Localisation précise d'un véhicule par couplage vision/capteurs embarqués/systèmes d'informations géographiques

Achkan Salehi

UNIVERSITE CLERMONT AUVERGNE

ECOLE DOCTORALE
SCIENCES POUR L'INGÉNIEUR DE CLERMONT-FERRAND

Thèse

Présentée par

ACHKAN SALEHI

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : Electronique et Systèmes (Vision artificielle)

Localisation précise d'un véhicule par couplage vision/capteurs embarqués/systèmes d'informations géographiques

Soutenue publiquement le 11 avril 2018 devant le jury :

Mme. Catherine ACHARD	Examinateur
M. Steve BOURGEOIS	Examinateur
M. Vincent CHARVILLAT	Examinateur
M. Frédéric CHAUSSE	Directeur de thèse
M. Michel DEVY	Rapporteur
M. Michel DHOME	Examinateur
M. Vincent GAY-BELLILE	Examinateur
Mme. Michèle ROMBAUD	Rapporteur

Résumé

La fusion entre un ensemble de capteurs et de bases de données dont les erreurs sont indépendantes est aujourd'hui la solution la plus fiable et donc la plus répandue de l'état de l'art au problème de la localisation. Les véhicules semi-autonomes et autonomes actuels, ainsi que les applications de réalité augmentée visant les contextes industriels exploitent des graphes de capteurs et de bases de données de tailles considérables, dont la conception, la calibration et la synchronisation n'est, en plus d'être onéreuse, pas triviale. Il est donc important afin de pouvoir démocratiser ces technologies, d'explorer la possibilité de l'exploitation de capteurs et bases de données bas-coûts et aisément accessibles. Cependant, ces sources d'information sont naturellement plus incertaines, et plusieurs obstacles subsistent à leur utilisation efficace en pratique. De plus, les succès récents mais fulgurants des réseaux profonds dans des tâches variées laissent penser que ces méthodes peuvent représenter une alternative peu coûteuse et efficace à certains modules des systèmes de SLAM actuels.

Dans cette thèse, nous nous penchons sur la localisation à grande échelle d'un véhicule dans un repère géoréférencé à partir d'un système bas-coût. Celui-ci repose sur la fusion entre le flux vidéo d'une caméra monoculaire, des modèles $3d$ non-texturés mais géoréférencés de bâtiments, des modèles d'élévation de terrain et des données en provenance soit d'un GPS bas-coût soit de l'odométrie du véhicule. Nos travaux sont consacrés à la résolution de deux problèmes. Le premier survient lors de la fusion par terme barrière entre le VSLAM et l'information de positionnement fournie par un GPS bas-coût. Cette méthode de fusion est à notre connaissance la plus robuste face aux incertitudes du GPS, mais est plus exigeante en matière de ressources que la fusion via des fonctions de coût linéaires. Nous proposons une optimisation algorithmique de cette méthode reposant sur la définition d'un terme barrière particulier. Le deuxième problème est le problème d'associations entre les primitives représentant la géométrie de la scène (*e.g.* points $3d$) et les modèles $3d$ des bâtiments. Les travaux précédents se basent sur des critères géométriques simples et sont donc très sensibles aux occultations en milieu urbain. Nous exploitons des réseaux convolutionnels profonds afin d'identifier et d'associer les éléments de la carte correspondants aux façades des bâtiments aux modèles $3d$. Bien que nos contributions soient en grande partie indépendantes du système de SLAM sous-jacent, nos expériences sont basées sur l'ajustement de faisceaux contraint basé images-clefs. Les solutions que nous proposons sont évaluées sur des séquences de synthèse ainsi que sur des séquence urbaines réelles sur des distances de plusieurs kilomètres. Ces expériences démontrent des gains importants en performance pour la fusion VSLAM/GPS, et une amélioration considérable de la robustesse aux occultations dans la définition des contraintes.

Mots-clefs : SLAM, VSLAM, fusion de capteurs, apprentissage profond, segmentation sémantique, GPS.

Abstract

The fusion between sensors and databases whose errors are independant is the most reliable and therefore most widespread solution to the localization problem. Current autonomous and semi-autonomous vehicles, as well as augmented reality applications targeting industrial contexts exploit large sensor and database graphs that are difficult and expensive to synchronize and calibrate. Thus, the democratization of these technologies requires the exploration of the possiblity of exploiting low-cost and easily accessible sensors and databases. These information sources are naturally tainted by higher uncertainty levels, and many obstacles to their effective and efficient practical usage persist. Moreover, the recent but dazzling successes of deep neural networks in various tasks seem to indicate that they could be a viable and low-cost alternative to some components of current SLAM systems.

In this thesis, we focused on large-scale localization of a vehicle in a georeferenced coordinate frame from a low-cost system, which is based on the fusion between a monocular video stream, $3d$ non-textured but georeferenced building models, terrain elevation models and data either from a low-cost GPS or from vehicle odometry. Our work targets the resolution of two problems. The first one is related to the fusion via barrier term optimization of VSLAM and positioning measurements provided by a low-cost GPS. This method is, to the best of our knowledge, the most robust against GPS uncertainties, but it is more demanding in terms of computational resources. We propose an algorithmic optimization of that approach based on the definition of a novel barrier term. The second problem is the data association problem between the primitives that represent the geometry of the scene (*e.g.* $3d$ points) and the $3d$ building models. Previous works in that area use simple geometric criteria and are therefore very sensitive to occlusions in urban environments. We exploit deep convolutional neural networks in order to identify and associate elements from the map that correspond to $3d$ building model façades. Although our contributions are for the most part independant from the underlying SLAM system, we based our experiments on constrained key-frame based bundle adjustment. The solutions that we propose are evaluated on synthetic sequences as well as on real urban datasets. These experiments show important performance gains for VSLAM/GPS fusion, and considerable improvements in the robustness of building constraints to occlusions.

Key-words : SLAM, VSLAM, sensor fusion, Deep learning, semantic segmentation, GPS.

