose relative entre les images-clés successives. Pour ce faire, on extrait d'abord la matrice d'information observée de Fisher $\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\Theta})$:

$$\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\Theta}) = \sum_{z \in \mathcal{Z}_{\mathrm{VI}}} \boldsymbol{J}^{\boldsymbol{\xi}_z}_{\Theta}(\hat{\Theta})^{\top} \cdot \boldsymbol{\Sigma}^{-1}_{\boldsymbol{\xi}_z} \cdot \boldsymbol{J}^{\boldsymbol{\xi}_z}_{\Theta}(\hat{\Theta}) \tag{5.2}$$

où $\boldsymbol{J}^{\boldsymbol{\xi}_z}_{\Theta}(\hat{\Theta})$ est la matrice Jacobienne du résidu $\boldsymbol{\xi}_z$ du facteur z par rapport aux variables incluses dans Θ et évaluée en $\hat{\Theta}$. On obtient ainsi une distribution Gaussienne équivalente :

$$p(\Theta|\mathcal{Z}_{\mathrm{VI}}) = \mathcal{N}\left(\boldsymbol{\xi}_{\hat{\Theta}}(\Theta); \mathbf{0}, \left[\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\Theta})\right]^{-1}\right) \tag{5.3}$$

où $\boldsymbol{\xi}_{\hat{\Theta}}(\Theta)$ désigne le résidu associé à l'estimée $\hat{\Theta}$. On calcule ensuite la matrice d'information $\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\mathcal{T}})$ sur les seules variables de pose $\mathcal{T}$. Elle s'obtient comme le complément de Schur des variables marginalisées :

$$\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\mathcal{T}}) = \left[\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\Theta})\right]_{\mathcal{T}\mathcal{T}} - \left[\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\Theta})\right]_{\mathcal{T}M} \cdot \left[\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\Theta})\right]^{-1}_{MM} \cdot \left[\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\Theta})\right]^{\top}_{\mathcal{T}M} \tag{5.4}$$

où l'indice M désignent les sous-blocs correspondants aux variables marginalisées, à savoir les états inertiels $\mathcal{E}$ et les positions des amers $\mathcal{L}$. La matrice de covariance sur les poses s'obtient par inversion : $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\hat{\mathcal{T}}}|\mathcal{Z}_{\mathrm{VI}}} = \left[\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\mathcal{T}})\right]^{-1}$.

Pour chaque paire d'images-clés successives, on calcule un facteur virtuel Gaussien de pose relative $p(\hat{T}_{\mathrm{B}_i\mathrm{B}_{\mathrm{i+1}}}|T_{\mathrm{LB}_i}, T_{\mathrm{LB}_{\mathrm{i+1}}})$. La mesure virtuelle correspond à la pose rela-

tive obtenue à l'issue de l'inférence : $\hat{T}_{\mathrm{B}_i\mathrm{B}_{i+1}} = \hat{T}_{\mathrm{LB}_i}^{-1} \oplus \hat{T}_{\mathrm{LB}_{i+1}}$. La matrice de covariance $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{T_{\mathrm{B}_i\mathrm{B}_{i+1}}}|\mathcal{Z}_{\mathrm{VI}}}$ du facteur est calculée en projetant l'information de la matrice de covariance dans le sous-espace correspondant aux nouveaux facteurs virtuels :

$$\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{T_{\mathrm{B}_i\mathrm{B}_{i+1}}}|\mathcal{Z}_{\mathrm{VI}}} = \left[\boldsymbol{J}^{T_{\mathrm{B}_i\mathrm{B}_{i+1}}}_{\{T_{\mathrm{LB}_i},T_{\mathrm{LB}_{i+1}}\}}(\hat{\mathcal{T}})\right] \cdot \boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\{T_{\mathrm{LB}_i},T_{\mathrm{LB}_{i+1}}\}}|\mathcal{Z}_{\mathrm{VI}}} \cdot \left[\boldsymbol{J}^{T_{\mathrm{B}_i\mathrm{B}_{i+1}}}_{\{T_{\mathrm{LB}_i},T_{\mathrm{LB}_{i+1}}\}}(\hat{\mathcal{T}})\right]^{\top} \tag{5.5}$$

La matrice Jacobienne $\boldsymbol{J}^{T_{\mathrm{B}_i,\mathrm{B}_{i+1}}}_{\{T_{\mathrm{LB}_i},T_{\mathrm{LB}_{i+1}}\}}$ est définie dans le chapitre d'Annexes C. Cette méthode correspond à la méthode d'éparsification proposée par (Mazuran et al. 2014) dont nous reparlerons plus spécifiquement dans la section 5.5.

Ajout de l'information visio-structurelle

L'étape cruciale de la synthèse des sous-carte est l'ajout de l'information de nature visio-structurelle qui permettra aux robots qui les recevront de détecter les correspondances inter-robot. Cette information prend la forme d'un nuage de points 3D constitué des amers observés au sein de la sous-carte. Leur estimée de position résulte de l'inférence locale et est exprimée relativement au référentiel inertiel attaché à l'image-clé de référence. On adjoint à chaque amer son descripteur médian, c'est-à-dire le descripteur dont la distance de Hamming à tous les autres est la plus faible. Formellement, on considère un amer l_i et l'ensemble de ses descripteurs $\mathcal{D}_i$. On choisit le descripteur d_i^* tel que :

$$d_i^* = \underset{d \in \mathcal{D}_i}{\arg\min} \sum_{j=1}^{|\mathcal{D}_i|} d_{\mathrm{Hamming}}(d, d_j) \tag{5.6}$$

où d_{Hamming} est la distance de Hamming. Dans la suite, nous travaillons avec des descripteurs binaires de type BRISK (Leutenegger et al. 2011). Enfin, dans le but d'utiliser le module de reconnaissance de lieux introduit par (Lynen et al. 2015), nous fournissons une table de covisibilité des amers qui associe à chaque image-clé associe la liste des indexes des amers qu'elle observe (sans leurs coordonnées 2D). Celles-ci sont nécessaires lors des vérifications topologiques qui consistent à regrouper toutes les images-clés candidates en groupes de covisibilité afin de filtrer les correspondances erronées.

Finalisation et sauvegarde du paquet calculé

L'étape précédente complète la synthèse du paquet de données à diffuser aux autres robots. Comme illustré par la Figure 5.4, le paquet final contient : une liste ordonnée de

facteurs de pose relative liant les images-clés consécutives ; une liste d'amers caractérisés par leur index, leur position et un descripteur médian ; pour chaque image-clé, une liste des indexes des amers qu'elle observe (i.e. une table de covisibilité).

Le paquet calculé est alors diffusé aux autres robots. En parallèle, nous sauvegardons dans la carte les facteurs de pose relative calculés durant l'opération. Ceux-ci seront d'une part utilisés pour l'inférence globale, et d'autre part durant la procédure de régularisation afin de relayer à d'autres robots des connaissances qui leur manqueraient, et notamment éviter de recalculer le paquet en question. On indexe enfin le paquet nouvellement calculé en incrémentant l'index courant qui ordonne chronologiquement les éléments d'informations communiqués, que ce soit des correspondances intra ou inter-robots ou des paquets.

5.4.4 Intégration d'un paquet

Désérialisation et reconstruction du paquet dans la carte

L'intégration d'un paquet reçu est directe : la trajectoire est complétée à l'aide des facteurs de pose relative consécutifs, tandis que le nuage d'amers associé est initialisé à l'aide des estimées de positions fournies pour les amers.

Détection des correspondances inter-robots rétrospectives

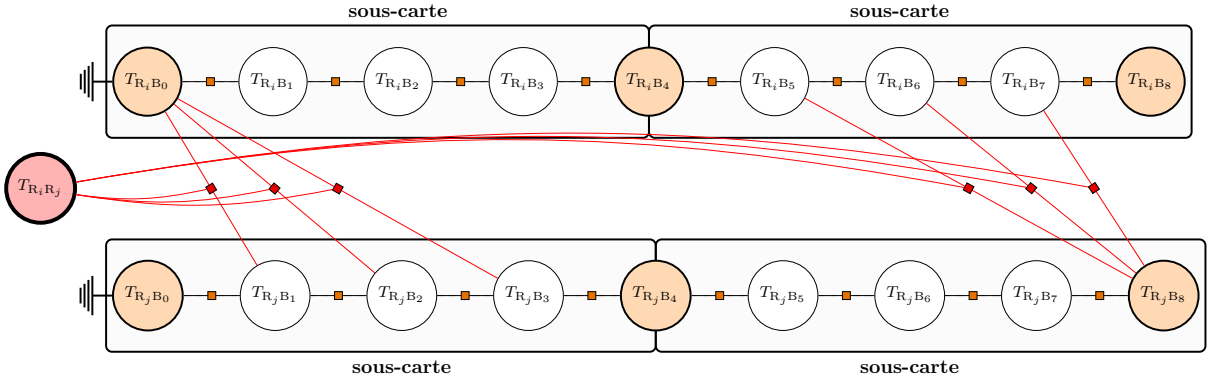
Le premier cas de figure concerne la détection rétrospective des correspondances inter-robot lors de l'intégration d'un paquet reçu. Chaque robot détecte uniquement les correspondances entre sa propre trajectoire et celles qu'il reçoit des autres robots. Cette détection s'effectue toujours dans le même sens afin d'éviter toute redondance dans la division des tâches entre les robots. Chaque robot tente alors de détecter des correspondances entre les images-clés de sa propre trajectoire et les informations visio-inertielles reçues [3]. Sitôt la nouvelle sous-carte reçue reconstruite, on ajoute ses informations visio-structurelle à une base de données visuelle spécifique contre laquelle nous comparerons les images-clés de la trajectoire hôte. Cependant, rechercher des correspondances entre toutes les images-clés de la trajectoire hôte et sous-carte reçue risquerait de compromettre l'extensibilité de la méthode par rapport aux nombre de robots, d'autant plus qu'il s'agirait d'une tâche de complexité infiniment croissante au fur et à mesure que la trajectoire hôte s'accroît. C'est pourquoi, dans un premier temps, nous ne recherchons les correspondances inter-robot

3. Nous verrons que le processus est inversé dans les deux autres méthodes présentées.

que sur un sous-ensemble des images-clés hôtes, sélectionnées au fil de l'ajout de nouvelles images-clés selon l'algorithme présenté dans la section 5.3.4 [4]. Cependant, dès lors qu'une correspondance inter-robot est détectée avec l'une de ces images-clés, toutes les images-clés intermédiaires avec les images-clés sélectionnées adjacentes sont également testées. Cela permet de mieux cibler l'affectation des ressources de calcul en vue de détecter les correspondances rétrospectives. Une fois les correspondances rétrospectives détectées, les informations visio-structurelles de la base de données exclusives à la nouvelle sous-carte sont transférées vers la base de données visuelle générale, regroupant les informations visio-structurelles de toutes les sous-cartes reçues du même robot.

Le second cas de figure consiste à détecter les correspondances rétrospectives dès lors que le robot récepteur ajoute une nouvelle image-clé à sa trajectoire. Une requête est alors formulée auprès des bases de données visio-structurelles de chaque autre robot. Une solution alternative, moins coûteuse, consiste à regrouper les informations de toutes les sous-cartes reçues au sein d'une unique base de données, ce qui permet de n'effectuer qu'une seule requête multi-robots pour chaque nouvelle image-clé, mais au risque de manquer des correspondances multiples avec plusieurs autres robots.

Inférence multi-robots



Les facteurs ■ désignent les facteurs de pose relative virtuels calculés durant la synthèse des sous-cartes, et les facteurs ■ dérivent des correspondances inter-robot détectées. Comme illustré ici, lorsqu'une correspondance est établie avec une sous-carte donnée, alors elle engendre une facteur avec sa pose de référence.

FIGURE 5.5 – Structure du graphe de facteurs multi-robots construit par SVIP

L'inférence multi-robots prend la forme d'une optimisation de graphe de poses qui

4. En considérant uniquement le critère de distance avec la dernière image-clé sélectionnée.

inclut les trajectoires de tous les robots dont les référentiels d'odométrie ont été préalablement ancrés avec celui du robot hôte. Durant cette optimisation, on ne considère que les facteurs de pose relatives reçus des autres robots ou calculés lors de la synthèse des sous-cartes, ainsi que les facteurs issus des correspondances inter-robot. On résout le problème suivant :

$$\hat{\Theta} = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\min} \left\{ \sum_{\mathrm{R}_i \in \mathcal{R}} \sum_{z_k^i \in \mathcal{Z}_{\mathrm{R}_i}} \left\| \boldsymbol{\xi}_{z_k^i} \left(T_{\mathrm{R}_i\mathrm{B}_k^{(i)}}, T_{\mathrm{R}_i\mathrm{B}_{k+1}^{(i)}} \right) \right\|^2_{\boldsymbol{\Sigma}_k^i} + \sum_{\mathrm{R}_i \in \mathcal{R}} \sum_{z_{kl}^i \in \mathcal{Z}^{\text{intra}}_{\mathrm{R}_i}} \rho \left(\left\| \boldsymbol{\xi}_{z_{kl}^i} \left(T_{\mathrm{R}_i\mathrm{B}_k^{(i)}}, T_{\mathrm{R}_i\mathrm{B}_l^{(i)}} \right) \right\|_{\boldsymbol{\Sigma}_{kl}^i} \right) + \sum_{\mathrm{R}_i, \mathrm{R}_{j \neq i} \in \mathcal{R}} \sum_{z_{kl}^{ij} \in \mathcal{Z}^{\text{inter}}_{\mathrm{R}_i, \mathrm{R}_j}} \rho \left(\left\| \boldsymbol{\xi}_{z_{kl}^{ij}} \left(T_{\mathrm{R}_i\mathrm{B}_k^{(i)}}, T_{\mathrm{R}_i\mathrm{B}_l^{(j)}}, T_{\mathrm{R}_i\mathrm{R}_j} \right) \right\|_{\boldsymbol{\Sigma}_{kl}^{ij}} \right) \right\} \quad (5.7)$$

Dans l'équation précédente, Θ désigne l'ensemble des variables inférées, $\mathcal{Z}_{\mathrm{R}_i}$ est l'ensemble des facteurs calculés lors de la synthèse des sous-cartes du robot R_i, $\mathcal{Z}^{\text{intra}}_{\mathrm{R}_i}$ est l'ensemble des facteurs de fermeture de boucle au sein de sa trajectoire, et $\mathcal{Z}^{\text{inter}}_{\mathrm{R}_i,\mathrm{R}_j}$ regroupe l'ensemble des correspondances inter-robot entre R_i et R_j. Il inclut les variables de pose $T_{\mathrm{R}_i\mathrm{B}_j} \in \mathbb{SE}_3$ où R_i désigne le référentiel d'odométrie attaché au robot i et B_j le référentiel inertiel attaché à sa $j^{\text{ème}}$ image-clé, ainsi que les poses relatives entre les référentiels d'odométrie $T_{\mathrm{R}_i\mathrm{R}_j}$. La fonction ρ désigne une fonction de pénalité d'Huber (Huber 1992) qui permet d'atténuer l'impact des potentielles correspondances erronées lors de l'inférence.

5.5 Échange de paquets visio-inertiels condensés

Si la méthode précédente permet de communiquer les informations visio-structurelles de façon exhaustive, elle pèche dans la communication des informations de localisation. En particulier, le graphe de facteurs sous-jacent aux sous-cartes résulte d'une méthode de marginalisation et d'éparsification rudimentaire qui projette toute l'information du graphe sur chaque facteur. Par conséquent, les facteurs sont *consanguins*, en ce sens qu'ils sont corrélés, sans pour autant que cette corrélation ne soit restituée. Dans cette section, nous présentons une méthode de calcul de paquets *condensés* qui repose sur un processus d'éparsification garantissant la consistance des paquets calculés et propose une gestion alternative des informations visuelles.

5.5.1 Travaux associés

Les techniques de marginalisation et d'éparsification permettent de condenser l'information contenue dans les facteurs visio-inertiels et d'en restituer la teneur à travers un jeu de facteurs plus simple. On y recourt notamment afin d'atténuer la complexité sans cesse croissante des problèmes d'inférence en SLAM (Hsiung et al. 2018).

Fondamentalement, le processus de marginalisation permet de projeter toute l'information contenue dans une distribution $p(\Theta)$ sur un sous-ensemble de ses variables $\Theta_s \subset \Theta$. Cependant, la distribution marginalisée $p(\Theta_s)$ est une distribution dense, car l'information apportée par les variables marginalisées se retranscrit sous la forme de corrélations croisées entre les variables restantes, ce que les anglo-saxons désignent sous le terme de *fill-in* pour caractériser les entrées supplémentaires qui en résultent dans la matrice d'information dans le cas Gaussien. Le processus d'éparsification consiste alors à approximer cette distribution dense par une distribution éparse – i.e. avec moins de corrélations partielles entre les variables – qui serait engendrée par un jeu de facteurs re-linéarisables. On parle alors de facteurs virtuels étant donné qu'il ne correspondent à aucune observation physique. Dans le cas Gaussien, cela suppose de calculer la moyenne et la matrice de covariance ou d'information associée à chaque facteur de la nouvelle topologie.

Idéalement, le processus d'éparsification doit minimiser la perte d'information qu'il induit par rapport à la distribution dense originale. On quantifie communément cette perte d'informations à l'aide de métriques issues de la théorie de l'information. L'entropie relative, ou divergence de Kullback-Leibler (KL) (Kullback et al. 1951), convient particulièrement [5]. Étant donné une distribution q approximant une "vraie" distribution p, la divergence KL de q calculée par rapport à p s'exprime comme suit :

$$\mathcal{D}_{\mathrm{KL}}(p, q) = \int p(\boldsymbol{x}) \log \left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \right) d\boldsymbol{x} \tag{5.8}$$

et quantifie l'écart d'information apportée par les deux distributions sous l'hypothèse que la distribution p décrit exactement l'état des connaissances. Dans le cas Gaussien où $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ et $q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$, la divergence KL vaut :

$$\mathcal{D}_{\mathrm{KL}}(p, q) = \frac{1}{2} \left[\mathrm{trace}\left(\boldsymbol{\Sigma}_q^{-1} \cdot \boldsymbol{\Sigma}_p - \boldsymbol{I}_n \right) + \| \boldsymbol{\mu}_p - \boldsymbol{\mu}_q \|^2_{\boldsymbol{\Sigma}_q} + \log \left(\frac{\det \boldsymbol{\Sigma}_q}{\det \boldsymbol{\Sigma}_p} \right) \right] \tag{5.9}$$

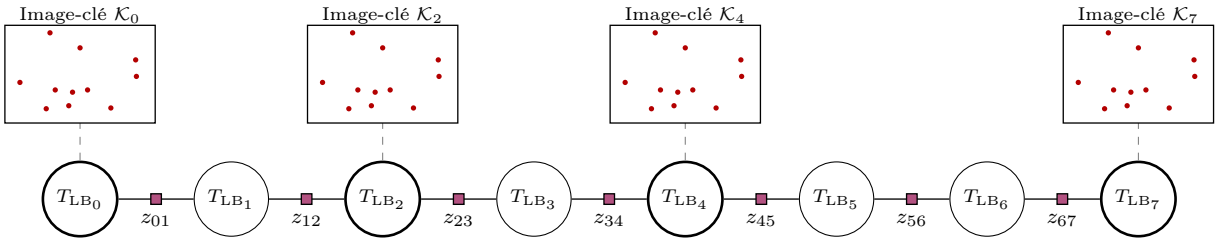
5. Il ne s'agit cependant pas d'une distance.

Les premiers choix qui conditionnent cette minimisation sont d'une part celui de la nature des facteurs virtuels qui approximeront la distribution dense et d'autre part leur topologie. Une première solution très employée dans la littérature consiste à adopter une topologie sous forme d'arbre de recouvrement. Elle présente l'avantage de fréquemment déboucher sur des solutions analytiques pour le calcul des matrices de covariances dans le cas Gaussien. C'est par exemple le cas des arbres de Chow-Liu (Chow et al. 1968). Ils se construisent de telles sortes à maximiser la somme des informations mutuelles entre les variables adjacentes selon la nouvelle topologie. Cette approche est par exemple adoptée par (Kretzschmar et al. 2011) afin d'éliminer des poses associées à des scans lasers. Les topologies en arbre sont reprises par les méthodes *Generic Linear Constraints* (Carlevaris-Bianco et al. 2014) et *Nonlinear Factor Revovery* (NFR) (Mazuran et al. 2014) qui proposent également des solutions analytiques. Cependant, les topologies de ce type ont une capacité limitée à restituer toute la teneur informative de la distribution dense initiale. En imposant de projeter l'information initiale sur un jeu aussi réduit de facteurs indépendants, elles échouent à résumer la complexité des interactions entre les variables du fait des indépendances conditionnelles fortes ainsi induites. C'est pourquoi d'autres techniques proposent de recourir à des topologies plus riches. Cette solution prohibe cependant toute solution analytique et impose l'emploi de méthodes itératives pour calculer les facteurs. Les contraintes structurelles sur les matrices de covariances sont alors assurées par des mécanismes complexes telles que les barrières logarithmiques pour les méthodes de points intérieurs (Eckenhoff et al. 2016a). D'autres méthodes (Mazuran et al. 2014) adoptent des techniques de type quasi-Newton et rectifient a posteriori le spectre des matrices de covariance obtenues pour les rendre définie-positives. (Vallvé et al. 2018) ont quant à eux proposé une méthode dite de descente de facteurs. Inspirée des techniques de descente de coordonnées et généralisant la méthode NFR, elle consiste à parcourir cycliquement les facteurs virtuels pour les optimiser individuellement tout en figeant les autres. La matrice d'information de chaque facteur est alors calculée en projetant l'information de la distribution dense sur le sous-espace associé, puis en retranchant l'information rétro-projetée depuis les autres facteurs. Enfin, une dernière technique consiste à coupler le choix de la topologie au calcul des facteurs eux-mêmes à l'aide de techniques de régularisation basées par exemple sur la norme $\mathcal{L}_1$. C'est en particulier l'approche adoptée par (Huang et al. 2013) et (Eckenhoff et al. 2016a).

Si le processus d'éparsification doit minimiser la perte d'informations, il doit se garder d'en rajouter. Dans le cas contraire, cela risquerait d'induire des biais d'estimation

et de surestimer la fiabilité des estimées obtenues. La distribution éparsifiée doit être consistante avec la distribution dense originale, en ce sens qu'elle ne peut témoigner d'un quelconque gain artefactuel de connaissances. Formellement, étant donné une distribution p approximée par une distribution q sur un ensemble de variables Θ, cela revient à s'assurer que nulle entropie marginale n'est réduite i.e. $\forall \Theta_S \in \mathcal{P}(\Theta)$, $H_q(\Theta_S) \geq H_p(\Theta_S)$, où H_q et H_p désignent des entropies de Shannon calculées par rapport aux distributions q et p. Dans le cas Gaussien, cette condition équivaut à imposer un ordre de Loewner sur les matrices de covariance respectives i.e. $\boldsymbol{\Sigma}_q \geq \boldsymbol{\Sigma}_p$. Géométriquement, cela implique que les ellipsoïdes d'incertitudes de p sont inclus dans ceux de q pour un même quantile d'erreur. Il s'agit d'une contrainte particulièrement complexe à imposer, ce qui explique que de fait, la plupart des méthodes d'éparsification publiées l'omettent. Quelques techniques existent néanmoins. (Eckenhoff et al. 2016a) utilisent pour ce faire une barrière logarithmique sur le déterminant de $\boldsymbol{\Sigma}_q - \boldsymbol{\Sigma}_p$ pour circonscrire le domaine de consistance, avec l'inconvénient de complexifier lourdement le problème d'optimisation. (Paull et al. 2015) et (Huang et al. 2013) adoptent une solution plus simple mais plus conservative. Elle consiste à d'abord résoudre le problème d'éparsification relaxé de la contrainte de consistance, puis à assurer a posteriori la consistance de la solution obtenue en opérant directement sur ses valeurs propres. C'est de cette dernière technique dont nous nous inspirons dans la méthode que nous proposons ci-après.

5.5.2 Vue d'ensemble de la méthode proposée



Le paquet comprend un graphe de facteurs de poses relatives ■ entre les images-clés successives et calculés par marginalisation et éparsification, ainsi qu'une sélection de points-clés • associés à leurs descripteurs sur un sous-ensemble des images-clés. L désigne un référentiel quelconque.

Figure 5.6 – Morphologie d'un paquet CVIP

Dans cette section, nous proposons une méthode de synthèse de paquets visio-inertiels principalement inspirée de l'algorithme *AUV-CSLAM* proposé par (Paull et al. 2015) pour le SLAM sous-marin acoustique. Nous la dénommons CVIP pour *Condensed Visual-*

Inertial Packets. Dans chaque paquet, tel que schématisé par la Figure 5.6, l'information de localisation est condensée selon des mécanismes de marginalisation et d'éparsification sous contrainte de consistance. Elle est ainsi restituée sous la forme d'un graphe de facteurs de poses relatives. Contrairement à la méthode précédente, aucune information structurelle n'est communiquée, et l'information visuelle ne l'est que façon locale sur un sous-ensemble des images-clés. Cela permet d'amorcer la détection de correspondances inter-robot, puis de l'étoffer à l'aide du système de requête d'images-clés décrit dans la section 5.3.2.

5.5.3 Calcul d'un paquet

Le calcul d'un paquet comprend deux étapes principales. La première condense l'information de localisation via la marginalisation locale du sous-graphe de facteurs initial suivi de son éparsification. La seconde étape consiste à ajouter une sélection de l'information visuelle associée au sous-graphe original.

Inférence visio-inertielle locale

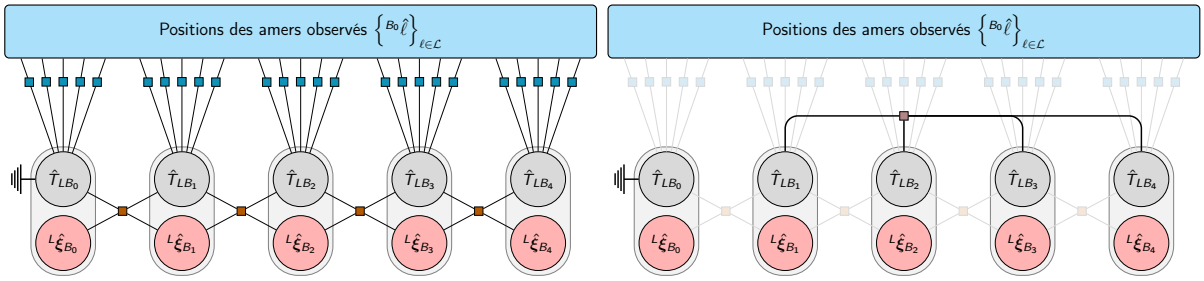
À l'instar de la méthode précédente, on extrait le sous-graphe de facteurs conformément à la politique de communication décrite dans la section 5.3.2 puis on réalise une inférence locale en considérant uniquement les facteurs visuels et inertiels $\mathcal{Z}_{\text{VI}}$ que contient ledit sous-graphe. On obtient alors une estimée *locale* $\hat{\Theta}$ des poses $\mathcal{T}$, des états inertiels $\mathcal{E}$ (vitesses et biais IMU) et des positions des amers $\mathcal{L}$:

$$\hat{\Theta} = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\min} \sum_{z \in \mathcal{Z}_{\text{VI}}} \|\boldsymbol{\xi}_z(\Theta)\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z}} \tag{5.10}$$

où, comme précédemment, $\boldsymbol{\xi}_z$ et $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z}$ désignent respectivement le résidu et la matrice de covariances associées au facteur z, tandis que $\mathcal{H}_\Theta$ désigne l'espace d'hypothèses dans lequel évolue Θ. Les degrés de libertés de jauge sont fixés en figeant la pose de la première image-clé. À l'issue de l'optimisation, les amers insuffisamment contraints sont éliminés selon les mêmes critères que précédemment.

Marginalisation locale

Une fois l'inférence locale accomplie, nous pouvons localement approximer la distribution sur les estimées locales à l'aide d'une distribution Gaussienne dont la matrice



(a) Sous-graphe visio-inertiel initial (b) Sous-graphe visio-inertiel marginalisé

La marginalisation produit un unique facteur n-aire ■ de pose absolue (c.f. section 2.5.3) conjoint sur toutes les variables de pose de $\mathcal{T}$. Il encode une vraisemblance $p(\mathcal{T}|\mathcal{Z}_{\text{VI}}) = \mathcal{N}\left(\boldsymbol{\xi}_{\hat{\mathcal{T}}}(\mathcal{T}); \mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\hat{\mathcal{T}}}|\mathcal{Z}_{\text{VI}}}\right)$ où $\boldsymbol{\xi}_{\hat{\mathcal{T}}}$ désigne le vecteur qui concatène les résidus de pose absolue, et $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\hat{\mathcal{T}}}|\mathcal{Z}_{\text{VI}}}$ est la matrice de covariance associée. On remarque que ce facteur ne porte pas sur la première pose T_{LB_0} du sous-graphe extrait, qui était fixée lors de l'optimisation visio-inertielle locale et ne fait donc pas partie des variables optimisées.

FIGURE 5.7 – Processus de marginalisation dans CVIP

d'information est la matrice d'information de Fisher évaluée en cette estimée :

$$p(\Theta|\mathcal{Z}_{\text{VI}}) = \mathcal{N}\left(\boldsymbol{\xi}_{\hat{\Theta}}(\Theta); \mathbf{0}, \left[\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\text{VI}}}(\hat{\Theta})\right]^{-1}\right) \tag{5.11}$$

où $\boldsymbol{\xi}_{\hat{\Theta}}(\Theta)$ est un vecteur de résidus et sa matrice d'information est calculée comme suit :

$$\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\text{VI}}}(\hat{\Theta}) = \sum_{z\in\mathcal{Z}_{\text{VI}}} \left[\boldsymbol{J}^{\boldsymbol{\xi}_z}_{\Theta}(\hat{\Theta})\right]^{\top} \cdot \boldsymbol{\Sigma}^{-1}_{\boldsymbol{\xi}_z} \cdot \left[\boldsymbol{J}^{\boldsymbol{\xi}_z}_{\Theta}(\hat{\Theta})\right] \tag{5.12}$$

et $\boldsymbol{J}^{\boldsymbol{\xi}_z}_{\Theta}(\hat{\Theta})$ est la matrice Jacobienne du résidu $\boldsymbol{\xi}_z$ par rapport aux variables Θ et évaluée en l'estimée locale $\hat{\Theta}$. L'étape de marginalisation consiste à projeter l'information sur les seules variables de pose $\mathcal{T} \subset \Theta$:

$$p(\mathcal{T}|\mathcal{Z}_{\text{VI}}) = \int p(\mathcal{T}, \mathcal{E}, \mathcal{L}|\mathcal{Z}_{\text{VI}})\, d\mathcal{E}\, d\mathcal{L} = \mathcal{N}\left(\boldsymbol{\xi}_{\hat{\mathcal{T}}}(\mathcal{T}); \mathbf{0}, \left[\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_{\text{VI}}}(\hat{\mathcal{T}})\right]^{-1}\right) \tag{5.13}$$

où la matrice d'information marginalisée se calcule comme le complément de Schur des variables marginalisées dans la matrice d'information complète :

$$\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_{\text{VI}}}(\hat{\mathcal{T}}) = \left[\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\text{VI}}}(\hat{\Theta})\right]_{\mathcal{T}\mathcal{T}} - \left[\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\text{VI}}}(\hat{\Theta})\right]_{\mathcal{T}M} \cdot \left[\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\text{VI}}}(\hat{\Theta})\right]^{-1}_{MM} \cdot \left[\boldsymbol{\mathcal{I}}^{\Theta}_{\mathcal{Z}_{\text{VI}}}(\hat{\Theta})\right]^{\top}_{\mathcal{T}M} \tag{5.14}$$

Dans l'équation précédente, les indices M renvoient aux indices des lignes et des colonnes associées aux variables marginalisées. Le vecteur des résidus $\boldsymbol{\xi}_{\hat{\mathcal{T}}}(\mathcal{T})$ concatène alors uni-

quement des résidus de poses relatives. Cette distribution est équivalente à un facteur n-aire $z_{\mathcal{T}}$ tel que $p(z_{\mathcal{T}}|\mathcal{T}) \triangleq p(\mathcal{T}|\mathcal{Z}_{\text{VI}})$ (c.f. Figure 5.7).

Éparsification consistante

La distribution marginale $p(\mathcal{T}|\mathcal{Z}_{\text{VI}})$ condense toute l'information du sous-graphe initial, mais elle est trop complexe pour être communiquée telle quelle. Nous utilisons alors des techniques d'éparsification afin de l'approximer de façon consistante par une distribution éparse $p(\mathcal{T}|\mathcal{Z}_S)$ qui serait engendrée par un jeu $\mathcal{Z}_S$ de facteurs virtuels Gaussiens de poses relatives (c.f. section 2.5.3) entre les images-clés successives. Nous privilégions une topologie liant les poses des images-clés consécutives de façon à considérer les paires de variables qui partagent le plus d'information mutuelle. Ces facteurs virtuels induisent la vraisemblance suivante vis-à-vis des variables de pose :

$$p(\mathcal{Z}_S|\mathcal{T}) = \mathcal{N}\left(\boldsymbol{\xi}_{\mathcal{Z}_S}(\mathcal{T}); \mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_S}\right) = \prod_{z\in\mathcal{Z}_S} p(z|\mathcal{T}) = \prod_{z\in\mathcal{Z}_S} \mathcal{N}\left(\boldsymbol{\xi}_z(\mathcal{T}); \mathbf{0}, \boldsymbol{\Lambda}^{-1}_{\boldsymbol{\xi}_z|z}\right) \tag{5.15}$$

où $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_S}$ désigne la matrice de covariance diagonale par blocs et donc chaque bloc diagonal correspond à la matrice de covariance d'un résidu de mesure $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z} = \boldsymbol{\Lambda}^{-1}_{\boldsymbol{\xi}_z}$. Les facteurs correspondants sont choisis de telles sortes qu'ils donnent la même estimée MLE :

$$\hat{\mathcal{T}} = \underset{\mathcal{T}\in\mathcal{H}_{\mathcal{T}}}{\arg\max}\ p(\mathcal{Z}_S|\mathcal{T}) = \underset{\mathcal{T}\in\mathcal{H}_{\mathcal{T}}}{\arg\max}\ p(\mathcal{Z}_{\text{VI}}|\mathcal{T}) \tag{5.16}$$

et dont la distribution est localement approximée par la distribution Gaussienne :

$$p(\mathcal{T}|\mathcal{Z}_S) = \mathcal{N}\left(\boldsymbol{\xi}_{\hat{\mathcal{T}}}(\mathcal{T}); \mathbf{0}, \left[\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_S}(\hat{\mathcal{T}})\right]^{-1}\right) \tag{5.17}$$

où $\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_S}(\hat{\mathcal{T}})$ est la matrice d'information de Fisher résultant des facteurs virtuels :

$$\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_S}(\hat{\mathcal{T}}) = \sum_{z\in\mathcal{Z}_S} \left[\boldsymbol{J}^{\boldsymbol{\xi}_z}_{\mathcal{T}}(\hat{\mathcal{T}})\right]^{\top} \cdot \boldsymbol{\Lambda}_{\boldsymbol{\xi}_z|z} \cdot \left[\boldsymbol{J}^{\boldsymbol{\xi}_z}_{\mathcal{T}}(\hat{\mathcal{T}})\right] = \left[\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}_S}}_{\mathcal{T}}(\hat{\mathcal{T}})\right]^{\top} \cdot \boldsymbol{\Lambda}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_S} \cdot \left[\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}_S}}_{\mathcal{T}}(\hat{\mathcal{T}})\right] \tag{5.18}$$

La matrice $\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}_S}}_{\mathcal{T}}(\hat{\mathcal{T}})$ désigne la matrice Jacobienne obtenue en concaténant les matrices Jacobiennes des facteurs virtuels de pose relative [6]. L'enjeu du processus d'éparsification est de calculer les matrices d'informations $\boldsymbol{\Lambda}_{\boldsymbol{\xi}_z|z}$ des facteurs virtuels $z \in \mathcal{Z}_S$ afin de

6. Cette matrice Jacobienne est construite à partir des matrices Jacobiennes de pose relative définies dans l'Annexe C

minimiser la perte d'information par rapport à la distribution initiale sous contrainte de consistance. Formellement, cela revient à résoudre le problème d'optimisation suivant :

$$\left\{\boldsymbol{\Sigma}^{*}_{\boldsymbol{\xi}_z|z}\right\}_{z\in\mathcal{Z}_S} = \underset{\left\{\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z|z}\right\}_{z\in\mathcal{Z}_s}}{\arg\min} \quad \mathcal{D}_{\mathrm{KL}}\left(p(\mathcal{T}|\mathcal{Z}_{\mathrm{VI}}), p(\mathcal{T}|\mathcal{Z}_S)\right) \tag{5.19a}$$

$$\text{tel que } \boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_S}(\hat{\mathcal{T}}) \leq \boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\mathcal{T}}) \tag{5.19b}$$

Le problème relaxé (5.19a) définit un problème convexe dont (Mazuran et al. 2014) ont montré qu'il admet une solution analytique dans le cas d'une topologie inversible (i.e. si $\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}_S}}_{\mathcal{T}}(\hat{\mathcal{T}})$ est inversible), ce qui est le cas ici par construction. La projection de la matrice de covariance dense sur le sous-espace associée aux facteurs virtuels donne :

$$\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}} = \left[\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}_S}}_{\mathcal{T}}(\hat{\mathcal{T}})\right] \cdot \left[\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\mathcal{T}})\right]^{-1} \cdot \left[\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}_S}}_{\mathcal{T}}(\hat{\mathcal{T}})\right]^{\top} \tag{5.20}$$

(Mazuran et al. 2014) montrent alors que la solution du problème relaxé de la contrainte de consistance affecte à chaque facteur virtuel $z \in \mathcal{Z}_S$, une matrice de covariance obtenue en marginalisant les autres facteurs dans la matrice $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}$. Ainsi, la matrice d'information $\boldsymbol{\Lambda}_{\boldsymbol{\xi}_z|z}$ correspond à l'inverse du bloc diagonal associé à son résidu $\boldsymbol{\xi}_z$ dans $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}$.

On doit finalement imposer la contrainte de consistance (5.19b) à partir de la solution du problème relaxé. Nous utilisons pour cela une méthode sous-optimale opérant directement sur le spectre des matrices d'information obtenues. Tout d'abord, nous reformulons la contrainte de consistance de façon équivalente dans l'espace des facteurs virtuels sous une forme covariationnelle :

$$\underbrace{\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}_S}}_{\mathcal{T}}(\hat{\mathcal{T}})^{\top} \cdot \boldsymbol{\Lambda}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_S} \cdot \boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}_S}}_{\mathcal{T}}(\hat{\mathcal{T}})}_{=\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_S}(\hat{\mathcal{T}})} \leq \boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\mathcal{T}})$$

$$\Leftrightarrow \underbrace{\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}_S}}_{\mathcal{T}}(\hat{\mathcal{T}}) \cdot \left[\boldsymbol{\mathcal{I}}^{\mathcal{T}}_{\mathcal{Z}_{\mathrm{VI}}}(\hat{\mathcal{T}})\right]^{-1} \cdot \boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}_S}}_{\mathcal{T}}(\hat{\mathcal{T}})^{\top}}_{=\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}} \leq \boldsymbol{\Lambda}^{-1}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_S} \triangleq \boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_S} \tag{5.21}$$

Une condition suffisante pour assurer cette contrainte est que la plus petite valeur propre de $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_S}$ excède la plus grande valeur propre de $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}$. La plus grande valeur propre $\lambda_{\max}(\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}})$ de $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}$ peut être calculée efficacement à l'aide d'algorithmes spécialisés tels que l'algorithme de Lanczos (Lanczos 1950), basé sur la méthode des puissances itérées. De plus, la matrice, $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_S}$ étant diagonale par blocs, ses valeurs propres sont

les valeurs propres de ses blocs diagonaux. On peut donc traiter chaque facteur $z \in \mathcal{Z}_S$ indépendamment :

$$\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z|z} = \boldsymbol{V} \cdot \mathrm{diag}(\boldsymbol{\lambda}_z) \cdot \boldsymbol{V}^\top \tag{5.22}$$

avec $\boldsymbol{V} \in \mathbb{SO}_6$. Une première solution consiste alors à corriger le vecteur des valeurs propres tel que $\boldsymbol{\lambda}_z \leftarrow \max\left(\boldsymbol{\lambda}_z, \lambda_{\max}(\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}).\mathbf{1}_n\right)$. Cependant, on perd alors les informations sur les incertitudes relatives entre les composantes du résidu. C'est pourquoi nous optons pour une solution plus conservative mais qui préserve cette information. Elle consiste à dilater le vecteur des valeurs propres d'un facteur α_z^* tel que :

$$\alpha_z^* = \underset{\alpha>1}{\arg\min}\ \alpha \text{ tel que } \min(\alpha_z.\boldsymbol{\lambda}_z) \geq \lambda_{\max}(\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_s}|\mathcal{Z}_{\mathrm{VI}}}) + \varepsilon \tag{5.23}$$

où $\varepsilon << 1$. La nouvelle matrice de covariance du facteur est alors calculée en dilatant la matrice originale du facteur α_z^* : $\boldsymbol{\Sigma}^*_{\boldsymbol{\xi}_z|z} = \alpha_z^*.\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z|z}$.

Dans le processus précédent, la valeur propre $\lambda_{\max}(\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}})$ est tributaire des valeurs numériques des coefficients de la matrice de covariance $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}$. Or celle-ci porte sur les résidus de translation et d'orientation relative. Ceux-ci s'expriment respectivement en mètres et en radians, et leurs ordres de grandeur numériques diffèrent. Afin de prévenir une éparsification trop conservative avec une valeur de $\lambda_{\max}(\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}})$ trop élevée, nous normalisons préalablement tous les coefficients diagonaux de $\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}$ en opérant une reparamétrisation d'échelle pour chaque facteur. Nous construisons donc la matrice jacobienne $\boldsymbol{J}$ correspondant à ce changement d'unité :

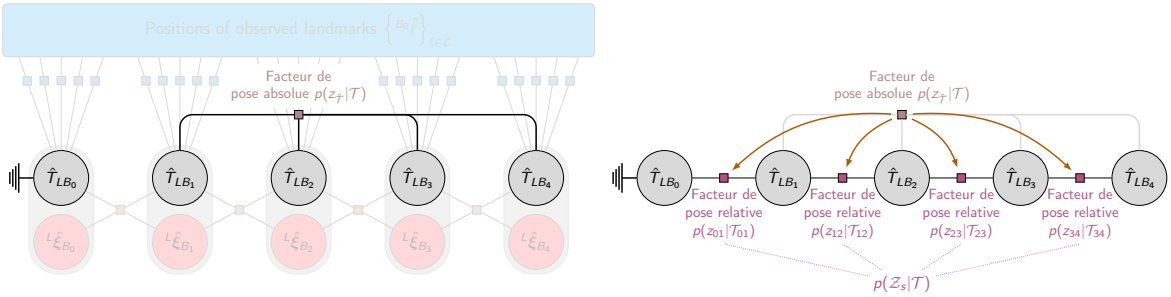
$$\boldsymbol{J} = \mathrm{diag}(\boldsymbol{s}) \text{ où } s_i = \sqrt{\frac{\min(\mathrm{diag}(\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}))}{\boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}(i,i)}} \tag{5.24}$$

Fondamentalement, cette standardisation revient à opérer sur un équivalent de matrice de corrélation au lieu d'une matrice de covariance. Nous appliquons ensuite la même procédure de façon à assurer la contrainte, de façon agnostique aux unités utilisées :

$$\underbrace{\boldsymbol{J} \cdot \boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}} \cdot \boldsymbol{J}^\top}_{=\boldsymbol{\Sigma}^{\mathrm{normed}}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_{\mathrm{VI}}}} \leq \underbrace{\boldsymbol{J} \cdot \boldsymbol{\Sigma}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_S} \cdot \boldsymbol{J}^\top}_{=\boldsymbol{\Sigma}^{\mathrm{normed}}_{\boldsymbol{\xi}_{\mathcal{Z}_S}|\mathcal{Z}_S}} \tag{5.25}$$

Les matrices de covariance obtenues selon l'algorithme précédent sont enfin restaurées

dans les unités standards. Ceci achève le calcul du facteur. Le graphe de facteurs résultant du processus d'éparsification est représenté par la Figure 5.8.



(a) Sous-graphe de facteurs marginalisé (b) Sous-graphe de facteurs éparsifié

L'éparsification approxime la distribution dense $p(\mathcal{T}|\mathcal{Z}_{\text{VI}})$, équivalente à celle engendrée par un unique facteur n-aire de poses absolues ■, par celle engendrée par un jeu de facteurs binaires ■ de poses relatives.

FIGURE 5.8 – Processus d'éparsification dans CVIP

Sélection de l'information visuelle

La seconde étape de la synthèse du paquet visio-inertiel consiste à ajouter l'information visuelle qui servira à la détection des correspondances inter-robot. Cette information est communiquée de façon localisée sur un sous-ensemble des images-clés contenues dans le sous-graphe considéré. On les sélectionne à l'aide de l'algorithme présenté dans la section 5.3.4. Pour chacune des images-clés sélectionnées, nous transmettons également un sous-ensemble de ses points d'intérêts et descripteurs, que nous sélectionnons à l'aide de l'algorithme de sélection de points caractéristiques présenté dans la section 5.3.4. Notons que contrairement à SVIP, nous ne communiquons pas d'information structurelle.

Finalisation et sauvegarde du paquet calculé

Le paquet final, représenté par la Figure 5.6, contient une liste ordonnée de facteurs de pose relative entre les images-clés successives, une liste de coordonnées de points caractéristiques et leur descripteurs associés sur un sous-ensemble d'images-clés. Lors du premier échange avec un robot, on communique également les paramètres du modèle de caméra associé (coefficients intrinsèques, de distorsion et de transformation extrinsèque IMU-caméra). Une fois le paquet finalisé, il est diffusé aux autres robots tandis qu'un index lui est attribué dans l'historique de communications. Les facteurs de pose relative calculés durant la synthèse sont sauvegardés dans la carte, afin de pouvoir être relayé à d'autres robots si besoin, et exploités pour l'inférence globale.

5.5.4 Intégration d'un paquet

Reconstruction du paquet

Lorsqu'un paquet est reçu d'un autre robot, il est d'abord désérialisé et reconstruit dans la carte, en initialisant les poses relatives entre les images-clés successives qui annulent les résidus des facteurs communiqués.

Détection des correspondances inter-robot rétrospectives

L'étape suivante implique d'utiliser les informations visuelles contenues dans le paquet reçu afin de détecter les correspondances inter-robot rétrospectives avec la trajectoire hôte. Pour cela, chaque image-clé contenue dans le paquet et associée à des informations visuelles est comparée à la base de données visuelle de la trajectoire hôte. Dans le cas où des fermetures de boucles sont détectées, alors une requête est formulée afin d'obtenir davantage d'informations visuelles sur les images-clés voisines selon la procédure décrite dans la section 5.3.2. Notons que contrairement à la méthode précédente, l'information structurelle nécessaire à la caractérisation des fermetures de boucles n'est pas échangée entre les robots, il s'agit du nuage d'amers estimé par le robot à partir de ses observations.