Notations

Table des notations

Sauf exception, les symboles suivants seront utilisés :

M^T	Transposée de la matrice M
A/B	complément de Schur du bloc B d'une matrice $M = \begin{pmatrix} A & C \\ C^T & B \end{pmatrix}$, *i.e.* $A/B = A - C^T B^{-1} C$.
$d(.,.)$	Distance Euclidienne entre deux vecteurs
$\rho(.)$	Noyau robuste (*i.e.* M-estimateur) de type Geman-McClure
α_i	coefficients des générateurs d'une algèbre de Lie
$\mathfrak{g}$	Algèbre de Lie d'un groupe de Lie G.
$[m]_\wedge$	matrice anti-symmétrique telle que $[m]_\wedge v$ soit le produit vectoriel entre les vecteurs m et v de $\mathbb{R}^3$.
$A(i\colon j, k\colon l)$	La sous-matrice de A composée des lignes i à j (inclues) et des colonnes k à l (inclues).
$A(:, i:j)$	la sous-matrice composées de toutes les lignes de A et des colonnes de i à j (inclues).
$\mathrm{Im}(f)$	Image de la fonction f.
$m/ = x$	remplaçement de la valeur d'une variable m par m/x (pour $x \in \mathbb{R}$)

Acronymes

SLAM	Simultaneous localization And Mapping
VSLAM	Visual SLAM
CNN	Convolutional Neural Network

k-BA	key-frame Bundle Adjustment
CPD	Contraintes de Poses Directes
CPM	Contraintes de Poses Multivues
SFM	Structure From Motion

Sommaire

Introduction

Problématique et contributions

L'évolution rapide et récente des algorithmes de SLAM, de localisation et de fusion de capteurs, ainsi que l'essor des réseaux de neurones profonds depuis quelques années ont eu pour résultat la conception et la commercialisation de véhicules semi-autonomes et dans de moindres mesures, de véhicules autonomes. Les systèmes de navigation de ces véhicules restent très complexes et onéreux, et reposent sur la fusion de données issues de plusieurs capteurs, tels que des bancs de caméras, de multiples centrales inertielles, le LIDAR, l'odométrie, le GPS RTK. Le coût élevé de ces capteurs s'accompagne en outre des difficultés de calibration et de synchronisation entre les différents nœuds du graphe de capteurs, ainsi que des problèmes liés à la complexité en terme de temps de calcul. Il est donc nécessaire, afin de rendre ces systèmes plus accessibles, d'explorer la faisabilité de l'utilisation de capteurs bas-coûts de précision inférieure. Il est clair qu'il faut donc réviser la conception du système afin d'y inclure de nouvelles sources d'informations qui pourront compenser les insuffisances de ceux-ci. Les mêmes arguments peuvent être sans grande modification employés pour toute application nécessitant une localisation précise de la caméra. Par exemple, la réalité augmentée dans le cadre d'applications industrielles comme l'aide à la manœuvre de systèmes mécaniques complexes ou le contrôle de qualité peut nécessiter une précision de localisation millimétrique atteignable avec les savoir faire actuel que via des processus complexes dont la réduction des coûts serait tout aussi souhaitable.

Le choix des capteurs et des bases de données, ainsi que les compromis nécessaires entre précision et temps d'exécution varient en fonction, entre autres, de l'environnement auquel le SLAM est destiné, des types de déplacements et de l'échelle. Dans ce mémoire, qui se situe dans la continuité des travaux de [56], nous nous concentrerons sur le problème du SLAM à grande échelle visant à estimer l'environnement et la position absolue (*e.g.* dans un repère géoréférencé) par rapport à celui-ci d'un véhicule se déplaçant dans un environnement urbain. Les capteurs que nous utilisons sont tous bas-coût : il s'agit d'une caméra monoculaire, d'un GPS non-différentiel et/ou de l'odométrie du véhicule. Afin d'apporter une information complémentaire, nous exploitons simultanément des bases de données géographiques gratuitement accessibles en ligne. Il s'agit de modèles $3d$ non texturés géoréférencés (sous forme de maillages), et dans certains cas, de modèles d'élévation de terrain (sous forme de segments). Deux problèmes majeurs surviennent dans ce contexte :

- ▷ Ainsi qu'il a été démontré par [67, 56], il est en pratique plus avantageux d'employer une fonction de coût non-linéaire, en particulier l'optimisation dite *par terme barrière* pour fusionner le SLAM visuel avec des données dont l'incertitude est inconnue. Cependant, ceci implique l'inclusion d'un plus grand nombre de caméras dans l'optimisation, et par conséquent une augmentation de la complexité en temps. Ceci rend le déploiement de la solution sur les plate-formes embarquées de faible capacité difficile.

- ▷ Afin de pouvoir exploiter les modèles des bâtiments, il est nécessaire de définir des contraintes entre les primitives du maillage (*e.g.* plans) et celles de l'estimation, indépendamment de la représentation choisie pour cette dernière (*e.g.* nuage de points 3d, carte de profondeurs, etc). L'utilisation de critères purement géométriques conduit à un grand nombre de fausses associations, en particulier en présence d'occultations telles que la végétation, qui peut être dense dans certaines zones urbaines. Il est donc important d'augmenter la robustesse des associations de données.

Une autre motivation s'ajoute à cette liste : les succès récents des réseaux de neurones profonds dans des tâches complexes et variées, en particulier en terme de segmentation sémantique et calcul de pose et reconstruction 3d soulèvent la question de la relation entre le SLAM et l'apprentissage profond. Au moment de la rédaction de ce mémoire, cette relation demeure encore peu explorée. Cependant, même si quelques efforts récents indiquent la possibilité d'une localisation et cartographie uniquement basée sur l'apprentissage, la fiabilité de leurs estimations reste inférieure à celle des méthodes géométriques. Les travaux sur la modélisation des incertitudes n'étant pas encore très développés, nous pensons qu'il est plus avantageux d'exploiter le deep learning soit dans une fusion avec les résultats d'un SLAM géométrique, soit comme un bloc pouvant être exploité comme un module entrant dans la composition d'un système de SLAM plus classique.

Dans cette thèse, nous nous concentrons sur les deux problèmes énoncés plus haut, et tentons d'explorer la possibilité d'exploiter des réseaux de neurones profonds dans le cadre d'un SLAM géométrique. Nos contributions principales peuvent être résumées ainsi :

- ▷ Proposition d'une méthode réduisant la complexité en temps de la fusion par terme barrière entre un VSLAM et le GPS, mais qui ne détériore pas la précision de manière significative.
- ▷ Amélioration de l'association de données entre le nuage de points 3d et les plans des maillages représentant les modèles $3d$ des bâtiments, ce qui permet d'obtenir des contraintes extrêmement robustes aux occultations.