Les informations visuelles reçues sont également utilisées pour détecter les correspondances inter-robot avec les nouvelles images-clés introduites par le robot hôte. Il s'agit des correspondances avec des portions de trajectoire du robot hôte introduites postérieurement à l'intégration du paquet. Pour ce faire, nous utilisons une base visuelle entièrement dédiée à la détection des correspondances immédiates. Pour chaque nouvelle image-clé introduite sur la trajectoire hôte, ses informations visuelles sont insérées dans cette base visuelles, jusqu'à atteindre à un certain nombre d'images-clés décidé par l'utilisateur. Lorsque cette capacité maximale est atteinte, alors toutes les images-clés communiquées de façon standard (c'est-à-dire transmises avec les paquets visio-inertielles) sont testées contre la base visuelle. Si une correspondance inter-robot est détectée, alors toutes les images-clés voisines connues sont également testées, et préalablement requises au robot associé si elles ne sont pas connues. Une fois tous les tests effectués, alors la base visuelle dédiée à la détection des correspondances immédiates est réinitialisée et translatée. Cette procédure saccadée est nécessaire pour permettre les vérifications topologiques telles que décrites dans la section 5.3.3, et qui nécessitent que la base visuelle contienne plusieurs images-clés consécutives sur une portion de sous-trajectoire significative.

Inférence globale multi-robots

L'inférence multi-robots consiste en l'optimisation du graphe de poses en incluant les facteurs de pose relatives reçus et calculés lors des synthèses des paquets, ainsi que les facteurs de pose relative dérivés des correspondances inter-robot. La Figure 5.9 montre la structure du graphe de facteurs optimisé. Comme précédemment, seules les trajectoires préalablement ancrées avec la trajectoire hôte sont considérées pour l'inférence jointe. La jauge est fixée en figeant la première pose de la trajectoire du robot hôte (symbolisé par le symbole ⊣ dans la figure).

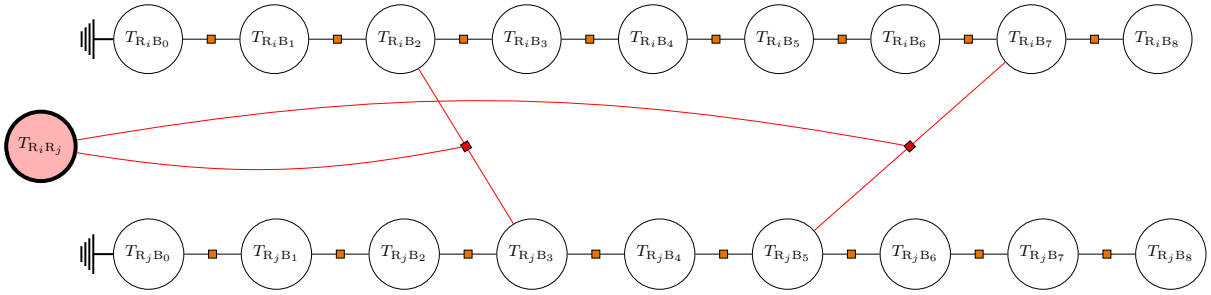


FIGURE 5.9 – Graphe de poses utilisé pour l'inférence multi-robots par CVIP

5.6 Échange de paquets visio-inertiels élagués

Les techniques d'éparsification permettent de calculer une approximation simple de distributions complexes. Néanmoins, elles induisent inéluctablement des pertes d'informations et pèchent éventuellement par conservativité. Dans cette section, nous investiguons une solution alternative qui consiste non plus à approximer la distribution dense initiale par celle qui serait engendrée par un ensemble de facteurs virtuels mais celle qui découlerait d'un sous-ensemble des facteurs visio-inertiels originaux.

5.6.1 Travaux associés

Les techniques d'élagage sélectionnent un sous-ensemble des variables et/ou des facteurs de façon à compacter la représentation originale sans compromettre excessivement la qualité de l'estimation qui en découle. Elles relèvent fondamentalement d'un problème

de recherche d'un sous-ensemble optimal qui se formule comme suit :

$$\mathcal{Z}_S^* = \underset{\mathcal{Z}_S \subseteq \mathcal{Z}}{\arg\max}\ u(\Theta; \mathcal{Z}_S) \text{ tel que } c(\mathcal{Z}_S) \leq 0 \tag{5.26}$$

où $\mathcal{Z}$ est l'ensemble des mesures originales. La fonction u mesure l'utilité d'une sélection donnée vis-à-vis de l'estimation des variables d'intérêt Θ, tandis que $c(\mathcal{Z}) \leq 0$ recouvre un ensemble de contraintes qui régularisent la sélection. Bien entendu, le problème (5.26) admet une formulation duale qui minimise une pénalité p au lieu de maximiser une utilité u. Il s'agit néanmoins d'un problème NP-difficile (Karp 1972). Sa difficulté découle en particulier de la combinatoire exponentielle de $\mathcal{P}(\mathcal{Z})$, et de la complexité à envisager la synergie des contributions des éléments sélectionnés à l'utilité globale. Le leitmotiv holistique selon lequel l'utilité globale ne se réduit pas à une somme d'utilités individuelles s'applique. En particulier, deux éléments individuellement utiles peuvent s'avérer mutuellement redondants.

La première étape de la conception d'une méthode d'élagage consiste à définir la fonction d'utilité utilisée. Idéalement, il s'agira d'une fonction d'utilité jointe qui privilégiera les éléments les plus informatifs tout en pénalisant la redondance mutuelle entre ces éléments. En pratique cependant, on approxime souvent ces fonctions d'utilité comme une somme de contributions individuelles. Si la théorie de l'information fournit un cadre idéal pour quantifier les utilités, les grandeurs dérivées de l'entropie de Shannon s'avèrent complexes à calculer, si bien que des métriques approchées en sont dérivées. Il est également possible de définir des utilités sur la base de critères heuristiques ou statistiques. Une fois les utilités définies, il faut déterminer la méthode d'optimisation. Quand les approches optimales – i.e. qui optimisent la fonction de coût définie telle quelle – ne sont pas applicables, une alternative consiste à concevoir des heuristiques qui procèdent par une succession de choix localement optimaux. En particulier, elles alternent entre l'évaluation des utilités individuelles et la sélection des éléments les plus utiles. Les applications des techniques d'élagage en SLAM sont multiples. La première concerne l'identification et l'élimination des redondances. La seconde consiste à sélectionner des amers observés qui restituent suffisamment l'information sur la pose courante ou la trajectoire.

La formulation des fonctions d'utilité et des contraintes

Les fonctions d'utilité doivent idéalement promouvoir les éléments qui contribuent fortement à l'estimation des variables d'intérêts, tout en pénalisant la sélection conjointe

d'éléments mutuellement redondants. La quantification des degrés de redondance a été particulièrement investiguée par les méthodes d'élagage des graphes de poses. C'est notamment le cas dans le contexte du SLAM LiDAR où chaque pose est associée à un scan laser dont le traitement et la mémorisation sont coûteux. L'information mutuelle entre deux variables s'est imposée comme une métrique particulièrement pertinente pour modéliser les redondances. Intuitivement, elle quantifie l'information (i.e. la réduction d'entropie) que l'on obtient sur l'une dès lors qu'on connaît l'autre. (Ila et al. 2009) en dérivent une distance qu'ils utilisent pour repérer les poses trop "proches" des autres. (Kretzschmar et al. 2011) exploitent également l'information mutuelle entre chaque scan et la carte – qu'ils approximent à l'aide d'un modèle de grille d'occupation – afin d'identifier puis d'éliminer les scans redondants. (Schmuck et al. 2019b) quantifient la redondance d'une image-clé en quantifiant son information mutuelle moyenne avec les images-clés de sa couverture de Markov dans le graphe de covisibilité, c'est-à-dire l'information moyenne que sa connaissance apporte sur les poses de ses images-clés voisines. Cependant, cette métrique s'avère trop coûteuse à calculer car elle requiert l'extraction et l'inversion de larges matrices de covariance. Les auteurs y substituent donc une métrique basée sur la capacité moyenne à la retriangulation des amers covisibles, évaluée à partir des nombres d'observations des amers observés par ladite image-clé. (Wang et al. 2013) se distinguent des méthodes précédentes en élaborant une métrique d'utilité à l'aide de la divergence de Kullback-Leibler, définie dans l'équation (5.8). Cette dernière permet de quantifier la perte d'information sur la carte occasionnée par l'élimination de scans. Cette métrique étant complexe à calculer, ils identifient alors les facteurs qui l'influent et en déduisent une heuristique de sélection des scans. Enfin, tirant parti de la théorie des graphes, (Khosoussi et al. 2014) adoptent une approche originale en reliant la fiabilité de l'estimée MLE à la structure du graphe de poses sous-jacent. Ils montrent notamment que celle-ci dépend du nombre d'arbres de recouvrements dans le graphe, et en dérivent une stratégie d'élagage des mesures, en quantifiant l'utilité de chacune comme le nombre d'arbres de recouvrement dont l'existence en dépend.

Le second volet d'une fonction d'utilité est de quantifier l'utilité d'un ensemble d'éléments pour l'estimation de variables d'intérêt. La première famille des fonctions d'utilité englobe les utilités jointes qui évaluent une sélection d'amers dans son ensemble. Par exemple, (Choudhary et al. 2015) définissent une fonction de pénalité globale qui formule un compromis entre la perte en précision d'estimation (quantifiée par l'information apportée par l'amer le moins informatif) et l'empreinte mémoire de la sélection. Une autre

approche consiste à optimiser des métriques sur la matrice d'information associée à la sélection par rapport aux variables d'intérêt Θ. (Lerner et al. 2007) minimisent la trace de la matrice de covariance engendrée par la sélection. Cette approche est reprise par (Beinhofer et al. 2013) ; ils ont proposé une fonction d'utilité qui évalue la réduction moyenne de la trace des matrices de covariance des poses le long de la trajectoire pour la sélection d'amers envisagée. Dans une démarche similaire, (Mu et al. 2015) minimisent le log-déterminant de la matrice de covariance engendrée et préalablement pondérée de façon à diriger préférentiellement l'information sur l'estimation des variables essentielles à une tâche spécifique. Cette idée est notamment reprise par (Beinhofer et al. 2013) et (Zhao et al. 2018). Une autre manière d'évaluer une sélection d'amers est de prêter attention aux contraintes visuelles qu'elles induisent sur les images-clés. S'inspirant de la méthode de compression introduite par (Soo Park et al. 2013), (Dymczyk et al. 2015a) ont formulé une fonction d'utilité dont la partie linéaire privilégie la sélection des amers les plus observés, tandis que sa partie quadratique pénalise la sélection d'amers redondants en ce sens qu'ils se projettent en moyenne proches l'un de l'autre dans les images-clés où ils sont covisibles. Cette fonction d'utilité est associée à une contrainte de visibilité qui astreint à garantir un nombre minimum d'observations sur chaque image-clé.

En pratique, et contrairement à l'approche holistique, l'approche la plus commode consiste à décomposer une utilité globale comme une somme d'utilités individuelles i.e. $u(\Theta; \mathcal{L}_S) = \sum_{l \in \mathcal{L}_S} u(\Theta; l)$. Notons que ces métriques individuelles sont rarement définies dans l'absolu et dépendent souvent de la sélection actuelle ; elles doivent alors être actualisées périodiquement lors d'un processus de sélection heuristique. L'approche classique consiste à classer les amers selon leur gain d'information (Zhang et al. 2005) i.e. la réduction d'entropie qu'ils induisent sur des variables d'intérêt Θ. (Hochdorfer et al. 2009) quantifient la contribution de chaque amer à la capacité du robot à se localiser dans l'environnement de travail. Ils regroupent les amers selon leur covisibilité puis retiennent les amers aux covariances les plus faibles au sein de ces groupes. Des métriques telles que la trace, le conditionnement (Cheein et al. 2009) ou le log-déterminant des matrices de covariance des amers peuvent également servir de métriques d'utilité. Il est également possible de paramétrer des fonctions d'utilité à l'aide de régressions statistiques. (Dymczyk et al. 2016) apprennent une fonction d'utilité à l'aide d'un modèle de régression logistique afin de prédire l'utilité d'un amer via une combinaison linéaire de métriques faiblement corrélées et dont les coefficients sont appris. Ces métriques incluent notamment le nombre d'observations, la distance parcourue tout en observant l'amer en question, son erreur

de re-projection moyenne ou encore un score quantifiant la qualité de ses descripteurs. (Strasdat et al. 2009) apprennent par renforcement une politique de sélection d'amers pour la navigation. (Bürki et al. 2016) réalisent un apprentissage continu en ligne en se basant sur l'apparence des amers ainsi que sur des métriques de covisibilité. D'autres fonctions d'utilité d'un amer incluent par exemple son nombre d'observateurs, ou encore le gain en couverture apporté aux images-clés vis-à-vis d'un objectif de couverture minimale.

Les méthodes de résolution heuristique

Une fois les métriques d'utilité définies, la seconde étape consiste à déterminer la sélection qui la maximise tout en respectant les contraintes associées. Une première approche consiste à utiliser des méthodes heuristiques qui procèdent en une succession de choix localement optimaux. Plus spécifiquement, elles alternent entre des étapes de ré-évaluation des éléments au regard de la sélection courante, et des étapes de sélection (resp. d'élimination) des éléments les plus (resp. les moins) utiles. Si les heuristiques s'accomodent particulièrement des fonctions d'utilité individuelles, elle peuvent également se déduire de fonctions d'utilité globale qui s'avèreraient trop complexes à optimiser. Ainsi, (Mu et al. 2015) sélectionnent itérativement l'observation z qui maximise la réduction de l'entropie $\Delta H(\Theta|z)$ sur un sous-ensemble de variables d'intérêt Θ. Pour cela, ils explicitent analytiquement une approximation de $\Delta H(\Theta|z)$. De même, (Choudhary et al. 2015) proposent une optimisation incrémentale par laquelle ils éliminent à chaque itération l'amer le moins informatif tant que cela fait décroire la fonction de coût qu'ils ont préalablement définie. Enfin, (Zhao et al. 2018) ont proposé une heuristique gloutonne stochastique (Mirzasoleiman et al. 2015) qui sélectionne à chaque itération l'amer qui accroît le plus le log-déterminant de la matrice d'information qui en résulte sur la pose courante.

Enfin, certaines techniques sélectionnent les amers uniquement sur la base de propriétés géométriques et de covisibilité. (Contreras et al. 2017) proposent une technique de compression de nuages d'amers en ligne : ils segmentent d'abord la trajectoire en se basant sur sa courbure, puis dans chaque segment, ils appliquent un *clustering* de type *k-means* sur le nuages d'amers, puis ne retiennent enfin que les amers les plus proches des centres des clusters ainsi calculés. Poursuivant les travaux de (Mühlfellner et al. 2016) qui ont introduit les cartes de résumé (*Summary Maps*), (Dymczyk et al. 2015b) proposent un algorithme itératif qui alterne entre des étapes d'évaluation des amers et des étapes d'échantillonnage. Ils introduisent différents scores basés sur les nombres d'observations des amers. À chaque itération, ils déciment l'amer le moins utile tout en vérifiant que

cela ne compromet pas le respect des contraintes de couverture des images-clés. Dans le contexte de la reconstruction par le mouvement,(Cao et al. 2014) utilisent une heuristique de type *Min-k-Cover* probabiliste afin d'assurer en espérance une couverture minimale sur toutes les images-clés, tout en maximisant la variance des descripteurs associés. (Choudhary et al. 2015) montrent d'ailleurs que l'heuristique Min-k-Cover permet d'obtenir des résultats comparables à leur propre méthode.

Les méthodes de résolution optimales

Contrairement aux méthodes heuristiques, les méthodes optimales s'attaquent au problème d'optimisation tel que défini par la fonction d'utilité et les contraintes considérées. En particulier, l'approche classique consiste à reformuler le problème introduit par l'équation (5.26) sous la forme d'un problème de programmation en nombres entiers. Ils associent alors à chaque élément une variable indicatrice x_i binaire qui vaut 1 s'il est sélectionné :

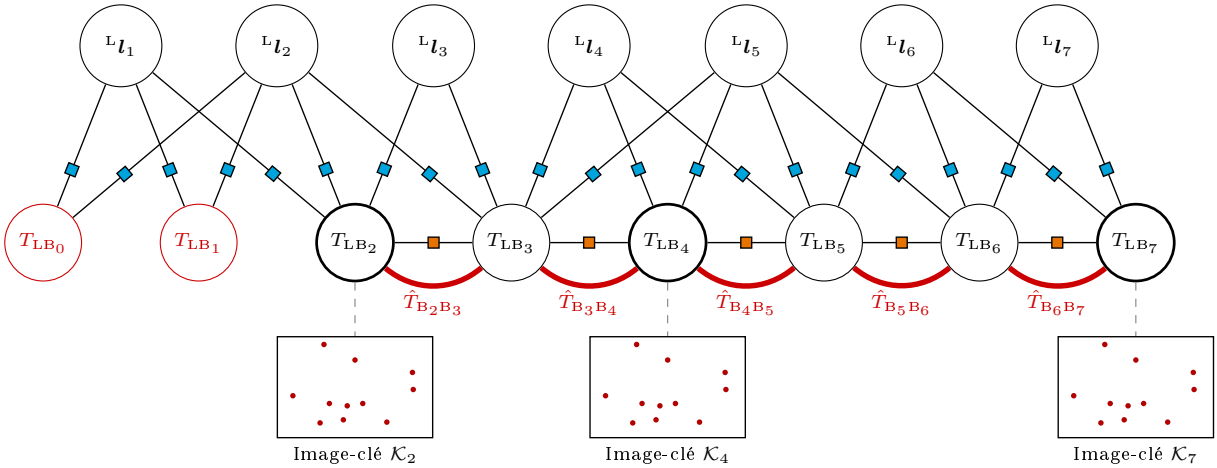
$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\{0,1\}^N}{\arg\min}\ u(\Theta;\boldsymbol{x}) \text{ tel que } c(\boldsymbol{x}) \leq 0 \tag{5.27}$$

Cette approche est adoptée par (Lerner et al. 2007) pour maximiser la trace de la matrice d'information engendrée par la sélection d'un sous-ensemble des amers. Plus tard, (Dymczyk et al. 2015a) formulent également un problème de programmation par entier quadratique dans sa version complète ou se restreignent à sa seule partie linéaire (la différence de temps de calcul entre les deux étant significative). En particulier, sa résolution est parallélisée sur des segments découpés le long de la trajectoire selon des techniques de *graph-cutting*. L'optimisation de la fonction d'utilité elle-même fait appel à des techniques de séparation et d'évaluation.

5.6.2 Vue d'ensemble de la méthode proposée

Les méthodes que nous proposons dans cette section présentent de nombreux points communs avec la méthode détaillée dans la section précédente, à l'exception que la synthèse des paquets repose sur des techniques d'élagage en lieu et place de la marginalisation et de l'éparsification. Le paquet calculé prend ici la forme d'un sous-graphe de facteurs visio-inertiels, aggrémenté d'informations visuelles communiquées sur un sous-ensemble des images-clés sélectionnées le long de la trajectoire. Les méthodes présentées dans cette section sont désignées sous l'acronyme PVIP pour *Pruned Visual-Inertial Packets*. Une

représentation des paquets communiqués est donnée par la Figure 5.10.



Le paquet comprend un graphe de facteurs visio-inertiels pour restituer les informations de localisation. Les symboles ■ représentent les facteurs inertiels tandis que ■ schématisent les facteurs visuels. Les variables ${}^{\mathrm{L}}\boldsymbol{l}_i$ sont les positions des amers, tandis que T_{LB_i} sont les poses attachées aux images-clés, où B est le référentiel inertiel, et L un référentiel local quelconque. Des informations visuelles sont localement communiquées sur une sélection d'images-clés. Les estimées de poses relatives sont également communiquées à des fins d'initialisation. Les variables de pose en rouge correspondent à un recouvrement avec le paquet précédent délimité par des critères de covisibilité afin de limiter les effets de bord.

FIGURE 5.10 – Morphologie d'un paquet PVIP.

5.6.3 Calcul d'un paquet

Le calcul d'un paquet comprend deux étapes principales. La première consiste sélectionner l'information visio-inertielle transmise, tandis que la seconde revient à sélectionner l'information visuelle qu'on y adjoint dans une démarche similaire à CVIP.

Extraction du sous-graphe de facteurs

La méthode d'extraction du sous-graphe de facteurs diffère quelque peu des méthodes précédentes. En effet, au sous-graphe correspondant à la nouvelle portion de la carte et de la trajectoire depuis le dernier paquet calculé, nous ajoutons également les images-clés antérieures qui observent au moins 10% des amers observés par sa première image-clé. Leurs amers sont également inclus dans les sélections qui suivent. Nous procédons ainsi afin de compenser les effets de bords lors de la sélection ultérieure des amers étant donné qu'elle repose sur des critères de covisibilité. Le sous-graphe extrait est représenté par la Figure 5.11.

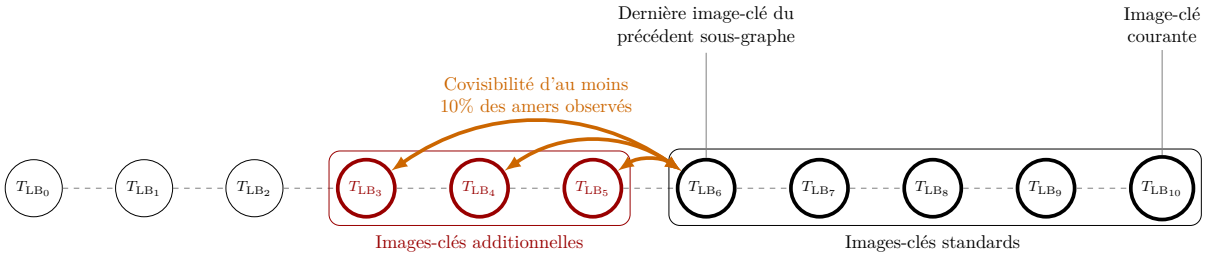


FIGURE 5.11 – Représentation du sous-graphe visio-inertiel extrait par PVIP

Contrairement aux méthodes précédentes, nous ne requérons pas que le sous-graphe de facteurs extrait soit optimisé localement, car seules les informations de covisibilité importeront dans la suite du processus. Cela dispense ainsi de l'une des étapes les plus coûteuses des méthodes précédentes.

Ajout de l'information inertielle

Dans la suite, nous retenons l'ensemble des facteurs inertiels qui n'ont pas encore été communiqués, c'est-à-dire entre les images-clés sélectionnées de façon standard comme représenté sur la Figure 5.11. Bien qu'il soit possible de communiquer des facteurs inertiels pré-intégrés tels que présentés dans la section 2.5.3, on préférera communiquer les mesures inertielles brutes. En effet, les facteurs pré-intégrés nécessitent de transmettre une quantité de données équivalente car ils communiquent également les matrices de covariance des pré-intégrations, les valeurs des biais en lesquels ces dernières ont été calculées, ainsi que les matrices Jacobiennes qui permettent leur correction au premier ordre pour des valeurs de biais différentes. Enfin, à des fins d'initialisation, nous communiquerons les estimées courantes des poses relatives entre les images-clés standards successives.

Sélection de l'information visuelle

L'étape essentielle de la synthèse du paquet est la sélection des amers. L'objectif est de maximiser l'information mutuelle induite entre les images-clés de la sous-trajectoire contenue dans le paquet. Cette étape pourrait se formuler comme précédemment dans le cadre de la théorie de l'information :

$$\mathcal{Z}_S^* = \underset{\mathcal{Z}_S \subseteq \mathcal{Z}_{\mathrm{VI}}}{\arg\min}\ \mathcal{D}_{\mathrm{KL}}\left(p(\mathcal{T}|\mathcal{Z}_{\mathrm{VI}}), p(\mathcal{T}|\mathcal{Z}_S)\right) \tag{5.28}$$

où $\mathcal{Z}_S$ désigne le sous-ensemble des facteurs visio-inertiels retenus de $\mathcal{Z}_{\text{VI}}$, et $\mathcal{T}$ est l'ensemble des poses de la sous-trajectoire couverte par le paquet. Si le problème formalise l'objectif pousuivi, la méthode que nous proposons ne dérive pas de cette formulation. Ici, nous avons fait le choix de préselectionner l'ensemble des facteurs inertiels. Concernant l'information visuelle, nous investiguons ici deux paradigmes de sélection d'amers. Le premier repose sur l'utilisation d'heuristiques de type Min-k-Cover tandis que la seconde relève de la programmation linéaire par entiers. Le problème de sélection de facteurs visuels devient un problème de sélection d'amers dont on retient les facteurs associés.

Sélection d'amers par résolution heuristique du problème Min-k-Cover. La première méthode est une solution heuristique au problème de la k-couverture minimale. Ce problème consiste à calculer le plus petit ensemble d'amers de telles sortes que chaque image-clé soit couverte par k observations. C'est un problème NP-complet dont la solution peut être approximée à l'aide d'une heuristique gloutonne. À chaque itération, le principe consiste à alors sélectionner l'amer qui couvre le plus d'images-clés partiellement couvertes. Lorsque deux amers sont à égalité, on sélectionne celui qui est associé au plus grand nombre d'observations afin de privilégier les amers les plus stables. L'amer sélectionné est alors retiré de l'ensemble des amers candidats tandis que les tables de correspondances, qui indiquent les couvertures atteintes sur les images-clés et les gains en couvertures de chaque amer, sont mises à jour. Chaque amer est sélectionné avec toutes ses observations. Cela peut néanmoins résulter en une couverture excédentaire sur certaines images-clés. Dans sa formulation simple, l'algorithme MKC ne permet cependant pas de tenir compte des éventuelles redondances entre les amers sélectionnés. C'est pourquoi, à chaque sélection d'un nouvel amer, nous retirons également de l'ensemble des candidats les amers qui lui sont mutuellement redondants. Deux techniques ont été envisagées. La première consistait à éliminer les amers proches situés dans un voisinage spatiale de l'amer sélectionné. On les repérait alors à l'aide d'un arbre KD construit au moment de l'extraction du sous-graphe. Cette méthode a été abandonnée au profit d'une seconde technique qui consiste à préalablement quantifier un degré de redondance sur chaque paire d'amers covisibles. Étant donné deux amers l_i et l_j, nous calculons la métrique suivante, inspirée des travaux de (Dymczyk et al. 2015a) :

$$R_{ij} = \frac{1}{|\mathcal{K}_{\text{ij}}|} \sum_{k \in \mathcal{K}_{\text{ij}}} \frac{\min\left(d_{\min}, d(\boldsymbol{p}_{k,i}, \boldsymbol{p}_{k,j})\right)}{d_{\min}} \in [0,1] \tag{5.29}$$

où $\boldsymbol{p}_{k,i}$ et $\boldsymbol{p}_{k,j}$ désignent leurs projections respectives sur l'image-clé k, et $\mathcal{K}_{\text{ij}}$ désigne l'ensemble des images-clés sur lesquelles les deux amers sont covisibles. Il s'agit de la distance moyenne de projection des deux amers sur les images-clés dans lesquelles ils sont conjointement observés, saturée et normalisée par une distance seuil $d_{\min}$. Lorsque R_{ij} est proche de 1, cela signifie que les deux amers se projettent en moyenne suffisamment loin l'un de l'autre par rapport à la distance d'alerte $d_{\min}$. Dans le cas contraire, lorsque R_{ij} est proche de 0, alors les deux amers peuvent être considérés comme mutuellement redondants. Ces métriques sont calculées au moment de l'extraction du sous-graphe, en parcourant chaque image-clé puis en considérant toutes les paires d'amers parmi ceux qu'elle observe, et enfin en incrémentant les entrées de la matrice $\boldsymbol{R} = [R_{ij}]_{i,j\in\mathcal{L}}$ ainsi construite. Il s'agit d'une matrice éparse car la plupart de ses entrées sont nulles en pratique. Pour chaque amer sélectionné, il suffit dès lors de parcourir les coefficients non-nuls de la matrice selon la ligne ou la colonne qui lui correspondent, et de supprimer tous les amers qui sont jugés trop proches. Cette technique garantit que les amers sélectionnés ne sont pas redondants les uns par rapport aux autres pour le problème de la localisation. L'algorithme cesse lorsque chaque image-clé a atteint son critère de couverture ou qu'il ne reste plus aucun amer à sélectionner. L'objectif de couverture de chaque image-clé a bien entendu été défini par rapport à son nombre d'observations afin d'éviter les contraintes infaisables. Nous désignerons cette variante par l'acronyme PVIP-MKC.

Sélection d'amers par programmation linéaire en nombres entiers. Nous proposons également une seconde technique d'élagage basée sur des techniques de programmation linéaire en nombres entiers. Inspirés par (Dymczyk et al. 2015a), nous proposons de résoudre le problème d'optimisation suivant :

$$\mathbb{1}^*_{\mathcal{L}} = \underset{\mathbb{1}_{\mathcal{L}}\in\{0,1\}^N}{\arg\min}\ \boldsymbol{q}^\top \cdot \mathbb{1}_{\mathcal{L}} \quad \text{tel que} \quad \boldsymbol{A}\cdot\mathbb{1}_{\mathcal{L}} \geq \min(k\cdot\mathbf{1}_N, \boldsymbol{o}) \tag{5.30}$$

où $\mathbb{1}_{\mathcal{L}}$ est un vecteur binaire de variables indicatrices. Si le i$^{\text{ème}}$ coefficient de $\mathbb{1}_{\mathcal{L}}$ vaut 1, alors l'amer correspondant est sélectionné, N étant le nombre d'amers inclus dans le problème. Le vecteur $\boldsymbol{q} \in \mathbb{R}^N$ est tel que q_i est l'inverse du nombre d'observations de l'amer i, afin de privilégier la sélection des amers les plus observés. La matrice $\boldsymbol{A}$ est une matrice de visibilité, telle que A_{ij} vaut 1 si l'image-clé i observe l'amer j, et $A_{ij} = 0$ sinon. Enfin, $\boldsymbol{o} \in \mathbb{R}^N$ tel que o_i est le nombre d'amers observés par l'image-clé i. Contrairement à la méthode de (Dymczyk et al. 2015a), nous n'appliquons pas de partionnement car

nous travaillons déjà sur un sous-graphe dont la taille est préalablement bornée. Cette méthode sera désignée par l'acronyme PVIP-ILP. (Dymczyk et al. 2015a) avaient également proposé une formulation quadratique qui afin de tenir compte des redondances :

$$\mathbb{1}_{\mathcal{L}}^{*} = \underset{\mathbb{1}_{\mathcal{L}} \in \{0,1\}^{N}}{\arg\min}\ \mathbb{1}_{\mathcal{L}}^{\top} \cdot \boldsymbol{Q} \cdot \mathbb{1}_{\mathcal{L}} + \boldsymbol{q}^{\top} \cdot \mathbb{1}_{\mathcal{L}} \quad \text{tel que} \quad \boldsymbol{A} \cdot \mathbb{1}_{\mathcal{L}} \geq \min(k \cdot \mathbf{1}_{N}, \boldsymbol{o}) \tag{5.31}$$

où $\boldsymbol{Q}$ est une matrice qui quantifie la redondance entre deux amers. Cependant, les tests préliminaires que nous avons menés en utilisant la librairie `Gurobi` (GUROBI 2014) ont montré que les temps d'exécution sont rédhibitoires pour une utilisation en temps-réel.

Ajout des informations visuelles

La dernière étape consiste à compléter le paquet avec les informations visuelles qui serviront à la détection des correspondances inter-robot. Comme pour la méthode précédente, on utilise l'algorithme présenté dans la section 5.3.4 afin de sélectionner un sous-ensemble des images-clés le long de la sous-trajectoire, et l'algorithme introduit dans la section 5.3.4 pour sélectionner les points caractéristiques et les descripteurs communiqués sur chacune des images sélectionnées. En revanche, ces informations ne sont prélevées que sur les nouvelles images-clés, non communiquées dans le paquet précédent.

Finaliation et sauvegarde du paquet calculé

Un paquet finalisé contient les facteurs inertiels entre les nouvelles images-clés successives ainsi que les estimées courantes de leurs poses relatives, et pour chaque image-clé les observations associées aux amers sélectionnés (associées à l'index de l'amer observé), ainsi qu'une sélection des points caractéristiques accompagnés de leurs descripteurs lorsque l'image en question contient des informations pour la détection des correspondances inter-robot. Enfin, on transmet également les paramètres des modèles de l'IMU et de la caméra.

Dès que le paquet est finalisé, les amers sélectionnés durant la synthèse du paquet sont marqués comme tels. L'inférence globale opérera uniquement sur ces amers. Comme précédemment, on indexe le nouveau paquet, et on mémorise ses caractéristiques dans la table de correspondances à laquelle on se référera pour transmettre à nouveau le paquet si besoin durant la phase de régularisation sans avoir besoin de le recalculer.

5.6.4 Intégration d'un paquet

Reconstruction du paquet

À la réception du paquet, sa trajectoire est d'abord reconstruite à l'aide de la donnée des poses relatives entre les images-clés communiquées. Nous commençons par initialiser les états inertiels en résolvant le problème d'optimisation suivant :

$$\mathcal{E}^* = \underset{\mathcal{E}\in\mathcal{H}_\mathcal{E}}{\arg\min} \left\{ \sum_{z\in\mathcal{Z}_I} \|\boldsymbol{\xi}_z(\mathcal{E}|\mathcal{T})\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{\xi}_z}} \right\} \tag{5.32}$$

où $\mathcal{E}$ désigne l'ensemble des états inertiels (vitesses et biais) associés aux images-clés reçues, $\mathcal{T}$ inclut l'ensemble des variables de poses reçues, $\mathcal{Z}_I$ désigne l'ensemble des facteurs inertiels reçus tandis que $\boldsymbol{\xi}_z$ désigne le facteur inertiel tel que défini dans la section 2.5.3. L'optimisation ne porte que sur les états inertiels tandis que la trajectoire est maintenue fixe. Une fois les états inertiels initialisés, on triangule les amers à partir des observations communiquées. Enfin, la troisième étape consiste à intégrer l'information visuelle reçues. Notons que ces deux dernières étapes peuvent conduire à mettre à jour des images-clés déjà connues et nécessiter des opérations de fusion.

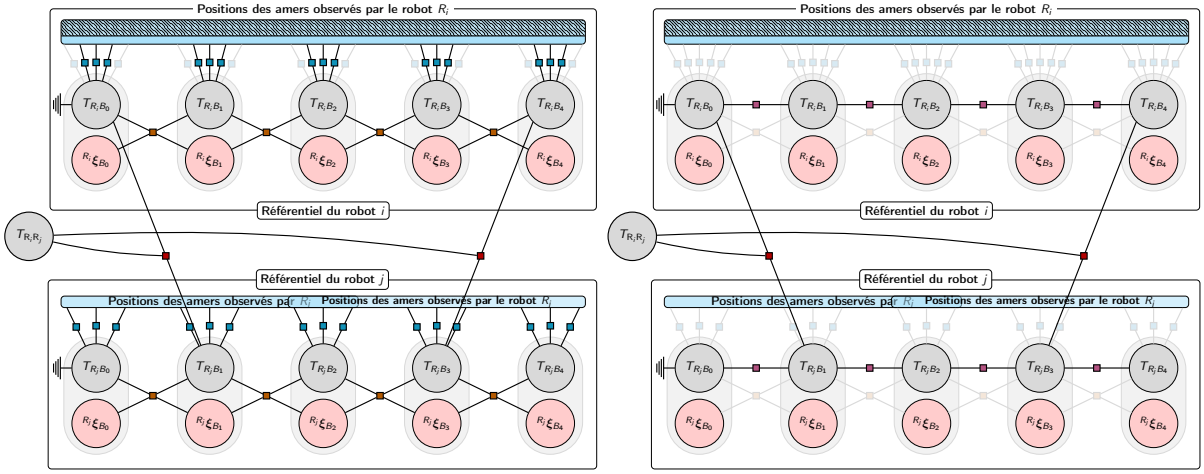
Détection des correspondances inter-robot rétrospectives

La détection des correspondances inter-robot rétrospectives reproduit la solution retenue pour CVIP. Nous renvoyons donc le lecteur aux paragraphes 5.5.4 et 5.5.4.

Inférence globale multi-robots

L'inférence multi-robots comprend deux facettes. Les informations reçues des autres robots permet de formuler un problème d'ajustement de faisceaux visio-inertiel multi-robots sur une portion réduite des amers, tel que représenté par la Figure 5.12a. Cette inférence considère l'ensemble des facteurs inertiels des trajectoires ancrées, les facteurs des correspondances inter-robot qui les lient ainsi que les facteurs visuels reçus ou associés aux amers sélectionnés le long de la trajectoire du robot hôte. Cette sélection d'amers sur la trajectoire hôte permet d'uniformiser l'information considérée avec l'information reçue sur les trajectoires des autres robots, sans pour autant compromettre la précision de l'estimation (Choudhary et al. 2015), et réduit la complexité du problème global.

Néanmoins, l'ajustement de faisceaux visio-inertiel est une opération de raffinement



(a) Ajustement de faisceaux visio-inertiel réduit (b) Relaxation de graphe de poses

L'ajustement de faisceaux visio-inertiel réduit optimise sur les facteurs inertiels ■ et les facteurs visuels associés ■ aux amers sélectionnés ainsi que les correspondances inter-robot ■, tandis que la relaxation de graphe de poses ne considère que des facteurs de pose relative ■ obtenus en entérinant des a priori sur les estimées courantes de poses relatives le long des trajectoires (facteurs séquentiels) et des correspondances précédemment intégrées.

FIGURE 5.12 – Inférence multi-robots sur les paquets PVIP

qui s'exécute en arrière-plan à une cadence réduite. C'est pourquoi nous opérons également des relaxations de graphe de poses, dont la structure est représentée par la Figure 5.12b. Ce type d'inférence permet d'impacter rapidement de nouvelles correspondances en encodant des a priori sur les estimées courantes de poses relatives le long des trajectoires et des correspondances précédemment intégrées, tandis que les facteurs associés aux nouvelles correspondances sont considérés tels quels. Les pondérations induites par les matrices de covariances sont en revanche ignorées (i.e. elles sont toutes considérées comme unitaires). La relaxation de graphe de poses permet de répercuter rapidement les nouvelles correspondances en distribuant uniformément l'erreur odométrique le long des boucles nouvelles fermées, et fournit une bonne initialisation au problème d'ajustement de faisceaux visio-inertiel sus-mentionné. En l'absence d'informations nouvelles, ce problème ne remet pas en question les estimées calculées par l'ajustement de faisceaux visio-inertiel.

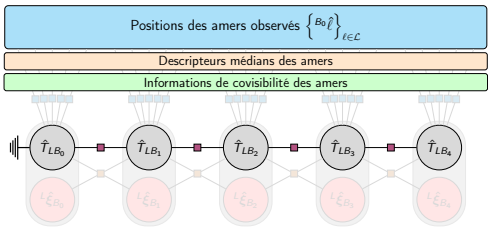
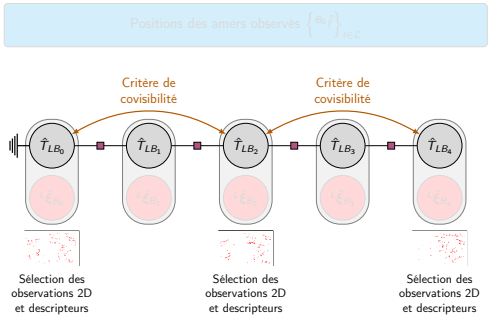
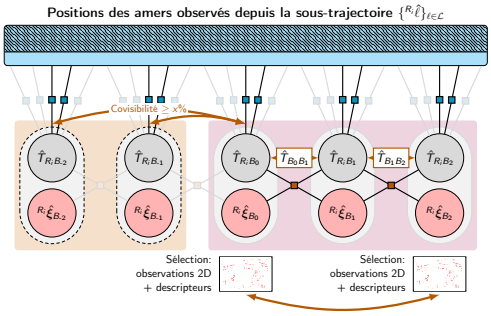
Contenu des paquets	Étapes de calcul des paquets
	**1** Optimisation visio-inertielle **2** Marginalisation locale : calcul d'un facteur de poses globales **3** Éparsification : calcul de facteurs de poses relatives **4** Ajout du nuage d'amers optimisés + descripteurs
	**1** Optimisation visio-inertielle **2** Marginalisation locale : calcul d'un facteur de poses absolues **3** Éparsification consistante : calcul de facteurs de pose relative **4** Sélection des images-clés et de leurs points-clés
	**1** Extraction d'image-clés supplémentaires **2** Ajout des facteurs inertiels **3** Sélection des amers (MKC ou ILP) **4** Sélection des images-clés et de leurs points-clés

TABLE 5.1 – Synthèse des méthodes proposées

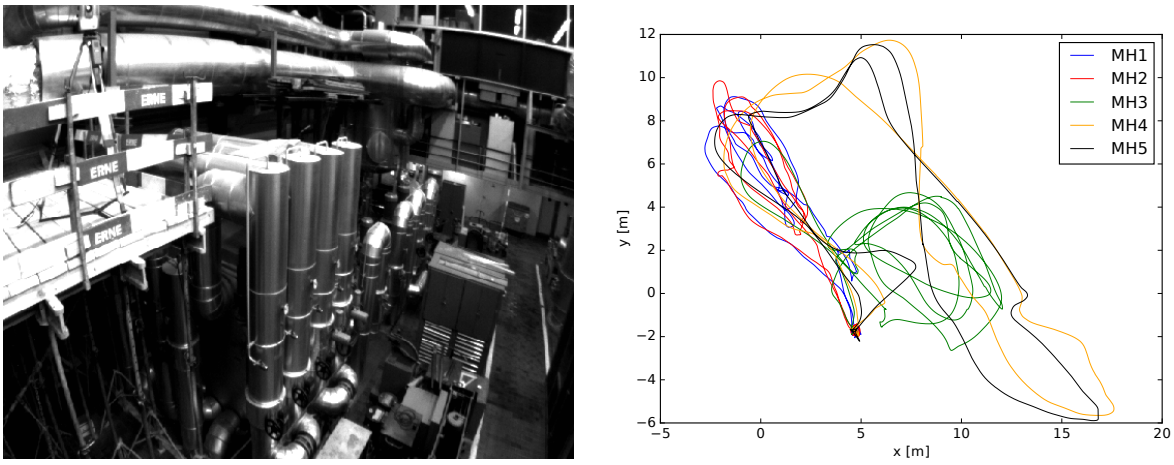
5.7 Évaluation des méthodes proposées

Dans la partie présente, nous évaluons les méthodes que nous avons proposées, et dont le Tableau 5.1 propose une synthèse. La problématique à laquelle nous tâchons de répondre consiste à proposer des méthodes de SLAM collaboratif visio-inertiel qui favorisent l'autonomie des robots tout en répondant aux contraintes d'information, de réseau et de ressources. Nous pouvons considérer que l'objectif d'autonomie est atteint par construction, celui-ci ayant guidé la conception des méthodes développées. Nous devons donc évaluer le comportement de ces méthodes vis-à-vis des trois contraintes mentionnées. Parviennent-elles à préserver l'information de façon consistante lors du calcul des paquets et à la valoriser lors de leur intégration ? Induisent-elles des coûts de communication et une consommation de bande-passante compatibles avec un déploiement sur plateformes

réelles ? Quel est le surcoût dû à la régularisation ? Enfin, le temps de calcul et d'intégration des paquets ainsi que des opérations d'inférences permettent-il d'envisager un fonctionnement en temps réel ?

5.7.1 Scénarios d'évaluation

Nous évaluons les méthodes proposées ci-dessus, et dont une synthèse est présentée par le Tableau 5.1, sur des scénarios multi-robots construits à l'aide des séquences *Machine Hall* d'EuRoC (Burri et al. 2016). Il s'agit de séquences acquises par un drone dans un environnement de type industriel, dont une illustration est fournie par la Figure 5.13a, et dont les trajectoires de vérité terrain sont représentées par la Figure 5.13b. Nous construisons 4 scénarios multi-robots à l'aide des séquences $\{MH1, MH2\}$, $\{MH1, MH3\}$, $\{MH4, MH5\}$ et $\{MH1, MH2, MH3\}$, ainsi que les scénarios 2, 3, 4 et 5 d'AirMuseum.



(a) Capture issue de la séquence MH1 (b) Trajectoires d'EuRoC

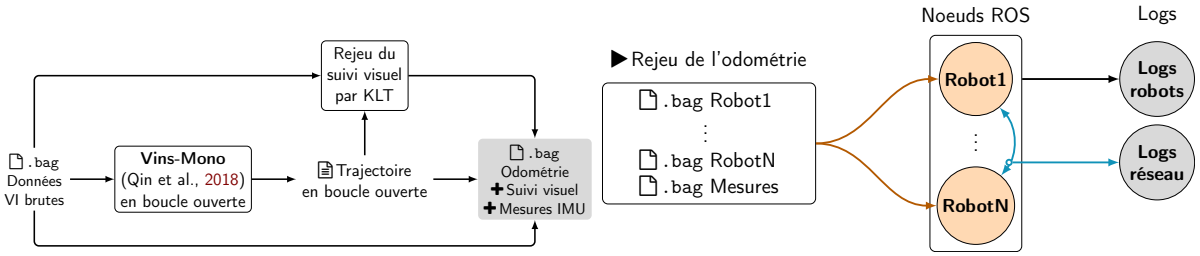
Figure 5.13 – Jeu de données EuRoC (Burri et al. 2016)

5.7.2 Détails d'implémentation

Les simulations ont été menées sur un ordinateur de type Intel® Core™ i7-8850H CPU 2.60GHz × 12 cœurs. Nous avons utilisé le *framework* Maplab (Schneider et al. 2018) pour gérer la structure de la carte, encoder les informations nécessaires à la formulation des problèmes d'optimisation, et réutiliser l'implémentation de la méthode proposée par (Lynen et al. 2015) pour la détection des fermetures de boucle. L'inférence globale est

effectuée à l'aide du solveur `Ceres` (Agarwal et al. 2012). Enfin, nous avons utilisé la bibliothèque `lp-solve` (Berkelaar 2007) pour la programmation du problème ILP.

Concernant la simulation en elle-même, nous simulons chaque robot sous la forme d'un nœud *ROS* (Quigley et al. 2009), avec ses processus *Front-End* et *Back-End*, ainsi que ses interfaces de communication cantonnés à des processus de type threads. En particulier, nous avons simulée les sorties d'un véritable module *Front-End* en utilisant les estimées en boucle ouverte de l'estimateur visio-inertiel de Vins-Mono (Qin et al. 2018b). Nous avons ensuite rejoué le suivi de Kanade-Lucas (Baker et al. 2004) sur les trajectoires estimées avec les images originales, afin d'en extraire les lignes de suivi associées



(a) Pré-traitement hors-ligne

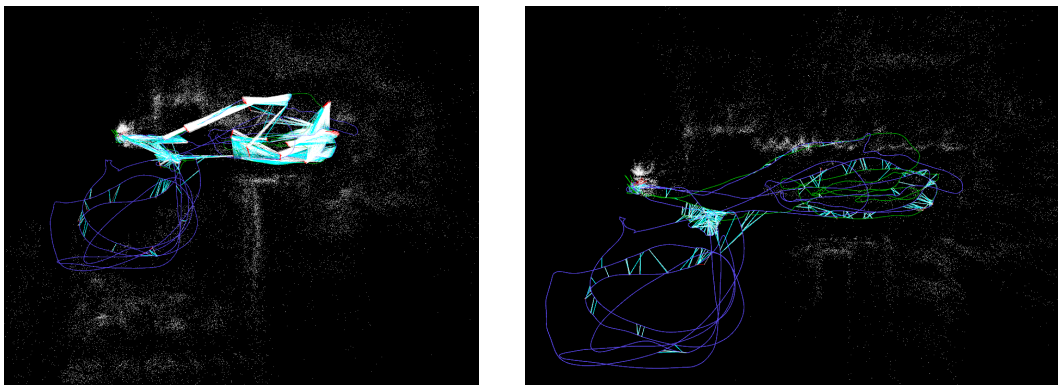
(b) Rejeu des sorties du module d'odométrie de Vins-Mono (Qin et al. 2018b) en boucle ouverte

Figure 5.14 – Architecture des simulations

Algorithme 2 – Simulation multi-robots en mode tour par tour

pour chaque *msg* ***in*** *messages_ordonnés_chronologiquement* **faire**
  `actualiser_horloge`(msg);
  robot ← robots[msg→robot_id]
  si *msg.type* = *image_clé* **alors**
    Traiter les messages reçus en attente dans les *subscribers*
    robot.`ajouter_les_paquets_reçus`();
    robot.`ajouter_les_correspondances_reçues`();
    robot.`traiter_les_requetes_d'informations_visuelles`();
    Ajouter la nouvelle image-clé
    robot.`ajouter_la_nouvelle_image_cle_à_la_carte`(msg);
    robot.`calculer_nouveau_paquet_si_necessaire`();
    robot.`detecter_fermetures_boucles_intra_robot`();
    robot.`detecter_correspondances_inter_robot`();
    robot.`ancrer_referentiels_si_possible`();
    robot.`executer_inference_multi_robot_si_necessaire`();
  sinon si *msg.type* = *correspondance_directe* **alors**
    robot.topic_correspondance_directe.`publier`(msg);
  Exécuter le tour des robots restés inactifs depuis un temps $t \geq t_{\max}$;

aux différents amers. Ces informations sont alors enregistrées sous la forme de messages ROS datés dans un fichier `.bag` qui est rejoué pour chaque robot lors des simulations. La Figure 5.14 illustre ce procédé. Un module *Front-End* interprète alors ces messages pour reconstruire la trajectoire telle qu'elle aurait été localement estimée par Vins-Mono tout en re-triangulant les amers et en leur associant les descripteurs désirés. Un module de fermetures de boucles est implémenté par dessus afin de détecter les correspondances inter-robot. La carte est accédée par différents threads, ce qui permet de reproduire le fonctionnement interne réaliste de l'algorithme de SLAM, tout en disposant d'un socle commun pour comparer les performances des méthodes évaluées au cas du SLAM standard. Les simulations sont menées selon un processus tour-par-tour tel que décrit par l'Algorithme 2. Cette architecture permet de simuler les scénarios multi-robots sur un seul ordinateur tout en respectant les contraintes liées au temps réel. La Figure 5.15 fournit des visuels des simulations des méthodes SVIP et CVIP sur le scénario MH123.



(a) Visuel de simulation de SVIP sur le scénario MH123

(b) Visuel de simulation de CVIP sur le scénario MH123

Les traits bleus représentants les fermetures de boucles et les correspondances inter-robot détectées.

FIGURE 5.15 – Visuels des simulations sur le scénario MH123

5.7.3 Paramètres de simulation

Dans les simulations qui suivent, nous fixons les paramètres suivants. Un nouveau paquet est calculé tous les 5 mètres. Pour la sélection des images-clés le long des trajectoires (c.f. Algorithme 1), nous fixons empiriquement les seuils de nombre et de quotient d'amers covisibles respectivement à $n_{\min} = 20$ et $q_{\min} = 0.5$. Le nombre de couples (point caractéristique + descripteur) communiqués sur chaque image-clé sélectionnée est porté à 100.

Enfin, lors des requêtes d'images-clés, seules les imates-clés adjacentes aux images-clés communiquées dans les paquets originaux sont éventuellement demandées.

5.7.4 Évaluation vis-à-vis des contraintes de ressources

Nous évaluons d'abord les performances des méthodes vis-à-vis des contraintes de ressoures. La Figure 5.16 présente les distributions des temps de calcul des paquets pour les différentes méthodes. Tout d'abord, nous notons que les temps de calcul vont du dixième

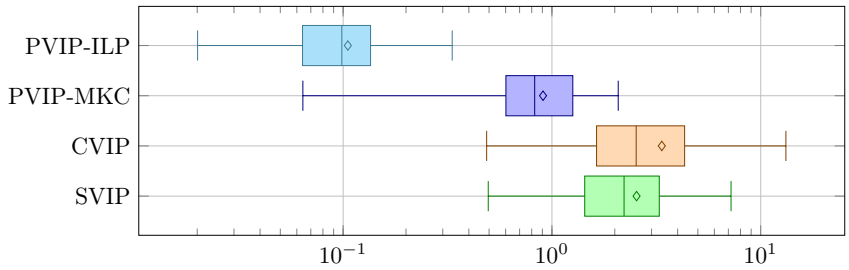


FIGURE 5.16 – Temps de calcul des paquets (en secondes)

de seconde à quelques secondes, ce qui est compatible avec les contraintes imposées par un fonctionnement temps-réel. Nous remarquons également que les méthodes SVIP et CVIP nécessitent plusieurs secondes afin de calculer les paquets correspondants, alors que la méthode PVIP ne requiert que quelques dixièmes de secondes. Cette nette différence d'ordres de grandeurs s'explique par le fait que les méthodes SVIP et CVIP nécessitent chacune une étape d'inférence locale sur le sous-graphe extrait, suivie d'une étape d'extraction de la matrice de covariance sur les poses de la sous-trajectoire. Au contraire, les méthodes PVIP n'exécutent qu'un algorithme de sélection d'amers. On note d'ailleurs que la méthode de sélection en programmation par nombres entiers surpasse l'heuristique MKC de presque un ordre de grandeur temporel. En revanche, dans les deux cas, la couverture exigée n'impacte pas le temps de calcul du paquet.

La Figure 5.17 montre les distributions de temps d'intégration des paquets dans les cartes des robots récepteurs. L'intégration des paquets recouvre leur désérialisation et leur reconstruction dans la carte du robot récepteur. Tout d'abord, nous notons que ces temps d'intégration sont très faibles. Ceux de CVIP et SVIP sont de l'ordre de la milliseconde, tandis que ceux de PVIP approchent le dixième de seconde. Cette différence d'ordre de grandeur s'explique par le besoin de trianguler les amers ainsi que d'initialiser les états inertiels dans le cas de PVIP, alors que l'intégration des paquets CVIP et SVIP

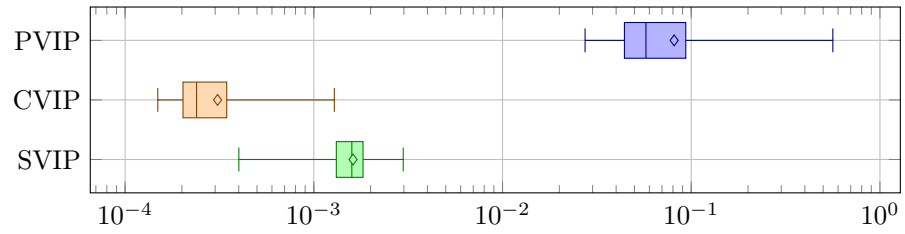
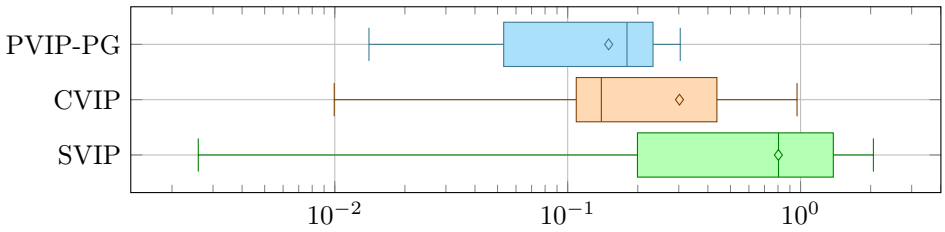


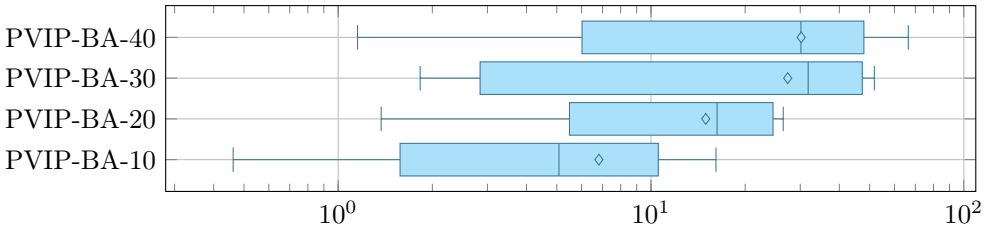
FIGURE 5.17 – Temps d'intégration des paquets reçus (en secondes)

ne nécessitent que de reconstruire la trajectoire et les image-clés le long des facteurs de pose relatives.

Les paquets reçus sont utilisés afin de détecter les correspondances inter-robot, ancrer les référentiels d'odométrie et procéder à des optimisations globales. La Figure 5.18 indique la distribution des temps nécessaires aux optimisations pour les différentes méthodes. On note qu'hormis pour l'ajustement de faisceaux visio-inertiel réduit de PVIP,



(a) Temps d'optimisation des graphes de poses (s)



(b) Temps d'optimisation (s) pour l'ajustement de faisceaux visio-inertiel réduit dans PVIP

FIGURE 5.18 – Temps d'optimisation

les autres problèmes d'optimisation globaux nécessitent des temps de calculs allant de quelques dixièmes de seconde à une seconde, ce qui permet de répercuter rapidement les nouvelles correspondances sur les trajectoires. Les optimisations des paquets SVIP requièrent un peu plus de temps de calcul car davantage de correspondances inter-robot sont trouvées grâce aux informations visio-structurelles transmises, ce qui complexifie légèrement le problème d'inférence. Le problème d'ajustement de faisceaux visio-inertiel réduit

nécessite lui plusieurs dizaines de secondes, ce qui impose qu'il soit résolu en arrière-plan. En guise de perspectives, nous pourrions réduire les temps de calcul nécessaires à l'ajustement de faisceaux en circonscrivant davantage les portions de la carte à ré-optimiser à l'aide par exemple d'une analyse préalable de la consistance des cycles de correspondances (Estrada et al. 2009) ou en ne ré-estimant qu'un sous-ensemble des positions des amers.

5.7.5 Évaluation vis-à-vis des contraintes de réseau

Nous évaluons maintenant les performances des méthodes vis-à-vis des contraintes de réseau. Les distributions des tailles des paquets calculés pour chaque méthode sont données par la Figure 5.19, tandis que leur composition est détaillée par la Figure 5.20. Nous

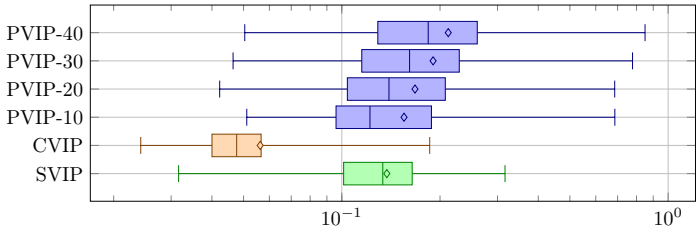


Figure 5.19 – Taille des paquets échangés (en kilo-octets)

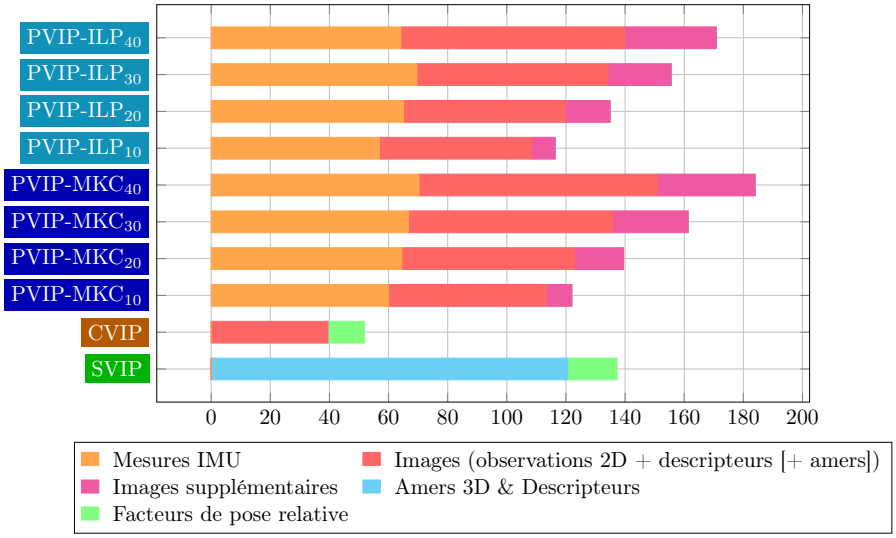


Figure 5.20 – Décomposition des paquets échangés (en kilo-octets)

notons que la méthode CVIP produit les paquets les plus légers, avec une médiane autour de 60 kilo-octets, étant donné qu'elle ne communique que des facteurs de pose relative

(estimée de pose et matrice de covariance) ainsi qu'une sélection des points caractéristiques et des descripteurs associés aux images-clés sélectionnées. Les tailles des paquets de SVIP et PVIP sont supérieures et équivalentes, entre 100 et 200 kilo-octets.Notons que l'information visuelle explique la majeure partie de la taille de ces paquets, que ce soient les informations communiquées sur les image-clés sélectionnées dans les paquets CVIP et PVIP, ou bien les informations visio-structurelles dans la méthode SVIP. Nous remarquons également que les informations inertielles représentent une part importante des paquets PVIP. De ces résultats, nous déduisons également l'ordre de grandeur des surcoût des opérations de régularisation. En effet, les robots qui recouvrent le contact échangent alors les paquets qu'ils ont manqué, si bien que le surcoût croît linéairement avec le nombre de paquets échangés, sachant que chaque paquet représente en moyenne une centaine de kilo-octets.

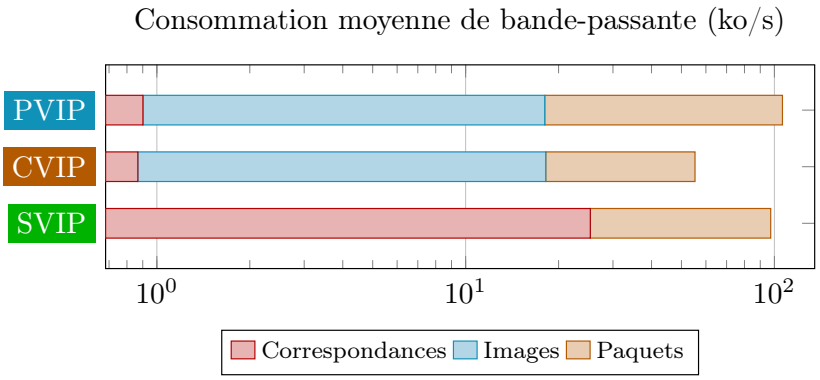
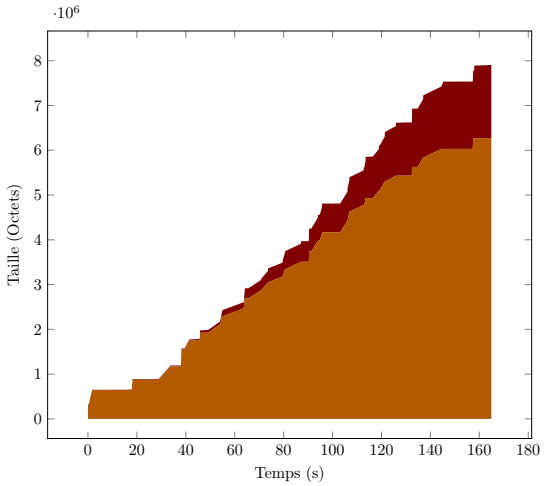
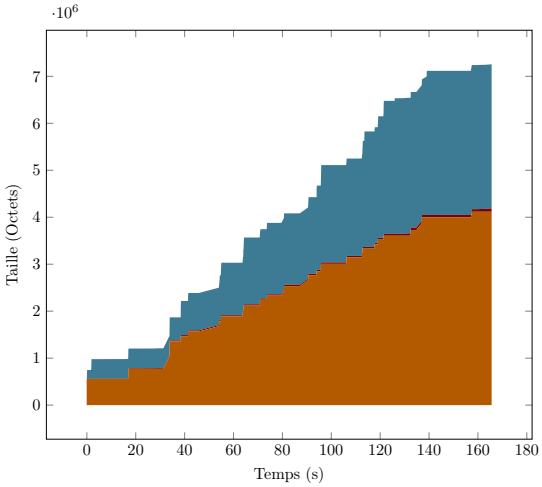


FIGURE 5.21 – Consommation moyenne de bande-passante dans le scénario MH123

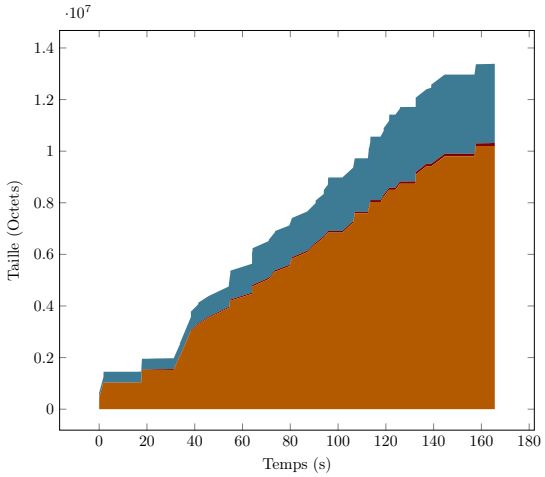
Enfin, nous évaluons la charge réseau induite par l'échange des paquets, mais également par la diffusion des correspondances intra et inter-robot détectées ainsi que les requêtes d'images-clés. Celle-ci est représentée par la Figure 5.22, pour chacune des méthodes dans le scénario EuRoC-MH12. Nous remarquons que la majeure partie de la charge réseau induite est due à l'échange des paquets visio-inertiels entre les robots, dont la quantité cumulée s'accroît à un rythme constant. À deux robots, on compte une consommation moyenne de la bande-passante de 50 kilo-octets par seconde. On note également que davantage de correspondances inter-robot sont échangées puisque les informations visuelles communiquées par SVIP permettent d'en détecter plus. Pour les deux autres méthodes s'ajoute la communication des images-clés sur requête. Pour CVIP, elle représente environ 40% de la bande-passante consommée, tandis que beaucoup moins de correspondances



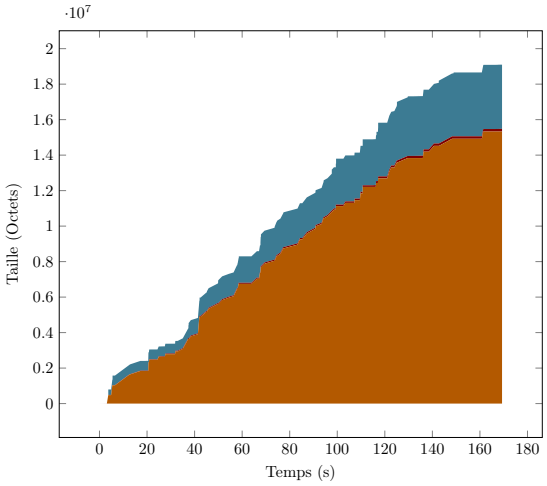
(a) Quantité de données échangées SVIP dans le scénario EuRoC-MH12



(b) Quantité de données échangées CVIP dans le scénario EuRoC-MH12



(c) Quantité de données échangées PVIP-ILP-20 dans le scénario EuRoC-MH12



(d) Quantité de données échangées PVIP-MKC-20 dans le scénario EuRoC-MH12

Légende : désigne l'échange de paquets, l'échange de correspondances inter-robot détectées, et les échanges d'images-clés supplémentaires sur requête.

FIGURE 5.22 – Quantités cumulées de données échangées (octets) entre les robots en fonction du temps (s)

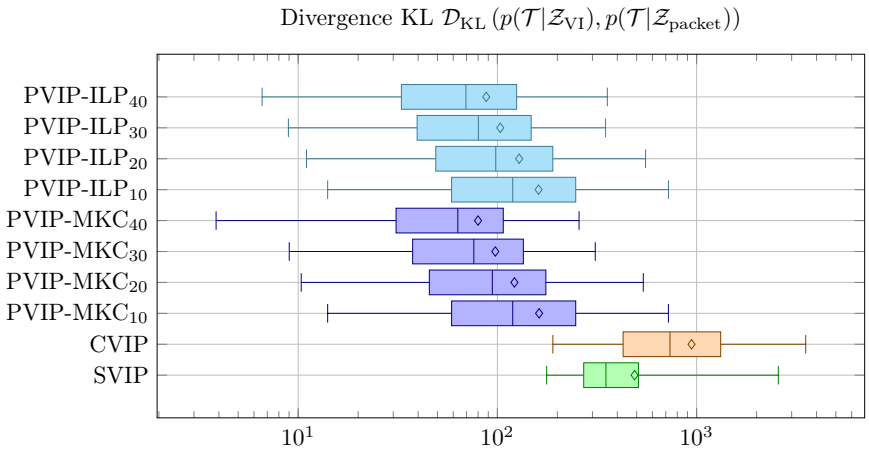
(70 kilo-octets sur toute la séquence). Cette méthode engendre une consommation de bande-passante de l'ordre de 39 kilo-octets par seconde. Enfin, un schéma similaire est de mise pour les méthodes PVIP-MKC et PVIP-ILP, qui consomment respectivement 88 kilo-octets et 113 kilo-octets par seconde de bande-passante. Dans le cas à trois robots du scénario MH123, les consommations moyennes en bande-passante ainsi que leur dé-

composition selon les types de données échangées sont données par la Figure 5.21. Les conclusions sont similaires : CVIP est la méthode qui consomme le moins de bande-passante (environ 70 kilo-octets par seconde) alors que SVIP et PVIP nécessite environ 100 kilo-octets par seconde pour les échanges cumulés des trois robots. Ces consommations de bande-passante sont largement abordables en contexte multi-robots étant donné les performances actuelles des réseaux Wifi.

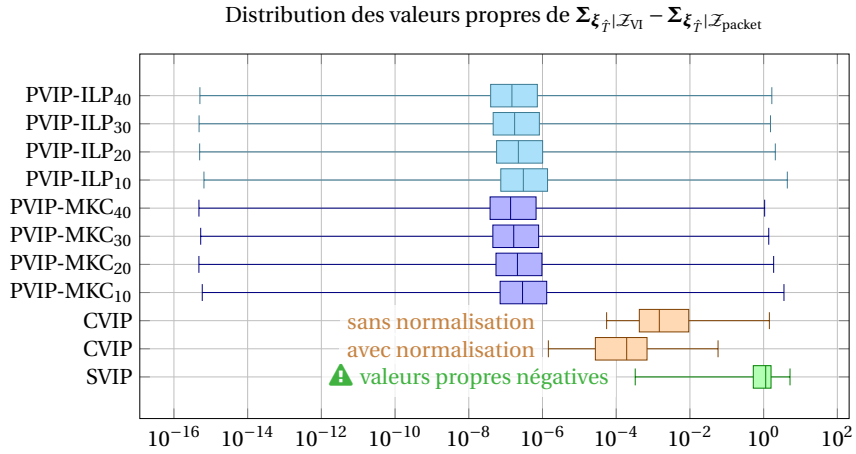
5.7.6 Évaluation vis-à-vis des contraintes d'information

Contenu informatif des paquets

La qualité des paquets ainsi calculés tient à la fidélité avec laquelle ils restituent dans toute sa teneur l'information de localisation contenue dans le sous-graphe extrait. Pour évaluer l'information contenue dans les paquets, nous calculons leur divergence KL avec le sous-graphe original, en comparant les distributions induites sur les variables de pose. Les valeurs de ces divergences sont rapportées par la Figure 5.23a. Nous remarquons que les paquets de CVIP sont ceux qui perdent le plus d'information par rapport à la distribution originale. Ce résultat illustre la conservativité de la méthode CVIP. La méthode SVIP semble mieux conserver l'information, mais au prix de sa consistance comme nous le verrons ci-dessous. Les méthodes PVIP basées sur la sélection d'amers sont celles qui restituent le mieux l'information, avec une divergence KL en moyenne plus faible de presque un ordre de grandeur par rapport à CVIP et SVIP. On note d'ailleurs que l'information est logiquement d'autant mieux préservée que la couverture exigée est importante. La divergence de Kullback-Leibler ne rend cependant pas compte de la consistance des paquets calculés. Pour ce faire, nous calculons les valeurs propres de la différence des matrices de covariances des paquets calculés et des sous-graphes originaux. Le paquet est consistant si et seulement si toutes ces valeurs propres sont positives sans exception aucune, et restituent d'autant mieux l'information que ces valeurs sont proches de zéro. Les distributions de ces valeurs propres sont représentées par la Figure 5.23b. Tout d'abord, nous remarquons que toutes les valeurs propres sont négatives dans le cas de SVIP (représentées en valeurs absolues sur le graphe). Cela est dû au fait que la totalité de l'information du sous-graphe est projetée sur chaque facteur de pose relative calculé. Au contraire, les méthodes CVIP et PVIP sont toutes deux parfaitement consistantes. Nous illustrons d'ailleurs dans ce graphe l'intérêt de la normalisation opérée sur la matrice de covariance lors de l'étape d'éparsification de CVIP, au sens qu'elle permet de réduire les valeurs propres. Enfin,



(a) Distribution des divergences de Kullback-Leibler des paquets calculés



(b) Distribution des valeurs propres de $\mathbf{\Sigma}_s - \mathbf{\Sigma}_d$ où $\mathbf{\Sigma}_s$ est la matrice de covariance du paquet et $\mathbf{\Sigma}_d$ la matrice de covariance du sous-graphe original.

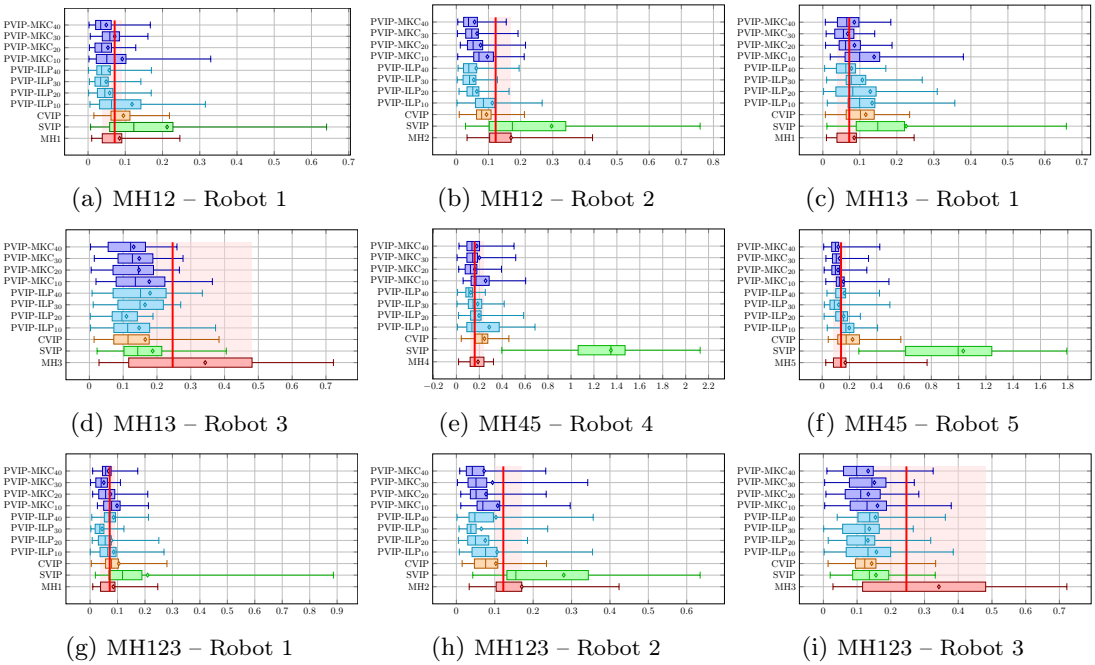
FIGURE 5.23 – Évaluation du contenu informatif des paquets calculés

nous remarquons que la méthode PVIP affiche les meilleures performances en termes de consistance avec une distribution des valeurs propres dont la médiane se situe plusieurs ordres de grandeur en dessous de celle de CVIP.

Performances d'estimation sur le jeu de données EuRoC

Enfin, nous évaluons les précisions d'estimation sur les trajectoires qui résultent de l'intégration des paquets reçus et de leur prise en compte dans les problèmes d'inférence globaux. La Figure 5.24 rapporte respectivement les distribution des erreurs en translations évaluées pour les différentes méthodes selon la procédure proposée par (ZHANG

et al. 2018b), sur les scénarios multi-robotss construits à partir des séquences EuRoC (Burri et al. 2016). On y reporte pour chaque robot dans chaque scénario la distribution de des erreurs en translations sur ses estimées de trajectoires. Les barres rouges réflètent les résultats dans le cas mono-robot. De façon générale, nous remarquons que



(a) MH12 – Robot 1 (b) MH12 – Robot 2 (c) MH13 – Robot 1

(d) MH13 – Robot 3 (e) MH45 – Robot 4 (f) MH45 – Robot 5

(g) MH123 – Robot 1 (h) MH123 – Robot 2 (i) MH123 – Robot 3

Figure 5.24 – Distribution des erreurs en translation (m) sur les scénarios EuRoC

les meilleures performances d'estimation sont obtenues avec les méthodes PVIP, suivi par CVIP et enfin par SVIP. Tout d'abord, les trois méthodes permettent à l'ensemble des robots d'estimer leur trajectoire et celles des autres robots avec des précisions allant du centimètre à quelques dizaines de centimètres, ces trajectoires mesurant chacune entre 100 et 200 mètres de long. Cependant, si estimer les trajectoires des autres robots est un objectif des méthodes de SLAM collaboratives, un second objectif consiste à exploiter ces informations afin d'améliorer les performances d'estimation par rapport au cas du SLAM mono-robot, ou du moins d'absorber les erreurs d'estimation sur les trajectoires reçues sans compromettre l'estimation de la trajectoire du robot hôte.

Nous remarquons tout d'abord que les méthodes CVIP et PVIP permettent des gains significatifs en performances d'estimation par rapport au cas mono-robot. Dans tous les scénarios, la méthode CVIP permet d'améliorer les précisions d'estimation sur certaines trajectoires, ou bien ne dégrade pas significativement la précision par rapport au cas mono-

robot. En particulier, on note qu'elle réduit l'erreur MSE et resserre notablement les distributions des erreurs sur les trajectoires des robots 2 et 3 dans les scénarios EuRoC-MH12 et EuRoC-MH13, et d'autant plus dans le EuRoC-MH123. Dans ce dernier scénario, elle permet notamment des gains de plusieurs centimètres pour le robot 2, et d'une dizaine de centimètres sur la trajectoire du robot 3. Sur les trajectoires des robots 1, 4 et 5, elle maintient des performances équivalentes au cas mono-robot. La méthode PVIP affichent les meilleurs résultats des trois méthodes. Elle engendre une amélioration quasi-systématique des précisions d'estimation dans tous les scénarios testés. C'est d'ailleurs la seule méthode qui parvient à améliorer légèrement la précision d'estimation de la trajectoire du robot 1 alors que cette trajectoire est déjà auto-suffisante en termes de fermetures de boucles dans le cas mono-robot. PVIP bénéficie notamment de son double niveau d"inférence : la relaxation de graphe de poses pour répercuter rapidement les correspondances détectées et reçues, et l'ajustement de faisceaux visio-inertiel réduit qui tourne en arrière plan avec des informations visio-inertielles brutes. L'influence des couvertures d'observation est visible, mais les différences ne sont pas significatives. Par ailleurs, on ne remarque pas de différences notables entre les deux variantes MKC et ILP.

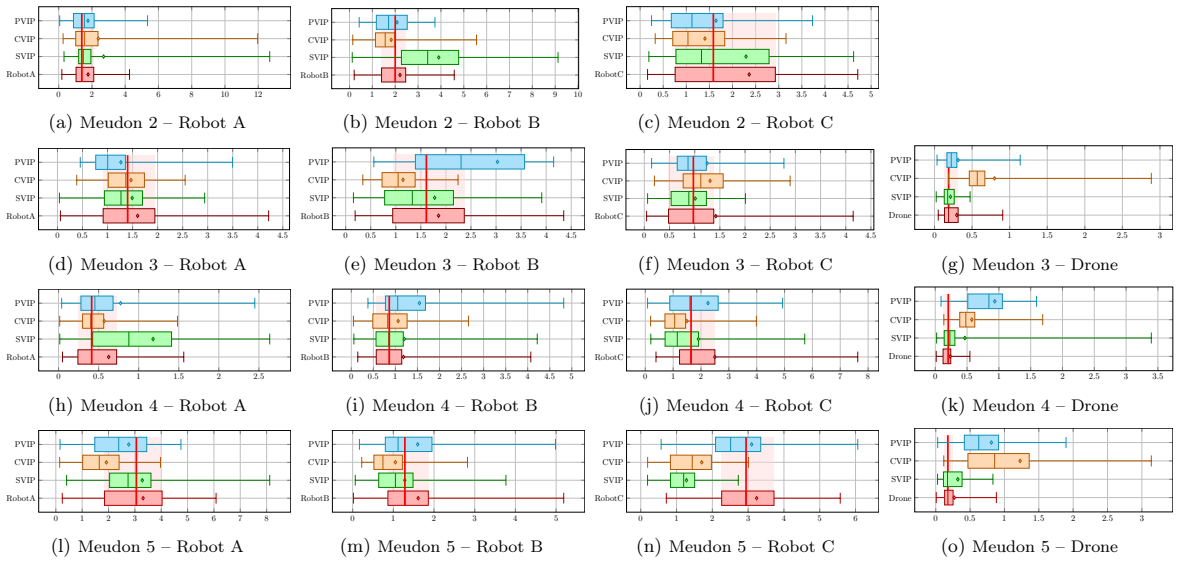
La méthode SVIP affiche des performances plus mitigées. En effet, on note qu'elle produit généralement des précisions d'estimation dégradées par rapport au cas mono-robot ; en particulier, elle élargit la distribution des erreurs et accroît l'erreur MSE. Dans le scénario EuRoC-MH12, l'intégration des sous-cartes reçues dégrade l'estimation par les robots de leur propre trajectoire. Le même constat est de mise sur le scénario EuRoC-MH13 pour le robot 1, bien qu'elle parvienne à améliorer l'estimation de la trajectoire du robot 3 aussi bien dans la carte du robot 1 que du robot 3. C'est sur le scénario EuRoC-MH45 que les performances sont les plus dégradées. Cette contre-performance est due à de fortes erreurs d'estimation provoquée par la mauvaise intégration des sous-cartes reçues au milieu de la séquence, là où les incertitudes sur les trajectoires des deux robots sont fortes, à laquelle les correspondances détectées ultérieurement ne parviennent pas à y remédier, occasionnant de très mauvaises initialisations pour l'intégration des correspondances et les optimisations globales. Dans le scénario EuRoC-MH123, on retrouve les constats faits pour les deux premiers scénarios. Seule l'estimation de la trajectoire du robot 3 bénéficie de l'intégration des sous-cartes. Une hypothèse concernant les performances médiocres de SVIP est qu'elle découle tout d'abord d'une mauvaise qualité des correspondances inter-robot détectées, puis de la topologie agressive des contraintes induite par ces correspondances qui, intégrées généralement assez brutalement par paquets, contribuent davantage à pro-

pager et amplifier des erreurs d'estimation. Les correspondances inter-robot sont en effet détectées en comparant les images-clés de la trajectoire du robot récepteur aux nuages de points des sous-cartes (ce qui conduit à intégrer les erreurs de la triangulation des amers résultant de l'inférence locale), et lient les poses des images-clés de requête à la pose de la sous-carte. Ce mécanisme peut manquer de précision lorsque les correspondances sont détectées avec des portions du nuage qui correspondent aux dernières images-clés de la sous-carte. De plus, beaucoup de correspondances inter-robot sont détectées, si bien qu'elles induisent davantage de contraintes que les fermetures de boucles au sein des trajectoires qui sont plus rares. Enfin, ces correspondances sont détectées et intégrées d'un coup, ce qui induit des corrections brutales, et nuisibles lorsque les portions de trajectoires impactées sont déjà sujettes à de larges erreurs, comme dans le scénario EuRoC-MH45.

Performances d'estimation sur le jeu de données AirMuseum

Dans un second temps, nous avons également évalué les méthodes proposées sur les scénarios du jeu de données AirMuseum que nous avons créé. Le but est de les confronter à des séquences exposant des difficultés complémentaires à celles rencontrées dans EuRoC. En particulier, on y trouve moins de fermetures de boucles intra et inter-robot que dans les séquences EuRoC. De plus, l'utilisation de robots terrestres introduit des difficultés supplémentaires pour l'estimation visio-inertielle comme en témoignent les précisions d'estimation obtenues avec Vins-Mono (Qin et al. 2018b) dans le tableau 4.3. En particulier, leurs trajectoires ont une cinématique beaucoup moins agressive que celle des drones. En conséquence, les IMU sont moins excitées, ce qui compromet localement l'observabilité des biais inertiels. Une autre difficulté est que les robots terrestres sont soumis à des vibrations beaucoup plus importantes que les drones. De plus, les IMU utilisées sont de moins bonne qualité que celles employées dans EuRoC (Burri et al. 2016). Enfin, contrairement aux drones, les robots terrestres ont un champ de vue plus restreint, ce qui engendre des contraintes visuelles plus faibles sur certaines portions lorsque les robots évoluent loins des obstacles.

Les erreurs en translations obtenues pour chaque robot dans les scénarios 2, 3, 4 et 5 sont rapportées dans la Figure 5.25. Étant donné que les variantes de PVIP ont montré des performances équivalentes, nous ne considérons ici que la version ILP avec une couverture exigée de 10 observations par image-clé. Globalement, bien que les scénarios soient intrinsèquement difficiles, les méthodes proposées affichent des performances moyennes.



(a) Meudon 2 – Robot A (b) Meudon 2 – Robot B (c) Meudon 2 – Robot C

(d) Meudon 3 – Robot A (e) Meudon 3 – Robot B (f) Meudon 3 – Robot C (g) Meudon 3 – Drone

(h) Meudon 4 – Robot A (i) Meudon 4 – Robot B (j) Meudon 4 – Robot C (k) Meudon 4 – Drone

(l) Meudon 5 – Robot A (m) Meudon 5 – Robot B (n) Meudon 5 – Robot C (o) Meudon 5 – Drone

FIGURE 5.25 – Distribution des erreurs en translation (m) sur les scénarios AirMuseum

Tout, on retrouve le fait que les méthodes PVIP et CVIP permettent des gains significatifs sur les précisions d'estimations bien que ceux-ci ne soient pas systématiques et que l'on parte de performances mono-robot considérablement dégradées par rapport aux scénarios d'EuRoC (même si cela résulte d'une volonté de conception du jeu de données AirMuseum pour laisser une plus grande marge d'amélioration aux méthodes collaboratives). Cependant, ces scénarios exhibent des limitations intrinsèques aux deux méthodes et qui ne transparaissaient pas sur les scénarios d'EuRoC. En particulier, PVIP montre sur AirMuseum des performances globalement dégradées. Deux facteurs peuvent l'expliquer. Tout d'abord, il arrive que très localement, les contraintes visuelles soient insuffisantes pour assurer la couverture désirée lors de la sélection des amers, et laissent les facteurs inertiels prévaloir lors des inférences. De plus, lors de la réception des paquets, les états inertiels sont initialisés à trajectoire fixée, bien que les biais ne soient pas systématiquement observables. Ces défaut locaux induisent alors des erreurs d'estimations difficilement recouvrables ultérieurement. Hormis quelques scénarios, les méthodes SVIP et CVIP permettent d'améliorer très légèrement les précisions d'estimation sur les trajectoires. Néanmoins, leur évaluation sur les séquences terrestres illustrent deux difficultés particulières qui ne transparaissaient pas dans EuRoC. Tout d'abord, dans les deux méthodes, toute l'information visio-inertielle des paquets est condensée sur des facteurs de pose relative, si bien que l'information absolue liée à la direction estimée du vecteur gravité est perdue. Ainsi, dans certaines séquences, la verticale n'est plus correctement estimée sur les por-

tions de trajectoires insuffisamment contraintes. Une seconde limitation tient à la faiblesse des contraintes visio-inertielles locales sur certaines portions des trajectoires terrestres. En effet, lors du calcul de certains paquets, l'inférence locale repose sur relativement peu de points caractéristiques tandis que l'IMU n'est pas suffisamment excitée pour rendre les biais inertiels observables. Cela oblige à fixer les biais inertiels associés à la première pose, et vérifier que l'échelle est conservée durant l'optimisation locale. Dans le cas contraire, on choisit l'estimation courante comme point de linéarisation pour le calcul des facteurs.

Notons également que l'alignement des trajectoires repose majoritairement sur l'exploitation des observations directes des marqueurs AprilTag car peu de correspondances indirectes inter-robot sont détectées. Ainsi, les trajectoires sont mutuellement moins contraintes, les transformations relatives estimées par l'observation des marqueurs AprilTag étant plus incertaines que celles déduites des correspondances indirectes.

Enfin, nous remarquons une disparité importante entre les performances d'estimation des robots terrestres et du drone. Alors que l'intégration des informations reçues des autres robots aide généralement à améliorer l'estimation des trajectoires dans le cas des robots terrestres, cela dégrade les estimées de trajectoires des drones. La fusion des informations multi-robots engendre davantage une propagation des erreurs qu'un véritable gain d'informations sur ces trajectoires. Le drone éprouve donc des difficultés à intégrer les paquets des autres robots sans compromettre ses propres estimées, ce qui appelle des mécanismes supplémentaires de régularisation, là encore éventuellement basée sur des analyse préalables de consistance le long des cycles de correspondances, afin de ne pas remettre en question trop facilement les estimées du robot lors de l'intégration des paquets reçus.

5.8 Conclusions et Perspectives

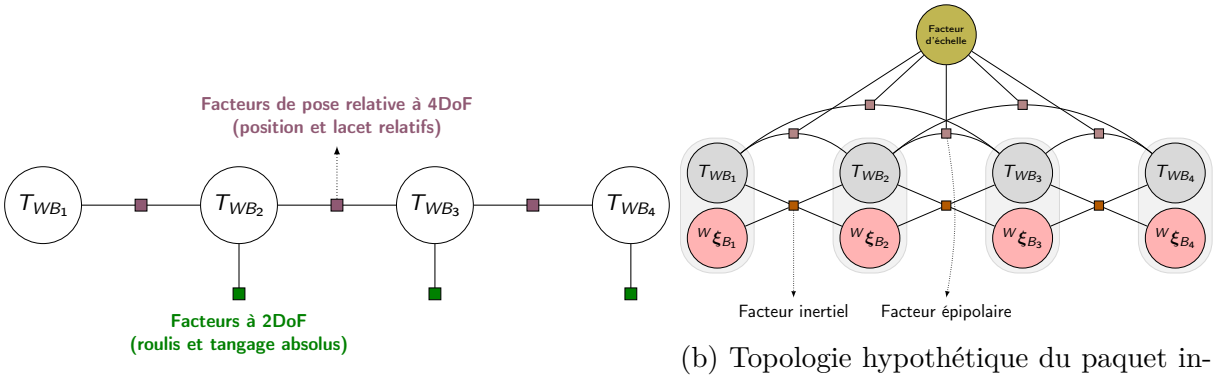
Le but poursuivi dans ce chapitre était de développer des méthodes de SLAM visio-inertielles collaboratives afin de permettre aux robots d'exploiter de façon autonome les informations acquises par les autres robots sur l'environnement de façon à améliorer la qualité de leur carte pour la navigation lorsqu'ils visitent les mêmes zones, tout en se conformant aux contraintes réseau et d'embarquabilité. Nous avons pour cela investigué trois paradigmes d'échange de données, à partir desquels nous avons construit trois méthodes de SLAM multi-robots. Ces méthodes, dans leurs versions préliminaires, ont fait l'objet d'une publication à la conférence nationale ORASIS 2019 (Dubois et al. 2019) et

à la conférence internationale IROS 2019 (Dubois et al. 2019).

Afin de maximiser l'autonomie des agents, nous avons retenu des architectures complètement décentralisées basées sur l'échange de paquets de données contenant toutes les informations nécessaires afin de détecter des correspondances inter-robots, ancrer les référentiels d'odométrie et ré-estimer l'ensemble des trajectoires. On choisit de communiquer une sélection d'informations visuelles sur une sélection d'images-clés afin de permettre aux robots de détecter les correspondances inter-robot avec leur propre trajectoire en toute autonomie. Cela consomme davantage de bande-passante que des méthodes telles que (Giamou et al. 2018) et (Lajoie et al. 2020) qui ne partagent respectivement que des vecteurs sac-de-mots et des descripteurs globaux NetVLAD (Arandjelovic et al. 2016) pour identifier les correspondances candidates, et procèdent à un second échange ciblé d'informations visuelles afin de les caractériser. La détection des correspondances inter-robot nécessite d'établir des correspondances visio-structurelles entre des images requête et une carte : c'est pourquoi il incombe à chaque robot de détecter les correspondances inter-robot avec sa propre carte, ce qui contribue à son autonomie. Il s'ensuit une division de cette tâche entre les robots, qui s'échangent ensuite les correspondances détectées. De même, les mécanismes de requêtes d'images-clés permettent opportunément de préciser des correspondances détectées. Enfin, les graphes de facteurs communiqués sont utilisés tels quels pour l'inférence multi-robots formulées par chaque agent avec une complexité abordable.

La première méthode proposée est SVIP, basée sur l'échange de sous-cartes visio-inertielles entre les robots. Leur calcul repose sur une optimisation locale et requiert l'extraction d'une matrice de covariance pour calculer les facteurs de poses relatives : c'est l'étape la plus complexe du processus. Des trois méthodes, SVIP est celle qui transmet les informations les plus exhaustives sur l'environnement observé et permet de détecter le plus de correspondances inter-robots. Cependant, ses performances sont moindres comparés aux deux autres méthodes, et elle échoue à améliorer la qualité de la carte des robots. Cela peut résulter de correspondances moins précises et de la topologie particulière des facteurs induits par ces correspondances et qui concentrent les contraintes sur les seules poses des sous-cartes. De plus, en l'état, les facteurs de pose relative calculés ne sont pas consistants ; cette étape pourrait bénéficier des méthodes d'éparsification utilisés dans CVIP. SVIP peut être choisie lorsque les robots utilisent les reconstructions éparses de l'environnement, par exemple à des fins de planification de trajectoires comme proposé par (Ruetz et al. 2019).

La seconde méthode, CVIP, repose sur des techniques de marginalisation locale et d'éparsification consistante. Bien qu'il s'agisse de la technique la plus complexe pour générer les paquets visio-inertiels, c'est la méthode qui consomme le moins de bande-passante, et sied donc aux contexte où cette dernière est réduite. C'est aussi elle qui condense l'information de la manière la plus synthétique. La consistance est garantie de façon conservative, mais cela évite les mécanismes complexes de soustraction d'informations (Cunningham et al. 2013). Ces paquets permettent notamment une intégration et des inférences multi-robots rapides, ce qui renforce son extensibilité avec le nombre d'agents, et l'adapte aux cas où la puissance de calcul des agents est réduite. Un inconvénient est cependant que chaque robot n'utilise qu'une partie de l'information dont il dispose sur sa propre trajectoires lors des inférences globales. Si cette méthode a montré de bonnes performances d'estimation sur EuRoC, elle peine davantage sur AirMuseum où elle pâtit de l'absence d'a priori sur la direction de la gravité, pourtant estimée dans le cadre visio-inertiel. Ce constat s'applique également pour SVIP. Ainsi, une solution alternative serait d'utiliser des facteurs alternatifs de la forme suivante. On remplacerait les facteurs de pose relative à 6 degrés de liberté entre les images-clés successives par des facteurs de pose relative à 4 degrés de liberté portant uniquement sur la translation relative et l'angle de lacet relatif. Ce facteur serait complété par un facteur unaire encodant un a priori sur les angles de roulis et de tangage par rapport au référentiel d'odométrie du robot. La matrice Jacobienne résultante serait également inversible par construction, ce qui ne modifierait donc en rien la procédure de calcul exposée. Ce schéma, tel que représenté par la Figure 5.26a, permettrait une projection plus utile de l'information visio-inertielle originale contenue dans le paquet.