Cette thèse a donné lieu à trois publications internationales. Le chapitre 3 est associé à la publication [105], et le chapitre 4 est associée aux deux publications [104] et [106].

Contexte

Cette thèse a été effectuée entre janvier 2015 et janvier 2018 au Laboratoire Vision et Intégration des Contenus (LVIC) du CEA LIST, à Saclay, en partenariat avec l'Institut Pascal de l'université Clermont Auvergne.

Organisation du mémoire

Le chapitre 1 présente les notions élémentaires nécessaires à la compréhension de ce mémoire. Le chapitre 2 est dédié à une discussion sur l'état de l'art des méthodes de SLAM visuel en lien avec nos ambitions, ainsi qu'au positionnement de nos travaux. Les différentes formulation du VSLAM (filtrage, ajustement de faisceaux, approches directes, indirectes, denses, semi-denses et éparses) ainsi que leurs avantages et inconvénients y seront présentés de manière succincte. Nos objectifs au cours de cette thèse étant d'une part l'amélioration du temps d'exécution d'une méthode particulière de fusion (fusion par terme barrière, §2.2.1), ainsi que l'exploitation des possibilités de l'exploitation d'algorithmes de Deep Learning comme composants d'un SLAM contraint aux bâtiments, la suite du chapitre en question aura principalement pour sujet les méthodes de fusion, les approches d'optimisation (en terme de temps d'execution) de l'ajustement de faisceaux, et un tour d'horizon des méthodes liant SLAM de apprentissage. Notre positionnement conclura ce chapitre. Nos contributions sont rapportées dans les chapitres 3 et 4. Le problème de dimensionalité de la fusion par terme barrière ainsi que notre solution à celui-ci sont présentés dans le chapitre 3. Nous expliquerons le lien entre le temps d'exécution et l'estimation de la covariance locale de la caméra la plus récente, et démontrerons de manière théorique que notre approche possède certaines propriété qui facilitent la fusion. Nous présenteront deux fonctions de coûts différentes et présenterons les résultats expérimentaux associés. Le chapitre 4 aura pour sujet nos contribution à la contrainte au bâtiment. Nous y exposerons les difficultés qui découlent des fausses associations points/plan, et détaillerons les deux approches que nous avons proposées afin de pallier ce problème. Le chapitre 5 regroupe les travaux en cours, n'ayant pas été finalisés au moment de la rédaction de cette thèse, ainsi que les perspectives.

CHAPITRE 1

Éléments de base et données utilisées

Dans ce chapitre, nous passons en revue les prérequis nécessaires à la compréhension du mémoire, et détaillons les specificités des données/capteurs utilisés.

1.1 Modèle de caméras et géométrie projective

1.1.1 Modèle de caméra et coordonnées homogènes

Plusieurs modèles de caméra sont utilisés dans la littérature ([115]). Dans le cadre de cette thèse, nous n'utiliserons que le plus simple d'entre eux, appelé le modèle sténopé (figure 1.1). Celui-ci correspond à une formation idéale de l'image et ne modélise pas les distorsions. C'est pourquoi dans cette thèse, les images fournies en entrée des algorithmes sont considérées comme ayant été, au préalable, corrigées en distorsion (§1.1.3).

La modélisation par le modèle sténopé repose sur l'hypothèse que l'image d'un point $\mathcal{Q}_c$ dans l'espace $3d$ se forme au point d'intersection $\mathcal{Q}_{proj}$ entre le plan image et le rayon passant par $\mathcal{Q}_c$ et un point unique, appelé le *centre optique* de la caméra, et que l'on notera en général O_c. Il s'agit de la *projection* du point $\mathcal{Q}_c$ dans le plan image. La droite passant par le centre optique et orthogonale au plan image sera désignée par le terme *axe optique*. Notons que l'information de profondeur est perdue lors de la formation de l'image par un capteur monoculaire typique. En effet, en prenant comme base pour notre raisonnement le modèle sténopé, nous savons que le point $3d$ dont la projection dans le plan image est $\mathcal{Q}_{proj}$ est donné par $\lambda\mathcal{Q}_c - O_c$, mais il nous est impossible de déterminer λ sans information supplémentaire. Il est donc raisonnable de considérer que l'espace dans lequel l'image se forme est la plan projectif $2d$, noté $\mathbb{P}^2$. Intuitivement, il s'agit de l'espace des points (a, b, c) où la profondeur n'a pas d'importance. Plus formellement, il s'agit de l'ensemble des classes d'équivalence de $\mathbb{R} - (0, 0, 0)^T$, où la relation d'équivalence est donnée par

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \equiv \lambda \begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad \lambda \in \mathbb{R}. \tag{1.1}$$

Un point (a, b) de l'image peut donc s'écrire $(a, b, 1)$. Dans le second cas, les coordonnées utilisées sont appelées *homogènes*.

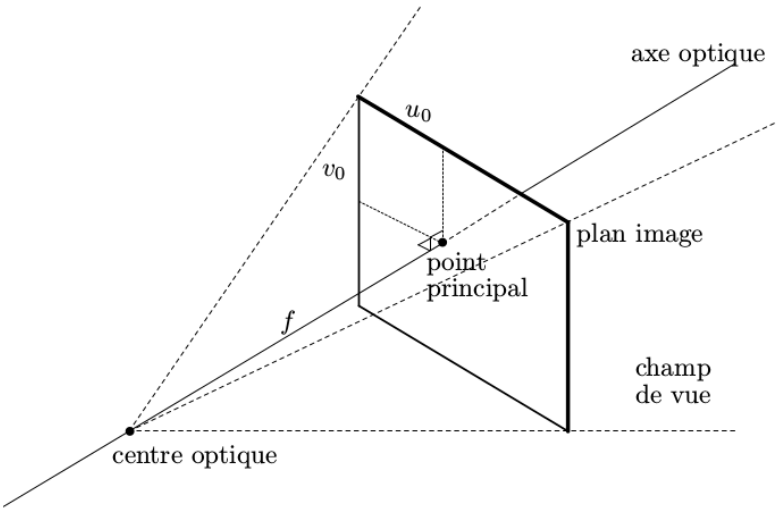


FIGURE 1.1 – Le modèle sténopé (Figure tirée de la thèse de [83]).

Dans ce texte, nous noterons

$$\pi\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right) = \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix} \tag{1.2}$$

1.1.2 Paramètres extrinsèques et intrinsèques, projection

Une caméra d'index i est exprimée dans le repère monde par un ensemble de paramètres dits extrinsèques : il s'agit de la rotation R_i et de la translation T_i du référentiel qui lui est attaché. Le vecteur T_i n'est autre que la position du centre optique de la caméra dans le repère monde, et chaque colonne de R_i est l'un des trois axes orthonormaux du système de coordonnées de la caméra. Il es clair que le passage d'un référentiel à l'autre se fera via

$$\mathcal{Q}_w = R_i \mathcal{Q}_c + T_i \tag{1.3}$$

où $\mathcal{Q}_w$ et $\mathcal{Q}_c$ désignent de manière respective les expression d'un point $3d$ dans le repère monde et dans le repère caméra.

Afin de simplifier les équations, nous respecterons la convention, populaire en vision par ordinateur, selon laquelle l'axe z du repère de la caméra coïncide avec l'axe optique. Dans ce cas, l'emplacement du plan image dans le repère de la caméra est donné par la distance focale f_0. La projection d'un point $\mathcal{Q}_c = (X_\mathcal{Q}, Y_\mathcal{Q}, Z_\mathcal{Q})$ sera alors

$$\begin{pmatrix} f_0 X_\mathcal{Q}/Z_\mathcal{Q} \\ f_0 Y_\mathcal{Q}/Z_\mathcal{Q} \\ f_0 \end{pmatrix} \tag{1.4}$$

Bien entendu, il se peut que le repère de l'image ne soit pas identique à celui dans lequel il est plongé (le repère caméra). Ceci peut être la conséquence d'une différence de taille entre

les pixels de l'image et les unités employées dans le référentiel de la caméra, d'un décalage de l'origine du système de coordonnées de l'image $(u_0, v_0)^T$ par rapport à celui du repère caméra, ou encore le résultat de pixels non rectangulaires. Compte tenu de nos images d'entrée, nous écartons cette dernière possibilité dans ce texte. Le changement de repère final se fera donc via

$$\begin{pmatrix} x_{\mathcal{Q}} \\ y_{\mathcal{Q}} \\ z_{\mathcal{Q}} \end{pmatrix} = \begin{pmatrix} (\frac{f_0}{du})(X_{\mathcal{Q}}/Z_{\mathcal{Q}}) + u_0 \\ (\frac{f_0}{dv})(Y_{\mathcal{Q}}/Z_{\mathcal{Q}}) + v_0 \\ f_0 \end{pmatrix} \tag{1.5}$$

où du et dv font (de manière respective) référence à la taille horizontale et à la taille verticale du pixel. Comme nous l'avons souligné plus tôt, l'information de profondeur est perdue, et les coordonnées finale que l'on obtient sont

$$\Big((\tfrac{f_0}{du})(X_{\mathcal{Q}}/Z_{\mathcal{Q}}) + u_0 \qquad (\tfrac{f_0}{dv})(Y_{\mathcal{Q}}/Z_{\mathcal{Q}}) + v_0\Big)^T \tag{1.6}$$

Nous pouvons ré-écrire ces relations sous forme matricielle et en coordonnées homogènes :

$$\begin{pmatrix} x_{\mathcal{Q}} \\ y_{\mathcal{Q}} \\ 1 \end{pmatrix} = \pi(\begin{pmatrix} \frac{f_0}{du} & 0 & u_0 \\ 0 & \frac{f_0}{dv} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_{\mathcal{Q}} \\ Y_{\mathcal{Q}} \\ Z_{\mathcal{Q}} \end{pmatrix}) = \pi(K\mathcal{Q}_c) \tag{1.7}$$

La matrice

$$K \triangleq \begin{pmatrix} \frac{f_0}{du} & 0 & u_0 \\ 0 & \frac{f_0}{dv} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.8}$$

est la matrice de *calibration des paramètres intrinsèques*. Il est clair que cette matrice est inversible, mais comme la fonction π ne l'est pas, nous ne pouvons obtenir à partir de $(x_{\mathcal{Q}}, y_{\mathcal{Q}}, 1)^T$ que le rayon $K^{-1}(x_{\mathcal{Q}}, y_{\mathcal{Q}}, 1)^T = \lambda(X_{\mathcal{Q}}, Y_{\mathcal{Q}}, 1)$ avec λ un coefficient réel, ce qui est conforme à nos attentes.

On sait que l'espace des matrices de rotations $SO(3)$ auquel appartiennent les matrices R_i est une variété différentielle de dimension 3. La démonstration est simple : on peut par exemple calculer une base de l'espace tangent en l'un des points de $SO(3)$ (section 1.4.3) et déduire que celui-ci est de dimension 3, ou encore de remarquer simplement que chaque perturbation infinitesimale dans $SO(3)$ peut s'exprimer sous forme d'un produit matriciel $R_x R_y R_z$ où les trois matrices commutent (car le mouvement est infinitesimal) et que par conséquent, $SO(3)$ ne possède que 3 degrés de liberté. Il est donc clair que $R_i \in \mathbb{R}^9$ est une paramétrisation redondante, dont l'utilisation dans le cadre d'une optimisation augmente artificiellement le nombre de paramètres et rend la convergence plus lente/difficile. Dans la littérature, les quaternions et les coefficients de l'algèbre de lie de $SO(3)$ sont des paramétrisations minimales populaires. Leurs efficacités semblent être équivalentes. Dans ce texte, nous utiliserons l'algèbre de Lie, principalement à cause des expressions élégantes et concises qui en découlent. Nous détaillerons cette paramétrisation à la section 1.4.

Les paramètres extrinsèques minimaux de chaque caméra i seront (sauf exception) notés $(\alpha_1, \alpha_2, \alpha_3, T)$ où les α_i correspondent au roulis, au lacet et au tanguage [1] et où $T \in \mathbb{R}^3$ est la translation. Notons au passage que chaque point $3d$ sera simplement paramétré par ses coordonnées Euclidiennes.