(a) Topologie du paquet CVIP modifié envisagé

(b) Topologie hypothétique du paquet intégrant des facteurs épipolaires

Figure 5.26 – Paquets modifiés envisagés

La dernière méthode proposée est PVIP, et exploite des techniques d'élagage. Il s'agit de la méthode la plus consistante des trois dans le sens qu'elle préserve le mieux l'information originale, et dont les paquets sont calculés le plus rapidement. Enfin, bénéficiant d'un double optimisation (relaxation de graphe de poses et ajustement de faisceaux visio-inertiel réduit), elle affiche les meilleures performances d'estimation des trois méthodes. C'est donc la méthode de partage qu'on choisira dans le cas général. En comparaison avec CVIP, elle induit cependant un surcoût de communication et l'intégration des paquets est plus complexe. De plus, si la relaxation de graphe de poses permet de répercuter rapidement les nouvelles correspondances sur l'ensemble des trajectoires, elle répartit les erreurs de façon uniforme le long des boucles fermées, et seul l'ajustement de faisceaux permet de rectifier cette approximation en arrière-plan. Ce type d'optimisation peut rapidement devenir complexe dès lors que le nombre de robots augmente. Le recours à CVIP semble alors plus approprié dans ce cas.

Une autre piste d'amélioration serait élaborer une politique d'intégration plus ciblée en identifiant plus précisément les portions des trajectoires à optimiser ou ré-optimiser, sur la base par exemple d'une analyse des cycles induits par les correspondances intra et inter-robots (Estrada et al. 2009). Cela pourrait notamment permettre de mieux gérer les ressources pour le problème d'ajustement de faisceaux réduit de PVIP. Afin d'améliorer l'efficacité de l'inférence multi-robots, nous pourrions exploiter les procédés d'optimisation incrémentale, tels qu'implémentés dans la bibliothèque g^{2}o (Kümmerle et al. 2011). Enfin, nous pourrions également envisager d'autres natures de paquets. Par exemple, une idée non explorée aurait consisté à constituer un paquet sur le modèle de ceux de PVIP : on transmettrait les facteurs inertiels, complétés par des facteurs visuels de pose relative qui exprimeraient des contraintes épipolaires formulées sur $\mathbb{S}\mathfrak{im}_3$. On les construirait en estimant la matrice essentielle entre les images successives puis en la décomposant en une matrice de rotation et un vecteur de translation unitaire (en l'absence d'informations sur l'échelle). Enfin, on calculerait la matrice de covariance associée à ces grandeurs (3 paramètres pour la rotation, 2 pour la translation unitaire) en calculant la matrice d'information de Fisher associée à la minimisation des distance des points aux lignes épipolaires engendrées par leurs points homologues sur l'autre image. De plus, de tels facteurs pourraient être calculés entre toute paire d'image-clé suffisamment covisibles tout en restituant l'information de façon consistante. Cette idée a simplement fait l'objet d'un prototypage qui a exposé des problématiques de sensibilité des vecteurs de translations estimés. Cependant, elle n'a pas été davantage investiguée.

Chapitre 6

SLAM dense stéréo-visuel décentralisé basé sur l'échange de sous-cartes TSDF

6.1 Introduction

Historiquement, les algorithmes de SLAM ont d'abord exclusivement envisagé la cartographie comme un support intermédiaire pour la localisation, avant qu'elle n'en deviennent une finalité à part entière. Le SLAM visuel a privilégié des cartographies éparses adaptées aux ressources de calcul alors disponibles en contexte embarqué. Celles-ci extraient des primitives géométriques dont le but originel se réduisait à formuler les erreurs géométriques impliquées dans l'estimation des poses. Peu portées sur le rendu visuel, elles ne sont pas davantage propices à la navigation car elles ne représentent pas explicitement les distances aux obstacles. L'augmentation récente des moyens de calcul embarqués a finalement permis d'envisager des reconstructions denses afin d'y remédier. Ces dernières se basent au choix sur des cartes de profondeur projetées, des nuages de points ou de surfels denses, des grilles d'occupation ou encore des maillages générés par triangulation ou à partir de champs de distances signés. Les méthodes de SLAM mono-robot se sont récemment approprié ces représentations à partir des années 2010. Cependant, leur extension aux méthodes de SLAM denses collaboratives constitue un domaine de recherche actif. En effet, les cartographies denses induisent des échanges de données plus volumineux, imposent une charge de calcul accrue et compliquent le maintien de la consistance globale de la carte au fur et à mesure que des boucles sont fermées.

Dans ce chapitre, nous nous intéressons à des méthodes de cartographie implicite basées sur des champs de distance signés et tronqués, abrégées TSDF pour *Truncated Signed Distance Fields*. L'objectif est d'appliquer l'architecture proposée dans le chapitre

précédent en contexte de cartographie dense. Plus spécifiquement, nous proposons une extension d'une méthode de SLAM dense collaborative développée à l'ONERA par Duhautbout et al. (Duhautbout et al. 2019), et qui fut, à notre connaissance, la première méthode de SLAM multi-robots basée sur l'échange de sous-cartes TSDF. Nous entamerons ce chapitre par une revue des représentations denses existantes puis étudierons comment elles s'intègrent au sein des algorithmes de SLAM existants. Dans un second temps, nous décrirons l'approche proposée module par module. Nous poursuivrons par l'évaluation de ses performances sur des scénarios multi-robots construits à partir du jeu de données EuRoC (Burri et al. 2016). Enfin, nous conclurons sur les perspectives ouvertes par les limites de la méthode exposée.

6.2 Travaux associés

Tout algorithme de SLAM adopte une représentation spécifique de l'environnement afin d'en restituer les propriétés géométriques et visuelles. On la choisit d'une part selon les possibilités offertes par le banc de capteurs, et d'autre part en fonction des applications visées au-delà de l'algorithme de SLAM lui-même. On oppose classiquement les représentations éparses aux cartographies denses. Les premières modélisent l'environnement comme une constellation de primitives géométriques. Les secondes s'efforcent au contraire d'en restituer la structure 3D à forte résolution et dans sa continuité. Comparativement aux cartographies éparses, elles poursuivent deux objectifs spécifiques : 1) offrir une représentation commode pour la planification de trajectoire et l'évitement d'obstacles, et 2) permettre un rendu visuel fin et réaliste. Dans un second temps, nous verrons comment ces représentations s'intègrent au sein d'algorithmes de SLAM denses.

6.2.1 Les cartographies éparses

Les représentations éparses extraient et localisent des primitives géométriques afin de rendre compte de la structure 3D de l'environnement. C'est le cas des des nuages de points éparses, communément employés pour les algorithmes de SLAM monoculaires et visio-inertiels, à l'instar de ORB-SLAM (Mur-Artal et al. 2015) et de Vins-Mono (Qin et al. 2018b). Les primitives correspondent le plus souvent à des amers 3D ponctuels, ou plus marginalement à des lignes (He et al. 2018). D'abord associées à des points d'intérêts extraits dans les images, elles sont suivies au fil de leurs observations successives, à par-

tir desquelles leur position est estimée puis raffinée. Les cartographies éparses présentent l'avantage de ne retenir que l'information 3D nécessaire à la formulation des erreurs géométriques intervenant dans l'estimation des poses. Cette information rudimentaire peut néanmoins se prêter à des alignements modèle-à-modèle comme l'ont montré (Cieslewski et al. 2016) en développant des descripteurs spécifiquement dédiés aux nuages de points éparses. Les représentations éparses s'avèrent toutefois insuffisantes au regard des objectifs mentionnés en introduction : elle n'esquissent qu'une ossature de la géométrie, et ne fournissent pas les informations adaptées à la planification de trajectoires.

6.2.2 Les cartographies denses non structurées

Face aux représentations éparses, les cartographies denses, portées par l'augmentation des moyens de calculs embarqués, fournissent des modèles plus résolus de la structure 3D de l'environnement, ou plus à même d'en restituer la continuité. On distingue les représentations denses structurées des représentations denses non-structurées. Ces dernières incluent notamment les nuages de primitives géométriques à forte résolution. Le point 3D est la primitive la plus employée. Des algorithmes de SLAM basés LiDAR exploitent par exemple cette représentation, tels que LOAM-SLAM (Zhang et al. 2014) ou encore IMLS-SLAM (Deschaud 2018). En SLAM visuel, ces nuages de points sont triangulés à partir des cartes de disparités dans le cas stéréo-visuel, ou dérivent de la rétro-projection dans l'espace de cartes de profondeurs dans le cas RGB-D. La densité des nuages de points obtenus invite alors à travailler directement sur la géométrie induite pour transposer des tâches usuellement exécutées sur la base de critères visuels. C'est par exemple le cas de la détection et de la caractérisation des fermetures de boucles via l'extraction, la description et l'appariement de primitives 3D. On trouve ainsi des extracteurs de coins de Harris 3D (Sipiran et al. 2011), de points ISS (*Intrinsic Shape Signature*) (Zhong 2009) et des analogues 3D des points SIFT (Flint et al. 2007), associés à un descripteur 3D du même nom. Similairement au cas 2D, des descripteurs 3D (Hana et al. 2018) caractérisent, identifient et discriminent les points 3D préalablement à leur appariement. À titre d'exemple, la bibliothèque *Point Cloud Library* (PCL) (Rusu et al. 2011) fournit une riche implémentation de ces outils. À côté du SLAM visuel indirect, les méthodes d'odométrie visuelle directes et RGB-D offrent un autre domaine de prédilection pour la cartographie dense. Enfin, les éléments de surface, ou *surfels*, tels qu'utilisés par Elastic Fusion (Whelan et al. 2015) (c.f. Figure 6.1b), introduisent un second type de primitives denses : il s'agit de petits disques orientés associés à des informations photométriques.

6.2.3 Les cartographie denses structurées

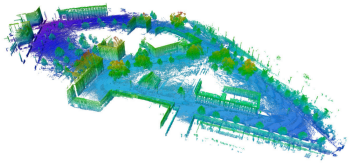
Les représentations par grilles d'occupation 3D

Si les représentations denses non-structurées ont accru le réalisme des rendus visuels, elles échouent cependant à fournir une solution adaptée à la planification de trajectoires sous contrainte d'évitement d'obstacles. À cette fin, des représentations denses structurées et focalisées sur la représentation des volumes et des surfaces ont été développées. Celles-ci ne résultent alors plus de la juxtaposition de primitives mais sont restituées dans leur continuité. Historiquement, les premières représentations explicites des volumes sont obtenues à partir de grilles d'occupation 3D (Moravec 1996), extension naturelle des grilles 2D puis des cartes d'élévations dites 2.5D (Lacroix et al. 2002). En dépit de leur incapacité à représenter correctement les surplombs, ces dernières s'avèrent cependant bien adaptées à certains contextes spécifiques comme le montrent (Käslin et al. 2016) qui font collaborer un drone et un robot terrestre en utilisant cette représentation pour la cartographie aérienne et détecter des correspondances inter-robot. Dans le cas des grilles d'occupation, on discrétise l'environnement en une grille 3D de voxels. Fondamentalement, chaque voxel observé est associé à une probabilité d'occupation, mais d'autre informations photométriques, voire sémantiques, peuvent y être adjointes. Contrairement aux représentations non structurées, les grilles d'occupation ont l'avantage de modéliser explicitement l'espace vacant ou inexploré. Cependant, la majorité de l'espace étant bien souvent libre, une allocation uniforme des voxels s'avère inutilement gourmande en ressources mémoire. C'est pourquoi Octomap (Wurm et al. 2010) adopte une allocation dynamique, hiérarchique et multi-résolue. La grille de voxels est alors structurée en interne par un Octree (Meagher 1982) et adapte dynamiquement sa résolution au niveau de détails rencontré localement. Cependant, les grilles d'occupations ne sont pas directement adaptées aux algorithmes de planification de trajectoires qui requièrent les distances aux obstacles. C'est pourquoi (Lau et al. 2013) proposent de les augmenter par une carte de distances régulièrement mise à jour. Des résolutions élevées permettent d'obtenir un rendu visuel correct des surfaces (voir la Figure 6.1a), bien qu'intrinsèquement limité par la voxellisation de l'espace.

Les représentations basées maillage 3D

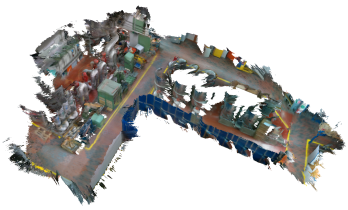
Les maillages 3D offrent une alternative aux grilles d'occupation afin de représenter explicitement les surfaces. En particulier, il est possible de générer des maillages à partir

de la triangulation de nuages de points, éventuellement éparses. C'est ce que permet la triangulation de Delaunay (Delaunay 1934), ou encore l'approche adoptée par (Ruetz et al. 2019) qui génère, à des fins de planification de trajectoires, un maillage des surfaces observées par le robot à partir des points non occlus. Cependant, de tels maillages sont difficiles à mettre à jour lorsque de nouvelles informations sont disponibles. Une solution alternative pour y pallier consiste alors à générer ces maillages à partir d'une représentation implicite intermédiaire basée sur des champs de distances signés (Oleynikova et al. 2016) (*Signed Distance Fields* (SDF) en anglais). Plusieurs champs de distances existent tels que les fonctions de base radiale (Carr et al. 2001), les champs de distances Euclidiens signés (*Euclidean Signed Distance Fields* (ESFD)) et les champs de distances signés et tronqués (*Truncated Signed Distance Fields* (TSDF)) (Curless et al. 1996). Leur formulation s'inscrit dans la continuité des grilles d'occupation 3D puisqu'elles reposent pareillement sur une voxellisation de l'espace. Dans le cas de l'ESDF, chaque voxel stocke la distance à l'obstacle le plus proche, tandis que dans le cas de la TSDF, il s'agit des distances projetées le long des axes d'observation. Les surfaces sont alors représentées implicitement par l'interpolation des zéros du champ de distances. Le maillage polygonal est ensuite généré par lancé de rayon (*raycasting*) – une technique empruntée aux milieux du jeu vidéo – ou plus communément à l'aide de l'algorithme *Marching Cube* (Lorensen et al. 1987). La Figure 6.1c montre un exemple de la représentation obtenue. L'intérêt de cette approche est triple : d'abord, les représentations implicites sont facilement mises à jour, ensuite le champs de distances fournit les informations nécessaires pour la navigation, et enfin, le maillage qui en résulte offre un visuel réaliste des surfaces. Les méthodes basées TSDF partagent avec les grilles d'occupation les problématiques d'allocation des voxels, étant donné que la majeure partie de l'espace est vacante. À l'instar d'Octomap, la solution adoptée est d'allouer les voxels dynamiquement et d'y accéder à l'aide de fonctions de hachage comme proposé par (Niessner et al. 2013). Cette astuce réduit les temps d'accès aux voxels en comparaison à Octomap qui, pour ce faire, doit parcourir l'arbre sous-jacent depuis sa racine. Cette idée est reprise par la bibliothèque OpenChisel (Klingensmith et al. 2015) qui organise les voxels en paquets alloués dynamiquement selon l'étendue de la cartographie. La proximité entre les grilles d'occupation et les champs de distances est illustrée par la méthode introduite par (Vespa et al. 2018) ; elle propose une représentation d'octree unificatrice capable de construire au choix une représentation TSDF ou une grille d'occupation.



(a) Grille d'occupation générée par Octomap

(b) Nuage de surfels généré par ElasticFusion



(c) Maillage généré par Voxblox

FIGURE 6.1 – Exemples de représentations cartographiques denses.

6.2.4 SLAM et cartographie dense

L'émergence des méthodes de SLAM dense coïncide avec l'augmentation des moyens de calculs, qui les a rendues compatibles avec les contraintes du temps réel. Dans le contexte du SLAM, les représentations cartographiques doivent idéalement vérifier les trois propriétés suivantes. Tout d'abord, elles doivent se prêter à une construction incrémentale. Ensuite, on souhaite qu'elles fournissent un modèle d'alignement pour l'estimation des poses relatives entre les acquisitions consécutives ou lors de la détection de fermetures de boucles. Enfin, on voudrait qu'elles soient suffisamment malléables pour maintenir leur consistance locale et globale. Dans les paragraphes qui suivent, nous verrons comment ces représentations denses satisfont ou non à ces critères et sont mises à profit par les méthodes de SLAM.

Cartographie dense en contexte mono-robot

Les représentations non structurées ont été parmi les premières à être adoptées en SLAM dense. Tout d'abord, les méthodes d'odométrie visuelle directes localisent et rétro-projettent des cartes de profondeurs dans l'espace. Elles les utilisent ensuite comme modèle de projection des pixels d'une image dans la suivante afin de formuler une erreur photométrique, qui elle-même dépend de la pose relative entre lesdites images. C'est le cas des algorithmes DTAM (NEWCOMBE et al. 2011a), REMODE (PIZZOLI et al. 2014), SVO (FORSTER et al. 2014) et LSD-SLAM (ENGEL et al. 2014). Dans ce dernier, la carte consiste en une collection de cartes de profondeur inverses semi-denses estimées en contexte monoculaire. Celles-ci sont raffinées localement à l'aide de méthodes de filtrage décrites dans (ENGEL et al. 2013), tandis que la consistance globale est maintenue à l'aide d'un graphe de poses dont les contraintes sont formulées sur $\mathbb{S}\mathrm{im}_3$. À côté des méthodes de SLAM directes, les SLAM basés LiDAR et stéréo travaillent sur des nuages

de points denses. On citait précédemment, à titre d'exemple, les méthodes LOAM-SLAM (Zhang et al. 2014) et IMLS-SLAM (Deschaud 2018) dans le cas LiDAR. La consistance locale est alors assurée par le recalage des scans successifs. Dans le cas stéréo, on se référera à la méthode proposée par (Brand et al. 2014). Étendue en une architecture basée sous-cartes dans (Brand et al. 2015), elle procède pareillement en détectant les correspondances entre les sous-cartes en appariant leurs primitives 3D puis en les recalant par *Iterative Closest Point* (ICP). Notons que la structuration de la carte globale en sous-cartes est un moyen de permettre sa consistance globale dans le cas d'une représentation rigide telle que les nuages de points denses. En revanche, d'autres méthodes misent sur la déformalité. C'est le cas d'ElasticFusion (Whelan et al. 2016), qui utilise des surfels, et introduit un graphe de déformation. La position de chaque surfel est alors influencée par celle de nœuds éparses avoisinants. Les auteurs introduisent également une fermeture de boucle locale opérant directement sur les modèles denses des surfaces, et qui se traduisent par des contraintes surfaciques sur les graphes de déformation. Les fermetures de boucles globales sont quant à elles détectées en comparant les prédictions des visuels des surfaces à partir de leurs modèles reconstruits. Elles débouchent également sur des contraintes surfaciques qui relaxent les graphes de déformations.

Dans le cas des grilles d'occupation, la consistance locale est recherchée par la mise à jour directe des probabilités d'occupation des voxels. Cependant, des méthodes telles qu'Octomap (Wurm et al. 2010) n'implémentent pas de fermetures de boucles globales. Le modèle de grille d'occupation est généré indépendamment de l'algorithme d'odométrie et ne sert pas de support aux modèles d'alignement locaux ou globaux. Toutefois, faisant office de structure sous-jacente à la formulation des champs de distances, les approches développées dans ce contexte en constituent une extension.

La mise à jour des maillages 3D est facilitée lorsqu'ils dépendent d'une représentation implicite sous-jacente. Les TSDF sont notamment propices au maintien de leur consistance locale à l'aide de leurs mécanismes de fusion qui seront décrits ultérieurement dans la section 6.4.2. Une première application au SLAM RGB-D en est proposée par l'algorithme KinectFusion (Newcombe et al. 2011b). Cependant, n'étant pas initialement développé par la navigation robotique, cet algorithme ne se prête pas à un fonctionnement embarqué : il fonctionne sur GPU et une cartographie même restreinte sature rapidement la mémoire disponible. Kintinous (Whelan et al. 2012) propose une extension de KinectFusion afin de cartographier des zones plus vastes en restreignant la TSDF à un volume glissant ; les parties qui sortent du volume sont alors figées en un maillage polygonal. Étendant les

travaux précédents, Voxblox (Oleynikova et al. 2017) propose de reconstruire l'environnement à l'aide d'une TSDF, à partir de laquelle une représentation ESDF est construite incrémentalement pour la navigation. Cependant, la détection de fermetures de boucles n'est pas assurée. Au contraire, dans BundleFusion (Dai et al. 2017), les auteurs gardent l'historique de l'intégration des cartes de profondeur impliquées dans la construction de la TSDF. Ils introduisent alors un mécanisme de dés-intégration et de ré-intégration pour impacter les changements induits par les optimisations globales. Néanmoins, ce mécanisme est relativement coûteux pour une application temps-réel. Une alternative consiste à segmenter le maillage global en sous-cartes recalées les unes par rapport aux autres. Par exemple, (Kähler et al. 2016) partitionnent l'environnement tout en minimisant les recouvrements entre les sous-cartes. La taille de la carte ne dépend alors que de l'étendue de l'environnement observé. Cependant, cette méthode s'avère complexe lors de la revisite des sous-cartes, d'autant plus qu'elle suppose une dérive négligeable. D'autres approches assurent la consistance globale en exploitant précisément les recouvrements entre les différentes sous-cartes. Cependant, la taille de la carte croît alors indéfiniment avec la longueur de la trajectoire (Wagner et al. 2014) ou le temps écoulé (Fioraio et al. 2015). C-Blox (Millane et al. 2018) s'inscrit dans cette logique mais limite la croissance de la carte en fusionnant les sous-cartes covisibles repérées selon la méthode présentée dans (Mei et al. 2010). Un graphe éparse est alors maintenu en parallèle pour la localisation et la détection des fermetures de boucles, basé sur une version modifiée d'ORB-SLAM 2 (Mur-Artal et al. 2017a). Couplant des représentations denses et des représentations éparses, C-Blox profite ainsi de leur atouts respectifs vis-à-vis de tâches spécifiques du SLAM. Ces travaux sont étendus par Voxgraph (Reijgwart et al. 2019) : contrairement à C-Blox, les contraintes de fermeture de boucles sont directement formulées sur les représentions ESDF.

Cartographie dense en contexte multi-robots

L'utilisation de la cartographie dense en contexte multi-robots a été relativement peu explorée dans la littérature, alors même que de nombreuses méthodes ont été introduites pour des cartographies éparses, telles que (Forster et al. 2013) et CCM-SLAM (Schmuck et al. 2017) dans le cas du SLAM monoculaire, ou encore de CVI-SLAM (Karrer et al. 2018) pour le SLAM visio-inertiel. En contexte dense, les premières méthodes multi-robots proposées étendent directement leurs analogues mono-robot. (Birk et al. 2006) et (Liu et al. 2013) proposent ainsi de fusionner des cartes mono-robot ba-

sées sur des grilles d'occupation. Dans le cas des nuages de points denses, (Nagatani et al. 2011) reprennent le concept de *manifold mapping* introduit dans (Howard 2004) et proposent de fusionner les cartes construites par plusieurs robots en détectant des correspondances avec l'algorithme *Iterative Closest Point* (ICP) entre des sous-cartes de taille limitée et associées à des nuages de points LiDAR. Cet algorithme est également utilisé par (Michael et al. 2012) afin de raffiner l'alignement de cartes terrestres et aériennes. (Schuster et al. 2019) appliquent une approche similaire au cas stéréo ; les auteurs combinent et étendent le *framework* multi-robots qu'ils ont introduits dans (Schuster et al. 2015) et la méthode fusion de sous-cartes exposée dans (Brand et al. 2015). Récemment, (Giubilato et al. 2020) ont proposé une méthode de relocalisation de sous-cartes basée sur des vecteurs sac-de-mots de descripteurs 3D SHOT binarisés. Enfin, dans le cas des cartographies par maillages, (Duhautbout et al. 2019) ont récemment proposé une architecture dans laquelle les robots s'échangent et recalent des sous-cartes basées sur des représentations TSDF, à l'aide d'ICP sur des nuages de points extraits des maillages polygonaux dès lors que leurs sous-cartes se recouvrent suffisamment.

6.3 Objectifs du chapitre

L'analyse de la littérature existante montre que les méthodes mono-robot de cartographie dense ont été profondément investiguées et maturées. Des méthodes multi-robots ont également été proposées dans le cas de cartographies basées sur des nuages de points denses à partir d'acquisitions LiDAR ou stéréo-visuelles. Concernant les cartographies denses basées TSDF, les méthodes de SLAM mono-robot exploitant les représentations TSDF ont bénéficié d'améliorations significatives et combinées par l'algorithme Voxgraph. Cette dernière laisse entrevoir la perspective de son extension directe en contexte multi-robots. À notre connaissance, la méthode (Duhautbout et al. 2019), développée à l'ONERA, est la première méthode multi-robots exploitant l'échange et la fusion de sous-cartes TSDF. Dans ce chapitre, nous en proposons une extension via des améliorations ponctuelles et l'adaptation de l'architecture multi-robots introduite dans le chapitre précédent, et dont les contributions sont les suivantes. Tout d'abord, nous rendons plus flexible la partie *Back-End* grâce au formalisme des graphes de facteurs qui permet d'une part une intégration unifiée des facteurs dérivés de l'odométrie, des correspondances inter-robot ainsi que des observations directes, et d'autre part de lisser l'ensemble de la trajectoire et de la cartographie à l'aide d'inférence globale, là où la méthode initiale se contentait de propager

des corrections cumulées au fil des correspondances détectées sur les patches ultérieures, et sans pour autant corriger les estimées de trajectoires. Nous proposons également des modifications sur l'algorithme initial de détection de correspondances afin de le rendre plus robuste face aux fausses détections. Enfin, en contexte multi-robots, nous proposons un protocole de communication basé sur le maintien d'historiques de communications afin de gérer les éventuelles pertes de contact. Nous complétons enfin l'ensemble d'un module d'alignement afin d'éliminer l'hypotèse de connaissance a priori des référentiels d'odométrie des robots. Nous discutons à la fin du chapitre des perspectives d'amélioration de la méthode proposée, et des façons dont elle pourrait s'enrichir des travaux récemment publiés dans le domaine.

6.4 Description de la méthode développée

6.4.1 Vue d'ensemble de la méthode dévéloppée

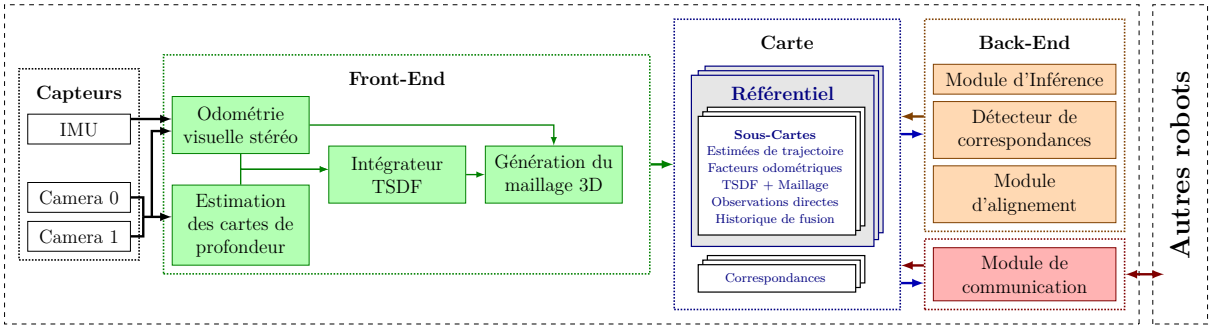


FIGURE 6.2 – Vue d'ensemble de la méthode de cartographie densée développée

Une vue d'ensemble de la méthode développée est proposée par la Figure 6.2. Chaque robot est muni d'un banc de caméras stéréo-visuel ainsi que d'une centrale inertielle. Le corps de l'algorithme se subdivise classiquement entre les processus *Front-End* et *Back-End* qui opèrent sur la carte à des cadences différentes. Les paires d'images stéréo sont d'abord utilisées par deux modules parallèles du *Front-End*. D'une part, un module d'odométrie stéréo-visuelle estime la trajectoire parcourue ; les mesures inertielles sont seulement utilisées à l'initialisation pour aligner la carte par rapport à la verticale. Nous utilisons l'algorithme eVO (SANFOURCHE et al. 2013). Cependant, tout autre algorithme d'odométrie stéréo pourrait s'y substituer, dès lors qu'il estime également les matrices de covariances des poses relatives entre deux images clés successives. En parallèle, les paires

d'images stéréos servent à estimer des cartes de profondeur. Nous utilisons l'algorithme *Efficient Large-Scale Stereo Matching* (Geiger et al. 2010). Les cartes de profondeur produites sont localisées à l'aide des estimées de pose fournies par eVO puis transmises à un intégrateur qui construit incrémentalement une représentation TSDF pour la sous-carte courante. Sur la base de critères sur la trajectoire estimée par eVO, la sous-carte courante est close tandis qu'une nouvelle sous-carte est entamée pour intégrer les mesures ultérieures. Le maillage associé à la sous-carte tout juste terminée est alors calculé, puis la sous-carte est incluse dans la carte. La carte consiste en une collection de sous-cartes, regroupées selon leur référentiel d'odométrie. Les modules *Back-End* permettent d'affiner la carte à l'aide d'inférences globale et en détectant des correspondances supplémentaires entre elles via le recalage de nuages de points extraits du maillage de leur surface. La détection de correspondances entre deux sous-cartes peut justifier leur fusion.

La communication entre les robots est gérée par un module dédié. La méthode multi-robots proposée implémente une allocation complètement décentralisées des tâches et des données. Elle distribue indirectement le calcul des correspondances inter-robot en ce sens que les robots se communiquent les correspondances détectées afin d'épargner d'éventuelles redondances de calculs. La politique de communication qui s'ensuit admet une topologie décentralisée. Les sous-cartes, diminuées de leur maillage, constituent l'unité d'échange élémentaire entre les robots. Les robots communiquent également les correspondances qu'ils détectent et indiquent si celles-ci donnent lieu à une fusion entre deux sous-cartes. On propose de réguler la communication sur la base d'un historique de communication afin de gérer les éventuels pertes et recouvrements de contact entre les robots. Enfin, la stratégie d'association et de fusion de données repose sur deux étapes essentielles. D'abord, un module d'alignement exploite les observations directes entre les robots afin d'estimer les transformations relatives entre les référentiels d'odométrie des robots, à l'instar de ce qui est fait dans (Schuster et al. 2015). Dès lors que ces transformations relatives sont connues, les correspondances entre les sous-cartes qui en dépendent peuvent être investiguées par le détecteur.

6.4.2 Le module d'odométrie et de cartographie locale

Odométrie et segmentation de la trajectoire en sous-cartes

L'estimation de la trajectoire à partir des paires d'images stéréo est assurée par l'algorithme eVO (Sanfourche et al. 2013). En guise d'initialisation, un nuage de points

3D est triangulé à partir d'une paire d'image stéréo, et celui-ci est aligné avec la verticale à l'aide des mesures inertielles éventuellement disponibles. Les coins de Harris (Shi 1994) extraits sur l'image de gauche sont ensuite suivis par la méthode de Kanade-Lucas (Baker et al. 2004) dans les images de gauche des paires suivantes. On obtient alors un jeu de correspondances 2D-3D à partir duquel une pose relative par rapport à la dernière image clé est estimée en résolvant un problème P3P. On calcule la covariance associée à partir de la matrice de Fisher dérivée des correspondances 2D-3D. Lorsque le nombre de points suivis passe en deçà d'un certain seuil, alors la paire d'images courante est sélectionnée pour trianguler un nouveau nuage de points de référence. Les estimées fournies par eVO définissent ainsi des facteurs odométriques qui permettent de construire incrémentalement un graphe de facteurs pour la sous-carte courante.

Nous adoptons une procédure de création de sous-cartes similaire au travail de (Duhautbout et al. 2019). Au regard de la littérature, les sous-cartes s'avèrent un moyen simple qui sied particulièrement aux représentations denses pour en garantir la consistance locale et globale. En effet, la dérive odométrique reste bornée en leur sein, et peut être corrigée en recalant les sous-cartes les unes par rapport aux autres. Ainsi, la trajectoire est segmentée en ligne en une succession de sous-cartes sur la base de critères cinématiques. Chaque nouvelle image clé ajoutée vient compléter la sous-carte courante, ou bien crée une nouvelle sous-carte. On crée une nouvelle sous-carte dès lors que la distance parcourue ou que les changements d'orientation cumulés au sein de la sous-carte courante excèdent des seuils respectifs $l_{\max}$ et $\theta_{\max}$, définis par l'utilisateur, afin de borner la dérive odométrique comme expliqué précédemment. La covariance associée à chaque nouvelle pose d'image clé est calculée en propageant les incertitudes le long des facteurs odométriques.

Estimation des cartes de profondeur

En parallèle de l'estimation de la trajectoire, les paires d'images stéréo sont utilisées pour estimer les cartes de profondeur nécessaires afin de construire ultérieurement le champ de distances associé à la sous-carte. À cette fin, on utilise l'algorithme *Efficient Large Scale Stereo Matching* (ELAS) (Geiger et al. 2010). À partir d'une paire d'images stéréo rectifiée, celui-ci extrait d'abord une carte de disparités qui associe à chaque pixel de l'image de gauche la distance algébrique d le séparant de son homologue sur l'image de droite le long de leur ligne épipolaire. Le calcul d'une carte de disparités dense est une tâche complexe. Dans l'optique d'applications en temps réel, ELAS propose d'approximer la carte de disparités dans un cadre Bayésien. Pour cela, il explicite et maximise parallè-

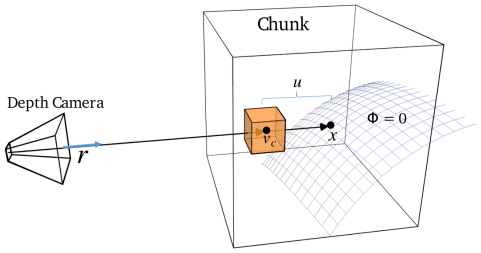
lement pour chaque observation $\boldsymbol{x}_n^{(g)}$ sur l'image de gauche une distribution *a posteriori* $p(d_n|\boldsymbol{x}_n^{(g)}, \mathcal{X}_n^{(d)}, \mathcal{S})$ où d_n est la disparité, $\mathcal{X}_n^{(d)}$ est l'ensemble des observations appartenant à la ligne épipolaire induite par $\boldsymbol{x}_n^{(g)}$ sur l'image de droite, et $\mathcal{S}$ est un sous-ensemble éparse de points supports sur lesquels la disparité est préalablement calculée. Ce réseau de points support est ensuite maillé par triangulation 2D de Delaunay (Delaunay 1934) afin de définir une distribution *a priori* de la disparité $p(d_n|\boldsymbol{x}_n^{(g)}, \mathcal{S})$ pour chaque pixel de l'image de gauche par interpolation linéaire. La carte de disparité obtenue est enfin convertie en une carte de profondeur.

Cartographie dense associée aux sous-cartes

Le maillage des surfaces associées à chaque sous-carte est généré à partir d'une représentation implicite de type TSDF. Nous utilisons l'implémentation de la bibliothèque OpenChisel (Klingensmith et al. 2015) et dont nous exposons le principe ci-dessous. Fondamentalement, un champ de distances associe à tout point $\boldsymbol{x} \in \mathbb{R}^3$ de l'espace sa distance $\Phi(\boldsymbol{x}) \in \mathbb{R}$ à la surface la plus proche. Par convention, cette distance est *signée* positivement si le point $\boldsymbol{x}$ se situe entre le capteur qui l'observe et la surface détectée, et négativement dans le cas contraire. On définit alors un champ de distances Euclidien (ESDF pour *Euclidean Signed Distance Field*). Les surfaces $\mathcal{S}$ sont alors définies implicitement comme l'iso-surface $\mathcal{S} = \Phi^{-1}(\{0\})$. En contexte robotique, construire incrémentalement une telle représentation suppose 1) de discrétiser et d'indexer adéquatement l'espace afin d'en permettre une itération efficace pour les mises à jour ; 2) de tenir compte du bruit de mesure des capteurs de distance pour fusionner les mesures multiples d'un même point ; et enfin 3) d'extraire les surfaces explicitement sous la forme d'un maillage polygonal à partir du champ de distances.

La première problématique est celle de la discrétisation et de l'indexation spatiale. La technique classique consiste à représenter l'espace à l'aide d'une grille de voxels de résolution donnée, et d'évaluer le champ de manière discrète sur chaque voxel. Afin d'optimiser la gestion de la mémoire, des techniques d'allocation dynamiques, basées sur des octrees (Wurm et al. 2010) ou des volumes glissants (Whelan et al. 2012) ont été proposées. Afin d'optimiser l'accès aux voxels, OpenChisel reprend la structure introduite par (Niessner et al. 2013) qui consiste à organiser les voxels en paquets cubiques appelés *chunks* et référencés via une fonction de hachage spatiale, permettant un accès aux voxels en temps constant. Les *chunks* sont alloués dynamiquement et mis à jour dès lors qu'ils intersectent un volume représentant de façon simplifié le champ de mesure de la caméra.

La seconde problématique concerne l'estimation des distances en chaque voxel. Chaque pixel de la carte de profondeur, de valeur z, définit un rayon rétro-projeté depuis le centre optique de la caméra vers la scène observée, porté par un vecteur unitaire $\boldsymbol{r}$. Il apporte une information de distance aux surfaces sur tous les voxels traversés par ledit rayon. Cependant, aussi bien z que $\boldsymbol{r}$ sont entachés d'une incertitude qui doit être prise en compte lors de la mise à jour du champ de distances. Pour ce faire, on associe à chaque voxel $\boldsymbol{v}$ un poids positif $\hat{w}(\boldsymbol{v})$ qui rend compte de la fiabilité de son estimation de distance $\hat{d}(\boldsymbol{v})$. À chaque nouvelle mesure d'un voxel est également associé un poids $w_m(\boldsymbol{v})$ qui dépend de la distance mesurée $d_m(\boldsymbol{v})$ du voxel $\boldsymbol{v}$ au point d'impact du rayon sur la surface (c.f. Figure 6.3). La valeur du voxel est alors mise à jour par moyenne pondérée glissante de façon à minimiser la distance de l'iso-surface à tous les points d'impacts (Curless et al. 1996) :



L'obstacle est représenté par l'iso-surface $\Phi^{-1}(\{0\})$. Le voxel v_c observé est associé à la distance u au point d'impact $\boldsymbol{x}$ sur l'iso-surface.

Figure 6.3 – Principe de la TSDF (extrait de (Klingensmith et al. 2015))

$$\begin{cases} \hat{d}(\boldsymbol{v}) & \leftarrow & \dfrac{\hat{w}(\boldsymbol{v}) \cdot \hat{d}(\boldsymbol{v}) + w_m(\boldsymbol{v}) \cdot d_m(\boldsymbol{v})}{\hat{w}(\boldsymbol{v}) + w_m(\boldsymbol{v})} \\ \hat{w}(\boldsymbol{v}) & \leftarrow & \hat{w}(\boldsymbol{v}) + w_m(\boldsymbol{v}) \end{cases} \tag{6.1}$$

Un second point est que la distance d_m déduite de la carte de profondeur est une distance projetée sur l'axe d'observation ; elle n'approxime la distance du voxel à la surface qu'au voisinnage de celle-ci. C'est pourquoi, le champ de distance est *tronqué* :

$$\Phi_{\text{TSDF}}(\boldsymbol{x}) = \begin{cases} \Phi(\boldsymbol{x}) & \text{si } |\Phi(\boldsymbol{x})| < \tau \\ \text{non défini} & \text{sinon} \end{cases} \tag{6.2}$$

avec τ la distance de troncature. Cette dernière est définie dynamiquement dans OpenChisel en fonction de la profondeur z et de l'écart-type du bruit de mesure $\sigma(z)$: $\tau(z) = \beta \cdot \sigma(z)$ où β est un facteur d'échelle. On définit alors une zone d'intégration telle que $d_m \in [-\tau, \tau]$. Seuls les voxels qui y appartiennent sont mis à jour selon l'équation (6.1). Afin d'atténuer les inconsistances dues à l'intégration de données bruitées ou à l'observation d'objets dynamiques, OpenChisel définit de plus une zone de "ciselage" (*space carving*) qui inclut les voxels suffisamment proches de la caméra en amont de la zone d'intégration. Les éven-

tuels voxels signés négativement qui s'y trouveraient sont alors réinitialisés. La nécessité de tronquer le champ de distances et l'approximation par la distance projective constituent les deux différences majeures entre la TSDF et l'ESDF. À des fins de navigation, certaines méthodes telles que (Oleynikova et al. 2017) proposent de reconstruire l'ESDF à partir de la TSDF. Enfin, il est possible d'ajouter des informations colorimétriques à chaque voxel ; elles sont gérées de la même façon que les informations de distance.

Enfin, la troisième problématique concerne la génération du maillage polygonal associé à la TSDF construite. OpenChisel utilise l'algorithme du "cube glissant" ou *MarchingCube* (Lorensen et al. 1987). Celui-ci approxime l'iso-surface de valeur nulle de la TSDF en parcourant la grille de voxels à l'aide d'un cube virtuel. Selon les valeurs du champ de distance en les coins du cube, il décide s'il doit créer un polygone pour représenter l'isosurface, puis le cas échéant il en définit la morphologie. Il existe alors $2^8 = 256$ configurations possibles selon que les coins sont signés positivement et négativement. Chaque cas est pré-calculé et correspond à un polygone particulier. Celui-ci est alors mis à l'échelle en positionnant ses sommets selon les arrêtes du cube via une interpolation linéaire des valeurs aux sommets du cube.

Dans la méthode proposée dans ce chapitre, l'intégrateur TSDF recupère les cartes de profondeur estimées par ELAS, puis les associe aux poses estimées par eVO préalablement ramenées dans le référentiel attaché à la sous-carte courante. Dès que la sous-carte est close selon les critères de distance et d'orientation cumulées précédemment définis et vérifiés sur la base des estimations d'eVO, la sous-carte est mise en attente afin que son maillage polgyonal soit calculé. Dès que c'est le cas, il peut alors servir pour la détection de correspondances. On identifie enfin chaque sous-carte de façon unique à l'aide d'un identifiant aggrégeant celui du robot qui l'a produite, ainsi que son index.

6.4.3 Le module de fermeture de boucles

Si la consistance locale de la carte est assurée par son découpage en sous-cartes successives, sa consistance globale est assurée en compensant la dérive odométrique accumulée le long de la trajectoire. À cette fin, on détecte des correspondances structurelles entre les sous-cartes afin d'estimer leur pose relative. La méthode d'alignement que nous proposons reprend les étapes introduites par (Duhautbout et al. 2019).

Chaque nouvelle sous-carte dont on a généré le maillage polygonal est signalée au module de fermetures de boucles sous forme d'une requête pour détecter ses éventuelles

Algorithme 3 – Détection de fermetures de boucles

Sous-carte source : $\mathcal{S}_i$;
Trouver la sous-carte $\mathcal{S}_j$ de plus fort recouvrement et vérifier que le recouvrement volumique excède $v_{\min}$;
Extraire les nuages de points $\mathcal{S}_i$ et $\mathcal{S}_j$ de la zone de recouvrement;
Estimer la transformation $T_{\mathrm{S}_i\mathrm{S}_j}$ par *Iterative Closest Point*;

si *l'ICP converge* **alors**
- Vérifier que le nombre de correspondances valides excède $\#_{\text{inliers}}$;
- Vérifier que l'erreur RMSE sur les correspondances valides est inférieure à $\text{rmse}_{\max}$;
- Vérifier que l'erreur angulaire moyenne sur les normales n'excède pas $\angle_{\max}$;
- Vérifier que $T^{\text{ICP}}_{\mathrm{S}_i\mathrm{S}_j}$ est consistante avec les incertitudes sur les poses des sous-cartes recalées;
- Calculer la matrice de covariance $\mathbf{\Sigma}_{T^{\text{ICP}}_{\mathrm{S}_i\mathrm{S}_j}}$;
- Ajouter le facteur de pose relative au graphe de facteurs;
- **si** *le recouvrement volumique excède* $v^{fusion}_{\min}$ **alors**
  - Fusionner la TSDF de $\mathcal{S}_i$ avec celle de $\mathcal{S}_j$;

correspondances avec d'autres sous-cartes. Les requêtes sont traitées par ordre d'arrivée. Considérons une sous-carte $\mathcal{S}_i$. La première étape consiste à identifier la sous-carte $\mathcal{S}_j$ présentant le plus fort recouvrement volumique avec $\mathcal{S}_i$. Nous excluons de cette recherche les sous-cartes consécutives $\mathcal{S}_{i-1}$ et $\mathcal{S}_{i+1}$. Nous ignorons de même les sous-cartes qui auraient déjà fait l'objet d'un alignement avec $\mathcal{S}_i$. Pour ce faire, on associe à chaque sous-carte une boîte englobante de son maillage, alignée selon les axes du repère global, abrégée AABB par la suite pour *Axis-Aligned Bounding Box*. Il s'agit certes d'une approximation grossière de l'étendue spatiale d'une sous-carte. Néanmoins, elle permet une estimation simple et rapide des recouvrements entre une paire de sous-cartes. On estime l'intersection de la boîte englobante de la sous-carte $\mathcal{S}_i$ avec les boîtes englobantes de l'ensemble des autres sous-cartes. Une méthode plus fine consisterait à se restreindre aux sous-cartes appartenant à un voisinage donné de $\mathcal{S}_i$ à l'aide d'un indexage spatial des sous-cartes. On retient la sous-carte $\mathcal{S}^{\max}_j$ qui maximise $v_{ij} = \text{volume}\left(\text{AABB}_i \cap \text{AABB}_j\right)$. La procédure d'alignement de la sous-carte n'est poursuivie que si $v_{ij} \geq v_{\min}$ où $v_{\min}$ est un seuil fixé par l'utilisateur.

Une fois la sous-carte de plus fort recouvrement identifiée, l'étape suivante consiste à recaler les deux sous-cartes les unes par rapport aux autres. La procédure retenue exploite directement la représentation dense des surfaces afin de détecter les correspondances. Elle repose sur un alignement modèle contre modèle (Fioraio et al. 2015) utilisant l'algorithme *Iterative Closest Point* (ICP) (Besl et al. 1992). Il s'agit d'un algorithme itératif

qui, étant donné deux nuages de points $\mathcal{P}_{\text{source}}$ et $\mathcal{P}_{\text{cible}}$ présentant des parties communes, estime la transformation $T_{\text{source}\to\text{cible}} \in \mathbb{SE}_3$ à appliquer au nuage source pour l'aligner sur le nuage cible (c.f. Figure 6.4). À chaque itération, l'algorithme construit un jeu de correspondances en appariant des points source à des points cibles, minimise un critère de distance entre les points appariés, puis applique la transformation estimée.

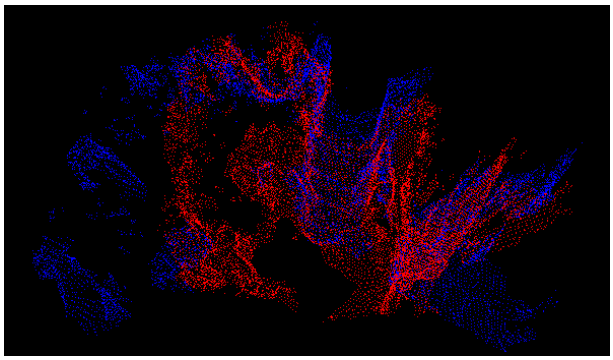


FIGURE 6.4 – Exemple d'une paire de nuages de points appariés dans EuRoC (BURRI et al. 2016)

L'étape préalable implique de construire les deux nuages de points à recaler. Connaissant la zone de recouvrement entre les deux sous-cartes $\mathcal{S}_i$ et $\mathcal{S}_j$ ainsi que leur pose initiale, nous identifions leurs *chunks* communs que nous extrayons de part et d'autre. L'alignement des TSDF est une opération analogue au mécanisme de fusion décrit à la fin de cette section. Nous extrayons les nœuds et les normales associées, de façon à former deux nuages de points $\mathcal{P}_i$ et $\mathcal{P}_j$, exprimés dans le référentiel attaché à $\mathcal{S}_j$. $\mathcal{P}_i$ désigne le nuage source tandis que $\mathcal{P}_j$ est le nuage cible. À chaque itération, les correspondances sont identifiées de façon réciproque en utilisant le critère du plus proche voisin. Nous filtrons ensuite ces correspondances en seuillant leur distances. Enfin, la transformation optimale est estimée dans une boucle de RANSAC (FISCHLER et al. 1981) en minimisant un critère point-à-plan (CHEN et al. 1992). Contrairement au critère point-à-point (BESL et al. 1992) qui minimise la distance quadratique entre les points appariés, ce critère minimise la distance quadratique du point source au plan tangent à la surface cible au point point cible correspondant. Cette formulation accélère la convergence de l'ICP, mais impose une résolution itérative là où le critère point-à-point admet une solution analytique (UMEYAMA 1991).