1. roll, pitch, yaw en Anglais

1.1.3 La distorsion

Il existe plusieurs types de distorsions. Les plus courantes sont les distorsions radiales et tangentielles induites par l'objectif de la caméra, et celle qui est introduite par le rolling shutter. Soit o_{orig} le centre de l'image I_{orig} et o_{dist} le centre d'une image I_{dist} ayant subi une distorsion. En pratique, on a en général $o_{orig} = o_{dist}$. La distorsion due à l'objectif peut donc être définie par le déplacement d'un point x_{dist} dans l'image I_{dist} par rapport à sa position d'origine x_{orig} dans l'image I. Si ce déplacement s'effectue le long de l'axe $x_{orig} - o_{orig}$, alors la distorsion est dite radiale. Dans le cas où elle possèderait une composante supplémentaire, la distorsion sera de plus tangentielle. Les images que nous prendrons en entrée dans nos travaux seront déjà corrigées en distorsion, c'est pourquoi nous invitons le lecteur intéressé à consulter [34, 62, 83].

Le *rolling shutter* est un mécanisme d'acquisition d'image où les pixels de l'images sont acquis ligne par ligne. La première ligne est enregistrée au temps t_{init}, la deuxième au temps $t_{init} + t_{acqui}$, et ainsi de suite. Par conséquent, la présence d'objets hautement dynamiques dans la scène peut produire une distorsion. Dans nos travaux, nous n'utilisons que des caméras *global shutter*, qui exposent l'intégralité de la surface photosensible de manière simultanée. Ce type de flou n'existe donc pas dans nos données d'entrée. Notons cependant que cela ne réduit en rien la généralité de nos contributions : cette distorsion, tout comme les distorsions radiales et tangentielles peut être corrigée par un pré-traitement séparé.

1.1.4 La Matrices Essentielle et la matrice Fondamentale

Soit $\mathcal{Q}_i$ l'expression d'un point $3d$ dans le repère d'une caméra d'index i. Considérons deux caméras d'indexes 1 et 2, et soit

$$\mathcal{Q}_1 = R\mathcal{Q}_2 + T \tag{1.9}$$

l'équation de passage du référentiel de la caméra 2 à celui de la caméra 1. Soit $\mathcal{Q}'_1$ un deuxième point. Nous voulons savoir si T, $\mathcal{Q}_1$ et $\mathcal{Q}'_1$ sont coplanaires. Supposons que ce soit le cas, dans ce cas il suffit que

$$det(\begin{pmatrix}\mathcal{Q}_1 & \mathcal{Q}'_1 & T\end{pmatrix}) = 0 \tag{1.10}$$

On a donc

$$\begin{aligned} det(\begin{pmatrix}\mathcal{Q}_1 & \mathcal{Q}'_1 & T\end{pmatrix}) & \\ &= det(\begin{pmatrix}\mathcal{Q}_1 & R\mathcal{Q}'_2 + T & T\end{pmatrix}) \\ &= \mathcal{Q}_1^T [T]_\wedge R\mathcal{Q}'_2 = 0 \end{aligned} \tag{1.11}$$

La matrice $E \triangleq [T]_\wedge R$ est la matrice *Essentielle*. Notons $\tilde{q}_i$ la projection en coordonnées homogènes du point $\mathcal{Q}_i$ dans l'image i. On remarque que $\mathcal{Q}_i = \lambda K^{-1}\tilde{q}_i$ et $\mathcal{Q}'_i = \lambda K^{-1}\tilde{q}'_i$, on peut donc réécrire l'équation 1.11 comme suit :

$$\tilde{q}_1^T K^{-T} [T]_\wedge R K^{-1} \tilde{q}_2 \tag{1.12}$$

La matrice $F \triangleq K^{-T}[T]_\wedge R K^{-1}$ est appelée la matrice *Fondamentale*.

Dans nos travaux, la matrice Essentielle sera utilisée pour l'initialisation du SLAM basé k-BA. Nous exploiterons la matrice Fondamentale au chapitre 3 dans les but de définir une contrainte de graphe de pose.

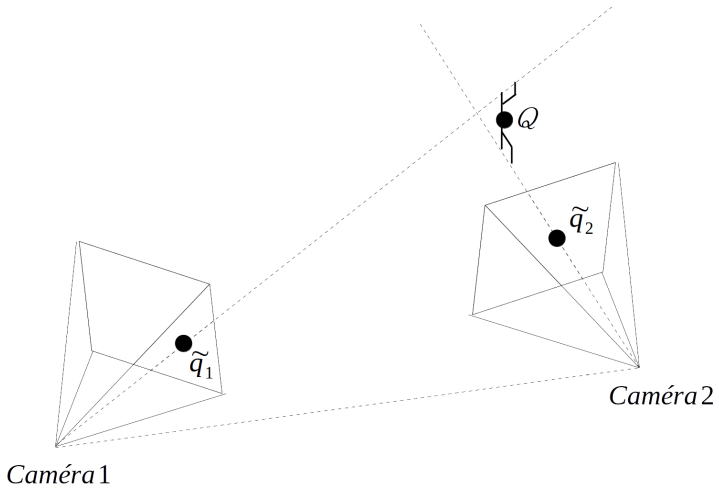


FIGURE 1.2 – Le principe de la triangulation.

1.1.5 La triangulation

Le terme *triangulation* désigne le calcul de la profondeur d'un point à partir de plusieurs observations (au minimum deux), lorsque les transformations relatives entre celle-ci sont connues. Le principe est illustré par la figure 1.2 pour les deux points $2d$ correspondants. Dans le cas idéal, le rayon passant par le centre optique de la première caméra et $\tilde{q}_1$ intersecterait le rayon passant par le centre optique de la deuxième caméra et le point $\tilde{q}_2$, et la profondeur du point pourrait simplement être déduite par des principes de géométrie élémentaires (loi des cosinus). Cependant, en pratique, les imprécisions des capteurs et les erreurs numériques font que ces deux rayons ne s'intersectent pas. Dans ce cas, on recherche le point Q dont la somme des distances avec les deux droites est minimale. Ce problème, qui peut facilement être généralisé au cas de plusieurs observations, n'est autre qu'un simple problème de moindres carrés dont l'expression est triviale. Nous nous abstenons donc de rapporter les équations. Pour plus de détails, le lecteur intéressé pourra se référer à la section 7.1 du chapitre 7 du livre de [117].

1.1.6 Le problème PnP

Le terme PnP (Perspective n point) désigne le problème du calcul de la pose relative d'une caméra par rapport à un autre repère à partir de correspondance $2d \leftrightarrow 3d$. Les points $2d$ sont exprimés dans le référentiel du plan image de la caméra et les points $3d$ dans le deuxième repère. Nous ne nous attarderons pas sur ce problème, car de nombreux algorithmes de calcul de pose performants (aussi bien symboliques que numériques) existent dans la littérature, et le choix de l'un d'entre eux en particulier n'aura qu'un effet négligeable sur les résultats présentés dans ce mémoire. Nous invitons donc le lecteur intéressé à consulter pour des exemples d'approches symboliques les travaux de [64, 65], et les travaux de [24] pour un exemple de méthode numérique. Ces trois altorithmes ont besoin de connaître la matrice de calibration. On pourra consulter [34] pour un exemple d'approche basée sur la DLT (Direct Linear Transform) qui opère sans aucun a priori sur les paramètres intrinsèques.

1.2 Optimisation non-linéaire

Dans le cadre de cette thèse, nous serons souvent intéressés par la minimisation d'une fonction réelle

$$\mathcal{B} : \mathbb{R}^s \to \mathbb{R}. \tag{1.13}$$

Nous utiliserons des outils standards d'optimisation non-linéaire itérative. Notons X_0 le point initial de la minimisation. Le développement de Taylor indique que pour X proche de X_0,

$$\mathcal{B}(X) = \mathcal{B}(X_0) + J_{\mathcal{B}}(X - X_0) + \frac{1}{2}(X - X_0)^T H_{\mathcal{B}}(X - X_0) \tag{1.14}$$

où $J_{\mathcal{B}}$ et $H_{\mathcal{B}}$ sont (respectivement) la Jacobienne et l'Hessienne de $\mathcal{B}$ par rapports aux paramètres de minimisation X, évaluées en X_0. On sait que la jacobienne évaluée en X est nulle si ce point est un extremum de $\mathcal{B}$. Par conséquent (lemmes 1 et 2, annexe A) :

$$H_{\mathcal{B}}(X - X_0) = -J_{\mathcal{B}}^T. \tag{1.15}$$

On notera que si $\mathcal{B}$ est une somme de i moindres carrés, alors $\mathcal{B} = e^T e$ où $e \in \mathbb{R}^i$. Dans ce cas, en notant J_e la Jacobienne de e par rapport à X, on a $H_{\mathcal{B}} \approx 2J_e^T J_e$ (annexe A, lemme 3). En prenant en compte l'équation A.3 du lemme A.3 de l'annexe A, on retombe dans ce cas sur l'équation plus répandue

$$\Delta X = (J_e^T J_e)^{-1} J_e^T e \tag{1.16}$$

1.2.1 Gauss-Newton

L'algorithme de Gauss-Newton consiste simplement en la minimisation itérative de l'équation 1.16. On choisit un point X_0 initial, On calcule l'incrément ΔX, et l'on remplace X_0 par $X + \Delta X$. L'intérêt de cette approche est la prise en compte de la courbure, cependant, elle est très sensible à l'initialisation et peut ne pas converger. En effet, il faut que l'Hessienne $H_{\mathcal{B}}$ soit définie positive.

1.2.2 Levenberg-Marquardt

L'algorithme de Levengberg-Marquardt (LM) oscille entre descente de gradient et Algorithme de Gauss-Newton. L'objectif de l'algorithme est de se comporter de manière semblable à la descente de gradient lorsque l'on est loin des minimas, et de manière similaire à Gauss-Newton lorsque l'on approche d'un minimum local. On résout donc à chaque étape

$$(H_{\mathcal{B}} + \lambda Id)(X - X_0) = -J_{\mathcal{B}}^T. \tag{1.17}$$

ou

$$(H_{\mathcal{B}} + \lambda diag(H_{\mathcal{B}}))(X - X_0) = -J_{\mathcal{B}}^T. \tag{1.18}$$

Si l'incrément obtenu à l'issue de cette étape conduit à une décroissance de la valeur de la fonction de côut, alors λ est remplacée par λ / v_{cst}. Dans le cas contraire, on remplace cette variable par $\lambda * v_{cst}$. Le plus souvent, il suffit de poser $v_{cst} = 10$.

1.2.3 L-BFGS

L'algorithme L-BFGS [70] est une version particulière de l'algorithme de Broyden-Fletcher-Goldfarb-Shanno [2]. Cette dernier méthode fait partie des approches "quasi-Newtoniennes", car elle approxime l'Hessienne afin de s'exploiter la structure locale de la fonction de coût. Elle est donc particulièrement adaptée aux problèmes où les approches de descente de gradient échouent, et où le nombre de paramètres est très élevé, ce qui rend le calcul exacte de l'Hessienne impossible en pratique. Cependant, l'algorithme BFGS représente l'inverse de l'Hessienne de manière dense. [70] ont donc proposé l'approche L-BFGS, qui limite la consommation en terme de mémoire de BGFS, en introduisant une représentation implicite et éparse de l'Hessienne. Nous ne nous attarderons pas sur les détails de cette approximation, et utiliserons la fonction `bgfsmin` du paquet `optim` de GNU Octave.