La troisième étape de l'alignement consiste à évaluer la solution résultat de l'ICP. La qualité du recalage par ICP est fortement tributaire de l'alignement initial. En effet, s'il n'appartient au bassin de convergence associé à l'optimum global, alors l'algorithme convergera vers un minimum local et débouchera sur un recalage erroné. C'est pourquoi on doit prendre garde de ne pas initialiser l'ICP trop loin de son minimum global. Dans la méthode proposée, nous justifions l'usage direct de l'ICP par l'hypothèse de faible dérive de l'odométrie stéréo-visuelle. L'ICP pourrait être précédée d'une étape de pré-

alignement. Néanmoins, les tests réalisés en utilisant des descripteurs 3D classiques tels que les coins de Harris 3D ou les points ISS ne se sont pas révélés concluants en raison de la faible densité du maillage. Une solution, non explorée, pourrait être d'utiliser les descripteurs de nuages de points éparses développés par (Cieslewski et al. 2016). Pour éviter d'intégrer des fermetures de boucles erronées, nous introduisons quatre tests réalisés en aval de l'ICP. Tout d'abord, nous vérifions que la pose relative estimée est corroborée par un minimum correspondances valides dont le seuil est noté $\#_{\text{inliers}}$. Le second test vérifie que l'erreur RMSE calculée sur les correspondances valides n'excède pas un seuil $\text{rmse}_{\max}$. Le troisième test vise à s'assurer de la cohérence des correspondances établies en vérifiant que la moyenne des angles entre leur normale est en deçà d'un seuil $\angle_{\max}$. Enfin, le dernier test consiste à vérifier que la pose relative estimée est cohérente avec l'incertitude estimée sur les poses des sous-cartes recalées. On considère les matrices de covariance globales $\mathbf{\Sigma}_{T_{\text{WS}_i}}$ et $\mathbf{\Sigma}_{T_{\text{WS}_j}}$ respectivement associées aux poses T_{WS_i} et T_{WS_j}. On définit $\hat{T}_{\text{S}_i\text{S}_j} = T_{\text{WS}_i}^{-1} \oplus T_{\text{WS}_j}$, et on approxime la matrice de covariance associée ainsi :

$$\mathbf{\Sigma}_{\hat{T}_{\text{S}_i\text{S}_j}} = \boldsymbol{J}_{T_{\text{WS}_i}}^{\hat{T}_{\text{S}_i\text{S}_j}} \cdot \mathbf{\Sigma}_{T_{\text{WS}_i}} \cdot \boldsymbol{J}_{T_{\text{WS}_i}}^{\hat{T}_{\text{S}_i\text{S}_j}\top} + \boldsymbol{J}_{T_{\text{WS}_j}}^{\hat{T}_{\text{S}_i\text{S}_j}} \cdot \mathbf{\Sigma}_{T_{\text{WS}_j}} \cdot \boldsymbol{J}_{T_{\text{WS}_j}}^{\hat{T}_{\text{S}_i\text{S}_j}\top} \tag{6.3}$$

où $\boldsymbol{J}_{T_{\text{WS}_i}}^{\hat{T}_{\text{S}_i\text{S}_j}}$ et $\boldsymbol{J}_{T_{\text{WS}_j}}^{\hat{T}_{\text{S}_i\text{S}_j}}$ désignent respectivement les matrices jacobiennes de $\hat{T}_{\text{S}_i\text{S}_j}$ par rapport à T_{WS_i} et T_{WS_j}, explicitée dans l'Annexe C. Une approximation supplémentaire est faite en négligeant l'influence des covariances croisées entre T_{WS_i} et T_{WS_j} qui sont donc traitées indépendamment. La consistance de l'estimée est vérifiée à l'aide d'un test du χ^2 :

$$\left\| T_{\text{S}_i\text{S}_i}^{\text{ICP}} \boxminus \hat{T}_{\text{S}_i\text{S}_j} \right\|^2_{\mathbf{\Sigma}_{T_{\text{S}_i\text{S}_j}}} \leq \chi^2_{6,95\%} \tag{6.4}$$

Dans l'équation précédente, $\chi^2_{6,95\%}$ désigne le fractile à 95% d'une distribution du chi-deux à 6 degrés de liberté. Si les quatres tests précédents sont passés avec succès, alors l'alignement est considéré comme valide.

L'étape suivante consiste à impacter la correspondance détectée sur le graphe de facteurs sous la forme d'une nouvelle contrainte de pose relative. Pour ce faire, on doit caractériser l'incertitude sur l'estimée de pose relative $T_{\text{S}_i\text{S}_j}^{\text{ICP}}$ à l'aide d'une matrice de covariance. En contexte robotique, une première catégorie de méthodes utilise des formes analytiques de la covariance basée sur la linéarisation de la fonction de coût de l'ICP au point de convergence. C'est le cas de la formule de Censi (Censi 2007). Cependant, elle omet de propager l'incertitude sur la transformation initiale, quant aux mauvaises

convergences qu'elle peut provoquer. C'est pourquoi (Brossard et al. 2019) proposent de propager cette incertitude à l'aide d'une transformée sans parfum (Uhlmann 1995). Toutefois, appliquer comme suggéré l'ICP sur les 12 points sigmas s'avèrerait rédhibitoire vis-à-vis des contraintes de temps réel, d'autant plus qu'on peut supposer qu'un mélange de Gaussiennes serait vraisemblablement plus adapté pour modéliser la distribution englobant plusieurs bassins de convergence potentiels qu'une seule loi Gaussienne. Dans la méthode proposée, nous reprenons la technique proposée par (Brand et al. 2015) dans le cas stéréo-visuel. Nous estimons alors la matrice de covariance comme une matrice diagonale, dont les coefficients sont calculés à partir de l'erreur RMSE calculée sur les distances des plus proches voisins entre les deux nuages recalés :

$$\sigma_{\{\mathrm{x,y,z}\}} = \max\left(\mathrm{RMSE}_{\mathrm{ICP}}, \sigma_t^{\min}\right) \tag{6.5}$$

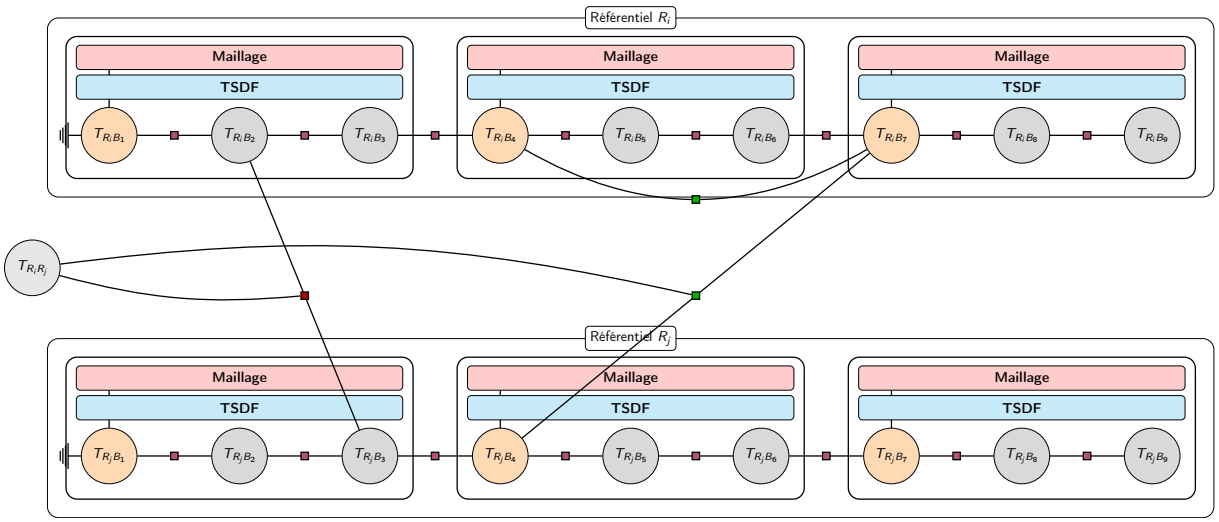
$$\sigma_{\{\mathrm{roulis,tangage,lacet}\}} = \max\left(\arctan\left(\frac{2 \cdot \mathrm{RMSE}_{\mathrm{ICP}}}{d_{\{\mathrm{yz,xz,xy}\}}}\right), \sigma_\theta^{\min}\right) \tag{6.6}$$

où d_{yz}, d_{xz} et d_{xy} sont les diamètres respectifs du recouvrement des boîtes englobantes des deux sous-cartes, et où $\sigma_t^{\min}$ et $\sigma_\theta^{\min}$ sont deux paramètres introduits pour éviter les estimations trop confiantes.

Enfin, la dernière étape consiste à éventuellement fusionner les maillages des deux sous-cartes. En cas d'ICP valide, et si le recouvrement volumique des deux sous-cartes excède un seuil $v_{\min}^{\mathrm{fusion}} > v_{\min}$, alors on fusionne la TSDF de la sous-carte source dans celle de la sous-carte cible. Nous reprenons la technique de fusion de TSDF introduite par (Duhautbout et al. 2019). Celui-ci nécessite d'abord de déterminer les voxels de la TSDF_j qui seront impactés par la fusion en projetant les coins et les centres des voxels de TSDF_i en utilisant la transformation $\hat{T}_{\mathrm{S}_i\mathrm{S}_j}^{\mathrm{ICP}}$. Enfin, les valeurs de TSDF_i aux centres des voxels de TSDF_j sont calculées par interpolation trilinéaire, puis fusionnées selon l'équation (6.1). Après la fusion, les deux sous-cartes demeurent des entités distinctes dans le graphe de facteurs. Cependant, les deux sous-cartes pointent vers la même TSDF pour les détections de correspondance futures. La TSDF fusionnée est ensuite supprimée de la mémoire. Chaque sous-carte conserve un historique des fusions qui liste les identifiant des sous-cartes dont les maillages ont été fusionnés avec le sien.

6.4.4 Le module d'inférence globale

Assurer la consistance globale de la carte impose d'une part de détecter les correspondances entre les sous-cartes, et d'autre part de les répercuter sur l'estimation des trajectoires, et en retour sur la cartographie elle-même. C'est le rôle du module d'inférence globale. Ce module n'existe pas dans la méthode proposée par (Duhautbout et al. 2019) : l'estimation de la trajectoire repose sur la seule odométrie stéréo-visuelle et ne bénéficie pas des correspondances détectées. Les corrections sur les poses des sous-cartes sont alors combinées en une pose de dérive cumulée au fil des correspondances détectées puis utilisées pour pré-corriger les poses des sous-cartes ultérieures. Outre le fait que la trajectoire ne soit pas corrigée, les poses des sous-cartes précédentes ne sont pas réestimées rétrospectivement au regard de l'information apportée par des correspondances impliquant des sous-cartes ultérieures En ce sens, la méthode proposée par (Duhautbout et al. 2019) relève davantage de la cartographie que du SLAM.



Légende : ■ facteurs odométriques, ■ les facteurs dérivés des correspondances entre sous-cartes, ■ les observations inter-robot.

Figure 6.5 – Structure du graphe de pose impliquant deux robots i et j

Nous proposons d'ajouter un module d'inférence globale reposant sur un modèle probabiliste sous-jacent classiquement matérialisé par un graphe de facteurs, tel que représenté par la Figure (6.5) dans un contexte multi-robots. Cette représentation jouit d'une grande flexibilité et fournit un formalisme unifié afin d'intégrer les facteurs dérivés de l'odométrie visuelle, des observations directes entre les robots ainsi que les détections entre les

sous-cartes. L'ensemble est classiquement résolu en utilisant l'algorithme de Levenberg-Marquardt. Par ailleurs, afin d'accroître la résilience de l'estimation face à d'éventuelles fermetures de boucles erronées, ces dernières sont mitigées à l'aide de fonctions de Cauchy de paramètres $\chi^2_{6,95\%}$. Cette technique, également utilisée par (Schuster et al. 2019), permet de réduire en aval l'influence des correspondances douteuses qui n'auraient pas été filtrés précédemment. Dans l'approche proposée, une optimisation de graphe de poses est exigée dès lors qu'une correspondance a été caractérisée entre deux sous-cartes, ou lorsque qu'une correspondance est reçue d'un autre robot. Les poses des sous-cartes et des maillages polygonaux sont mis à jour en conséquence à l'issue de l'optimisation. Enfin, une fois l'inférence terminée, on estime les matrices de covariance sur les poses des sous-cartes à partir du calcul de la matrice de Fisher associée à la fonction de coût évaluée aux estimées résultantes. Ces matrices de covariances sont nécessaires pour le test de consistance lors des détections de correspondances.

La structure du graphe présenté sur la Figure 6.5 propose une topologie séquentielle, ici permise par le fait qu'eVO fournit les estimées de poses relatives entre les images clés successives ainsi que leur covariance associée. On pourrait cependant aussi bien adopter une topologie non séquentielle, à l'instar de celle proposée par (Schuster et al. 2015), et qui sied particulièrement aux modules d'odométrie basées filtrage.

6.4.5 Extension en contexte multi-robots

Les sections précédentes ont décrit le fonctionnement de la méthode dans une optique mono-robot. En contexte collaboratif, nous devons préciser quelle architecture nous retenons afin d'organiser le calcul, l'échange et l'intégration des paquets de données entre les robots. Dans les paragraphes suivants, nous précisions donc l'allocation des tâches et des données choisie, quelle politique de communication régit les échanges entre les robots, et enfin comment les données reçues sont associées puis fusionnées.

Allocation des tâches et des données

La méthode proposée conserve l'allocation complètement décentralisée proposée par (Duhautbout et al. 2019). Tout d'abord, chaque robot maintient sa propre version de la carte globale dans laquelle il inclut les sous-cartes qu'il reçoit des autres robots. De même, chaque robot recherche indépendamment les correspondances entre ses sous-cartes et optimise l'ensemble de sa carte. Cette décentralisation se justifie par le gain d'autonomie

qu'elle procure aux agents, d'autant plus que les informations cartographiques peuvent s'avérer utiles pour chacun d'entre eux dans une optique de planification de trajectoires. Dans ce cas, la représentation TSDF doit alors être étendue en une représentation ESDF. Une architecture centralisée se justifierait cependant par le fait que la recherche des correspondances est une opération coûteuse qui, comme nous le verrons, empêche les robots de tester toutes les correspondances potentielles.

Politique de communication

La seconde composante d'une méthode de SLAM multi-robots est la politique de communication. Nous proposons une politique basée sur deux régimes de communication : un régime standard qui prévaut pour les robots évoluant à portée de communication, et un régime de régularisation qui prend le relais lorsque deux robots recouvrent le contact après une période d'interruption. Dans les deux cas, les sous-cartes constituent l'unité d'échange élémentaire. Les paquets standards comprennent pour chaque sous-carte le sous-graphe de facteurs associé, l'horodatage associé à chaque pose estimée ainsi que la TSFD sérialisée. Nous y adjoignons également les facteurs de pose relative dérivés des observations directes d'autres robots, ainsi que l'historique de fusions. Afin de limiter la quantité de données transmises et d'éviter les redondances, on ne communique pas le maillage associé, qui devra être reconstruit par le robot récepteur. Seule la représentation implicite de type TSDF est communiquée, l'avantage étant que celle-ci contient les informations suffisantes pour des opérations ultérieures de fusion. En régime standard, chaque robot diffuse chaque sous-carte qu'il clôt aux robots avec lesquels il est en contact. En plus des sous-cartes, il diffuse également aux autres robots les correspondances qu'il détecte entre les sous-cartes et si elles ont donné lieu à des fusions entre des sous-cartes, afin d'éviter de recherches redondantes.

Lorsqu'ils couvrent de vastes zones, les robots peuvent occasionnellement perdre contact les uns avec les autres. Quand ils se rencontrent de nouveau et ré-établissent la communication, alors ils doivent régulariser leurs connaissances de part et d'autre. Dans ce but, la même politique de recouvrement que celle décrite dans la section 5.3.2 peut être appliquée.

Association et fusion de données multi-robots

Quand un robot reçoit une sous-carte d'un autre robot, il l'intègre dans sa carte à la suite des autres sous-cartes reçues de ce même robot. Le sous-graphe de pose est connecté

au précédent, la covariance de la pose de la sous-carte est calculée par propagation de la covariance de la sous-carte précédente le long des facteurs odométriques. Enfin, la TSDF est désérialisée dans le référentiel attaché à la sous-carte et son maillage polygonal est généré avec l'algorithme *Marching Cube*. Les correspondances et les observations inter-robot directes associées sont ajoutées si possible, ou mises en attente dans un buffer dans le cas contraire. Ce même buffer est parcouru à la recherche de correspondances et des observations impliquant la sous-carte nouvellement reçue afin de vérifier si elles peuvent dès lors être intégrées à la carte.

Dans (Duhautbout et al. 2019), on intègre les sous-cartes reçues dans une carte publique, distincte de la carte privée qui regroupe exclusivement les sous-cartes du robot hôte. L'architecture de carte que nous adoptons ignore cette distinction entre carte publique et carte privée. Nous initialisons un nouveau référentiel par robot, puis nous fusionnons ces référentiels dès que nous disposons de suffisamment de correspondances inter-robot afin d'estimer la transformation relative qui les unit. Le module d'alignement fonctionne comme suit. Chaque nouvelle correspondance inter-robot intégrée à la carte lui est notifiée afin qu'il comptabilise le nombre de correspondances disponibles entre chaque paire de référentiels. S'il existe suffisamment de correspondances entre deux référentiels, alors on tente de les aligner. Chaque correspondance induit une hypothèse sur la transformation relative entre les deux référentiels. Pour chaque hypothèse, on calcule sa distance aux autres hypothèses pour identifier celles qui la corroborent. On retient finalement la transformation la plus consensuelle. Dans ce cas, les deux référentiels sont fusionnés, et le décompte des nombres de correspondances entre les référentiels est mis à jour.

Les opérations de détection de correspondances et d'inférence ne sont pratiquées sur le référentiel (hôte) du robot récepteur. Cependant, pour des raisons de temps de calculs, les correspondances ne sont pas recherchées de façon rétrospective, les sous-cartes nouvellement alignées sont simplement considérées dans les recherches ultérieures de correspondances avec les nouvelles sous-cartes du robot hôte. En effet, pour chaque nouvelle sous-carte hôte $\mathcal{S}_i$, on recherche la sous-carte $\mathcal{S}_j$ de plus fort recouvrement produite prar le robot lui-même, ainsi que la sous-carte reçue de plus fort recouvrement, quel que soit le robot auquel elle se rattache. Il s'agit d'une limitation significative qui occasionne la perte d'informations portentiellement intéressantes. En revanche, le fait que chaque robot se restreigne à calculer les correspondance entre ses propres sous-cartes et celles des autres robots puis les communique aux autres agents induit un partage tacite des tâches entre les robots. Enfin, les opérations d'inférence sont uniquement menées sur le référentiel hôte

dès lors qu'une nouvelle correspondence est intégrée. Elles pourraient cependant être déclenchées de façon ciblée, spécifiquement sur certains référentiels sur la base des mêmes conditions de déclenchement.

6.5 Résultats expérimentaux

6.5.1 Scénarios d'évaluation

Nous avons évalué l'algorithme proposé sur des scénarios multi-robots construit à partir des séquences du jeu de données EuRoC (Burri et al. 2016). Pour ce faire, nous avons d'une part synchronisé les séquences individuelles, puis d'autre part simulé les observations directes entre les robots. Nous avons équippé les drones de petits marqueurs AprilTags virtuels (Olson 2011), puis nous avons simulé les interactions à partir des trajectoire de la vérité terrain. Au temps t, nous identifions la pose T_{WC_i} de la caméra C_i du robot i utilisée pour détecter les marqueurs – W étant le référentiel global –, puis nous calculons la pose $T_{\mathrm{WT_j}}$ de chaque marqueur T_j. Pour chaque robot i et chaque marqueur j, nous en déduisons la pose $T_{\mathrm{C}_i\mathrm{T}_j}$, puis procédons à plusieurs tests pour vérifier l'observabilité du marqueur par la caméra. Connaissant les dimensions du marqueur, nous vérifions d'abord que chacun des coins du marqueur se projette dans le plan image de la caméra. Si c'est le cas, nous vérifions que $\|{}^{C_i}\boldsymbol{p}_{C_iT_j}\| \leq d_{\max}$. Enfin, nous imposons une condition sur l'orientation relative $\angle\left({}^{T_j}\boldsymbol{z}, {}^{T_j}\boldsymbol{u}_{T_jC_i}\right) \leq \theta_{\max}$ où ${}^{T_j}\boldsymbol{u}_{T_jC_i}$ dirige l'axe d'observation. Si ces trois conditions sont vérifiées, alors nous simulons la mesure AprilTag $T^{(m)}_{\mathrm{C}_i\mathrm{T}_j} = \boldsymbol{\xi}_{\mathrm{ij}} \boxplus T_{\mathrm{C}_i\mathrm{T}_j}$ où $\boldsymbol{\xi}_{\mathrm{ij}} = \begin{bmatrix}\boldsymbol{\delta\theta} & \boldsymbol{\delta p}\end{bmatrix}^\top \in \mathbb{R}^6 \cong \mathfrak{so}_3 \times \mathbb{R}^3$, $\boldsymbol{\delta\theta} \sim \mathcal{N}(\mathbf{0}, \sigma_\theta^2 \boldsymbol{I}_3)$ et $\boldsymbol{\delta p} \sim \mathcal{N}\left(\mathbf{0}, \sigma_p^2 \boldsymbol{I}_3\right)$. L'inconvénient de cette méthode est qu'elle modélise les erreurs de mesures comme des bruits Gaussiens, alors qu'en pratique, ces mesures sont sujettes à des biais et des ambigüités pouvant causer de grandes erreurs de mesures. Leur distribution a été analysée empiriquement par (Schuster 2019) et analytiquement par (Schweighofer et al. 2006). Il y est d'ailleurs montré que l'incertitude dépend de l'angle d'incidence de l'observation ainsi que de la distance relative entre le marqueur et la caméra. La simulation des mesures AprilTag constitue donc ici une simplification du cas réel. Nous avons pris les valeurs suivantes pour les paramètres cités ci-dessus : $d_{\max} = 5$ mètres, $\theta_{\max} = 60$ degrés, $\sigma_p = 10$ centimères, $\sigma_\theta = 5$ degrés.

Le tableau suivant montre les performances d'estimation obtenues par l'algorithme eVO (Sanfourche et al. 2013) seul sur les différentes séquences. Nous indiquons égale-

Seq.	Longueur [m]	Durée [s]	ATE [m]	ARE [deg]	Échelle [%]	Scenarios 1	2	3	4
Jeu de données EuRoC									
MH1	80.6	182	0.620	14.04	0.490	✗	✗		✗
MH2	73.5	150	0.515	10.66	1.211	✗			✗
MH3	130.9	132	1.018	14.46	1.943		✗		✗
MH4	91.7	99	1.069	10.15	3.245			✗	
MH5	97.6	111	1.049	10.28	3.351			✗	

TABLE 6.1 – Propriétés des séquences EuRoC utilisées et précisions d'estimation de l'algorithme eVO (SANFOURCHE et al. 2013) sur chaque séquence

ment quelles séquences ont été utilisées dans les différents scénarios. Nous avons construit 4 scénarios multi-robots. Les trois premiers n'impliquent que deux robots tandis que le dernier en implique trois. Le scénario 1 utilise les deux séquences MH1 et MH2 dont les trajectoires sont relativement identiques. Dans le second scénario, on utilise la séquence MH3, plus aggressive et couvrant une zone plus vaste que la seule séquence MH1. Les trois séquences MH1, MH2 et MH3 sont d'ailleurs utilisées pour le dernier scénario. Enfin, le troisième scénario utilise les séquences MH4 et MH5 qui couvre l'ensemble de la zone d'acquisition via des trajectoires amples et véloces.

6.5.2 Métriques d'évaluation

Nous évaluons la méthode proposée quant aux trois composantes d'une méthode de SLAM multi-robots. Nous évaluons l'allocation des tâches et des données en quantifiant la charge de calcul requise par chacun des modules. En particulier, nous estimons les fréquences et les temps de calculs associés aux différentes tâches telles que la génération des maillages polygonaux des sous-cartes, la recherche puis la caractérisation éventuelle des correspondances entre les sous-cartes, et l'optimisation du graphe de poses multi-robots. La politique de communication est évaluée au regard de la quantité de données transmise et la charge réseau qui en résulte. Durant les simulations, nous avons supposé que les robots restaient en permanence à portée de communication, si bien que seule la politique régulière est évalée ici, et non la politique de régularisation. Nous mesurons également les temps de traitement nécessaires à la création des paquets envoyés, ainsi qu'à leur intégration dans la carte du récepteur. Enfin, nous étudions l'association et la fusion des données en dénombrant d'une part les correspondances détectées entre les robots,

ainsi qu'en évaluant d'autre part la précision des estimations des trajectoires pour les différents robots. Pour chaque robot, nous évaluons ces métriques individuellement sur chaque trajectoire, puis de façon jointe. Dans chaque scénario, nous alignons d'abord les trajectoires sur la vérité terrain en trouvant la similarité de $\mathbb{S}\mathrm{im}_3$ qui minimise les erreurs de translation des poses en correspondance. Nous évaluons les trajectoires individuelles des robots, ainsi que les erreurs jointes afin de fournir une erreur d'ensemble.

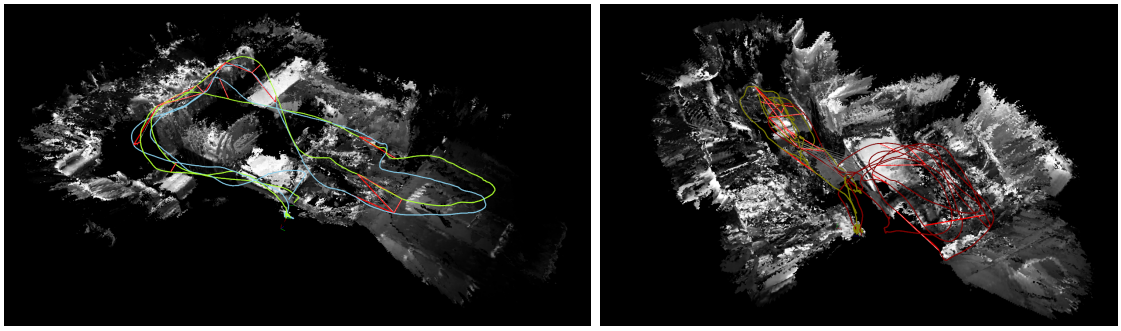
6.5.3 Détails d'implémentation

Nous avons mené les simulations à l'aide du middleware ROS (Quigley et al. 2009). Celui-ci structure la simulation en un graphe de processus parallèles, appelés *nœuds*. Chaque nœud instancie un exécutable spécifique, et peut interagir avec les autres nœuds par l'échange de messages ou via des services selon une logique client-serveur, le tout selon un protocole TCP-IP. Nous adaptons cette architecture à la simulation multi-robots. Chaque robot est représenté par un nœud, instanciant lui-même plusieurs sous-processus en parallèles (*threads*) pour chaque de ses modules tels que son *Front-End*, la recherche des correspondances, l'inférence globale, le calcul et l'intégration des paquets. ROS sied particulièrement aux simulations par rejeu de séquence réelles qu'il facilite par la lecture de fichiers `.bag`. Ceux-ci permettent d'enregistrer et de rejouer des messages en temps réel. Ainsi, les sorties d'eVO et d'ELAS sont enregistrées dans ce format et rejouées en entrée. Les exécutables implémentés traitent ledits messages et implémentent les modules restants. Les simulations sont menées à l'aide d'un processeur Inter® Xeon(R) W-2123 CPU 3.60GHz × 8 cœurs.

Comme mentionné précédemment, nous utilisons la bibliothèque OpenChisel (Klingensmith et al. 2015) pour gérer les représentations TSDF ainsi que les maillages qui en résultent. Nous reprenons en particuliers les objets implémentant les patches TSDF du code source de (Duhautbout et al. 2019), lui-même basé sur OpenChisel et *Point Cloud Library* (Rusu et al. 2011). L'implémentation de la carte et du graphe de poses est basée sur le *framework* Maplab (Schneider et al. 2018). Enfin, nous utilisons la bibliothèque `Ceres` (Agarwal et al. 2012) pour le module d'inférence.

6.5.4 Résultats

Paramètres de simulation Dans les tests, nous avons utilisé un seuil de distance $l_{\max} = 3$ mètres pour les cartes ainsi qu'un seuil d'orientation cumulée de $\theta_{\max} = 90$ de-



Les correspondances entre les sous-cartes sont figurées en rouge.

FIGURE 6.6 – Visuels des cartes reconstruites et des maillages du robot 4 dans le scénario 3 et du robot 1 dans le scénario 2.

grés. Nous avons utilisé une résolution de 8cm pour la grille de voxels pour la TSDF, qui apparaissait comme un bon compromis entre la charge de calculs induite et la qualité du rendu visuel. Pour les appariements, nous avons imposé un nombre minimal de correspondances $\#_{\text{inliers}} = 1000$, un seuil sur le RMSE $\text{rmse}_{\text{ICP}} = 5$ cm, un angle d'erreur maximal sur les normales de $\angle_{\text{max}} = 30$ degrés. Pour le calcul des covariances des correspondances ICP, nous avons empiriquement attribué les valeurs $\sigma_t = 0.2$ cm et $\sigma_\theta = 5.0$ degrés.

Évaluation de l'allocation des tâches et des données Le tableau 6.7 récapitule les statistiques sur les temps d'exécution pour les différentes tâches. L'optimisation du graphe de poses global, ainsi que l'intégration des sous-cartes reçues et le calcul des maillages TSDF sont réalisés en des temps compatibles avec les exigences temps-réel. L'appariement des sous-cartes requiert cependant plus de temps, avec une médiane autour de 5 secondes. Certains appariements ont nécessité des temps conséquents comme en témoigne la valeur maximale de 16 secondes pour l'un des robots. De telles durées incitent à imposer un mécanisme d'abandon de l'alignement courant lorsque celui-ci dépasse une certaine valeur. L'alignement ICP reste l'étape la plus coûteuse de l'alignement.

Évaluation de la politique de communication En moyenne, les paquets des sous-cartes ont un poids en mémoire de 3.9 Méga-Octets avec un écart-type de 1.2 Méga-Octets. La majeure partie de ce poids est expliqué par la représentation TSDF. La période moyenne des échanges est de 7.4 secondes, ce qui est abordable dans un cadre multi-robots. Les correspondances entre sous-cartes, échangées régulièrement entre les robots au gré de leur détection, ont une empreinte mémoire de 288 Kilo-Octets, ce qui est marginal par rapport à la taille des paquets des sous-cartes, chaque message n'incluant que les

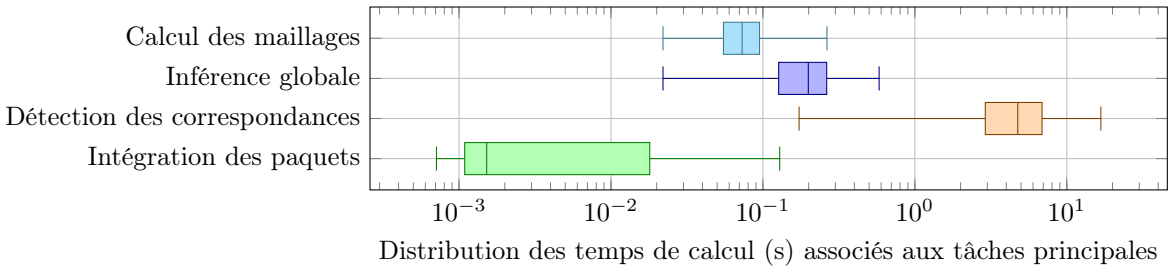


FIGURE 6.7 – Statistiques sur les temps de calculs associés aux tâches principales

identifiants des sous-cartes appariées, la pose relative estimée ainsi que la matrice de covariance associée.

Évaluation de l'association et de la fusion de données multi-robots Les nombres de correspondances détectées pour chaque robot sont reportés dans la Figure 6.8, tandis que les statistiques sur les précisions de ces correspondances sont données par la Figure 6.8. On note que les robots parviennent à détecter une part significative des correspondances inter-robot. La précision d'estimation moyenne des poses relatives entre les sous-cartes reste cependant relativement importante puisqu'elle est de 10 centimètres. Cette faible précision peut résulter du caractère éparse des nuages de points extraits des maillages es sous-cartes. Dans l'implémentation actuelle, la méthode d'alignement souffre de l'absence d'une étape de pré-alignement, ce qui restreint son application aux cas de faibles dérives. En effet, le bassin de convergence de l'ICP vers la bonne solution est en général étroit. Elle requiert donc une bonne initialisation. Pour le moment, les nuages de points sont trop éparses pour permettre un usage de primitives et de descripteurs 3D pour obtenir un pré-alignement.

FIGURE 6.8 – Statistiques de détection des correspondances entre les sous-cartes

Les précisions d'estimations sur les trajectoires jointes dans les cas mono et multi-robots sont rapportées par la Figure 6.10 respectivement pour les cas mono-robot et

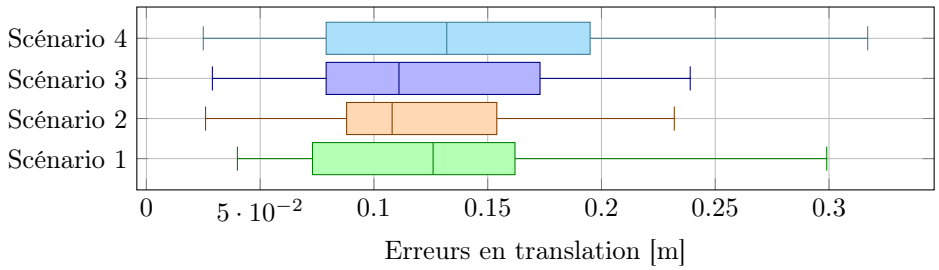


FIGURE 6.9 – Précision des correpondances détectées.

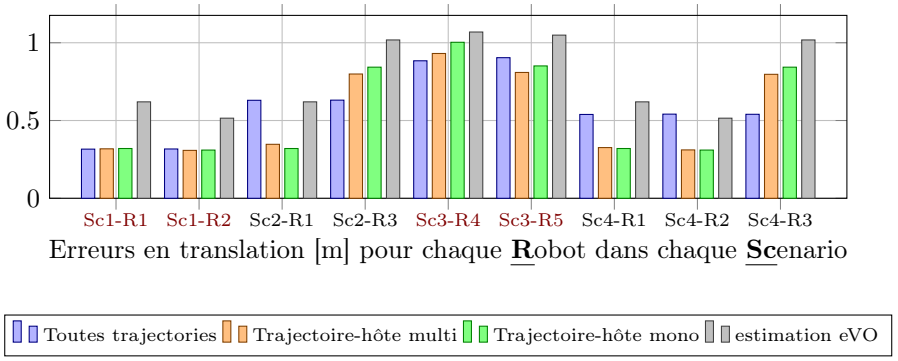


FIGURE 6.10 – Précision d'estimation des trajectoires jointes et des trajectoires hôtes en contexte multi-robots

multi-robots. Bien que l'algorithme permet de légères améliorations sur les précisions d'estimation dans le cas mono-robot, la détection des correspondances et leur intégration dans l'optimisation du graphe de poses n'apporte pas d'améliorations significatives dans le cas multi-robots, car les précisions sur les trajectoires hôtes s'avèrent comparables. Néanmoins, les précisions jointes témoignent que les sous-cartes reçues sont correctement intégrées dans les cartes des receveurs.

6.6 Conclusions et Perspectives

Dans ce chapitre, nous avons proposé une méthode de SLAM pour la cartographie dense 3D en contexte multi-robots. Cette méthode étend l'algorithme de cartographie collaborative originellement développé par (DUHAUTBOUT et al. 2019). Nous l'avons enfin évaluée sur des scénarios multi-robots construits à partir des séquences du jeu de données EuRoC (BURRI et al. 2016) afin de démontrer son aptitude à construire une carte consistante en temps réel tout en y intégrant les informations communiquées par les autres robots. La méthode décrite a fait l'objet d'une publication (DUBOIS et al. 2020b)

à la conférence IROS 2020.

Ces simulations ont également mis en lumières diverses limitations de la méthode développée. Ces limitations, qui concernent toutes la procédure de recherche des correspondances entre les sous-cartes, sont de deux ordres. Tout d'abord, le temps nécessaire pour chaque alignement est significatif, ce qui peut poser des problèmes d'échelle lorsque le nombre de robots augmente. Pour le moment, cette contrainte temporelle conduit à restreindre pour chaque sous-carte la recherche des correspondances aux seules sous-cartes de plus fort recouvrement parmi les sous-carte produites par le robot lui-même et les sous-cartes reçues des autres robots. Quelques solutions pourraient contribuer à y remédier. On pourrait tout d'abord amener les robots à se communiquer non pas uniquement les correspondances qu'ils ont détectées, mais également les couples de sous-cartes qu'ils ont échoué à aligner. Une perspective d'amélioration serait également de prioriser les correspondances potentielles entre les sous-cartes dont les poses relatives sont les plus incertaines afin d'en maximiser l'impact, à la manière de la priorisation effectuée par (Schuster et al. 2015).

La seconde limitation tient à la qualité des correspondances détectées. La méthode retenue complète l'ICP par quatre tests destinés à filtrer les fausses correspondances résultant de minima locaux de l'ICP en amont de l'inférence, tandis que les fonctions de coût robustes doivent les mitiger durant l'inférence. En pratique, le filtrage des correspondances reste délicat. Bien qu'on puisse espérer que les alignements initiaux des sous-cartes soient relativement bon en raison de la faible dérive de l'odométrie stéréo-visuelle, la procédure proposée devrait fortement bénéficier d'une étape de pré-alignement préalable, en particulier pour la recherche de correspondances inter-robot. Cependant, la résolution adoptée pour la TSDF ne permet pas de générer des maillages suffisamment denses en vue d'un pré-alignement par appariement de primitives 3D, et nous avons constaté qu'accroître la résolution de la TSDF pour en extraire des maillages plus fins et donc des nuages plus denses en vue des recalages ICP détériore vainement les performances d'ensemble. Une possibilité serait d'utiliser les descripteurs de nuages de points éparses développés par (Cieslewski et al. 2016) et qui caractérisent la distribution des points dans un voisinage 3D cylindrique de la primitive et aligné avec la verticale pour compenser l'orientation – cette information étant fournie par les mesures inertielles. De tels descripteurs pourraient éventuellement être utilisés en complément des critères de recouvrement pour identifier les couples de sous-cartes prometteurs en termes de correspondances. La technique de reconnaissance de lieux proposée par (Giubilato et al. 2020) et basée sur des vecteurs sac-de-mots construits à partir de descripteurs SHOT binarisés pourrait également être

mise à profit afin d'affiner la recherche des correspondances potentielles. Enfin, une autre perspective d'amélioration – la plus prometteuse – serait *in fine* de changer le paradigme d'alignement. On pourrait remplacer l'ICP par un alignement ne nécessitant pas d'établir des correspondances point-à-point, à l'instar de la méthode proposée par Voxgraph (Reijgwart et al. 2019). Celle-ci adapte le concept de l'ICP aux spécificités des représentations par champs de distances. Le principe consiste à minimiser la distance entre les points extraits du maillage de la surface de la sous-carte à aligner, de minimiser leur distance à l'iso-surface de la sous-carte cible en se basant sur sa représentation ESDF associée. Cette approche élégante implique cependant d'étendre préalablement la représentation TSDF en une représentation ESDF.

Chapitre 7

Conclusion et Perspectives

Problématiques

Dans cette thèse, nous avons investigué le problème de la localisation et de la cartographie simultanées dans un cadre collaboratif. L'objectif consistait à concevoir des méthodes de cartographie et de localisation simultanées collaboratives et décentralisées en contexte visio-inertiel et stéréo-visuel de façon à maximiser l'autonomie des robots et leur permettre d'accroître la qualité de leurs estimations, tout en tenant compte des contraintes d'information, de réseau et de ressources ou d'embarquabilité. Constatant que les méthodes centralisées ainsi que les solutions distribuées adoptées dans le cas décentralisé restreignent l'autonomie des robots, nous nous sommes intéressés à des méthodes complètement décentralisées minimisant les dépendances entre les agents pour la gestion des interactions multi-robot. Nous avons travaillé dans le cas spécifique du SLAM visio-inertiel pour l'estimation de trajectoires durant la navigation, puis nous avons proposé une extension de l'architecture développée dans le cas du SLAM dense stéréo-visuel dans une approche davantage orientée sur la reconstruction et la cartographie multi-robot.

La problématique principale de cette thèse a consisté à permettre aux robots de tirer parti du contexte collaboratif pour enrichir leurs connaissances de l'environnement et in fine améliorer leur propre navigation et/ou cartographie lorsqu'ils visitent des portions communes de l'environnment. Nous nous sommes particulièrement intéressés à la nature des données à transmettre entre les robots afin de permettre de détecter les correspondances inter-robot, puis d'ancrer les référentiels d'odométrie sous l'hypothèse d'absence d'a priori sur les transformations relatives qui les lient, et enfin de ré-estimer l'ensemble des trajectoires au sein d'un problème d'inférence multi-robot, idéalement de façon consistante et dans le but d'accroître les précisions d'estimation sur les trajectoires ou la carte. La conception de ces paquets de données devait en particulier tenir compte des contraintes imposées par les algorithmes de SLAM. Il s'agit notamment des contraintes d'informa-

tion, de réseau et d'embarquabilité : la synthèse et l'intégration des paquets reçus doit être rapide et ne pas lorgner sur les ressources nécessaires aux autres modules du SLAM mais également aux autres processus amonts et avals. Nous devions également prêter attention à la taille des paquets ainsi conçus afin de ne pas saturer la bande-passante. Dans un contexte multi-robot se pose également la question de l'extensibilité des méthodes proposées lorsque le nombre de robots augmente.

Contributions

Tout d'abord, nous avons conçu un nouveau jeu de données dédié au SLAM collaboratif stéréo-visuel et inertiel, et que nous avons nommé AirMuseum (Dubois et al. 2020a). La principale motivation était de pouvoir tester les méthodes développées en les confrontant à des séquences plus difficiles que les scénarios multi-robot constructibles à partir d'EuRoC, et évaluer leurs performances en contexte hétérogène incluant des plateformes terrestres.

Dans le chapitre 5, nous avons proposé trois méthodes de partage de données visio-inertielles reposant sur une architecture commune. Périodiquement quant à la distance parcourue, les robots calculent des paquets résumant leurs informations de localisation et leurs observations sur la sous-trajectoire correspondante, utilisent les paquets reçus pour détecter les correspondances inter-robot impliquant leur propre trajectoire, s'échangent les correspondances détectées, et formulent éventuellement des requêtes ciblées pour compléter les informations visuelles déjà reçues afin d'étoffer leur détection des correspondances. Nous avons proposé une méthode basée sur l'échange de sous-cartes visio-inertielles rigides (SVIP), une méthode empruntant à des techniques de marginalisation et d'éparsification consistante (CVIP), et enfin une méthode exploitant des techniques d'élagage afin de sélectionner des sous-ensembles d'amers qui, adjoint aux mesures inertielles brutes, restituent une portion significative de l'information de localisation. Ces trois méthodes sont exclusivement dédiées à l'estimation des trajectoires en contexte multi-robot : aucune information cartographique n'est échangée, et les cartes construites servent de support à la détection des correspondances inter-robot. Elles se distinguent donc d'autres méthodes de SLAM visio-inertiel telles celles publiées par (Sartipi et al. 2019) et (Lajoie et al. 2020) (DOOR-SLAM) : la première est dédiée à l'identification de correspondances inter-robot simultanées pour des applications de réalité augmentée, tandis que dans la seconde distribue la détection des correspondances et les opérations d'inférence entre les robots pour estimer leur propre trajectoire uniquement. L'évaluation de ces méthodes sur les

scénarios multi-robot construits à partir du jeu de données EuRoC (Burri et al. 2016) a montré qu'elles permettent d'intégrer les informations reçues des autres robots, de détecter les correspondances et de formuler des problèmes d'inférence multi-robot. En dépit de performances moyennes de SVIP quant aux précisions d'estimation qui en résultent, CVIP et PVIP estiment correctement les trajectoires des autres robots, et permettent, lorsque l'information disponible le permet, d'accroître la précision des estimations sur les trajectoires des robots. Néanmoins, leur évaluation sur des séquences plus difficiles du jeu de données AirMuseum (Dubois et al. 2020a) a exposé différentes limites de ces méthodes qui n'étaient pas visibles sur EuRoC, notamment quant à la nécessité d'intégrer les informations absolues sur l'estimation de la direction de la gravité, et sur le besoin de gérer plus explicitement les cas où peu d'informations visuelles sont disponibles.

Dans le chapitre 6, nous nous sommes intéressés au cas du SLAM stéréo-visuel dans une optique davantage orientée sur la cartographie multi-robot que l'estimation des trajectoires. Étendant les travaux de (Duhautbout et al. 2019) qui avaient proposé une méthode de cartographie collaborative basée sur l'échange et la fusion de sous-cartes exploitant des représentations denses de type TSDF, nous avons conçu une méthode de SLAM reposant sur le même principe en apportant des améliorations ponctuelles aux modules *Back-End* et aux trois dimensions de l'architecture multi-robot (i.e. allocation des tâches et des données, politique de communication, association et fusion de données multi-robot). Nous avons en particulier repris et rendu plus robuste le mécanisme de recalage des sous-cartes TSDF et remplacé les mécanismes initiaux de propagation des corrections cumulées par une inférence sur un graphe de facteurs intégrant les correspondances détectées. Comme précédemment, les robots se partagent les correspondances détectées. L'évaluation de cette méthode a montré sa capacité à reconstruire des cartes consistantes. Cependant, si le mécanisme de détection des correspondances se montre plus robuste et éliminent les faux positifs, il manque encore de précision dans sa caractérisation des correspondances. Cela semble être intrinsèquement à l'utilisation de l'ICP sur les nuages de points extraits des maillages qui ne sont pas suffisamment denses pour utiliser les mécanismes de pré-alignement classiques.