1.3 SLAM, (k-)BA et SLAM basé images clefs

Le terme SLAM est l'acronyme Anglais de *Simultaneous localization and mapping*, qui peut se traduire directement par *Localisation et cartographie simultanée*. Il s'agit de l'estimation conjointe d'une carte de l'environnement et de la localisation de l'agent par rapport à celle-ci. Un grand nombres de capteurs proprioceptifs (*e.g.* centrales inertielles) et extéroceptifs (*e.g.* Caméras, Lidar, sonar) peuvent être utilisés comme source d'information pour la résolution de ce problème. On parlera de VSLAM (pour *Visual SLAM*) lorsque le SLAM reposera principalement sur des capteurs visuels, tels que les caméras RGB ou RGB-D. En général, le problème du SLAM peu être formulé de manière probabiliste comme l'estimation conjointe de la loi postérieure (voir [121])

$$P(x_t, x_{t-1}, ..., x_0 | z_t, ..., z_0, u_0, ..., u_t) \tag{1.19}$$

où x_i est le vecteur d'état estimé à l'instant i, et z_i désigne une observation (*i.e.* mesure) apportant une information directe ou indirecte sur le problème. La variable u_i représente les commandes transmises à l'agent à l'instant i. La variable x_i peut contenir différents paramètres en fonction des applications et de la méthode choisie. Il peut s'agir de variables exprimant l'état de l'agent (*e.g.* pose, accélération, vitesse), ou de paramètres modélisant des éléments de l'environnement (*e.g.* points 3d, droite ou toute autre structure), ou d'un mélange des deux. Les observations z_i permettent d'inférer la vraisemblance des nos estimations, et il peut s'agir, par exemple, de détections de points d'intérêt $2d$ dans une image, ou d'une profondeur estimé par un capteur RGB-D. Les variables u_i qui correspondent en général aux commandes envoyées à un acteur de l'agent ayant une influence sur son mouvement (*e.g.* roues, bras robotique, etc) ne seront pas utilisées dans le cadre de cette thèse.

Le théorème de Bayes nous permet de résoudre le problème équivalent

$$P(x_t, x_{t-1}, ..., x_0 | z_t, ..., z_0, u_0, ..., u_t) \propto P(z_t, ..., z_0 | x_t, ..., x_0, u_0, ..., u_t) P(x_t, ..., x_0 | u_0, ..., u_t) \tag{1.20}$$

Notons que la plupart des méthodes n'estiment que l'état x_t, car la ré-estimation de l'historique complet des états n'est pas envisageable en temps-réel, ou est impossible si la quantité de

2. l'algorithme a été découvert et publié indépendamment par chacun des auteurs. Voir par exemple le livre de [101] pour plus de détail.

données peut croître de manière illimitée, comme dans le cas d'un agent autonome opérant sur de longs intervalles de temps et de longues distances. Notons cependant que x_t peut encapsuler des informations sur l'état de l'agent et de son environnement sur une fenêtre temporelle. Dans cette thèse, nous serons intéressé par ce cas, c'est à dire que nous estimerons x_t à l'instant t et non pas l'historique complet.

Cependant, même en retirant les $x_0, .., x_{t-1}$ de l'équation 1.20, cette formulation demeure extrêmement générale, et la plupart des méthodes de SLAM résolvent donc un cas particulier de ce problème. A titre d'exemple, de nombreuses variantes du filtre de Kalman reposent sur les hypothèses que l'observation z_i ne dépend que de x_i, et que l'état est complet ([121]), c'est à dire qu'il suffit de connaître l'état à l'instant $i-1$ pour pouvoir prédire l'état à l'instant i [3]. Plus formellement, ces approches se penche sur la résolution du problème

$$P(x_t|z_t, ..., z_0, u_0, ..., u_t) \propto P(z_t|x_t)P(x_t|u_t). \tag{1.21}$$

Le calcul de $P(x_t|u_t)$ et de la vraisemblance $P(z_t|x_t)$ sont respectivement appelés la prédiction et la mise à jour. On notera que $P(x_t|u_t) = \int_{x_{t-1}} P(x_t, x_{t-1})dx_{t-1}$, ce qui implique qu'inclure la prédiction dans la mise à jour conduit à une marginalisation implicite des anciens vecteurs d'état. Notons que l'on parle ici de marginalisation au sens probabiliste. Il s'agit de cette propriété élémentaire pour toute variables aléatoires A et B :

$$\int_A P(A, B)dA = \int_A P(A|B)P(B)dA = P(B)\int_A P(A|B) = P(B). \tag{1.22}$$

Ainsi qu'on le verra dans la section suivante, l'ajustement de faisceaux peut être vu comme l'étape de calcul de vraisemblance de l'équation.

1.3.1 Ajustement de faisceaux

Supposons que le nombre de caméras à optimiser soit de l, et que le nombre de points à optimiser soit de N. Soit $\Phi(j)$ le nombre de points $3d$ visibles depuis la caméra j, Dans ce mémoire, nous noterons en général

$$e = \begin{pmatrix} e_{11} & ... & e_{1\Phi(1)} & e_{21} & ... & e_{2\Phi(2)} & ... & e_{l\Phi(l)} \end{pmatrix}^T \tag{1.23}$$

le vecteur des erreurs de reprojections, tel que e_{ij} est la différence entre la projection du point j et son observation dans le plan image de la caméra j.

$$e_{ij} = \pi(K(R_i^T P_j - R_i^T T_i)) - \begin{pmatrix} m_{ij}^1 \\ m_{ij}^2 \end{pmatrix} \in \mathbb{R}^2 \tag{1.24}$$

L'ajustement de faisceaux, que nous noterons BA (pour Bundle Adjustment en Anglais) désigne la minimisation des erreurs de reprojections, *i.e.* la minimisation de

$$B(X) = e^T \Lambda e \tag{1.25}$$

où Λ est une matrice de pondération. Le vecteur X concatène les paramètres extrinsèques des caméras et les points 3d optimisés. Nous noterons qu'en posant l'hypothèse que le vecteur e est une variable aléatoire Gaussienne, on remarque que la minimisation de $B(X)$ revient à maximiser $\exp(-e^T \Lambda e)$, ce qui n'est autre que l'expression d'une loi Gaussienne multivariables où Λ

3. Le terme "hypothèse Markovienne" est parfois employé pour désigner l'hypothèse de la complétude de l'état

est l'inverse de la covariance. On voit donc que cette expression n'est en fait un cas particulier de la vraisemblance de l'équation 1.3.

Ainsi que nous l'avons constaté en §1.2, la minimisation de 1.25 revient à résoudre un système de semblable (en terme de structure)

$$H_B(X - X_0) = -J_B^T. \tag{1.26}$$

(avec H_B l'Hessienne de B et J_B sa Jacobienne) de manière itérative. Les terme λId ou $\lambda diag H_B$ employés par l'algorithme LM ne modifient pas la structure de l'Hessienne. On s'intéressera donc, sans perte de généralité, à la résolution de l'équation 1.26. Les résultat qui découleront de cette discussion seront directement transférable à l'algorithme de LM.