Perspectives

Les méthodes visio-inertielles décentralisées que nous avons proposées pourraient être étendues de multiples manières. Tout d'abord, les méthodes SVIP et CVIP devraient béné-

ficier de l'intégration explicite des a priori déduits des mesures inertielles sur la direction de la gravité, cette information étant perdue dans la configuration actuelle qui n'encode que des facteurs de poses relatives. Dans les trois méthodes, une attention particulière doit être portée aux cas où la cinématique des trajectoires ne garantit pas l'observabilité des biais inertiels ou du facteur d'échelle, ou encore que les contraintes visuelles sont trop faibles. La méthode PVIP s'y avère particulièrement sensible. Une autre piste d'amélioration serait de concevoir des paquets de natures distinctes. Un exemple serait de répliquer la méthode PVIP en remplaçant la sélection des amers par des facteurs visuels épipolaires sur $\mathbb{S}\mathfrak{im}_3$ dérivés de l'estimation de matrices essentielles entre les poses successives. Concernant la méthode CVIP, nous pourrions envisager des topologies d'éparsification plus denses et retiendraient davantage d'informations que la topologie séquentielle actuelle. Cependant, la plupart des méthodes d'éparsification adaptées aux topologies non-inversibles (e.g. la descente de facteurs (Vallvé et al. 2018)) sont prévues pour s'appliquer à l'échelle des couvertures de Markov des nœuds marginalisés, et deviennent complexes lorsqu'on les applique à l'échelle de sous-graphes. Par ailleurs, nous pourrions adopter une autre politique de communication des informations visuelles, en se restreignant à des descripteurs globaux, et dédier le mécanisme de requêtes d'images-clés à la caractérisation des correspondances potentielles. Cela permettrait notamment de transmettre un descripteur global pour chaque image et ainsi, potentiellement, de cibler davantage l'échange d'informations visuelles. Cependant, cela exigerait des robots qu'ils maintiennent le contact les uns avec les autres afin de caractériser ces correspondances, et amoindrirait ainsi l'autonomie recherchée. Une autre piste d'améloration serait de mieux gérer l'intégration de trajectoires affectées par de larges erreurs d'estimation, ce cas n'étant pas spécifiquement traité par les méthodes actuelles ; cela risque notamment de compromettre fortement les estimées de trajectoire du robot récepteur. Enfin, le développement des méthodes de SLAM sémantiques telles que (Salas-Moreno et al. 2013) et (Rosinol et al. 2019) ouvre de nouvelles perspectives quant aux informations cartographiques à communiquer pour permettre la détection des correspondances inter-robot.

Enfin, concernant la méthode de SLAM stéréo-visuelle proposée pour la cartographie dense multi-robot, la principale piste d'amélioration consiste à substituer à l'ICP une méthode de recalage de sous-cartes exploitant directement les représentations implicites sous-jacentes i.e. le champ de distances signées, tel que proposé par Voxgraph (Reijgwart et al. 2019). Ce mécanisme nécessiterait préalablement l'extraction d'une représentation ESDF à partir de la représentation TSDF, qui pourrait alors être générée à la suite du

maillage de la sous-carte. Ce mécanisme devrait permettre d'accroître la précision des correspondances inter-robot détectées.

Troisième partie

Annexes

Annexe A

Géométrie dans l'espace

Afin de représenter les trajectoires des robots et de modéliser la structure de l'environnement qu'ils observent, nous recourrons aux principaux outils de la géométrie dans l'espace. Dans cette section, nous nous intéresserons d'abord à la représentation des rotations 3D ainsi que des transformations affines Euclidienne. Enfin, nous verrons en quoi les groupes de Lie fournissent un formalisme unifié pour appréhender ces différents objets.

A.1 Interprétations actives et passives

Considérons un vecteur $\boldsymbol{x} \in \mathbb{R}^n$ et un automorphisme T de $\mathbb{R}^n$. On note ${}^A\boldsymbol{x}$ les coordonnées de $\boldsymbol{x}$ dans un système de coordonnées A. La quantité $\boldsymbol{y} = T({}^A\boldsymbol{x})$ peut s'interpréter de deux manières différentes et ainsi représenter deux objets distincts. L'interprétation active considère que la transformation T s'est appliquée au point $\boldsymbol{x}$ dans le système de coordonnées A, et qu'ainsi les coordonnées $\boldsymbol{y}$ sont celles de son image par T dans ce même système de coordonnées. On peut alors noter $\boldsymbol{y} = {}^AT({}^A\boldsymbol{x})$. Au contraire, l'interprétation passive considère que la transformation T s'est appliquée à un système de coordonnées B et dont le système de coordonnées A est l'image. Dans ce cas, les coordonnées de $\boldsymbol{y}$ sont celle du même vecteur $\boldsymbol{x}$ mais exprimées dans le système de coordonnées B. On note alors $T = T_{BA}$ et $\boldsymbol{y} = {}^B\boldsymbol{x} = T_{BA}({}^A\boldsymbol{x})$, T_{BA} désignant la transformation appliquée au système de coordonnées B (et non pas aux vecteurs exprimé dans B) pour obtenir le nouveau système de coordonnées A. Ces deux interprétations duales s'appliqueront à l'ensemble des transformations que nous introduirons dans les sections suivantes pour modéliser les rotations, les transformations et les similitudes affines Euclidiennes.

A.2 Représentation des rotations et des orientations

Mathématiquement, on définit les rotations dans l'espace comme les automorphismes orthogonaux de $\mathbb{R}^3$ de déterminant 1. Ceux-ci forment le *Groupe Spécial Orthogonal*

d'ordre 3 $\mathbb{SO}_3$. La description numérique de ces rotations et des opérations qui leur sont associées constitue l'une des difficultés majeures de la géométrie 3D. Pour ce faire, on doit les identifier à des objets mathématiques que l'on sait manipuler numériquement, et sur lesquels on peut éventuellement transporter les propriétés topologiques de $\mathbb{SO}_3$ ainsi que des opérations structurantes telles que la composition, l'inversion et des actions de groupe. Dans cette section, nous étudierons quatre représentations des rotations. D'un coté, il existe des représentations minimales et intuitives telles que les angles d'Euler et les vecteurs d'Euler, mais celles-ci présentent des singularités et/ou ne permettent pas modéliser les opérations élémentaires mentionnées précédemment. C'est pourquoi on utilise d'autres représentations plus ésotériques et sur-paramétrées telles que les matrices de rotation et les quaternions qui gomment ces singularités et permettent de modéliser numériquement les opérations associées aux rotations.

A.2.1 Les angles d'Euler

De la même façon qu'une translation se décompose en la somme de ses projections selon les trois axes d'une base orthogonale, on peut de façon analogue décomposer les rotations en une succession de trois rotations élémentaires autour de ces mêmes axes. Trois angles, dit *angles d'Euler*, suffisent alors pour les décrire. Lorsque les axes demeurent fixes, on parle d'*angles d'Euler extrinsèques* ou d'*angles de Tait-Bryant*. En contexte aéronautique, les rotations longitudinales, transversales et verticales sont respectivement nommées roulis, tangage et lacet (*pitch*, *roll* et *yaw* en anglais). Lorsqu'au contraire, les axes de la base sont mobiles et accompagnent les mouvements de rotation, on parle d'*angles d'Euler intrinsèques* : ce sont dans l'ordre la *précession*, la *nutation* et la *rotation propre*. Qu'elles soient extrinsèques ou intrinsèques, l'ordre de ces rotations importe car elles ne commutent pas [1].

Si les angles d'Euler offrent une représentation intuitive et minimale des rotations, ils présentent trois inconvénients majeurs. Tout d'abord, pour une même séquence d'axes, plusieurs enchaînements peuvent représenter la même rotation : il n'y donc pas unicité de la représentation. Deuxièmement, elle ne fournit pas un cadre commode pour représenter numériquement les opérations de composition ou d'inversion des rotations, ni leur action de groupe sur les vecteurs 3D. Enfin, cette représentation souffre de singularités appe-

1. Dans les cas des angles d'Euler extrinsèques, les trois rotations élémentaires successives r_1 r_2 et r_3 décomposant la rotation r sont composées de droite à gauche i.e. $r = r_3 \circ r_2 \circ r_1$ (approche globale), et inversement dans le cas des angles d'Euler intrinsèques $r = r_1 \circ r_2 \circ r_3$ (approche locale).

lées *blocage de Cardan* dans une analogie au système mécanique de trois cardans qu'elle modélise par ailleurs. Ces singularités correspondent localement à la perte d'un degré de liberté [2] lorsque deux axes sont alignés. Si l'on ne déplore aucun blocage au sens mécanique du terme, ces singularités compliquent l'interpolation et l'encodage continu d'une trajectoire de rotations dans les problèmes de navigation.

A.2.2 Le vecteur rotation

Le vecteur rotation, également appelé *vecteur d'Euler* ou *représentation angle-axe*, identifie une rotation par un vecteur $\boldsymbol{\omega} \in \mathbb{R}^3$ qui dirige l'axe de rotation, et dont la norme $\theta = \|\boldsymbol{\omega}\|$ est l'angle de la rotation directe dans le plan orthogonal à cet axe. Si cette représentation intuitive ne permet pas de transporter commodément les lois de composition et les actions de groupe des rotations, nous verrons cependant qu'elle tire son intérêt des correspondances qui la lient aux quaternions et aux matrices de rotations.

A.2.3 Les matrices de rotation

En tant qu'application linéaire en dimension finie, toute rotation r peut être représentée par une matrice $\boldsymbol{R} \in \mathcal{M}_3(\mathbb{R})$. En qualité d'automorphisme orthogonal de $\mathbb{R}^3$, $\boldsymbol{R}$ admet sa transposée pour inverse i.e. $\boldsymbol{R}^{-1} = \boldsymbol{R}^\top$, tandis que par définition des rotations, son déterminant vaut 1 i.e. $\det(\boldsymbol{R}) = 1$. Cette représentation s'appréhende plus délicatement que les angles ou le vecteur d'Euler, notamment parce qu'elle est sur-paramétrée et que la contrainte du déterminant unitaire induit des relations complexes entre ces paramètres. Cependant, elle restitue fidèlement les propriétés topologiques des rotations, tandis que l'inversion et la multiplication matricielles fournissent une représentation numériquement de l'inversion et la composition des rotations, ainsi que leur action de groupe sur $\mathbb{R}^3$. Étant donné deux rotations $\boldsymbol{R}_{\mathrm{AB}}$ et $\boldsymbol{R}_{\mathrm{BC}}$ entre des référentiels A, B et C, on vérifie les relations suivantes : $\boldsymbol{R}_{\mathrm{AC}} = \boldsymbol{R}_{\mathrm{AB}} \cdot \boldsymbol{R}_{\mathrm{BC}}$, tandis que $\boldsymbol{R}_{\mathrm{BA}} = \boldsymbol{R}_{\mathrm{AB}}^{-1}$.

Soulignons enfin qu'il existe des correspondances entre les matrices de rotations et les deux représentations que nous avons évoquées précédemment. En particulier, la formule de Rodrigues lie les vecteurs rotations aux matrices de rotation. Notons $\boldsymbol{\omega} = \theta \boldsymbol{u}$ un vecteur

2. Mathématiquement, le blocage de Cardan se traduit par l'existence de points en lesquels la représentation des rotations par les angles d'Euler ne définit plus un homéomorphisme local en raison d'une perte de rang.

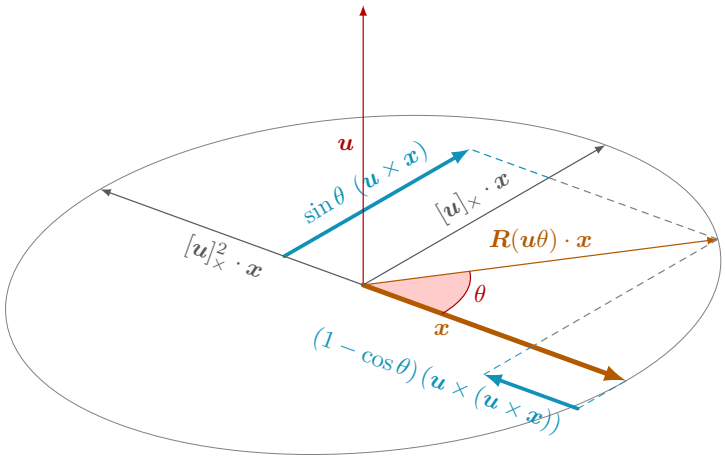


FIGURE A.1 – Visualisation de la formule de Rodrigues

rotation avec $\boldsymbol{u}\in\mathbb{R}^3$ un vecteur unitaire :

$$\boldsymbol{R}(\boldsymbol{\omega}) = \boldsymbol{I}_3 + \sin\theta\,[\boldsymbol{u}]_\times + (1-\cos\theta)\,[\boldsymbol{u}]_\times^2 \text{ avec } [\boldsymbol{u}]_\times = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \tag{A.1}$$

où la matrice $[\boldsymbol{u}]_\times$ représente l'application $\boldsymbol{x}\in\mathbb{R}^3 \mapsto \boldsymbol{u}\times\boldsymbol{x}$. La Figure A.1 fournit une représentation visuelle de cette formule. Nous verrons notamment que cette relation correspond à l'exponentielle de $\mathbb{SO}_3$ considéré en tant que groupe de Lie. Des correspondances existent également entre les matrices de rotation et les angles d'Euler, bien qu'elles soient tributaires des séquences choisies pour les rotations. Ces relations sont complexes mais on les retrouve en composant les matrices des rotations élémentaires.

A.2.4 Les quaternions

Introduits au XIX$^{\text{ème}}$ siècle par le mathématicien William Rowan Hamilton dans le but de généraliser les nombres complexes en dimension 3, les quaternions offrent une représentation originale et numériquement très commode des rotations. Un quaternion est un nombre de la forme :

$$\boldsymbol{q} = q_w + q_x\cdot\boldsymbol{i} + q_y\cdot\boldsymbol{j} + q_z\cdot\boldsymbol{k}\in\mathbb{H} \quad \text{tel que} \quad \boldsymbol{i}^2=\boldsymbol{j}^2=\boldsymbol{k}^2=-1 \quad \text{et} \quad \boldsymbol{ij}=\boldsymbol{k} \tag{A.2}$$

où $\boldsymbol{i}$, $\boldsymbol{j}$ et $\boldsymbol{k}$ sont trois nombres imaginaires distincts [3]. Isomorphe à $\mathbb{R}^4$, l'ensemble des quaternions emprunte néanmoins beaucoup au formalisme des nombres complexes. Ainsi, on peut scinder tout quaternion $\boldsymbol{q}$ en une partie réelle $\mathrm{Re}(\boldsymbol{q}) = q_w$ et une partie imaginaire $\mathrm{Im}(\boldsymbol{q}) = q_x \cdot \boldsymbol{i} + q_y \cdot \boldsymbol{j} + q_z \cdot \boldsymbol{k}$. Dans le cas des quaternions cependant, cette partie imaginaire peut s'identifier à un vecteur de $\mathbb{R}^3$. Ceci motive l'abus de notation suivant – mais fort pratique – qui consiste à considérer un quaternion comme la somme d'un réel et d'un vecteur : $\boldsymbol{q} = q_w + \boldsymbol{q_v}$. En corollaire, tout vecteur de $\mathbb{R}^3$ peut s'identifier à un quaternion imaginaire pur. De façon analogue aux nombres complexes, on définit le conjugué d'un quaternion $\bar{\boldsymbol{q}} = \mathrm{Re}(\boldsymbol{q}) - \mathrm{Im}(\boldsymbol{q})$, sa norme $\|\boldsymbol{q}\| = \sqrt{\boldsymbol{q} \cdot \bar{\boldsymbol{q}}}$ ou encore son inverse $\boldsymbol{q}^{-1} = \bar{\boldsymbol{q}}/\|\boldsymbol{q}\|^2$. Enfin, à l'instar des nombres complexes, les quaternions admettent également une forme polaire :

$$\boldsymbol{q} = \|\boldsymbol{q}\| \cdot \mathrm{Exp}_{\mathbb{H}}(\boldsymbol{u}\theta) = \|\boldsymbol{q}\| \cdot (\cos\theta + \boldsymbol{u}\sin\theta) \tag{A.3}$$

avec $\boldsymbol{u}$ un vecteur unitaire de $\mathbb{R}^3$ et $\theta \in \mathbb{R}$. L'analogie avec la représentation angle-axe n'est pas fortuite, et nous verrons que l'exponentielle introduite est l'analogue de la formule de Rodrigue des matrices de rotations pour les quaternions.

Enfin, la relation A.2 permet de définir un produit entre deux quaternions. Considérons deux quaternions $\boldsymbol{q}_1 = q_w^1 + \boldsymbol{q}_{\boldsymbol{v}}^1$ et $\boldsymbol{q}_2 = q_w^2 + \boldsymbol{q}_{\boldsymbol{v}}^2$. Leur produit s'obtient ainsi :

$$\boldsymbol{q_1} * \boldsymbol{q_2} = \left(q_w^1 \cdot q_w^2 - \boldsymbol{q}_{\boldsymbol{v}}^{\boldsymbol{1}} \cdot \boldsymbol{q}_{\boldsymbol{v}}^{\boldsymbol{2}}\right) + \left(q_w^1 \cdot \boldsymbol{q}_{\boldsymbol{v}}^{\boldsymbol{2}} + q_w^2 \cdot \boldsymbol{q}_{\boldsymbol{v}}^{\boldsymbol{1}} + \boldsymbol{q}_{\boldsymbol{w}}^{\boldsymbol{1}} \times \boldsymbol{q}_{\boldsymbol{w}}^{\boldsymbol{2}}\right) \tag{A.4}$$

où apparait notamment le produit vectoriel $\times$ entre les parties imaginaires. Ce produit est associatif mais non commutatif.

Un intérêt primordial des quaternions est leur habilité à représenter les rotations dans l'espace. Munis du produit des quaternions, ils forment un groupe dont la structure reproduit la topologie de $\mathbb{SO}_3$. La contrainte de norme unitaire est l'analogue de la contrainte sur le déterminant des matrices de rotation. Considérons une rotation d'axe dirigé par un vecteur unitaire $\boldsymbol{u}$ et d'angle θ. On note alors $\boldsymbol{\omega} = \theta\boldsymbol{u}$ son vecteur d'Euler et on l'associe

3. La relation $\boldsymbol{ij} = \boldsymbol{k}$ est une convention, dite de Hamilton, et que nous utiliserons exclusivement. Signalons cependant la convention JPL (en référence au *Jet Propulsory Lab* de la NASA), qui utilise la relation opposée $\boldsymbol{ij} = -\boldsymbol{k}$.

au quaternion ainsi défini :

$$\boldsymbol{q}(\boldsymbol{\omega}) = \mathrm{Exp}_{\mathbb{H}}(\boldsymbol{\omega}) = \cos\left(\frac{\theta}{2}\right) + \boldsymbol{u}\sin\left(\frac{\theta}{2}\right) \tag{A.5}$$

Étant donné un vecteur $\boldsymbol{x} \in \mathbb{R}^3$, on vérifie la relation :

$$\boldsymbol{R}(\boldsymbol{\omega}) \cdot \boldsymbol{x} = \boldsymbol{q}(\boldsymbol{\omega}) * \boldsymbol{x} * \bar{\boldsymbol{q}}(\boldsymbol{\omega}) \triangleq \boldsymbol{q}(\boldsymbol{\omega}) \cdot \boldsymbol{x} \tag{A.6}$$

Cette opération correspond à l'enchaînement de deux rotations conjuguées en dimension 4, et dont le résultat, projeté en dimension 3, correspond à une rotation vectorielle. Notons d'ailleurs qu'un quaternion et son opposé représentent la même rotation ; par commodité, on choisira le quaternion de partie réelle positive. Via cette correspondance, les quaternions offrent ainsi une représentation plus compacte et plus stable numériquement que les matrices de rotation. Pour un exposé plus exhaustif des propriétés des quaternions, le lecteur intéressé pourra se référer à (Sola 2012).

A.3 Représentation des transformations affines Euclidiennes

À l'instar des orientations et des rotations, la pose d'un objet – i.e. la donnée de sa position et de son orientation – et la transformations affine entre deux référentiels se représentent indifféremment. Mathématiquement, leur ensemble est le *Groupe Spécial Euclidien* $\mathbb{SE}_3$, de dimension 6. Leur caractère affine rend cependant délicate la gestion de leur composition en chaîne. C'est pourquoi on utilisera préférentiellement le formalisme des coordonnées *homogènes* pour les représenter. Celui-ci consiste à augmenter la représentation matricielle des vecteurs et des transformations d'une quatrième composante afin de traiter les translations dans un formalisme linéaire. La transformation T_{AB} est ainsi représentée par une matrice de transformation homogène $\boldsymbol{T}_{\mathrm{AB}}$ ainsi construite par blocs :

$$\boldsymbol{T}_{\mathrm{AB}} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{AB}} & {}^{A}\boldsymbol{t}_{\mathrm{AB}} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{A.7}$$

où le vecteur ${}^{A}\boldsymbol{t}_{\mathrm{AB}}$ et la matrice $\boldsymbol{R}_{\mathrm{AB}}$ désignent respectivement la translation et la rotation appliquées au référentiel A pour obtenir le référentiel B. Cette astuce permet ainsi de

représenter la composition et l'inversion des transformations affines, ainsi que leur action de groupe sur les vecteurs 3D, dans un cadre linéaire à l'aide de de la multiplication et de l'inversion matricielle. Étant donné trois référentiels A, B et C, on vérifie les relations suivantes : $T_{\mathrm{AC}} = T_{\mathrm{AB}} \cdot T_{\mathrm{BC}}$ et $T_{\mathrm{BA}} = T_{\mathrm{AB}}^{-1}$. On dispose d'ailleurs d'une formule explicite de l'inverse d'une matrice de transformation homogène :

$$\boldsymbol{T}_{\mathrm{BA}} = \boldsymbol{T}_{\mathrm{AB}}^{-1} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{AB}}^{\top} & -\boldsymbol{R}_{\mathrm{AB}}^{\top} \cdot {}^{A}\boldsymbol{t}_{\mathrm{AB}} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{A.8}$$

L'action d'une transformation affine Euclidienne T sur un vecteur $\boldsymbol{x}$ correspond au produit matriciel de leurs représentations homogènes :

$$\boldsymbol{T} \cdot \boldsymbol{x} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}.\boldsymbol{x} + \boldsymbol{t} \\ 1 \end{bmatrix} \tag{A.9}$$

et son résultat s'interpréte activement ou passivement comme décrit dans la section A.1.

A.4 Représentation des similitudes Euclidiennes

Les similitudes 3D, dont le groupe est noté $\mathbb{Sim}_3$ et comporte 7 dimensions, sont des transformations affines qui dilatent ou contractent les distances d'un facteur d'échelle donné. La gestion des similitudes est en tout point similaire à celle des transformations affines. Leur représentation matricielle homogène intègre néanmoins explicitement le facteur d'échelle. Considérons une similitude T_{AB} entre deux référentiels A et B d'échelle différente. Sa matrice homogène et son inverse sont :

$$T_{\mathrm{AB}} = \begin{bmatrix} s_{\mathrm{AB}}.\boldsymbol{R}_{\mathrm{AB}} & {}^{\mathrm{A}}\boldsymbol{t}_{\mathrm{AB}} \\ \boldsymbol{0}_{3\times 1} & 1 \end{bmatrix} \qquad T_{\mathrm{AB}}^{-1} = \begin{bmatrix} s_{\mathrm{AB}}^{-1} \cdot \boldsymbol{R}_{\mathrm{AB}}^{\top} & -s_{\mathrm{AB}}^{-1} \cdot \boldsymbol{R}_{\mathrm{AB}}^{\top} \cdot {}^{\mathrm{A}}\boldsymbol{t}_{\mathrm{AB}} \\ \boldsymbol{0}_{3\times 1} & 1 \end{bmatrix} \tag{A.10}$$

où s_{AB} est le facteur d'échelle (ou la dilatation) appliquée au référentiel A pour obtenir le référentiel B.

A.5 Les groupes de Lie comme cadre unificateur

Les ensembles des rotations, des transformations affines et des similitudes partagent un grand nombre de propriétés structurelles en tant que groupes de Lie. Ce cadre mathé-

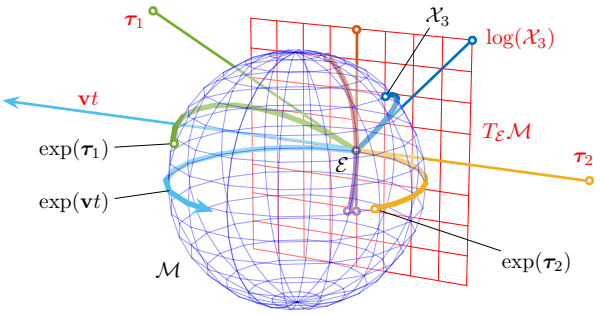
matique unifie ces propriétés et en dérive des paramétrisations adéquates. Cette section présente les rudiments mathématiques nécessaires à l'estimation d'états dans les groupes de Lie. Pour une présentation plus exhaustive des propriétés des groupes de Lie, on pourra se reporter à (Sola et al. 2018 ; Eade 2014 ; Barfoot 2017).

A.5.1 Définition des groupes de Lie

Un groupe de Lie est une variété différentielle dotée d'une structure de groupe, une variété différentielle étant un espace topologique localement homéomorphe à $\mathbb{R}^n$. $\mathbb{SO}_3$, $\mathbb{SE}_3$ et $\mathbb{Sim}_3$ sont des exemples de groupes de Lie. Cette structure, introduite au XIX$^{\text{ème}}$ siècle par le mathématicien norvégien Sophus Lie, joue aujourd'hui un rôle fondamental pour l'estimation d'états en robotique car elle permet de représenter des états qui n'évoluent pas dans des espaces Euclidiens, à l'instar des poses et des rotations.

Groupe de Lie et Algèbre de Lie

On considère un groupe de Lie $\mathcal{M}$ de dimension n. La propriété fondamentale d'un groupe de Lie est qu'on peut lui associer une algèbre de Lie $\mathfrak{m}$. Il s'agit d'un espace vectoriel Euclidien, isomorphe à $\mathbb{R}^n$, et dont tout élément correspond à un élément de $\mathcal{M}$ via une application exponentielle $\exp_{\mathcal{M}} : \mathfrak{m} \to \mathcal{M}$, dont l'inverse est une application logarithme $\log_{\mathcal{M}} : \mathcal{M} \to \mathfrak{m}$. L'algèbre de Lie de $\mathcal{M}$ se définit comme son espace tangent à l'identité ε i.e. $\mathfrak{m} \triangleq T_\varepsilon\mathcal{M}$. Sa structure dérive de la différenciation temporelle de la contrainte de groupe $x \circ x^{-1} = \varepsilon$, où $\circ$ est la loi de composition interne, en ce sens qu'elle comprend tous les éléments de la forme $\boldsymbol{\tau}^\wedge = \dot{x} \circ x^{-1} = -x \circ \dot{x}^{-1}$. L'algèbre de Lie étant isomorphe à $\mathbb{R}^n$, il existe également deux opérateurs $(\cdot)^\wedge : \mathbb{R}^n \mapsto \mathfrak{m}$ et $(\cdot)^\vee : \mathfrak{m} \mapsto \mathbb{R}^n$ qui permettent de passer de l'un à l'autre. Par commodité, on considère directement l'application composée $\mathrm{Exp}_{\mathcal{M}}(\boldsymbol{\tau}) \triangleq \exp_{\mathcal{M}}(\boldsymbol{\tau}^\wedge)$ et son inverse $\mathrm{Log}_{\mathcal{M}}(x) \triangleq \log_{\mathcal{M}}(x)^\vee$. Un grand nombre d'opérations sur le groupe de Lie se



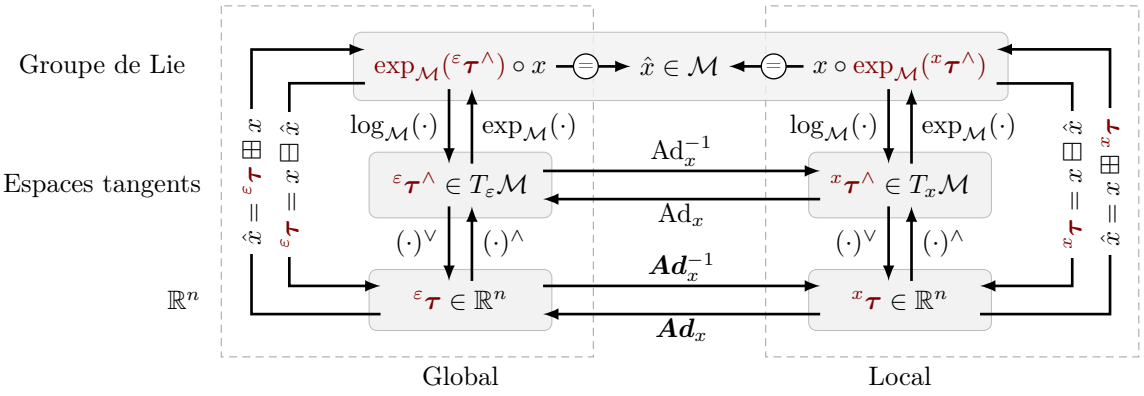
Cette figure extraite de (Sola et al. 2018) illustre les relations entre un groupe de Lie et son algèbre de Lie. La sphère unitaire de $\mathbb{R}^3$, n'étant pas un groupe de Lie, n'est utilisée ici qu'à des fins de représentation.

Figure A.2 – Relations entre un groupe de Lie et son algèbre de Lie

transportent sur l'algèbre de Lie.

Espaces tangents et opérateurs adjoints

Tout élément x du groupe de Lie admet un espace tangent $T_x\mathcal{M}$ dont la structure est identique à celle de l'algèbre de Lie. Cette propriété permet de définir un système de coordonnées local à x et d'exprimer dans un cadre Euclidien l'incrément d'un élément y par rapport à x. Étant donné deux éléments $x, y \in \mathcal{M}$, on peut ainsi définir deux incréments ${}^{\varepsilon}\boldsymbol{\tau} \in T_\varepsilon\mathcal{M}$ et ${}^{x}\boldsymbol{\tau} \in T_x\mathcal{M}$ tels que : $y = \mathrm{Exp}\left({}^{\varepsilon}\boldsymbol{\tau}\right) \circ x \ = x \circ \mathrm{Exp}_{\mathcal{M}}\left({}^{x}\boldsymbol{\tau}\right)$. Vis à vis de x, l'incrément ${}^{x}\boldsymbol{\tau}$ est dit *local* car il est exprimé dans son espace tangent, tandis que ${}^{\varepsilon}\boldsymbol{\tau}$ est dit *global* car il est exprimé dans celui de l'identité, l'algèbre de Lie (c.f. Figure A.2). On



On considère un élément $\hat{x}$ d'un groupe de Lie $\mathcal{M}$ de dimension n. Son incrément par rapport à $x \in \mathcal{M}$ peut s'exprimer dans l'algèbre de Lie $T_\varepsilon\mathcal{M}$ (approche globale) ou bien dans l'espace tangent $T_x\mathcal{M}$ de x (approche locale) qui a la même structure que l'algèbre de Lie. Les incréments dans ces espaces tangents sont liés entre eux par des opérateurs linéaires dits adjoints. Les opérateurs $\exp_{\mathcal{M}}$ et $\log_{\mathcal{M}}$ lient les espaces tangents au groupe de Lie, et les opérateurs $(\cdot)^{\wedge}$ et $(\cdot)^{\vee}$ lient les espaces tangents à $\mathbb{R}^n$.

FIGURE A.3 – Relations entre un groupe de Lie et ses espaces tangents

introduit alors deux opérateurs $\boxplus$ et $\boxminus$ pour exprimer ces incréments. La relation précédente se réécrit $y = {}^{\varepsilon}\boldsymbol{\tau} \boxplus x = x \boxplus {}^{x}\boldsymbol{\tau}$. L'opérateur inverse $\boxminus$ est défini tel que $y \boxminus x$ renvoie l'incrément de y par rapport à x. Dans le cas local ${}^{x}\boldsymbol{\tau} = y \boxminus x \triangleq \log_{\mathcal{M}}\left(x^{-1} \circ y\right)^{\vee}$, tandis que dans le cas global ${}^{\varepsilon}\boldsymbol{\tau} = y \boxminus x \triangleq \log_{\mathcal{M}}\left(y \circ x^{-1}\right)^{\vee}$. Les espaces tangents définissent ainsi des paramétrisations locales canoniques qui permettent de manipuler les objets du groupe de Lie dans un cadre Euclidien sans déroger aux propriétés topologiques structurelles qui les unissent. Néanmoins, l'addition dans l'algèbre de Lie ne transporte pas la composition dans le groupe de Lie : la relation $\log(x_i \circ x_j) = \log(x_i) + \log(x_j)$ n'est ainsi pas vérifiée dans le cas général.

Enfin, il est possible de passer d'un espace tangent à l'autre à l'aide d'opérateurs linéaires appelés *opérateurs adjoints*. Ceux-ci permettent en particulier de trouver l'incrément local correspondant à un incrément global et vice-versa. En reprenant l'égalité ${}^{\varepsilon}\boldsymbol{\tau} \boxplus x = x \boxplus {}^{x}\boldsymbol{\tau}$, l'opérateur adjoint associé à un élément x est tel que :

$$ {}^{\varepsilon}\boldsymbol{\tau}^{\wedge} = \mathrm{Ad}_x \left({}^{x}\boldsymbol{\tau}^{\wedge} \right) = x \circ {}^{x}\boldsymbol{\tau}^{\wedge} \circ x^{-1} \tag{A.11} $$

L'opérateur adjoint transporte ainsi un incrément local dans $T_x\mathcal{M}$ en un incrément global de $T_\varepsilon\mathcal{M}$, et définit une action de groupe des éléments de $\mathcal{M}$ sur ceux de $\mathfrak{m}$. Ces relations sont récapitulées par la Figure A.3.

A.5.2 Calcul différentiel sur les groupes de Lie

Les correspondances entre un groupe de Lie et ses espaces tangents permettent d'exprimer les opérations différentielles directement dans ces derniers. Ainsi, étant donné deux groupes de Lie $\mathcal{M}_1$ et $\mathcal{M}_2$, ainsi qu'une fonction $f : \mathcal{M}_1 \mapsto \mathcal{M}_2$ de classe au moins $\mathcal{C}^1$, la dérivée *globale* de f par rapport à x évaluée en un élément $\hat{x}$ se définit comme suit :

$$ \left. \frac{{}^{\varepsilon}\partial f}{{}^{\varepsilon}\partial x} \right|_{\hat{x}} = \left. \frac{\partial}{\partial\, {}^{\varepsilon}\boldsymbol{\tau}} f({}^{\varepsilon}\boldsymbol{\tau} \boxplus \hat{x}) \boxminus f(\hat{x}) \right|_{{}^{\varepsilon}\boldsymbol{\tau}=\mathbf{0}} \tag{A.12} $$

Ainsi, ce n'est pas directement l'application $f : \mathcal{M}_1 \mapsto \mathcal{M}_2$ que l'on dérive, mais une application qui à un incrément global ${}^{\varepsilon}\boldsymbol{\tau}$ sur $\hat{x}$ associe l'incrément global image ${}^{\varepsilon}\boldsymbol{\tau}'$ sur $f(\hat{x})$ tel que $f({}^{\varepsilon}\boldsymbol{\tau} \boxplus \hat{x}) = {}^{\varepsilon}\boldsymbol{\tau}' \boxplus f(\hat{x})$. Notons que si l'équation A.12 présente une dérivée *globale*, c'est-à-dire différenciant un incrément image global par rapport à un incrément global, il est possible de définir de façon analogue des dérivées *locales* différenciant un incrément local par rapport à un incrément local, voire des dérivées *mixtes* qui différencient un incrément local par rapport à un incrément global et inversement. Toutes ces dérivées sont notamment liées entre elles grâce aux opérateurs adjoints.

Les règles classiques de dérivations s'appliquent. C'est en particulier le cas de la règle de dérivation des fonctions composées. Soit $\mathcal{M}_1$, $\mathcal{M}_2$ et $\mathcal{M}_3$ trois groupes de Lie, et $f : \mathcal{M}_1 \mapsto \mathcal{M}_2$ et $g : \mathcal{M}_2 \mapsto \mathcal{M}_3$ deux fonctions de classe au moins $\mathcal{C}^1$, alors :

$$ \left. \frac{{}^{\varepsilon}\partial (g \circ f)}{{}^{\varepsilon}\partial x} \right|_{\hat{x}} = \left. \frac{{}^{\varepsilon}\partial g}{{}^{\varepsilon}\partial y} \right|_{f(\hat{x})} \circ \left. \frac{{}^{\varepsilon}\partial f}{{}^{\varepsilon}\partial x} \right|_{\hat{x}} \tag{A.13} $$

Deux dérivées particulières d'un groupe de Lie sont le Jacobien à gauche et le Jacobien à droite, qui sont respectivement des dérivées des opérateurs $\boxplus$ dans le cadre global et local. Enfin, notons que les matrices des opérateurs adjoints interviennent fréquemment dans les formes analytiques des dérivées sur les groupes de Lie en raison de la relation qui lie incréments locaux et globaux $x \boxplus \boldsymbol{\tau} = \boldsymbol{Ad}_x \cdot \boldsymbol{\tau} \boxplus x$ où $\boldsymbol{Ad}_x$ est la matrice associée à l'opérateur adjoint de x.

A.5.3 Applications aux groupes $\mathbb{SO}_3$ et $\mathbb{SE}_3$

Les groupes des rotations $\mathbb{SO}_3$ et des transformations Euclidiennes $\mathbb{SE}_3$ sont des groupes de Lie. On peut ainsi définir les cartes exponentielles et logarithmiques qui leur sont associées. On montre que l'algèbre de Lie de $\mathbb{SO}_3$, notée $\mathfrak{so}_3$, est constituée de l'ensemble des matrices anti-symétriques en dimension 3 et dont les éléments s'identifient aux vecteurs de $\mathbb{R}^3$ via l'opérateur $[\,\cdot\,]_\times$ défini dans l'équation A.1. On peut donc paramétrer $\mathfrak{so}_3$ par des vecteurs d'Euler de la forme $\boldsymbol{\omega} = \theta \boldsymbol{u}$ avec $\theta \in \mathbb{R}^+$ et $\boldsymbol{u}$ un vecteur unitaire de $\mathbb{R}^3$. La formule de Rodrigues définit donc l'exponentielle de $\mathbb{SO}_3$:

$$\mathrm{Exp}_{\mathbb{SO}_3}(\boldsymbol{\omega}) = \boldsymbol{I_3} + \sin\theta\,[\boldsymbol{u}]_\times + (1 - \cos\theta)\,[\boldsymbol{u}]_\times^2 \tag{A.14}$$

Cette application s'identifie d'ailleurs à la transformation de Cayley qui définit une application entre des matrices anti-symétriques et les matrices de rotations [4]. Étant donné une matrice $\boldsymbol{R} \in \mathbb{SO}_3$, le logarithme est défini comme suit :

$$\mathrm{Log}_{\mathbb{SO}_3}(\boldsymbol{R}) = \frac{\Phi}{2 \cdot \sin(\Phi)} \cdot (\boldsymbol{R} - \boldsymbol{R}^\top) \quad \text{où} \quad \Phi = \arccos\left(\frac{\mathrm{tr}(\boldsymbol{R}) - 1}{2}\right) \tag{A.15}$$

L'algèbre de Lie de $\mathbb{SE}_3$, notée $\mathfrak{se}_3$, est identifiée à $\mathbb{R}^6$, et comprend les éléments de la forme $\boldsymbol{\tau} = \begin{bmatrix}\boldsymbol{\omega} & \boldsymbol{\rho}\end{bmatrix}^\top$, où $\boldsymbol{\omega} \in \mathbb{R}^3$ est un vecteur rotation, tandis que $\boldsymbol{\rho} \in \mathbb{R}^3$ désigne l'incrément en translation. Ces deux éléments sont couplés dans l'exponentielle de $\mathbb{SE}_3$:

$$\mathrm{Exp}_{\mathbb{SE}_3}(\boldsymbol{\tau}) = \begin{bmatrix} \mathrm{Exp}_{\mathbb{SO}_3}(\boldsymbol{\omega}) & \boldsymbol{V}(\boldsymbol{\omega}) \cdot \boldsymbol{\rho} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{A.16}$$

4. Étant donné une matrice anti-symétrique $\boldsymbol{A}$ en dimension n, la transformation de Cayley définit la matrice $\boldsymbol{Q} = (\boldsymbol{I} - \boldsymbol{A}) \cdot (\boldsymbol{I} + \boldsymbol{A})^{-1} = (\boldsymbol{I} + \boldsymbol{A})^{-1} \cdot (\boldsymbol{I} - \boldsymbol{A})$ dont on montre qu'il s'agit d'une matrice orthogonal de $\mathbb{SO}_n$.

où $\boldsymbol{V}(\boldsymbol{\omega})$ désigne le Jacobien à gauche de $\mathbb{SE}_3$ ainsi défini :

$$\boldsymbol{V}(\boldsymbol{\omega}) = \boldsymbol{I_3} + \frac{1-\cos\theta}{\theta}[\boldsymbol{u}]_\times + \frac{\theta-\sin\theta}{\theta}\cdot[\boldsymbol{u}]_\times^2 \tag{A.17}$$

Le logarithme de $\mathbb{SE}_3$ est construit comme suit :

$$\mathrm{Log}_{\mathbb{SE}_3}\left(\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\\ \boldsymbol{0}_{1\times 3} & 1\end{bmatrix}\right) = \begin{bmatrix}\mathrm{Log}_{\mathbb{SO}_3}(\boldsymbol{R})\\ \boldsymbol{V}(\boldsymbol{\omega})^{-1}\cdot\boldsymbol{t}\end{bmatrix} \tag{A.18}$$

où intervient l'inverse du Jacobien à gauche de $\mathbb{SE}_3$. Pour contourner le couplage entre les incréments de rotation et de translation via le Jacobien à gauche, on pourra paramétrer $\mathbb{SE}_3$ à l'aide du groupe de Lie composite $\mathbb{SO}_3 \times \mathbb{R}^3$ dont la loi exponentielle associée est :

$$\mathrm{Exp}_{\{\mathbb{SO}_3\times\mathbb{R}^3\}}(\boldsymbol{\omega},\boldsymbol{t}) = \begin{bmatrix}\mathrm{Exp}_{\mathbb{SO}_3}(\boldsymbol{\omega})\\ \boldsymbol{t}\end{bmatrix} \cong \begin{bmatrix}\mathrm{Exp}_{\mathbb{SO}_3}(\boldsymbol{\omega}) & \boldsymbol{t}\\ \boldsymbol{0_{3\times 1}} & 1\end{bmatrix} \tag{A.19}$$

et dans lequel on transporte la loi de composition interne de $\mathbb{SE}_3$:

$$\begin{bmatrix}\boldsymbol{R}_1\\ \boldsymbol{t}_1\end{bmatrix} \oplus \begin{bmatrix}\boldsymbol{R}_2\\ \boldsymbol{t}_2\end{bmatrix} = \begin{bmatrix}\boldsymbol{R}_1\cdot\boldsymbol{R}_2\\ \boldsymbol{R}_1\cdot\boldsymbol{t}_2+\boldsymbol{t}_1\end{bmatrix} \tag{A.20}$$

Le chapitre d'Annexes C décrit le calcul de matrices Jacobiennes associées à l'opérateur $\oplus$ de $\mathbb{SE}_3$ en utilisant $\mathbb{SO}_3 \times \mathbb{R}^3$ comme paramétrisation sous-jacente. On pourra se référer à (Blanco 2010) pour une étude plus exhaustive des paramétrisations de $\mathbb{SE}_3$.

Annexe B

Estimation par maximum de vraisemblance

B.1 Modélisation et propagation des incertitudes

B.1.1 Modélisation générale des incertitudes

En théorie des probabilités, on définit une variable aléatoire X comme une application mesurable définie sur un espace probabilisé [1] Ω appelé univers ou espace des possibles.On parle de vecteur aléatoire lorsque X est à valeurs vectorielles. La variable aléatoire transporte ainsi la mesure de probabilité originelle sur son espace image Ω_X, et définit une mesure de probabilité image P_X qui caractérise l'incertitude associée à ses réalisations. En particulier, elle traduit numériquement le degré de connaissances de X. On associe généralement à P_X une densité de probabilité p_X telle que :

$$\forall A \in \mathcal{P}(\Omega_X),\ \ P_X(A) = \int_{x \in A} p_X(x) dx \tag{B.1}$$

L'interprétation fréquentiste considère que la mesure de probabilités quantifie la fréquence relative d'occurrence d'un événement, tandis que l'interprétation Bayésienne l'interprète comme la vraisemblance relative d'une hypothèse.

Les moments d'une variable aléatire fournissent des indicateurs de tendance et de dispersion qui permettent de la décrire. Étant donné une variable aléatoire X associée à une mesure de probablités P_X, on définit son espérance ou moyenne comme suit :

$$\mathbb{E}_{P_X}(X) = \int_{\Omega_X} X \, dP_X \tag{B.2}$$

Cet opérateur permet de construire des indicateurs de dispersion tels que la variance qui

1. C'est-à-dire muni d'une tribu et d'une mesure de probabilités.

caractérise l'espérances des écarts quadratiques autour de la moyenne :

$$\mathrm{var}_{P_X}(X) = \mathbb{E}_{P_X}\left((X - \mathbb{E}_{P_X}(X))^2\right) \tag{B.3}$$

L'opérateur de covariance étend la notion de variance pour caractériser les corrélations linéaires conjointes de deux variables aléatoires X et Y autour de leur moyenne :

$$\mathrm{cov}_{P_{XY}}(X, Y) = \mathbb{E}_{P_{XY}}\left((X - \mathbb{E}_{P_X}(X))(Y - \mathbb{E}_{P_Y}(Y))\right) \tag{B.4}$$

En particulier, l'opérateur de covariance définit un produit scalaire sur l'espace Hilbertien de dimension infinie $\mathcal{L}^2(\Omega, \mathcal{A}, P)$ [2] des variables aléatoires centrées de carré intégrable. Dans cet espace, l'orthogonalité entre deux variables aléatoires s'interprète comme l'absence de corrélations linéaires entre elles. Cela n'exclut pas en revanche d'autres corrélations fonctionnelles, et n'implique pas l'indépendance dans le cas général. De plus, la projection orthogonale $P_{Y_1,...,Y_n}(X)$ d'une variable aléatoire X sur le sous-espace engendré par une famille de variables aléatoires $Y_1, ..., Y_n$ correspond à la régression linéaire de X par les variables $Y_1, ..., Y_n$:

$$P_{Y_1,...,Y_n}(X) = \sum_{k=1}^{n} \frac{\mathrm{cov}(X, Y_k)}{\mathrm{var}(Y_k)} Y_k \tag{B.5}$$

Étant donné une famille de variables aléatoires $\mathcal{X} = (X_1, ..., X_n)$, on peut ainsi construire leur matrice de variance-covariance $\mathbf{\Sigma}$, qui correspond à leur matrice de Gram pour le produit scalaire définit par l'opérateur de covariance. Ainsi, l'inversibilité de la matrice de covariance équivaut à l'indépendance linéaires des X_i qui engendrent alors un sous-espace vectoriel de dimension n dans $\mathcal{L}^2(\Omega, \mathcal{A}, P)$. La matrice de covriance encode ainsi la structure linéaire de l'incertitude jointe, et comment l'information (ou l'incertitude) se propage entre les variables au premier ordre. La version standardisée de la matrice de covariance correspond à la matrice de corrélation, dont les coefficients quantifient l'intensité de la corrélation linéaire entre les variables. La matrice inverse $\mathbf{\Lambda}$ de la matrice de covariance est couramment appelée la matrice de précision. Elle corrrespond à la matrice de Gram de la base duale $\mathcal{X}^* = (X_1^*, ..., X_n^*)$ de $\mathcal{X}$. Par définition de la base duale, chaque élément X_i^* est proportionnel au résidu de la régression linéaire de X_i par toutes les autres variables $X_{j\neq i}$, via une constante de normalisation qui assure la relation $\mathrm{cov}(X_i^*, X_j) = \delta_{ij}$

2. On considère l'espace quotient dans lequel on confond les variables aléatoires qui presque sûrement égales.

où δ_{ij} est le symbole de Kronecker, et garantit que X_i^* est homogène à l'inverse de X_i. Alors que la matrice de covariance quantifie des corrélations complètes, la matrice de précision quantifie des corrélations partielles. Ces dernières quantifie les corrélations résiduelles entre deux variables lorsque l'on retire explicitement les corrélations induites par l'intermédiaire de tierces variables. En effet, ses coefficients diagonaux correspondent aux inverses des variances partielles, tandis que le coefficient de corrélation entre X_i^* et X_j^* correspond à l'opposé du coefficient de corrélation partielle $\rho_{X_i,X_j \cdot X_{k\neq i,j}}$ entre X_i et X_j en contrôlant toutes les autres variables. Étant donné un ensemble de variables de contrôle Z, la covariance partielle entre deux variables X et Y se définit comme suit :

$$\operatorname{cov}(X, Y \cdot Z) = \operatorname{cov}\left(X - P_Z(X), Y - P_Z(Y)\right) \tag{B.6}$$

Dans les cas particuliers des distributions elliptiques et Gaussiennes, la covariance partielle $\operatorname{cov}(X, Y \cdot Z)$ coïncide avec la covariance conditionnelle $\operatorname{cov}(X, Y|Z)$ mais ce n'est pas vrai dans le cas général (Baba et al. 2004). La covariance conditionnelle se définit ainsi :

$$\operatorname{cov}(X, Y|Z) = \operatorname{cov}\left(X - \mathbb{E}(X|Z), Y - \mathbb{E}(Y|Z)\right) \tag{B.7}$$

B.1.2 Mesure et distribution Gaussiennes

Bien qu'il existe une grande variété de lois de probabilités, nous utiliserons classiquement les lois Gaussiennes (ou lois normales) pour représenter l'incertitude sur les grandeurs vectorielles rencontrées. On caractérise une loi normale par ses deux premiers moments que sont sa moyenne $\boldsymbol{\mu}_X = \mathbb{E}_P(\boldsymbol{X})$ et sa matrice de variance-covariance $\boldsymbol{\Sigma}_X = \mathbb{E}_P\left((\boldsymbol{X} - \hat{\boldsymbol{X}})(\boldsymbol{X} - \hat{\boldsymbol{X}})^\top\right)$, et on lui associe la densité suivante :

$$\forall \boldsymbol{x} \in \mathbb{R}^n, \qquad \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X\right) = \frac{1}{\sqrt{(2\pi)^n \cdot \det \boldsymbol{\Sigma}_X}} \exp\left(-\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{\mu}_X\|^2_{\boldsymbol{\Sigma}_X}\right) \tag{B.8}$$

où $\|\boldsymbol{x}\|^2_{\boldsymbol{\Sigma}} = \boldsymbol{x}^\top.\boldsymbol{\Sigma}^{-1}.\boldsymbol{x}$ est la norme de Mahalanobis. L'incertitude encodée par la matrice de covariance peut d'ailleurs se représenter géométriquement par un ellipsoïde de confiance $\mathcal{E}_\alpha = \left\{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x}\|^2_{\boldsymbol{\Sigma}} \leq \chi^2_{n,\alpha}\right\}$ où $\chi^2_{n,\alpha}$ est le fractile d'ordre α pour une distribution de chi-deux à n degrés de liberté, avec par exemple $\alpha = 95\%$. Une autre façon de représenter l'incertitude est d'adopter la forme informationnelle de la distribution Gaussienne $\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \triangleq \mathcal{N}^{-1}\left(\boldsymbol{x}; \boldsymbol{\eta}, \boldsymbol{\Lambda}\right)$ avec $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ la matrice d'information et $\boldsymbol{\eta} = \boldsymbol{\Sigma}^{-1} \cdot \hat{\boldsymbol{x}}$ est le vecteur d'information.

B.1.3 La propagation des incertitudes

Un objet d'étude fondamental des probabilités est la propagation des incertitudes i.e. le transport des mesures de probabilités via des applications mesurables. Il s'agit de caractériser la loi P_Y d'une variable $Y = f(X)$ où f est une application mesurable, connaissant la loi P_X de X :

$$\forall \mathcal{A} \in \Omega_Y, \quad P_Y(\mathcal{A}) = P_X\left(f^{-1}(\mathcal{A})\right) \tag{B.9}$$

Supposons que sa densité soit $p_X(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$, alors P_Y préservera le caractère Gaussien de P_X si f est linéaire, mais pas dans le cas général. On pourra néanmoins approximer p_Y par une distribution Gaussienne en approximant f par sa partie linéaire en $\boldsymbol{\mu}_X$. C'est la loi de propagation des incertitudes :

$$p_Y(\boldsymbol{y}) \approx \mathcal{N}(\boldsymbol{y}; f(\boldsymbol{\mu}_X), \boldsymbol{\Sigma}_Y) \quad \text{avec} \quad \boldsymbol{\Sigma_Y} = \boldsymbol{J}^f_{\boldsymbol{x}}(\boldsymbol{\mu}_X) \cdot \boldsymbol{\Sigma_X} \cdot \boldsymbol{J}^f_{\boldsymbol{x}}(\boldsymbol{\mu}_X)^\top \tag{B.10}$$

où $\boldsymbol{J}^f_{\boldsymbol{x}}(\boldsymbol{\mu}_X)$ désigne la matrice Jacobienne de f par rapport à $\boldsymbol{x}$ évaluée en $\boldsymbol{\mu}_X$. D'autres techniques de propagation existent, telle que la transformée sans parfum (*Unscented Transform* en anglais) (Uhlmann 1995) qui adopte l'approche inverse : au lieu d'approximer la fonction f, elle approche la distribution initiale qu'elle caractérise par des points *sigmas*, et reconstitue une distribution Gaussienne à partir des images de ces points par f. Néanmoins, il faut noter que dans le cas de transformations fortement non-linéaires, l'approximation de la distribution par une densité Gaussienne peut résulter en d'importantes erreurs, et qu'une meilleure description nécessiterait l'emploi de densités plus complexes, telles que les mélanges de Gaussiennes par exemple.

Tout comme les dérivées, la matrice de covariance $\boldsymbol{\Sigma}_X$ d'une variable aléatoire x appartenant à un groupe de Lie $\mathcal{M}$ est caractérisée par une distribution gaussienne de densité $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$, s'exprime dans son espace tangent :

$$\boldsymbol{\Sigma}_X = \mathbb{E}\left((x \boxminus \hat{x}) \cdot (x \boxminus \hat{x})^\top\right) \tag{B.11}$$

Si $x \boxminus \hat{x} \in T_x\mathcal{M}$, alors la covariance est dite *locale* et notée ${}^x\boldsymbol{\Sigma}_X$, et l'incertitude sur x est caractérisée ainsi : $x = \hat{x} \circ \mathrm{Exp}({}^x\boldsymbol{\tau})$ où $p({}^x\boldsymbol{\tau}) = \mathcal{N}({}^x\boldsymbol{\tau}; \mathbf{0}, {}^x\boldsymbol{\Sigma}_X)$. Dans le cas contraire où $x \boxminus \hat{x}$ est un incrément global dans $T_\mathcal{E}\mathcal{M}$, alors elle est dite *globale*, elle est notée ${}^\mathcal{E}\boldsymbol{\Sigma}_X$ et $x = \mathrm{Exp}({}^\mathcal{E}\boldsymbol{\tau}) \circ \hat{x}$ avec $p({}^\mathcal{E}\boldsymbol{\tau}) = \mathcal{N}({}^\mathcal{E}\boldsymbol{\tau}; \mathbf{0}, {}^\mathcal{E}\boldsymbol{\Sigma}_X)$. La loi de propagation des incertitudes précédemment définie s'applique pareillement sur les covariances dans les espaces tangents.

B.1.4 La théorie de l'information

La théorie de l'information vise à quantifier l'information contenue dans une loi de probabilités. L'idée fondamentale est qu'une mesure de probabilités est d'autant plus informative que sa masse est concentrée, c'est-à-dire qu'elle diverge d'une loi uniforme par laquelle chaque hypothèse est équiprobable. Cette idée est concrétisée globalement par la notion d'entropie de Shannon dans le cas discret. On considère dans la suite que la variable X peut prendre N valeurs discrète distinctes. On associe à toute réalisation x_i d'une variable aléatoire X un degré de surprise $S_P(x_i) = -\log_b P(\{x_i\})$, d'autant plus fort que sa probabilité selon P était faible. Elle s'exprime en bits si $b = 2$ et en nats si $b = e$. On définit alors l'entropie de Shannon $H_P(P)$ comme l'espérance de la surprise :

$$H_P(P) = \mathbb{E}_{X\sim P}\left(S_P(X)\right) = -\sum_{i=}^{n} P(\{x_i\}) \log_b P(\{x_i\}) \tag{B.12}$$

On remarque qu'elle est maximale (et vaut alors $\log_b(N)$) pour une distribution uniforme (information nulle), et minimale (nulle) pour une distribution ponctuelle (information infinie). C'est en ce sens une mesure globale de l'incertitude. L'entropie croisée $H_P(Q)$ est l'espérance de la surprise si l'on considère la loi Q alors que la "vraie" loi est P :

$$H_P(Q) = \mathbb{E}_{X\sim P}\left(S_Q(X)\right) = -\sum_{i=1}^{n} P(\{x_i\}) \log_b Q(\{x_i\}) \geq 0 \tag{B.13}$$

La divergence de Kullback-Leibler, ou entropie relative, est l'espérance de la différence des surprises selon P et Q si P est la "vraie" loi de probabilités :

$$\mathcal{D}_{\mathrm{KL}}(P, Q) = \mathbb{E}_{X\sim P}\left(S_P(X) - S_Q(X)\right) = \sum_{i=1}^{n} P(\{x_i\}) \log_b \left(\frac{P(\{x_i\})}{Q(\{x_i\})}\right) \geq 0 \tag{B.14}$$

Attention, il ne s'agit pas d'une distance. On remarque en particulier qu'entropies croisées et relatives sont liées : $H_P(Q) = H_P(P) + \mathcal{D}_{\mathrm{KL}}(P, Q)$. L'entropie peut notamment s'exprimer en fonction de l'entropie relative avec la distribution uniforme U :

$$\mathcal{D}_{\mathrm{KL}}(P, U) = \log_b(N) - H_P(P) \tag{B.15}$$

Ainsi, l'entropie relative avec la loi uniforme semble correspondre au concept d'information. Les grandeurs précédentes sont étendues dans le cas continu où l'entropie différen-

tielle se substitue à l'entropie de Shannon :

$$h_p(p) = -\int p(x) \log_b p(x) dx \tag{B.16}$$

L'entropie différentielle est définie par analogie mais ne généralise exactement pas l'entropie de Shannon au cas continu. En particulier, elle peut prendre des valeurs négatives du fait que les densités de probabilités peuvent excéder la valeur 1. Une seconde difficulté à laquelle se confronte la définition de l'information comme la divergence avec la loi uniforme découle de l'impossibilité de définir des lois propres (i.e. de masse finie) sur des espaces non bornés. Pour une introduction plus détaillée à la théorie de l'information, on pourra se référer à (Cover et al. 2012).

B.2 Estimation MLE et graphes de facteurs

On suppose disposer d'un ensemble d'observations $\mathcal{Z}$ générées par un modèle d'observation $p(\mathcal{Z}|\Theta)$ paramétré par des variables cachées Θ. La loi de Bayes permet d'exprimer le problème d'estimation qui en découle :

$$p(\Theta|\mathcal{Z}) = \frac{p(\mathcal{Z}|\Theta) \cdot p(\Theta)}{p(\mathcal{Z})} \tag{B.17}$$

où $p(\Theta)$ est un a priori sur Θ, $p(\mathcal{Z})$ est un terme de normalisation sur l'ensemble des mesures, et $p(\mathcal{Z}|\Theta)$ est la probabilité conditionnelle des mesures par rapport aux variables, également appelée *vraisemblance* et notée $\mathcal{L}(\mathcal{Z}|\Theta)$. Lorsque les observations $\boldsymbol{z} \in \mathcal{Z}$ sont indépendantes les unes des autres, celle-ci se décompose comme suit :

$$\mathcal{L}(\mathcal{Z}|\Theta) = \prod_{z \in \mathcal{Z}} p(z|\Theta) \tag{B.18}$$

L'équation de Bayes permet ainsi de formuler un problème d'inférence qui consiste à estimer les paramètres $\Theta \in \mathcal{H}_\Theta$ qui corroborent le plus l'ensemble des observations, ce qui se traduit par le problème d'optimisation suivant :

$$\hat{\Theta}_{\text{MAP}} = \arg\max_{\Theta} \; p(\Theta|\mathcal{Z}) = \arg\max_{\Theta} \; \frac{p(\mathcal{Z}|\Theta) \cdot p(\Theta)}{p(\mathcal{Z})} = \arg\max_{\Theta} \; p(\mathcal{Z}|\Theta) \cdot p(\Theta) \tag{B.19}$$

Il s'agit d'un problème d'estimation dit MAP (*Maximum A Posteriori*) car il maximise la distribution postérieure. Cependant, dans le cas où l'on ne dispose d'aucune information a priori sur les variables, on peut alors se contenter d'optimiser uniquement la vraisemblance des observations, d'où le nom d'*estimation par maximisation de la vraisemblance* ou *Maximum Likelihood Estimation* (MLE) en anglais [3] :

$$\hat{\Theta}_{\text{MLE}} = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\max}\ p(\mathcal{Z}|\Theta) = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\max}\ \mathcal{L}(\mathcal{Z}|\Theta) \tag{B.20}$$

On peut alors de façon équivalente minimiser la log-vraisemblance négative pour transformer le produit en somme, et obtenir le problème d'optimisation suivant :

$$\hat{\Theta}_{\text{MLE}} = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\min}\ -\sum_{z \in \mathcal{Z}} \log p(z|\Theta) \tag{B.21}$$

Dans le cas gaussien, chaque mesure est associée à un facteur gaussien :

$$p(z|\Theta) = \mathcal{N}\left(\boldsymbol{\xi}_z(\Theta); \mathbf{0}, \boldsymbol{\Sigma}_z\right) \tag{B.22}$$

où $\boldsymbol{\xi}_z(\Theta)$ est un résidu associée à la mesure et $\boldsymbol{\Sigma}_z$ est la matrice de covariance qui le caractérise. On obtient alors un problème d'optimisation au sens des moindres carrés :

$$\hat{\Theta}_{\text{MLE}} = \underset{\Theta \in \mathcal{H}_\Theta}{\arg\min}\ \sum_{z \in \mathcal{Z}} \|\boldsymbol{\xi}_z(\Theta)\|^2_{\boldsymbol{\Sigma}_z} = \|\boldsymbol{\xi}_{\mathcal{Z}}(\Theta)\|^2_{\boldsymbol{\Sigma}_{\mathcal{Z}}} \tag{B.23}$$

où $\boldsymbol{\xi}_{\mathcal{Z}}(\Theta)$ désigne la concaténation des résidus des facteurs de $\mathcal{Z}$ et $\boldsymbol{\Sigma}_{\mathcal{Z}}$ est la matrice de covariance associée. On résout usuellement ce type de problème à l'aide des algorithmes de Gauss-Newton ou de Levenberg-Marquardt expliqués ci-dessous. L'estimation par maximum de vraisemblance s'adapte particulièrement aux problèmes sur-déterminés tels que le SLAM. Elle requiert cependant d'encoder la vraisemblance des mesures $\mathcal{L}(\mathcal{Z}|\Theta)$. Les représentations de types *graphes de facteurs* introduisent pour cela un formalisme graphique particulièrement adapté et flexible. Il s'agit de graphes bipartites liant deux types de noeuds : les variables et les facteurs. Chaque facteur exprime une contrainte probabiliste induite par une mesure $z \in \mathcal{Z}$ sur les variables $\Theta_z \subset \Theta$ qui lui sont adjacentes dans le graphe. Le formalisme des graphes de facteurs permet d'une part d'intégrer des contraintes de natures diverses déduites de modèles dynamiques ou d'observation, et ainsi

3. Le terme a priori est alors éventuellement remplacé par une mesure uniforme, éventuellement impropre sur les espaces d'hypothèse non bornés.

de fusionner des capteurs variés, et d'autre part de mettre à profit les propriétés issues de la théorie des graphes pour l'estimation. Pour une introduction plus détaillée aux graphes de facteurs dans le contexte de l'estimation robotique, on pourra se référer à (Grisetti et al. 2010a) et (Dellaert et al. 2017).

B.3 Algorithme de Levenberg-Marquardt

L'estimation MLE se traduit par un problème d'optimisation en général non-linéaire tel que présenté dans l'équation B.23. Classiquement, on résout ce type de problème itérativement en calculant à chaque itération k un incrément $\delta\Theta_k$ qui minimise une approximation du problème original en l'estimée courante Θ_k. Il s'agit généralement d'approximations linéaires ou quadratiques de la fonction de coût, mais des approximations plus spécifiques à la structure du problème sont possibles comme dans le cas des problèmes aux moindres carrés. L'algorithme de Gauss-Newton consiste à optimiser l'approximation quadratique de la fonction de coût engendrée par une linéarisation des facteurs :

$$\boldsymbol{\xi}_{\mathcal{Z}}(\Theta_k + \delta\Theta_k) = \boldsymbol{\xi}_{\mathcal{Z}}(\Theta_k) + \boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}}}_{\Theta_k}(\Theta_k) \cdot \delta\Theta_k + o\left(\|\delta\Theta_k\|^2\right) \tag{B.24}$$

Ainsi, la fonction de coût approximée s'exprime :

$$\begin{aligned}
\mathcal{L}(\Theta_k + \delta\Theta_k, \mathcal{Z}) &\approx \left\|\boldsymbol{\xi}_{\mathcal{Z}}(\Theta_k) + \boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}}}_{\Theta_k}(\Theta_k) \cdot \delta\Theta_k\right\|^2_{\boldsymbol{\Sigma}_{\mathcal{Z}}} && \text{(B.25a)} \\
&\approx \left\|\boldsymbol{\xi}_{\mathcal{Z}}(\Theta_k)\right\|^2_{\boldsymbol{\Sigma}_{\mathcal{Z}}} + 2\left\langle \boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}}}_{\Theta_k}(\Theta_k) \cdot \delta\Theta_k \mid \boldsymbol{\xi}_{\mathcal{Z}}(\Theta_k)\right\rangle_{\boldsymbol{\Sigma}_{\mathcal{Z}}} + \left\|\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}}}_{\Theta_k}(\Theta_k) \cdot \delta\Theta_k\right\|^2_{\boldsymbol{\Sigma}_{\mathcal{Z}}} && \text{(B.25b)}
\end{aligned}$$

où $\langle\cdot|\cdot\rangle$ désigne le produit scalaire associé à $\boldsymbol{\Sigma}_{\mathcal{Z}}^{-1}$. Cette approximation quadratique se minimise en annulant son gradient, dont on dérive les *équations normales* :

$$\left(\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}}}_{\Theta_k}(\Theta_k)^\top \cdot \boldsymbol{\Sigma}_{\mathcal{Z}}^{-1} \cdot \boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}}}_{\Theta_k}(\Theta_k)\right) \delta\Theta_k^* = -\boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}}}_{\Theta_k}(\Theta_k)^\top \cdot \boldsymbol{\Sigma}_{\mathcal{Z}}^{-1} \cdot \boldsymbol{\xi}_{\mathcal{Z}}(\Theta_k) \tag{B.26}$$

La structure de la matrice Hessienne approximée $\boldsymbol{H}_k = \boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}}}_{\Theta_k}(\Theta_k)^\top \cdot \boldsymbol{\Sigma}_{\mathcal{Z}}^{-1} \cdot \boldsymbol{J}^{\boldsymbol{\xi}_{\mathcal{Z}}}_{\Theta_k}(\Theta_k)$, en particulier son *éparsité*, guide le choix des algorithmes utilisés pour résoudre ce système linéaire, telles que la factorisation QR ou la factorisation de Cholesky. L'incrément $\delta\Theta_k^*$ ainsi calculé est ensuite utilisé pour mettre à jour l'estimée courante $\Theta_{k+1} = \Theta_k + \delta\Theta_k^*$. (Grisetti et al. 2020) présentent une revue détaillée des méthodes de résolution des

problèmes aux moindres carrés en contexte robotique.

L'algorithme de Levenberg-Marquardt étend l'algorithme de Gauss-Newton en une méthode dite *à région de confiance*. Contrairement aux méthodes de recherche linéaire qui déterminent d'abord une direction de descente puis la taille du pas, les méthodes à région de confiance déterminent préalablement un voisinage de l'estimée courante sur lequel l'approximation minimisée est jugée suffisamment fidèle à la fonction de coût, puis recherche l'incrément dans cette région. Dans l'algorithme de Levenberg-Marquardt, cela se traduit par l'ajout d'un facteur d'amortissement aux équations normales :

$$\left(\boldsymbol{J}_{\Theta_k}^{\boldsymbol{\xi}_\mathcal{Z}}(\Theta_k)^\top \cdot \boldsymbol{\Sigma}_\mathcal{Z}^{-1} \cdot \boldsymbol{J}_{\Theta_k}^{\boldsymbol{\xi}_\mathcal{Z}}(\Theta_k) + \lambda \cdot \operatorname{diag}\left(\boldsymbol{J}_{\Theta_k}^{\boldsymbol{\xi}_\mathcal{Z}}(\Theta_k)^\top \cdot \boldsymbol{\Sigma}_\mathcal{Z}^{-1} \cdot \boldsymbol{J}_{\Theta_k}^{\boldsymbol{\xi}_\mathcal{Z}}(\Theta_k)\right)\right) \delta\Theta_k^* = -\boldsymbol{J}_{\Theta_k}^{\boldsymbol{\xi}_\mathcal{Z}}(\Theta_k)^\top \cdot \boldsymbol{\Sigma}_\mathcal{Z}^{-1} \cdot \boldsymbol{\xi}_\mathcal{Z}(\Theta_k) \quad \text{(B.27)}$$

Cette amortissement est propotionnel à la courbure de la fonction de coût que reflète la diagonale de la Hessienne approximée via un facteur λ. Un amortissement important rapproche d'une descente de gradient classique tandis qu'un amortissement faible rapproche d'une itération de typss-Newton. À chaque itération, il est respectivement augmenté ou diminué (en général d'un facteur 2) selon que l'accroissement effectif de la focntion de coût est correctement prédit par l'approximation. Si l'accroissement est insuffisant, l'itération n'est pas appliquée et elle refaite avec un amortissement accru.

Les algorithmes de Gauss-Newton et de Levenverg-Marquardt se transposent naturellement sur les groupes de Lie en calculant les incréments dans les espaces tangents, de façon à préserver la structure des objets estimés. Cette approche impose de modifier la fonction de coût ainsi que la mise à jour de l'estimée courante à l'aide des opérateurs $\boxplus$ et $\boxminus$. Il s'agit du l'approche *lift-solve-retract* détaillée dans (Absil et al. 2007).

B.4 Matrice d'information de Fisher

Le but de l'estimation MLE est de caractériser la distribution sur les paramètres Θ qui maximise la vraisemblance des mesures $\mathcal{Z}$ selon la loi de Bayes. Les résultats issus de la théorie de l'information permettent d'approximer cette distribution localement autour de $\hat{\Theta}_{\text{MLE}}$ par une loi Gaussienne $p(\Theta|\mathcal{Z}) = \mathcal{N}\left(\Theta; \hat{\Theta}_{\text{MLE}}, \left[\boldsymbol{\mathcal{I}}_\mathcal{Z}^\Theta(\hat{\Theta}_{\text{MLE}})\right]^{-1}\right)$ où $\boldsymbol{\mathcal{I}}_\mathcal{Z}^\Theta(\hat{\Theta}_{\text{MLE}})$

est la matrice d'information observée de Fisher évaluée en $\hat{\Theta}_{\text{MLE}}$. En effet, à l'ordre 2 :

$$\log p(\Theta|\mathcal{Z}) \approx \log p(\hat{\Theta}_{\text{MLE}}|\mathcal{Z}) + \underbrace{\left\{ \frac{\partial}{\partial \Theta} \log p(\Theta|\mathcal{Z}) \Big|_{\hat{\Theta}_{\text{MLE}}} \right\}}_{=0 \text{ à l'optimum}} (\Theta \boxminus \hat{\Theta}_{\text{MLE}}) + (\Theta \boxminus \hat{\Theta}_{\text{MLE}})^\top \left\{ \frac{1}{2} \frac{\partial^2}{\partial \Theta^2} \log p(\Theta|\mathcal{Z}) \Big|_{\hat{\Theta}_{\text{MLE}}} \right\} (\Theta \boxminus \hat{\Theta}_{\text{MLE}}) \quad \text{(B.28)}$$

Ainsi, en définissant :

$$\boldsymbol{\mathcal{I}}_{\mathcal{Z}}^{\Theta}(\hat{\Theta}_{\text{MLE}}) = -\mathbb{E}_{\mathcal{Z}} \left(\frac{\partial^2}{\partial^2 \Theta} \log \mathcal{L}(\mathcal{Z}|\Theta) \Big|_{\hat{\Theta}_{\text{MLE}}} \right) \quad \text{(B.29)}$$

on obtient l'approximation suivante :

$$p(\Theta|\mathcal{Z}) \propto \exp\left(-\frac{1}{2} (\Theta \boxminus \hat{\Theta}_{\text{MLE}})^\top \left[\boldsymbol{\mathcal{I}}_{\mathcal{Z}}^{\Theta}(\hat{\Theta}_{\text{MLE}}) \right] (\Theta \boxminus \hat{\Theta}_{\text{MLE}}) \right) \quad \text{(B.30)}$$

ce qui correspond à une distribution Gaussienne. La matrice d'information de Fisher se définit ainsi comme la courbure de la log-vraisemblance négative. On considère donc que les observations apportent d'autant plus d'information sur les paramètres qu'elles façonnent une vraisemblance dont l'optimum est piqué. En pratique, on calcule la matrice de Fisher observée en approximant les espérances par des sommes sur les observations disponibles. Dans le cas Gaussien où la vraisemblance vaut $\mathcal{L}(\mathcal{Z}|\Theta) = \mathcal{N}\left(\boldsymbol{\xi}_{\mathcal{Z}}(\Theta); \mathbf{0}, \boldsymbol{\Sigma}_{\mathcal{Z}}\right)$, la matrice d'information de Fisher se calcule explicitement avec la formule suivante :

$$\boldsymbol{\mathcal{I}}_{\mathcal{Z}}^{\Theta}(\hat{\Theta}_{\text{MLE}}) = \boldsymbol{J}_{\Theta}^{\boldsymbol{\xi}_{\mathcal{Z}}}(\hat{\Theta}_{\text{MLE}})^\top \cdot \boldsymbol{\Sigma}_{\mathcal{Z}}^{-1} \cdot \boldsymbol{J}_{\Theta}^{\boldsymbol{\xi}_{\mathcal{Z}}}(\hat{\Theta}_{\text{MLE}}) \quad \text{(B.31a)}$$

$$= \sum_{z \in \mathcal{Z}} \boldsymbol{J}_{\Theta}^{\boldsymbol{\xi}_{z}}(\hat{\Theta}_{\text{MLE}})^\top \cdot \boldsymbol{\Sigma}_{z}^{-1} \cdot \boldsymbol{J}_{\Theta}^{\boldsymbol{\xi}_{z}}(\hat{\Theta}_{\text{MLE}}) \quad \text{(B.31b)}$$

Cette forme explicite comment l'information de chaque facteur est projetée sur les paramètres estimés via la matrice Jacobienne correspondante. On retrouve dans cette formule le caractère additif de la matrice d'information. La matrice de covariance associée se calcule en inversant la matrice de Fisher. On remarque également qu'elle s'identifie à la matrice Hessienne approximée des équations normales.

Annexe C

Calcul des matrices Jacobiennes de l'opérateur $\oplus$ dans $\mathbb{SO}_3 \times \mathbb{R}^3$

Ce chapitre expose le calcul de la matrice Jacobienne du produit $\oplus$ de poses de $\mathbb{SE}_3$ transporté sur le groupe de Lie composite $\mathbb{SO}_3 \times \mathbb{R}^3$ que nous utilisons dans ce mémoire afin de paramétrer $\mathbb{SE}_3$. Dans la suite, on considère deux poses $T_1, T_2 \in \mathbb{SO}_3 \times \mathbb{R}^3$:

$$T_1 = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{t}_1 \end{bmatrix} \qquad T_2 = \begin{bmatrix} \boldsymbol{R}_2 \\ \boldsymbol{t}_2 \end{bmatrix} \tag{C.1}$$

Dans la suite, on considère un incrément $\boldsymbol{\delta\tau} = \begin{bmatrix} \boldsymbol{\delta\theta}^\top & \boldsymbol{\delta p}^\top \end{bmatrix}^\top$ tel que $\boldsymbol{\delta\theta}^\wedge \in \mathfrak{so}_3$ est un incrément en rotation et $\boldsymbol{\delta p} \in \mathbb{R}^3$ un incrément en translation. On rappelle la définition des opérateurs globaux $\boxplus$ et $\boxminus$ sur $\mathbb{SO}_3 \times \mathbb{R}^3$:

$$\boldsymbol{\delta\tau} \boxplus T_1 = \begin{bmatrix} \exp_{\mathbb{SO}_3}(\delta\theta^\wedge) \cdot \boldsymbol{R}_1 \\ \boldsymbol{t}_1 + \boldsymbol{\delta p} \end{bmatrix} \qquad T_2 \boxminus T_1 = \begin{bmatrix} \log_{\mathbb{SO}_3}\left(\boldsymbol{R}_2 \cdot \boldsymbol{R}_1^\top\right)^\vee \\ \boldsymbol{t}_2 - \boldsymbol{t}_1 \end{bmatrix} \tag{C.2}$$

L'opérateur $\oplus$ se définit ainsi sur $\mathbb{SO}_3 \times \mathbb{R}^3$:

$$T_1 \oplus T_2 = \begin{bmatrix} \boldsymbol{R}_1 \cdot \boldsymbol{R}_2 \\ \boldsymbol{t}_1 + \boldsymbol{R}_1 \cdot \boldsymbol{t}_2 \end{bmatrix} \tag{C.3}$$

La matrice Jacobienne $\boldsymbol{J}^{T_1^{-1} \oplus T_2}_{\{T_1, T_2\}}$ [1] se décompose comme suit :

$$\boldsymbol{J}^{T_1^{-1} \oplus T_2}_{\{T_1, T_2\}} = \begin{bmatrix} \boldsymbol{J}^{T_1^{-1} \oplus T_2}_{T_1} & \boldsymbol{J}^{T_1^{-1} \oplus T_2}_{T_2} \end{bmatrix} \tag{C.4}$$

1. Afin d'alléger les notations dans la suite, on confond les variables par rapport auxquelles les dérivations sont faites et les valeurs en lesquelles les matrices Jacobiennes sont évaluées i.e. $\boldsymbol{J}^{T_1^{-1} \oplus T_2}_{\{T_1, T_2\}} = \boldsymbol{J}^{T_1^{-1} \oplus T_2}_{\{T_1, T_2\}}(T_1, T_2)$.

où $\boldsymbol{J}_{T_1}^{T_1^{-1}\oplus T_2}$ et $\boldsymbol{J}_{T_2}^{T_1^{-1}\oplus T_2}$ sont les matrices Jacobiennes de $T_1^{-1} \oplus T_2$ par rapport à T_1 et T_2 respectivement. La règle de composition en chaîne permet de décomposer la matrice $\boldsymbol{J}_{T_1}^{T_1^{-1}\oplus T_2}$ ainsi :

$$\boldsymbol{J}_{T_1}^{T_1^{-1}\oplus T_2} = \boldsymbol{J}_{T_1^{-1}}^{T_1^{-1}\oplus T_2} \cdot \boldsymbol{J}_{T_1}^{T_1^{-1}} \tag{C.5}$$

Nous devons donc calculer les trois matrices jacobiennes suivantes : $\boldsymbol{J}_{T_1}^{T_1\oplus T_2}$ (produit à droite), $\boldsymbol{J}_{T_1}^{T_2\oplus T_1}$ (produit à gauche) et $\boldsymbol{J}_{T_1}^{T_1^{-1}}$ (inverse).

Calcul de $\boldsymbol{J}_{T_1}^{T_1\oplus T_2}$

$$\boldsymbol{J}_{T_1}^{T_1\oplus T_2} = \left.\frac{\partial}{\partial \boldsymbol{\delta\tau}}\right|_{\boldsymbol{\delta\tau}=\mathbf{0}} ((\boldsymbol{\delta\tau} \boxplus T_1) \oplus T_2) \boxminus (T_1 \oplus T_2) \tag{C.6}$$

où

$$(\boldsymbol{\delta\tau} \boxplus T_1) \oplus T_2 = \begin{bmatrix} \exp_{\mathbb{SO}_3}(\boldsymbol{\delta\theta}^\wedge) \cdot \boldsymbol{R}_1 \cdot \boldsymbol{R}_2 \\ \exp_{\mathbb{SO}_3}(\boldsymbol{\delta\theta}^\wedge) \cdot \boldsymbol{R}_1 \cdot \boldsymbol{t}_2 + \boldsymbol{t}_1 + \boldsymbol{\delta p} \end{bmatrix} \tag{C.7a}$$

$$T_1 \oplus T_2 = \begin{bmatrix} \boldsymbol{R}_1 \cdot \boldsymbol{R}_2 \\ \boldsymbol{R}_1 \cdot \boldsymbol{t}_2 + \boldsymbol{t}_1 \end{bmatrix} \tag{C.7b}$$

d'où

$$\boldsymbol{J}_{T_1}^{T_1\oplus T_2} = \left.\frac{\partial}{\partial \boldsymbol{\delta\tau}}\right|_{\boldsymbol{\delta\tau}=\mathbf{0}} \begin{bmatrix} \boldsymbol{\delta\theta} \\ \left(\exp_{\mathbb{SO}_3}(\boldsymbol{\delta\theta}^\wedge) - \boldsymbol{I}_3\right) \cdot \boldsymbol{R}_1 \cdot \boldsymbol{t}_2 + \boldsymbol{\delta p} \end{bmatrix} \tag{C.8a}$$

$$= \left.\frac{\partial}{\partial \boldsymbol{\delta\tau}}\right|_{\boldsymbol{\delta\tau}=\mathbf{0}} \begin{bmatrix} \boldsymbol{\delta\theta} \\ [\boldsymbol{\delta\theta}]_\times \cdot \boldsymbol{R}_1 \cdot \boldsymbol{t}_2 + \boldsymbol{\delta p} \end{bmatrix} \tag{C.8b}$$

$$= \left.\frac{\partial}{\partial \boldsymbol{\delta\tau}}\right|_{\boldsymbol{\delta\tau}=\mathbf{0}} \begin{bmatrix} \boldsymbol{\delta\theta} \\ -[\boldsymbol{R}_1 \cdot \boldsymbol{t}_2]_\times \cdot \boldsymbol{\delta\theta} + \boldsymbol{\delta p} \end{bmatrix} \tag{C.8c}$$

en utilisant le développement au premier ordre $\exp_{\mathbb{SO}_3}(\boldsymbol{\delta\theta}^\wedge) \approx \boldsymbol{I}_3 + [\boldsymbol{\delta\theta}]_\times$ et la relation $[\boldsymbol{a}]_\times \cdot \boldsymbol{b} = -[\boldsymbol{b}]_\times \cdot \boldsymbol{a}$. Ainsi :

$$\boxed{\boldsymbol{J}_{T_1}^{T_1\oplus T_2} = \begin{bmatrix} \boldsymbol{I}_3 & \mathbf{0}_3 \\ -[\boldsymbol{R}_1 \cdot \boldsymbol{t}_2]_\times & \boldsymbol{I}_3 \end{bmatrix}} \tag{C.9}$$

Calcul de $\boldsymbol{J}_{T_1}^{T_2 \oplus T_1}$

$$\boldsymbol{J}_{T_1}^{T_2 \oplus T_1} = \left. \frac{\partial}{\partial \boldsymbol{\delta\tau}} \right|_{\boldsymbol{\delta\tau}=\mathbf{0}} (T_2 \oplus (\boldsymbol{\delta\tau} \boxplus T_1)) \boxminus (T_2 \oplus T_1) \tag{C.10}$$

où

$$T_2 \oplus (\boldsymbol{\delta\tau} \boxplus T_1) = \begin{bmatrix} \boldsymbol{R}_2 \cdot \exp_{\mathbb{SO}_3}(\boldsymbol{\delta\theta}^\wedge) \cdot \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \cdot (\boldsymbol{t}_1 + \boldsymbol{\delta p}) + \boldsymbol{t}_2 \end{bmatrix} \tag{C.11a}$$

$$T_2 \oplus T_1 = \begin{bmatrix} \boldsymbol{R}_2 \cdot \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \cdot \boldsymbol{t}_1 + \boldsymbol{t}_2 \end{bmatrix} \tag{C.11b}$$

d'où

$$\boldsymbol{J}_{T_1}^{T_2 \oplus T_1} = \left. \frac{\partial}{\partial \boldsymbol{\delta\tau}} \right|_{\boldsymbol{\delta\tau}=\mathbf{0}} \begin{bmatrix} \left(\boldsymbol{R}_2 \cdot \boldsymbol{\delta\theta}^\wedge \cdot \boldsymbol{R}_2^\top \right)^\vee \\ \boldsymbol{R}_2 \cdot \boldsymbol{\delta p} \end{bmatrix} \tag{C.12a}$$

$$= \left. \frac{\partial}{\partial \boldsymbol{\delta\tau}} \right|_{\boldsymbol{\delta\tau}=\mathbf{0}} \begin{bmatrix} \boldsymbol{Ad}_{\boldsymbol{R}_2} \cdot \boldsymbol{\delta\theta} \\ \boldsymbol{R}_2 \cdot \boldsymbol{\delta p} \end{bmatrix} \tag{C.12b}$$

L'opérateur adjoint sur $\mathbb{SO}_3$ est tel que $\boldsymbol{Ad}_{\boldsymbol{R}_2} = \boldsymbol{R}_2$. Ainsi :

$$\boxed{\boldsymbol{J}_{T_1}^{T_2 \oplus T_1} = \begin{bmatrix} \boldsymbol{R}_2 & \mathbf{0}_3 \\ \mathbf{0}_3 & \boldsymbol{R}_2 \end{bmatrix}} \tag{C.13}$$

Calcul de $\boldsymbol{J}_{T_1}^{T_1^{-1}}$

$$\boldsymbol{J}_{T_1}^{T_1^{-1}} = \left. \frac{\partial}{\partial \boldsymbol{\delta\tau}} \right|_{\boldsymbol{\delta\tau}=\mathbf{0}} (\boldsymbol{\delta\tau} \boxplus T_1)^{-1} \boxminus T_1^{-1} \tag{C.14}$$

où

$$(\boldsymbol{\delta\tau} \boxplus T_1)^{-1} = \begin{bmatrix} \boldsymbol{R}_1^\top \cdot \exp_{\mathbb{SO}_3}(-\boldsymbol{\delta\theta}^\wedge) \\ -\boldsymbol{R}_1^\top \cdot \exp_{\mathbb{SO}_3}(-\boldsymbol{\delta\theta}^\wedge) \cdot (\boldsymbol{t}_1 + \boldsymbol{\delta p}) \end{bmatrix} \tag{C.15}$$

d'où

$$\boldsymbol{J}_{T_1}^{T_1^{-1}} = \frac{\partial}{\partial \boldsymbol{\delta\tau}}\bigg|_{\boldsymbol{\delta\tau}=\mathbf{0}} \begin{bmatrix} \left(\boldsymbol{R}_1^\top \cdot \exp_{\mathbb{SO}_3}(-\boldsymbol{\delta\theta}^\wedge) \cdot \boldsymbol{R}_1\right)^\vee \\ -\boldsymbol{R}_1^\top \cdot \exp_{\mathbb{SO}_3}(-\boldsymbol{\delta\theta}^\wedge) \cdot (\boldsymbol{t}_1 + \boldsymbol{\delta p}) \end{bmatrix} \tag{C.16a}$$

$$= \frac{\partial}{\partial \boldsymbol{\delta\tau}}\bigg|_{\boldsymbol{\delta\tau}=\mathbf{0}} \begin{bmatrix} -\boldsymbol{Ad}_{\boldsymbol{R}_1}^{-1} \cdot \boldsymbol{\delta\theta} \\ -\boldsymbol{R}_1^\top \cdot (\boldsymbol{I}_3 - [\boldsymbol{\delta\theta}]_\times) \cdot (\boldsymbol{t}_1 + \boldsymbol{\delta p}) \end{bmatrix} \tag{C.16b}$$

$$= \frac{\partial}{\partial \boldsymbol{\delta\tau}}\bigg|_{\boldsymbol{\delta\tau}=\mathbf{0}} \begin{bmatrix} -\boldsymbol{Ad}_{\boldsymbol{R}_1}^{-1} \cdot \boldsymbol{\delta\theta} \\ \boldsymbol{R}_1^\top \cdot [\boldsymbol{\delta\theta}]_\times \cdot (\boldsymbol{t}_1 - \boldsymbol{R}_1^\top \cdot \boldsymbol{\delta p}) \end{bmatrix} \tag{C.16c}$$

Ainsi :

$$\boxed{\boldsymbol{J}_{T_1}^{T_1^{-1}} = \begin{bmatrix} -\boldsymbol{R}_1^\top & \mathbf{0}_3 \\ -\boldsymbol{R}_1^\top \cdot [\boldsymbol{t}_1]_\times & -\boldsymbol{R}_1^\top \end{bmatrix}} \tag{C.17}$$

Calcul de $\boldsymbol{J}_{T_1}^{T_1^{-1} \oplus T_2}$

$$\boldsymbol{J}_{T_1}^{T_1^{-1} \oplus T_2} = \boldsymbol{J}_{T_1^{-1}}^{T_1^{-1} \oplus T_2} \cdot \boldsymbol{J}_{T_1}^{T_1^{-1}} \tag{C.18a}$$

$$= \begin{bmatrix} \boldsymbol{I}_3 & \mathbf{0}_3 \\ -[\boldsymbol{R}_1^\top \cdot \boldsymbol{t}_2]_\times & \boldsymbol{I}_3 \end{bmatrix} \cdot \begin{bmatrix} -\boldsymbol{R}_1^\top & \mathbf{0}_3 \\ -\boldsymbol{R}_1^\top \cdot [\boldsymbol{t}_1]_\times & -\boldsymbol{R}_1^\top \end{bmatrix} \tag{C.18b}$$

donc

$$\boxed{\boldsymbol{J}_{T_1}^{T_1^{-1} \oplus T_2} = \begin{bmatrix} -\boldsymbol{R}_1^\top & \mathbf{0}_3 \\ -\boldsymbol{R}_1^\top \cdot [\boldsymbol{t}_2 - \boldsymbol{t}_1]_\times & -\boldsymbol{R}_1^\top \end{bmatrix}} \tag{C.19}$$

BIBLIOGRAPHIE

ABSIL, P-A et al. (2007), « Trust-region methods on Riemannian manifolds », in : *Foundations of Computational Mathematics* 7.*3*, p. 303-330.

ACHTELIK, Markus W et al. (2011), « Collaborative stereo », in : *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 2242-2248.

AGARWAL, Sameer et al. (2012), « Ceres solver », in :

AGRAWAL, Motilal et al. (2008), « Censure : Center surround extremas for realtime feature detection and matching », in : *European Conference on Computer Vision*, Springer, p. 102-115.

ALAHI, Alexandre et al. (2012), « Freak : Fast retina keypoint », in : *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Ieee, p. 510-517.

ANDERSSON, Lars AA et al. (2008), « C-SAM : Multi-robot SLAM using square root information smoothing », in : *2008 IEEE International Conference on Robotics and Automation*, IEEE, p. 2798-2805.

ANTONINI, Amado et al. (2018), « The blackbird dataset : A large-scale dataset for UAV perception in aggressive flight », in : *arXiv preprint arXiv :1810.01987*.

ARANDJELOVIC, Relja et al. (2016), « NetVLAD : CNN architecture for weakly supervised place recognition », in : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 5297-5307.

BABA, Kunihiro et al. (2004), « Partial correlation and conditional correlation as measures of conditional independence », in : *Australian & New Zealand Journal of Statistics* 46.*4*, p. 657-664.

BAILEY, Tim et al. (2006a), « Consistency of the EKF-SLAM algorithm », in : *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 3562-3568.

BAILEY, Tim et al. (2006b), « Simultaneous localization and mapping (SLAM) : Part II », in : *IEEE robotics & automation magazine* 13.*3*, p. 108-117.

BAKER, Simon et al. (2004), « Lucas-kanade 20 years on : A unifying framework », in : *International journal of computer vision* 56.*3*, p. 221-255.

BARFOOT, Timothy D (2017), *State estimation for robotics*, Cambridge University Press.

Barrau, Axel et al. (2015), « An EKF-SLAM algorithm with consistency properties », in : *arXiv preprint arXiv :1510.06263*.

Barreto, Joao Pedro et al. (2001), « Issues on the geometry of central catadioptric image formation », in : *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, t. 2, IEEE, p. II-II.

Bay, Herbert et al. (2006), « Surf : Speeded up robust features », in : *European conference on computer vision*, Springer, p. 404-417.

Beinhofer, Maximilian et al. (2013), « Robust landmark selection for mobile robot navigation », in : *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 3637-2643.

Bencina, Ross et al. (2005), « Improved topological fiducial tracking in the reactivision system », in : *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, IEEE, p. 99-99.

Bentley, Jon Louis (1975), « Multidimensional binary search trees used for associative searching », in : *Communications of the ACM* 18.*9*, p. 509-517.

Berkelaar, Michel (2007), « The lpsolve package », in : *URL : http ://www. lpsolve. sourceforge. net*.

Besl, Paul J et al. (1992), « Method for registration of 3-D shapes », in : *Sensor fusion IV : control paradigms and data structures*, t. 1611, International Society for Optics et Photonics, p. 586-606.

Birk, Andreas et al. (2006), « Merging occupancy grid maps from multiple robots », in : *Proceedings of the IEEE* 94.*7*, p. 1384-1397.

Blanco, Jose-Luis (2010), « A tutorial on se (3) transformation parameterizations and on-manifold optimization », in : *University of Malaga, Tech. Rep* 3.

Blanco, Jose-Luis et al. (2009), « A collection of outdoor robotic datasets with centimeter-accuracy ground truth », in : *Autonomous Robots* 27.*4*, p. 327.

Blanco-Claraco, José-Luis et al. (2014), « The Málaga urban dataset : High-rate stereo and LiDAR in a realistic urban scenario », in : *The International Journal of Robotics Research* 33.*2*, p. 207-214.

Bloesch, Michael et al. (2015), « Robust visual inertial odometry using a direct EKF-based approach », in : *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, p. 298-304.

Bosse, Michael et al. (2012), « Zebedee : Design of a spring-mounted 3-d range sensor with application to mobile mapping », in : *IEEE Transactions on Robotics* 28.*5*, p. 1104-1119.

Brand, Christoph et al. (2014), « Stereo-vision based obstacle mapping for indoor/outdoor SLAM », in : *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 1846-1853.

— (2015), « Submap matching for stereo-vision based indoor/outdoor SLAM », in : *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 5670-5677.

Bresson, Guillaume et al. (2017), « Simultaneous localization and mapping : A survey of current trends in autonomous driving », in : *IEEE Transactions on Intelligent Vehicles* 2.*3*, p. 194-220.

Brossard, Martin et al. (2019), « A New Approach to 3D ICP Covariance Estimation for Mobile Robotics », in : *arXiv preprint arXiv :1909.05722*, p. 2502-2509.

Buoncompagni, Simone et al. (2015), « Saliency-based keypoint selection for fast object detection and matching », in : *Pattern Recognition Letters* 62, p. 32-40.

Burri, Michael et al. (2016), « The EuRoC micro aerial vehicle datasets », in : *The International Journal of Robotics Research* 35.*10*, p. 1157-1163.

Byröd, Martin et al. (2010), « Conjugate gradient bundle adjustment », in : *European Conference on Computer Vision*, Springer, p. 114-127.

Bürki, M. et al. (2016), « Appearance-based landmark selection for efficient long-term visual localization », in : *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 4137-4143.

Cadena, Cesar et al. (2016), « Past, present, and future of simultaneous localization and mapping : Toward the robust-perception age », in : *IEEE Transactions on robotics* 32.*6*, p. 1309-1332.

Calonder, Michael (2006), *EKF SLAM vs. FastSLAM–A comparison*, rapp. tech.

Calonder, Michael et al. (2010), « Brief : Binary robust independent elementary features », in : *European conference on computer vision*, Springer, p. 778-792.

Cao, Song et al. (2014), « Minimal scene descriptions from structure from motion models », in : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 461-468.

Carlevaris-Bianco, Nicholas et al. (2014), « Generic node removal for factor-graph SLAM », in : *IEEE Transactions on Robotics* 30.*6*, p. 1371-1385.

Carlone, Luca et al. (2014), « From angular manifolds to the integer lattice : Guaranteed orientation estimation with application to pose graph optimization », in : *IEEE Transactions on Robotics* 30.*2*, p. 475-492.

Carr, Jonathan C et al. (2001), « Reconstruction and representation of 3D objects with radial basis functions », in : *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, p. 67-76.

Carrillo-Arce, Luis C et al. (2013), « Decentralized multi-robot cooperative localization using covariance intersection », in : *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 1412-1417.

Cartwright, Julyan HE et al. (1992), « The dynamics of Runge–Kutta methods », in : *International Journal of Bifurcation and Chaos* 2.*03*, p. 427-449.

Caruso, David et al. (2017), « Robust indoor/outdoor navigation through magneto-visual-inertial optimization-based estimation », in : *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 4402-4409.

Censi, Andrea (2007), « An accurate closed-form estimate of ICP's covariance », in : *Proceedings 2007 IEEE international conference on robotics and automation*, IEEE, p. 3167-3172.

Ceriani, Simone et al. (2009), « Rawseeds ground truth collection systems for indoor self-localization and mapping », in : *Autonomous Robots* 27.*4*, p. 353.

Chebrolu, Nived et al. (2015), « Collaborative visual slam framework for a multi-robot system », in : *PPNIV* 2, p. 4.

Cheein, Fernando Auat et al. (2009), « Feature selection criteria for real time EKF-SLAM algorithm », in : *International Journal of Advanced Robotic Systems* 6.*3*, p. 21.

Chen, Yang et al. (1992), « Object modeling by registration of multiple range images. », in : *Image Vis. Comput.* 10.*3*, p. 145-155.

Choudhary, Siddharth et al. (2015), « Information-based reduced landmark SLAM », in : *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 4620-4627.

Choudhary, Siddharth et al. (2017), « Distributed mapping with privacy and communication constraints : Lightweight algorithms and object-based models », in : *The International Journal of Robotics Research* 36.*12*, p. 1286-1311.

Chow, C et al. (1968), « Approximating discrete probability distributions with dependence trees », in : *IEEE transactions on Information Theory* 14.*3*, p. 462-467.

Cieslewski, Titus et al. (2015), « Map api-scalable decentralized map building for robots », in : *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 6241-6247.

Cieslewski, Titus et al. (2016), « Point cloud descriptors for place recognition using sparse visual information », in : *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 4830-4836.

Cieslewski, Titus et al. (2017), « Efficient decentralized visual place recognition from full-image descriptors », in : *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, IEEE, p. 78-82.

Cieslewski, Titus et al. (2018), « Data-efficient decentralized visual SLAM », in : *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 2466-2473.

Civera, Javier et al. (2008), « Inverse depth parametrization for monocular SLAM », in : *IEEE transactions on robotics* 24.*5*, p. 932-945.

Contreras, Luis et al. (2017), « O-poco : Online point cloud compression mapping for visual odometry and slam », in : *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 4509-4514.

Cover, Thomas M et al. (2012), « Elements of information theory », in :

Cummins, Mark et al. (2008), « FAB-MAP : Probabilistic localization and mapping in the space of appearance », in : *The International Journal of Robotics Research* 27.*6*, p. 647-665.

Cunningham, Alexander et al. (2010), « DDF-SAM : Fully distributed SLAM using constrained factor graphs », in : *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 3025-3030.

Cunningham, Alexander et al. (2013), « DDF-SAM 2.0 : Consistent distributed smoothing and mapping », in : *2013 IEEE international conference on robotics and automation*, IEEE, p. 5220-5227.

Curless, Brian et al. (1996), « A volumetric method for building complex models from range images », in : *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, p. 303-312.

Dai, Angela et al. (2017), « Bundlefusion : Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration », in : *ACM Transactions on Graphics (ToG)* 36.*4*, p. 1.

DELAUNAY, Boris et al. (1934), « Sur la sphere vide », in : *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.*793-800*, p. 1-2.

DELLAERT, Frank (2012), *Factor graphs and GTSAM : A hands-on introduction*, rapp. tech., Georgia Institute of Technology.

DELLAERT, Frank et al. (2017), « Factor graphs for robot perception », in : *Foundations and Trends® in Robotics* 6.*1-2*, p. 1-139.

DELMERICO, Jeffrey et al. (2019), « Are we ready for autonomous drone racing ? the UZH-FPV drone racing dataset », in : *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, p. 6713-6719.

DESCHAUD, Jean-Emmanuel (2018), « IMLS-SLAM : scan-to-model matching based on 3D data », in : *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 2480-2485.

DEUTSCH, Isaac et al. (2016), « A framework for multi-robot pose graph SLAM », in : *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, IEEE, p. 567-572.

DOITSIDIS, Lefteris et al. (2011), « 3d surveillance coverage using maps extracted by a monocular slam algorithm », in : *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 1661-1667.

DOSOVITSKIY, Alexey et al. (2017), « CARLA : An open urban driving simulator », in : *arXiv preprint arXiv :1711.03938*.

DUBOIS, Rodolphe et al. (2019), « Évaluation de deux algorithmes de partage de données en Cartographie et Localisation Simultanées visuelles-inertielles multi-robot et décentralisées », in : *(ORASIS)*.

DUBOIS, Rodolphe et al. (2019), « On Data Sharing Strategy for Decentralized Collaborative Visual-Inertial Simultaneous Localization And Mapping », in : *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

— (2020a), « AirMuseum : a heterogeneous multi-robot dataset for stereo-visual and inertial Simultaneous Localization And Mapping », in : *2020 IEEE International Conference on Multisensor Fusion and Integration (MFI)*.

DUBOIS, Rodolphe et al. (2020b), « Dense Decentralized Multi-robot SLAM based on locally consistent TSDF submaps », in : *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Duhautbout, Thibaud et al. (2019), « Distributed 3D TSDF Manifold Mapping for Multi-Robot Systems », in : *2019 European Conference on Mobile Robots (ECMR)*, IEEE, p. 1-8.

Durrant-Whyte, Hugh et al. (2006), « Simultaneous localization and mapping : part I », in : *IEEE robotics & automation magazine* 13.*2*, p. 99-110.

Dymczyk, Marcin et al. (2015a), « Keep it brief : Scalable creation of compressed localization maps », in : *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 2536-2542.

Dymczyk, Marcin et al. (2015b), « The gist of maps-summarizing experience for lifelong localization », in : *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 2767-2773.

Dymczyk, Marcin et al. (2016), « Erasing bad memories : Agent-side summarization for long-term mapping », in : *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 4572-4579.

Eade, Ethan (2014), « Lie groups for computer vision », in : *Cambridge Univ., Cambridge, UK, Tech. Rep.*

Eckenhoff, Kevin et al. (2016a), « Decoupled, consistent node removal and edge sparsification for graph-based SLAM », in : *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 3275-3282.

Eckenhoff, Kevin et al. (2016b), « High-accuracy preintegration for visual-inertial navigation », in : *Proc. of the International Workshop on the Algorithmic Foundations of Robotics*, p. 13-16.

Egodagamage, Ruwan et al. (2017), « Distributed monocular SLAM for indoor map building », in : *Journal of Sensors* 2017.

El-Sheimy, Naser et al. (2007), « Analysis and modeling of inertial sensors using Allan variance », in : *IEEE Transactions on instrumentation and measurement* 57.*1*, p. 140-149.

Elibol, Armagan et al. (2014), « Efficient image mosaicing for multi-robot visual underwater mapping », in : *Pattern Recognition Letters* 46, p. 20-26.

Engel, Jakob et al. (2013), « Semi-dense visual odometry for a monocular camera », in : *Proceedings of the IEEE international conference on computer vision*, p. 1449-1456.

Engel, Jakob et al. (2014), « LSD-SLAM : Large-scale direct monocular SLAM », in : *European conference on computer vision*, Springer, p. 834-849.

ENGEL, Jakob et al. (2016), « A photometrically calibrated benchmark for monocular visual odometry », in : *arXiv preprint arXiv :1607.02555*.

ESTRADA, Carlos et al. (2009), « Finding good cycle constraints for large scale multi-robot SLAM », in : *2009 IEEE International Conference on Robotics and Automation*, IEEE, p. 395-402.

FALLON, Maurice F et al. (2011), « Efficient AUV navigation fusing acoustic ranging and side-scan sonar », in : *2011 IEEE International Conference on Robotics and Automation*, IEEE, p. 2398-2405.

FALLON, Maurice F et al. (2013), « Relocating underwater features autonomously using sonar-based SLAM », in : *IEEE Journal of Oceanic Engineering* 38.*3*, p. 500-513.

FEDER, Hans Jacob S et al. (1999), « Adaptive mobile robot navigation and mapping », in : *The International Journal of Robotics Research* 18.*7*, p. 650-668.

FERRERA, Maxime et al. (2019a), « AQUALOC : An underwater dataset for visual–inertial–pressure localization », in : *The International Journal of Robotics Research* 38.*14*, p. 1549-1559.

FERRERA, Maxime et al. (2019b), « Real-time monocular visual odometry for turbid and dynamic underwater environments », in : *Sensors* 19.*3*, p. 687.

FIALA, Mark (2005), « ARTag, a fiducial marker system using digital techniques », in : *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, t. 2, IEEE, p. 590-596.

FIORAIO, Nicola et al. (2015), « Large-scale and drift-free surface reconstruction using online subvolume registration », in : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 4475-4483.

FISCHLER, Martin A et al. (1981), « Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography », in : *Communications of the ACM* 24.*6*, p. 381-395.

FLINT, Alex et al. (2007), « Thrift : Local 3d structure recognition », in : *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*, IEEE, p. 182-188.

FORSTER, Christian et al. (2013), « Collaborative monocular slam with multiple micro aerial vehicles », in : *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 3962-3970.

Forster, Christian et al. (2014), « SVO : Fast semi-direct monocular visual odometry », in : *2014 IEEE international conference on robotics and automation (ICRA)*, IEEE, p. 15-22.

Forster, Christian et al. (2016), « On-Manifold Preintegration for Real-Time Visual–Inertial Odometry », in : *IEEE Transactions on Robotics* 33.*1*, p. 1-21.

Fraundorfer, Friedrich et al. (2012), « Visual odometry : Part ii : Matching, robustness, optimization, and applications », in : *IEEE Robotics & Automation Magazine* 19.*2*, p. 78-90.

Fryer, John G et al. (1986), « Lens distortion for close-range photogrammetry », in : *Photogrammetric engineering and remote sensing* 52.*1*, p. 51-58.

Furgale, P et al. (2014), *Kalibr*.

Furrer, Fadri et al. (2016), « RotorS—A modular Gazebo MAV simulator framework », in : *Robot Operating System (ROS)*, Springer, p. 595-625.

Gaidon, Adrien et al. (2016), « Virtual worlds as proxy for multi-object tracking analysis », in : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 4340-4349.

Gallego, Guillermo et al. (2019), « Event-based vision : A survey », in : *arXiv preprint arXiv :1904.08405*.

Galvez-Lopez, Dorian et al. (2011), « Real-time loop detection with bags of binary words », in : *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 51-58.

Gálvez-López, Dorian et al. (2012), « Bags of binary words for fast place recognition in image sequences », in : *IEEE Transactions on Robotics* 28.*5*, p. 1188-1197.

Geiger, Andreas et al. (2010), « Efficient large-scale stereo matching », in : *Asian conference on computer vision*, Springer, p. 25-38.

Geiger, Andreas et al. (2013), « Vision meets robotics : The kitti dataset », in : *The International Journal of Robotics Research* 32.*11*, p. 1231-1237.

Geneva, Patrick et al. (2019), « Openvins : A research platform for visual-inertial estimation », in : *IROS 2019 Workshop on Visual-Inertial Navigation : Challenges and Applications, Macau, China*.

Giamou, Matthew et al. (2018), « Talk resource-efficiently to me : Optimal communication planning for distributed loop closure detection », in : *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 1-9.

GIUBILATO, Riccardo et al. (2020), « Relocalization with submaps : Multi-session mapping for planetary rovers equipped with stereo cameras », in : *IEEE Robotics and Automation Letters* 5.*2*, p. 580-587.

GRABE, Volker et al. (2013), « A comparison of scale estimation schemes for a quadrotor UAV based on optical flow and IMU measurements », in : *2013 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, p. 5193-5200.

GRISETTI, Giorgio et al. (2010a), « A tutorial on graph-based SLAM », in : *IEEE Intelligent Transportation Systems Magazine* 2.*4*, p. 31-43.

GRISETTI, Giorgio et al. (2010b), « Hierarchical optimization on manifolds for online 2D and 3D mapping », in : *2010 IEEE International Conference on Robotics and Automation*, IEEE, p. 273-278.

GRISETTI, Giorgio et al. (2012), « Robust optimization of factor graphs by using condensed measurements », in : *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 581-588.

GRISETTI, Giorgio et al. (2020), « Least Squares Optimization : from Theory to Practice », in : *arXiv preprint arXiv :2002.11051*.

GUIVANT, Jose et al. (2002), « Simultaneous localization and map building : Test case for outdoor applications », in : *IEEE Int. Conference on Robotics and Automation*.

GUROBI (2014), « INC. Gurobi optimizer reference manual, 2015 », in : *URL : http ://www. gurobi. com*, p. 29.

GUTMANN, J-S et al. (1999), « Incremental mapping of large cyclic environments », in : *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No. 99EX375)*, IEEE, p. 318-325.

HAHNEL, Dirk et al. (2003), « An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements », in : *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, t. 1, IEEE, p. 206-211.

HANA, Xian-Feng et al. (2018), « A comprehensive review of 3d point cloud descriptors », in : *arXiv preprint arXiv :1802.02297*.

HARRIS, Christopher G et al. (1988), « A combined corner and edge detector. », in : *Alvey vision conference*, t. 15, 50, Citeseer, p. 10-5244.

HARTLEY, Richard et al. (2003), *Multiple view geometry in computer vision*, Cambridge university press.

He, Yijia et al. (2018), « PL-VIO : Tightly-coupled monocular visual–inertial odometry using point and line features », in : *Sensors* 18.*4*, p. 1159.

Hewitt, Robert A et al. (2018), « The Katwijk beach planetary rover dataset », in : *The International Journal of Robotics Research* 37.*1*, p. 3-12.

Hochdorfer, Siegfried et al. (2009), « Landmark rating and selection according to localization coverage : Addressing the challenge of lifelong operation of SLAM in service robots », in : *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 382-387.

Holder, Martin et al. (2019), « Real-time pose graph SLAM based on radar », in : *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, p. 1145-1151.

Howard, Andrew (2004), « Multi-robot mapping using manifold representations », in : *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, t. 4, IEEE, p. 4198-4203.

— (2006), « Multi-robot simultaneous localization and mapping using particle filters », in : *The International Journal of Robotics Research* 25.*12*, p. 1243-1256.

Hsiung, Jerry et al. (2018), « Information sparsification in visual-inertial odometry », in : *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 1146-1153.

Huang, Albert S et al. (2010a), « A high-rate, heterogeneous data set from the darpa urban challenge », in : *The International Journal of Robotics Research* 29.*13*, p. 1595-1601.

Huang, Guoquan et al. (2013), « Consistent sparsification for graph optimization », in : *2013 European Conference on Mobile Robots*, IEEE, p. 150-157.

Huang, Guoquan P et al. (2010b), « Observability-based rules for designing consistent EKF SLAM estimators », in : *The International Journal of Robotics Research* 29.*5*, p. 502-528.

Huber, Peter J (1992), « Robust estimation of a location parameter », in : *Breakthroughs in statistics*, Springer, p. 492-518.

Ila, Viorela et al. (2009), « Information-based compact Pose SLAM », in : *IEEE Transactions on Robotics* 26.*1*, p. 78-93.

Indelman, Vadim (2015), « Distributed perception and estimation : a short survey », in : *Principles of Multi-Robot Systems, workshop in conjunction with Robotics Science and Systems*.

INDELMAN, Vadim et al. (2016), « Incremental distributed inference from arbitrary poses and unknown data association : Using collaborating robots to establish a common reference », in : *IEEE Control Systems Magazine* 36.*2*, p. 41-74.

JULIER, Simon et al. (2017), « General decentralized data fusion with covariance intersection », in : *Handbook of multisensor data fusion*, CRC Press, p. 339-364.

KAESS, Michael et al. (2008), « iSAM : Incremental smoothing and mapping », in : *IEEE Transactions on Robotics* 24.*6*, p. 1365-1378.

KÄHLER, Olaf et al. (2016), « Real-time large-scale dense 3D reconstruction with loop closure », in : *European Conference on Computer Vision*, Springer, p. 500-516.

KALMAN, Rudolph Emil (1960), « A new approach to linear filtering and prediction problems », in :

KANNALA, Juho et al. (2006), « A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses », in : *IEEE transactions on pattern analysis and machine intelligence* 28.*8*, p. 1335-1340.

KARP, Richard M (1972), « Reducibility among combinatorial problems », in : *Complexity of computer computations*, Springer, p. 85-103.

KARRER, Marco et al. (2018), « CVI-SLAM—collaborative visual-inertial SLAM », in : *IEEE Robotics and Automation Letters* 3.*4*, p. 2762-2769.

KÄSLIN, Roman et al. (2016), « Collaborative localization of aerial and ground robots through elevation maps », in : *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, IEEE, p. 284-290.

KEIVAN, Nima et al. (2016), « Asynchronous adaptive conditioning for visual-inertial SLAM », in : *Experimental Robotics*, Springer, p. 309-321.

KERL, Christian et al. (2013), « Robust odometry estimation for RGB-D cameras », in : *2013 IEEE international conference on robotics and automation*, IEEE, p. 3748-3754.

KHOSOUSSI, Kasra et al. (2014), « Novel insights into the impact of graph structure on SLAM », in : *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 2707-2714.

KIM, Ayoung et al. (2013), « Real-time visual SLAM for autonomous underwater hull inspection using visual saliency », in : *IEEE Transactions on Robotics* 29.*3*, p. 719-733.

KIM, Been et al. (2010), « Multiple relative pose graphs for robust cooperative mapping », in : *2010 IEEE International Conference on Robotics and Automation*, IEEE, p. 3185-3192.

KLEIN, Georg et al. (2007), « Parallel tracking and mapping for small AR workspaces », in : *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, IEEE, p. 225-234.

KLINGENSMITH, Matthew et al. (2015), « Chisel : Real Time Large Scale 3D Reconstruction Onboard a Mobile Device using Spatially Hashed Signed Distance Fields. », in : *Robotics : science and systems*, t. 4, p. 1.

KOENIG, Nathan et al. (2004), « Design and use paradigms for gazebo, an open-source multi-robot simulator », in : *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, t. 3, IEEE, p. 2149-2154.

KRETZSCHMAR, Henrik et al. (2011), « Efficient information-theoretic graph pruning for graph-based SLAM with laser range finders », in : *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 865-871.

KSCHISCHANG, Frank R et al. (2001), « Factor graphs and the sum-product algorithm », in : *IEEE Transactions on information theory* 47.*2*, p. 498-519.

KULLBACK, Solomon et al. (1951), « On information and sufficiency », in : *The annals of mathematical statistics* 22.*1*, p. 79-86.

KÜMMERLE, Rainer et al. (2011), « g 2 o : A general framework for graph optimization », in : *2011 IEEE International Conference on Robotics and Automation*, IEEE, p. 3607-3613.

LACROIX, Simon et al. (2002), « Autonomous rover navigation on unknown terrains : Functions and integration », in : *The International Journal of Robotics Research* 21.*10-11*, p. 917-942.

LAJOIE, Pierre-Yves et al. (2020), « DOOR-SLAM : Distributed, online, and outlier resilient SLAM for robotic teams », in : *IEEE Robotics and Automation Letters* 5.*2*, p. 1656-1663.

LAMARRE, Olivier et al. (2020), « The Canadian Planetary Emulation Terrain Energy-Aware Rover Navigation Dataset », in : *The International Journal of Robotics Research*, p. 0278364920908922.

LANCZOS, Cornelius (1950), *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, United States Governm. Press Office Los Angeles, CA.

LATIF, Yasir et al. (2013), « Robust loop closing over time for pose graph SLAM », in : *The International Journal of Robotics Research* 32.*14*, p. 1611-1626.

Lau, Boris et al. (2013), « Efficient grid-based spatial representations for robot navigation in dynamic environments », in : *Robotics and Autonomous Systems* 61.*10*, p. 1116-1130.

Lazaro, Maria Teresa et al. (2013), « Multi-robot SLAM using condensed measurements », in : *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 1069-1076.

Leclet-Groux, Dominique et al. (2018), « Efficient restitution of heritage buildings : a new way to exploit 3D point clouds slicing, An example with the Amiens Gothic Cathedral », in :

Lerner, Ronen et al. (2007), « Landmark selection for task-oriented navigation », in : *IEEE transactions on robotics* 23.*3*, p. 494-505.

Leung, Keith et al. (2017), « Chilean underground mine dataset », in : *The International Journal of Robotics Research* 36.*1*, p. 16-23.

Leung, Keith YK et al. (2011), « The UTIAS multi-robot cooperative localization and mapping dataset », in : *The International Journal of Robotics Research* 30.*8*, p. 969-974.

Leutenegger, Stefan et al. (2011), « BRISK : Binary robust invariant scalable keypoints », in : *2011 International conference on computer vision*, Ieee, p. 2548-2555.

Leutenegger, Stefan et al. (2015), « Keyframe-based visual–inertial odometry using nonlinear optimization », in : *The International Journal of Robotics Research* 34.*3*, p. 314-334.

Li, Fu et al. (2017), « Corb-slam : a collaborative visual slam system for multiple robots », in : *International Conference on Collaborative Computing : Networking, Applications and Worksharing*, Springer, p. 480-490.

Liu, Yanli et al. (2013), « A fast map merging algorithm in the field of multirobot SLAM », in : *The Scientific World Journal* 2013.

Longuet-Higgins, H Christopher (1981), « A computer algorithm for reconstructing a scene from two projections », in : *Nature* 293.*5828*, p. 133-135.

Lorensen, William E et al. (1987), « Marching cubes : A high resolution 3D surface construction algorithm », in : *ACM siggraph computer graphics* 21.*4*, p. 163-169.

Lowe, David G (2004), « Distinctive image features from scale-invariant keypoints », in : *International journal of computer vision* 60.*2*, p. 91-110.

Lowry, Stephanie et al. (2015), « Visual place recognition : A survey », in : *IEEE Transactions on Robotics* 32.*1*, p. 1-19.

Lu, Feng et al. (1997), « Globally consistent range scan alignment for environment mapping », in : *Autonomous robots* 4.*4*, p. 333-349.

Luft, Lukas et al. (2016), « Recursive Decentralized Collaborative Localization for Sparsely Communicating Robots. », in : *Robotics : Science and Systems*, New York, NY, USA.

Lupton, Todd et al. (2011), « Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions », in : *IEEE Transactions on Robotics* 28.*1*, p. 61-76.

Lynen, Simon et al. (2013), « A robust and modular multi-sensor fusion approach applied to mav navigation », in : *2013 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, p. 3923-3929.

Lynen, Simon et al. (2015), « Get out of my lab : Large-scale, real-time visual-inertial localization. », in : *Robotics : Science and Systems*, t. 1.

Maddern, Will et al. (2017), « 1 year, 1000 km : The Oxford RobotCar dataset », in : *The International Journal of Robotics Research* 36.*1*, p. 3-15.

Madhavan, Raj et al. (2004), « Distributed cooperative outdoor multirobot localization and mapping », in : *Autonomous Robots* 17.*1*, p. 23-39.

Maimone, Mark et al. (2007), « Two years of visual odometry on the mars exploration rovers », in : *Journal of Field Robotics* 24.*3*, p. 169-186.

Majdik, András L et al. (2017), « The Zurich urban micro aerial vehicle dataset », in : *The International Journal of Robotics Research* 36.*3*, p. 269-273.

Mangelson, Joshua G et al. (2018), « Pairwise consistent measurement set maximization for robust multi-robot map merging », in : *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 2916-2923.

Mazuran, Mladen et al. (2014), « Nonlinear Graph Sparsification for SLAM. », in : *Robotics : Science and Systems*, p. 1-8.

Meagher, Donald (1982), « Geometric modeling using octree encoding », in : *Computer graphics and image processing* 19.*2*, p. 129-147.

Mei, Christopher et al. (2010), « Closing loops without places », in : *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 3738-3744.

Michael, Nathan et al. (2012), « Collaborative mapping of an earthquake-damaged building via ground and aerial robots », in : *Journal of Field Robotics* 29.*5*, p. 832-841.

Millane, Alexander et al. (2018), « C-blox : A scalable and consistent TSDF-based dense mapping approach », in : *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 995-1002.

Mirzasoleiman, Baharan et al. (2015), « Lazier than Lazy Greedy », in : *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, Austin, Texas : AAAI Press, 1812–1818, isbn : 0262511290.

Montemerlo, Michael et al. (2003), « FastSLAM 2.0 : An improved particle filtering algorithm for simultaneous localization and mapping that provably converges », in : *IJCAI*, p. 1151-1156.

Montijano, Eduardo et al. (2013), « Distributed data association in robotic networks with cameras and limited communications », in : *IEEE Transactions on Robotics* 29.*6*, p. 1408-1423.

Moon, Todd K (1996), « The expectation-maximization algorithm », in : *IEEE Signal processing magazine* 13.*6*, p. 47-60.

Moosmann, Frank et al. (2011), « Velodyne slam », in : *2011 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, p. 393-398.

Moravec, H (1996), « Robot spatial perceptionby stereoscopic vision and 3d evidence grids », in : *Perception*.

Morel, Jean-Michel et al. (2009), « ASIFT : A new framework for fully affine invariant image comparison », in : *SIAM journal on imaging sciences* 2.*2*, p. 438-469.

Morrison, John G et al. (2016), « Moarslam : Multiple operator augmented rslam », in : *Distributed autonomous robotic systems*, Springer, p. 119-132.

Mourikis, Anastasios I et al. (2007), « A multi-state constraint Kalman filter for vision-aided inertial navigation », in : *Proceedings 2007 IEEE International Conference on Robotics and Automation*, IEEE, p. 3565-3572.

Mu, Beipeng et al. (2015), « Two-stage focused inference for resource-constrained collision-free navigation », in :

Mueggler, Elias et al. (2018), « Continuous-time visual-inertial odometry for event cameras », in : *IEEE Transactions on Robotics* 34.*6*, p. 1425-1440.

Mühlfellner, Peter et al. (2016), « Summary maps for lifelong visual localization », in : *Journal of Field Robotics* 33.*5*, p. 561-590.

Mur-Artal, Raul et al. (2015), « ORB-SLAM : a versatile and accurate monocular SLAM system », in : *IEEE transactions on robotics* 31.*5*, p. 1147-1163.

Mur-Artal, Raul et al. (2017a), « Orb-slam2 : An open-source slam system for monocular, stereo, and rgb-d cameras », in : *IEEE Transactions on Robotics* 33.*5*, p. 1255-1262.

Mur-Artal, Raúl et al. (2017b), « Visual-inertial monocular SLAM with map reuse », in : *IEEE Robotics and Automation Letters* 2.*2*, p. 796-803.

Nagatani, Keiji et al. (2011), « Multirobot exploration for search and rescue missions : A report on map building in RoboCupRescue 2009 », in : *Journal of Field Robotics* 28.*3*, p. 373-387.

Nerurkar, Esha D et al. (2009), « Distributed maximum a posteriori estimation for multi-robot cooperative localization », in : *2009 IEEE International Conference on Robotics and Automation*, IEEE, p. 1402-1409.

Nerurkar, Esha D et al. (2014), « C-KLAM : Constrained keyframe-based localization and mapping », in : *2014 IEEE international conference on robotics and automation (ICRA)*, IEEE, p. 3638-3643.

Nettleton, Eric et al. (2003), « Decentralised SLAM with low-bandwidth communication for teams of vehicles », in : *Field and Service Robotics*, Springer, p. 179-188.

Newcombe, Richard A et al. (2011a), « DTAM : Dense tracking and mapping in real-time », in : *2011 international conference on computer vision*, IEEE, p. 2320-2327.

Newcombe, Richard A et al. (2011b), « KinectFusion : Real-time dense surface mapping and tracking », in : *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, IEEE, p. 127-136.

Niessner, Matthias et al. (2013), « Real-time 3D reconstruction at scale using voxel hashing », in : *ACM Transactions on Graphics (ToG)* 32.*6*, p. 1-11.

Nistér, David (2004), « An efficient solution to the five-point relative pose problem », in : *IEEE transactions on pattern analysis and machine intelligence* 26.*6*, p. 756-770.

Nistér, David et al. (2010), *Scalable object recognition using hierarchical quantization with a vocabulary tree*, US Patent 7,725,484.

Oleynikova, Helen et al. (2015), « Real-time visual-inertial localization for aerial and ground robots », in : *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 3079-3085.

Oleynikova, Helen et al. (2016), « Signed distance fields : A natural representation for both mapping and planning », in : *RSS 2016 Workshop : Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*, University of Michigan.

Oleynikova, Helen et al. (2017), « Voxblox : Incremental 3d euclidean signed distance fields for on-board mav planning », in : *2017 Ieee/rsj International Conference on Intelligent Robots and Systems (iros)*, IEEE, p. 1366-1373.

Olson, Edwin (2011), « AprilTag : A robust and flexible visual fiducial system », in : *2011 IEEE International Conference on Robotics and Automation*, IEEE, p. 3400-3407.

Olson, Edwin et al. (2013a), « Exploration and mapping with autonomous robot teams », in : *Communications of the ACM* 56.*3*, p. 62-70.

Olson, Edwin et al. (2013b), « Inference on networks of mixtures for robust robot mapping », in : *The International Journal of Robotics Research* 32.*7*, p. 826-840.

Pandey, Gaurav et al. (2011), « Ford campus vision and lidar data set », in : *The International Journal of Robotics Research* 30.*13*, p. 1543-1552.

Paull, Liam et al. (2015), « Communication-constrained multi-AUV cooperative SLAM », in : *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, p. 509-516.

Pfrommer, Bernd et al. (2017), « Penncosyvio : A challenging visual inertial odometry benchmark », in : *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 3847-3854.

Pfrommer, Bernd et al. (2019), « TagSLAM : Robust SLAM with fiducial markers », in : *arXiv preprint arXiv :1910.00679*.

Piao, Jin-Chun et al. (2017), « Adaptive monocular visual–inertial SLAM for real-time augmented reality applications in mobile devices », in : *Sensors* 17.*11*, p. 2567.

Pizzoli, Matia et al. (2014), « REMODE : Probabilistic, monocular dense reconstruction in real time », in : *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 2609-2616.

Qin, Tong et al. (2018a), *Collaborative Localization for Multiple Monocular Vision-Based MAVs*, rapp. tech., Aerial Robotics Group, The Hong-KOng University of Science et Technology, Clear Water Bay, Kowloon, Hong Kong.

Qin, Tong et al. (2018b), « Vins-mono : A robust and versatile monocular visual-inertial state estimator », in : *IEEE Transactions on Robotics* 34.*4*, p. 1004-1020.

Qin, Tong et al. (2019), « A general optimization-based framework for local odometry estimation with multiple sensors », in : *arXiv preprint arXiv :1901.03638*.

Queralta, Jorge Peña et al. (2020), « Collaborative Multi-Robot Systems for Search and Rescue : Coordination and Perception », in : *arXiv preprint arXiv :2008.12610*.

Quigley, Morgan et al. (2009), « ROS : an open-source Robot Operating System », in : *ICRA workshop on open source software*, t. 3, 3.2, Kobe, Japan, p. 5.

Quraishi, Anwar et al. (2016), « Robustness to connectivity loss for collaborative mapping », in : *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 4580-4585.

Reijgwart, Victor et al. (2019), « Voxgraph : Globally Consistent, Volumetric Mapping Using Signed Distance Function Submaps », in : *IEEE Robotics and Automation Letters* 5.*1*, p. 227-234.

Riazuelo, Luis et al. (2014), « C2tam : A cloud framework for cooperative tracking and mapping », in : *Robotics and Autonomous Systems* 62.*4*, p. 401-413.

Rogers III, John G et al. (2017), « Distributed subterranean exploration and mapping with teams of UAVs », in : *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR VIII*, t. 10190, International Society for Optics et Photonics, p. 1019017.

Rohou, Simon et al. (2019), *Reliable robot localization : a constraint-programming approach over dynamical systems*, John Wiley & Sons.

Rosinol, Antoni et al. (2019), « Kimera : an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping », in : *arXiv preprint arXiv :1910.02490*.

Rosten, Edward et al. (2006), « Machine learning for high-speed corner detection », in : *European conference on computer vision*, Springer, p. 430-443.

Roumeliotis, Stergios I et al. (2000), « Distributed multi-robot localization », in : *Distributed Autonomous Robotic Systems 4*, Springer, p. 179-188.

Rublee, Ethan et al. (2011), « ORB : An efficient alternative to SIFT or SURF », in : *2011 International conference on computer vision*, Ieee, p. 2564-2571.

Ruetz, Fabio et al. (2019), « OVPC Mesh : 3D Free-space Representation for Local Ground Vehicle Navigation », in : *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, p. 8648-8654.

Rusu, Radu Bogdan et al. (2011), « 3d is here : Point cloud library (pcl) », in : *2011 IEEE international conference on robotics and automation*, IEEE, p. 1-4.

Saeedi, Sajad et al. (2016), « Multiple-robot simultaneous localization and mapping : A review », in : *Journal of Field Robotics* 33.*1*, p. 3-46.

Salas-Moreno, Renato F et al. (2013), « Slam++ : Simultaneous localisation and mapping at the level of objects », in : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 1352-1359.

Sanfourche, Martial et al. (2013), « evo : A realtime embedded stereo odometry for mav applications », in : *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 2107-2114.

Sartipi, Kourosh et al. (2019), « Decentralized visual-inertial localization and mapping on mobile devices for augmented reality », in : *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 2145-2152.

Sattler, Torsten et al. (2012), « Improving image-based localization by active correspondence search », in : *European conference on computer vision*, Springer, p. 752-765.

Sayre-McCord, Thomas et al. (2018), « Visual-inertial navigation algorithm development using photorealistic camera simulation in the loop », in : *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 2566-2573.

Scaramuzza, Davide et al. (2011), « Visual odometry [tutorial] », in : *IEEE robotics & automation magazine* 18.*4*, p. 80-92.

Schmuck, Patrik et al. (2017), « Multi-uav collaborative monocular slam », in : *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 3863-3870.

— (2019a), « CCM-SLAM : Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams », in : *Journal of Field Robotics* 36.*4*, p. 763-781.

— (2019b), « On the Redundancy Detection in Keyframe-based SLAM », in : *2019 International Conference on 3D Vision (3DV)*, IEEE, p. 594-603.

Schneider, Thomas et al. (2018), « maplab : An open framework for research in visual-inertial mapping and localization », in : *IEEE Robotics and Automation Letters* 3.*3*, p. 1418-1425.

Schonberger, Johannes L et al. (2016), « Structure-from-motion revisited », in : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 4104-4113.

Schubert, David et al. (2018), « The TUM VI benchmark for evaluating visual-inertial odometry », in : *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 1680-1687.

Schuster, Martin J et al. (2015), « Multi-robot 6D graph SLAM connecting decoupled local reference filters », in : *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 5093-5100.

Schuster, Martin J et al. (2019), « Distributed stereo vision-based 6D localization and mapping for multi-robot teams », in : *Journal of Field Robotics* 36.*2*, p. 305-332.

Schuster, Martin Johannes (2019), « Collaborative Localization and Mapping for Autonomous Planetary Exploration », thèse de doct., PhD thesis, University of Bremen & German Aerospace Center (DLR).

Schweighofer, Gerald et al. (2006), « Robust pose estimation from a planar target », in : t. 28, 12, IEEE, p. 2024-2030.

Shah, Shital et al. (2018), « Airsim : High-fidelity visual and physical simulation for autonomous vehicles », in : *Field and service robotics*, Springer, p. 621-635.

Shen, Shaojie et al. (2013), « State estimation for indoor and outdoor operation with a micro-aerial vehicle », in : *Experimental Robotics*, Springer, p. 273-288.

Shi, Jianbo et al. (1994), « Good features to track », in : *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, IEEE, p. 593-600.

Shim, Jae Hong et al. (2017), « A Visual Localization Technique for Unmanned Ground and Aerial Robots », in : *2017 First IEEE International Conference on Robotic Computing (IRC)*, IEEE, p. 399-403.

Shoemake, Ken (1985), « Animating rotation with quaternion curves », in : *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, p. 245-254.

Sipiran, Ivan et al. (2011), « Harris 3D : a robust extension of the Harris operator for interest point detection on 3D meshes », in : *The Visual Computer* 27.*11*, p. 963.

Sivic, Josef et al. (2003), « Video Google : A text retrieval approach to object matching in videos », in : *null*, IEEE, p. 1470.

Smith, Mike et al. (2009), « The new college vision and laser data set », in : *The International Journal of Robotics Research* 28.*5*, p. 595-599.

Sola, Joan (2012), « Quaternion kinematics for the error-state KF », in : *Laboratoire dAnalyse et dArchitecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Toulouse, France, Tech. Rep.*

Sola, Joan et al. (2018), « A micro Lie theory for state estimation in robotics », in : *arXiv preprint arXiv :1812.01537*.

Soo Park, Hyun et al. (2013), « 3d point cloud reduction using mixed-integer quadratic programming », in : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, p. 229-236.

Stachniss, Cyrill et al. (2016), « Simultaneous localization and mapping », in : *Springer Handbook of Robotics*, Springer, p. 1153-1176.

Strasdat, Hauke et al. (2009), « Which landmark is useful ? Learning selection policies for navigation in unknown environments », in : *2009 IEEE International Conference on Robotics and Automation*, IEEE, p. 1410-1415.

Sturm, Jürgen et al. (2012), « A benchmark for the evaluation of RGB-D SLAM systems », in : *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 573-580.

Suger, Benjamin et al. (2014), « An approach to solving large-scale slam problems with a small memory footprint », in : *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, p. 3632-3637.

Sünderhauf, Niko et al. (2013), « Switchable constraints vs. max-mixture models vs. rrr-a comparison of three approaches to robust pose graph slam », in : *2013 IEEE International Conference on Robotics and Automation*, IEEE, p. 5198-5203.

Tardioli, Danilo et al. (2015), « Visual data association in narrow-bandwidth networks », in : *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 2572-2577.

Teixeira, Lucas et al. (2018), « VI-RPE : Visual-inertial relative pose estimation for aerial vehicles », in : *IEEE Robotics and Automation Letters* 3.*4*, p. 2770-2777.

Teixeira, Pedro V et al. (2016), « Underwater inspection using sonar-based volumetric submaps », in : *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 4288-4295.

Thrun, Sebastian et al. (2004), « Simultaneous localization and mapping with sparse extended information filters », in : *The international journal of robotics research* 23.*7-8*, p. 693-716.

Thrun, Sebastian et al. (2005), « Multi-robot SLAM with sparse extended information filers », in : *Robotics Research. The Eleventh International Symposium*, Springer, p. 254-266.

Tian, Yulun et al. (2018), « Near-optimal budgeted data exchange for distributed loop closure detection », in : *arXiv preprint arXiv :1806.00188*.

Tomasi, Carlo et al. (1991), « Detection and tracking of point features », in :

Tombari, Federico et al. (2011), « A combined texture-shape descriptor for enhanced 3D feature matching », in : *2011 18th IEEE international conference on image processing*, IEEE, p. 809-812.

Triggs, Bill et al. (1999), « Bundle adjustment—a modern synthesis », in : *International workshop on vision algorithms*, Springer, p. 298-372.

Uhlmann, Jeffrey K (1995), « Dynamic map building and localization : New theoretical foundations », thèse de doct., University of Oxford Oxford.

Umeyama, Shinji (1991), « Least-squares estimation of transformation parameters between two point patterns », in : *IEEE Transactions on Pattern Analysis & Machine Intelligence 4*, p. 376-380.

Vallvé, Joan et al. (2018), « Graph SLAM sparsification with populated topologies using factor descent optimization », in : *IEEE Robotics and Automation Letters* 3.*2*, p. 1322-1329.

Vayugundla, Mallikarjuna et al. (2018), « Datasets of Long Range Navigation Experiments in a Moon Analogue Environment on Mount Etna », in : *ISR 2018 ; 50th International Symposium on Robotics*, VDE, p. 1-7.

Vespa, Emanuele et al. (2018), « Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping », in : *IEEE Robotics and Automation Letters* 3.*2*, p. 1144-1151.

Vidal-Calleja, Teresa A et al. (2011), « Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain », in : *Robotics and Autonomous Systems* 59.*9*, p. 654-674.

Wagner, René et al. (2014), « Graph SLAM with signed distance function maps on a humanoid robot », in : *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 2691-2698.

Walter, Viktor et al. (2018), « Fast mutual relative localization of uavs using ultraviolet led markers », in : *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, p. 1217-1226.

Wang, Wenshan et al. (2020), « TartanAir : A Dataset to Push the Limits of Visual SLAM », in : *arXiv preprint arXiv :2003.14338*.

Wang, Yue et al. (2013), « Kullback-leibler divergence based graph pruning in robotic feature mapping », in : *2013 European Conference on Mobile Robots*, IEEE, p. 32-37.

Weiss, Stephan et al. (2012), « Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments », in : *2012 IEEE international conference on robotics and automation*, IEEE, p. 957-964.

Whelan, Thomas et al. (2012), « Kintinuous : Spatially extended kinectfusion », in :

Whelan, Thomas et al. (2015), « ElasticFusion : Dense SLAM without a pose graph », in : Robotics : Science et Systems.

Whelan, Thomas et al. (2016), « ElasticFusion : Real-time dense SLAM and light source estimation », in : *The International Journal of Robotics Research* 35.*14*, p. 1697-1716.

Wurm, Kai M et al. (2010), « OctoMap : A probabilistic, flexible, and compact 3D map representation for robotic systems », in : *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, t. 2.

Yamada, Taichi et al. (2020), « Development of a Dataset to Evaluate SLAM for Fukushima Daiichi Nuclear Power Plant Decommissioning », in : *2020 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, p. 7-11.

Zhang, Hui et al. (2018a), « Distributed and collaborative monocular simultaneous localization and mapping for multi-robot systems in large-scale environments », in : *International Journal of Advanced Robotic Systems* 15.*3*, p. 1729881418780178.

Zhang, Ji et al. (2014), « LOAM : Lidar Odometry and Mapping in Real-time. », in : *Robotics : Science and Systems*, t. 2, 9.

Zhang, Sen et al. (2005), « Entropy based feature selection scheme for real time simultaneous localization and map building », in : *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, p. 1175-1180.

Zhang, Zichao et al. (2018b), « A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry », in : *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 7244-7251.

Zhao, Yipu et al. (2018), « Good feature selection for least squares pose optimization in VO/VSLAM », in : *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, p. 1183-1189.

Zhong, Yu (2009), « Intrinsic shape signatures : A shape descriptor for 3d object recognition », in : *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, IEEE, p. 689-696.

Zhou, Xun S et al. (2006), « Multi-robot SLAM with unknown initial correspondence : The robot rendezvous case », in : *2006 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, p. 1785-1792.

Zhu, Alex Zihao et al. (2018), « The multivehicle stereo event camera dataset : An event camera dataset for 3D perception », in : *IEEE Robotics and Automation Letters* 3.*3*, p. 2032-2039.

Zlot, Robert et al. (2014), « Efficient large-scale 3D mobile mapping and surface reconstruction of an underground mine », in : *Field and service robotics*, Springer, p. 479-493.

Zou, Danping et al. (2012), « Coslam : Collaborative visual slam in dynamic environments », in : *IEEE transactions on pattern analysis and machine intelligence* 35.*2*, p. 354-366.

Zuiderveld, Karel (1994), « Contrast limited adaptive histogram equalization », in : *Graphics gems IV*, Academic Press Professional, Inc., p. 474-485.

CENTRALE NANTES

Titre : Méthodes de partage d'informations visuelles et inertielles pour la localisation et la cartographie simultanées décentralisées multi-robots

Mot clés : Robotique, SLAM multi-robot, estimation visio-inertielle

Résumé : En robotique mobile, les méthodes de cartographie et de localisation simultanées (SLAM) constituent une brique algorithme essentielle afin de percevoir l'environnement et y naviguer de façon autonome. En contexte visuel, les méthodes de SLAM mono-robot ont aujourd'hui atteint un haut degré de maturité, ce qui a permis l'essor de méthodes collaboratives. Néanmoins, les problématiques d'autonomie des agents couplées aux contraintes d'information, de réseau et de ressources interrogent sur la nature des données à transmettre entre les robots. L'objectif de cette thèse est de concevoir des méthodes de partage d'informations visuelles et inertielles qui favorisent l'autonomie des robots et leur permettent d'affiner leur navigation dès lors qu'ils visitent des zones communes. Dans ce but, nous investiguons différents paradigmes d'échanges pour des architectures décentralisées de SLAM visio-inertiel et stéréo-visuel. Tout d'abord, nous proposons trois façons de synthétiser des données visio-inertielles, et développons une architecture de SLAM collaboratif décentralisée chargée d'en organiser le calcul, l'échange et l'intégration. Ces méthodes exploitent respectivement l'échange de sous-cartes visio-inertielles rigides, de paquets condensés par marginalisation et éparsification consistante, et de paquets élagués via la sélection de facteurs visio-inertiels bruts. Nous les évaluons sur des jeux de données standards, ainsi que sur notre jeu de données AirMuseum, spécifiquement conçu à cette fin. Enfin, nous appliquons l'architecture développée pour la cartographie dense en étendant une méthode de cartographie collaborative reposant sur l'échange, le recalage et la fusion de sous-cartes localement consistantes associées à des représentations de type TSDF.

Title: Sharing visual-inertial information for collaborative decentralized simultaneous localization and mapping

Keywords: Robotics, collaborative SLAM, visual-inertial Estimation

Abstract: In mobile robotics, simultaneous mapping and localization (SLAM) methods are an essential algorithmic brick in order to perceive the environment and autonomously navigate within it. In a visual context, single-robot SLAM methods have now reached high maturity, which has allowed the development of collaborative SLAM methods. However, enhancing the agents' autonomy while facing informational, network and resource constraints raises the question about the nature of the data to be transmitted between the robots. The objective of this thesis is to design methods for sharing visual and inertial information for collaborative SLAM, which enforce the autonomy of agents and allow them to refine their navigation when they visit common areas. To this end, we have investigated multiple exchange paradigms for decentralized visual-inertial and stereo-visual SLAM architectures. First, we proposed three ways of summarizing visual-inertial data, from which ones we built collaborative SLAM methods. Those are respectively based on the exchange of rigid visual-inertial submaps, condensed packets computed through marginalization and consistent sparsification techniques, and pruned packets built through the selection of raw visual-inertial factors. We evaluated those methods on standard datasets, as well as on our custom AirMuseum dataset, which we purposively designed. Finally, from a mapping perspective, we applied the developped architecture to multi-robot dense SLAM, by extending a dense collaborative mapping method based on the exchange, the registration and the fusion of locally consistent submaps and associated with a TSDF representation.