La résolution de 1.26 via l'inversion de H_B est impossible en pratique, car le nombre élevé de points $3d$ (même lorsque le nombre de caméra observé est limité) confère une dimensionnalité non-négligeable au problème (*i.e.* H_B est d'une taille trop importante), ce qui rend la complexité en temps de calcul trop élévée pour des applications temps-réelles. Cependant, plusieurs techniques et stratégies permettent de pallier le problème. Dans ces travaux, nous nous baserons sur l'ensemble de techniques communément appelées ajustement de faisceaux basé images clefs.

1.3.2 Ajustement de faisceaux local basé images clefs

L'ajustement de faisceaux basé images clefs local limite le nombre de paramètres à optimiser à ceux qui correspondent aux l caméras les plus récentes [4] et aux points observés depuis celles-ci. Par soucis de concision, nous omettrons (sauf si le contexte l'impose) le terme "local", et désignerons ce type d'ajustement de faisceaux par l'acronyme Anglais k-BA (pour key-frame Bundle Adjustment). Outre cette réduction du nombre de paramètres en limitant l'optimisation à une fenêtre glissante, le k-BA standard exploite

▷ La redondance des données

▷ La structure particulière d'un ré-ordonnement de l'Hessienne

Concernant le premier point, il s'agit simplement de constater que l'information d'images successives est très redondante, et qu'il est donc possible d'échantilloner l'ensemble des images afin de réduire le temps d'exécution sans que le précision ne diminue de manière significative. On choisira donc un nombre "d'images clefs", *i.e.* que l'on inclura dans la minimisation en ignorant toutes les autres images. Une image est donc selectionnée comme telle si elle apporte une information suffisante par rapport à l'image clef précédente. Un critère de choix d'image clef possible est, par exemple, le nombre d'inliers lors du calcul de pose PnP.

Concernant le deuxième point, il nous ré-ordonnons l'Hessienne et d'exploiter le complément de Schur (annexe A, 1)afin de résoudre le problème en temps réel. Nous ré-ordonnons donc la Jacobienne J_e de la manière suivante :

$$J_e = \begin{pmatrix} \frac{\partial e}{\partial X_{cams}} & \frac{\partial e}{\partial X_{points}} \end{pmatrix} \tag{1.27}$$

avec $X_{cams} \in \mathbb{R}^{6\times l}$ le vecteur des paramètres extrinsèques de toutes les caméras et $X_{points} \in \mathbb{R}^{3*N}$ le vecteur concaténant tout les paramètres des points $3d$. Par conséquent

4. Il est plus avantageux de déterminer les caméras à optimiser en utilisant un critère de covisibilité [86], mais ceci n'est pas indispensable dans le cadre de ce tapuscrit.

$$H_B = \begin{pmatrix} (\frac{\partial e}{\partial X_{cams}})^T \frac{\partial e^T}{\partial X_{cams}} & (\frac{\partial e}{\partial X_{cams}})^T \frac{\partial e}{\partial X_{points}} \\ (\frac{\partial e}{\partial X_{points}})^T \frac{\partial e}{\partial X_{cams}} & (\frac{\partial e}{\partial X_{points}})^T \frac{\partial e}{\partial X_{points}} \end{pmatrix} \triangleq \begin{pmatrix} U & W \\ W^T & V \end{pmatrix}. \tag{1.28}$$

Il est clair que les erreurs de reprojections d'une caméra i ne sont pas affectées par les changements des paramètres (extrinsèques et intrinsèques) d'une caméra $j \neq i$. De même manière, les erreur de reprojection d'un point j ne sont pas affectées par les déplacements d'une point $k \neq j$. Donc, d'après le définition de l'Hessienne (comme étant la matrice contenant les dérivées secondes), que U est une matrice bloc-digonale, où chaque bloc est de taille 6×6. De la même manière, V est une matrice bloc-diagonale où chaque bloc est de taille 3×3. Un exemple d'Hessienne de BA, avec l'ordre spécifié, est illustré par la figure 1.4.

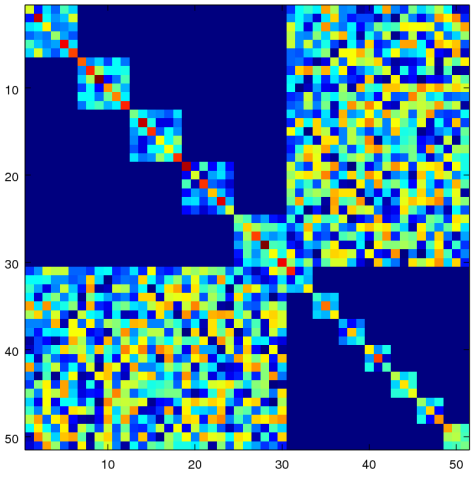


FIGURE 1.3 – Un exemple illustratif de matrice de BA, avec $l = 5$ caméras et $N = 7$ points.

Le complément de Schur (annexe A, 1) permet d'arriver à la solution suivante

$$(U - WV^{-1}W^T)\Delta X_{cams} = (\frac{\partial e}{\partial X_{cams}})^T - WV^{-1}(\frac{\partial e}{\partial X_{points}})^T \tag{1.29}$$

$$\Delta X_{points} = V^{-1}((\frac{\partial e}{\partial X_{points}})^T - W^T \Delta X_{cams}) \tag{1.30}$$

On notera que $(U - WV^{-1}W^T)$ est une matrice de la même taille que U, elle est donc de taille négligeable dans le cadre d'une k-BA, où le nombre de keyframes est en générale limité (~ 10, et dans de rares cas que nous rencontrerons au chapitre 3, de taille ~ 40). On peut donc l'inverser de manière performante en temps réel, en utilisant par exemple la factorisation de Cholesky. Pour plus de détails, nous invitons le lecteur à consulter [83].

1.3.3 Quelques notes sur les covariances et l'effet de la marginalisation

Nous utiliserons les deux théorèmes suivants pour la propagation de la covariance à travers une fonction. Les démonstrations peuvent être retrouvées dans le livre de [34].

***Théorème 1.* 1.** *Soit v un vecteur aléatoire dans $\mathbb{R}^m$ de moyenne $\bar{v}$ et de matrice de covariance Σ, et soit la fonction $f : \mathbb{R}^m \to \mathbb{R}^n$. Dans ce cas, la covariance de $f(v)$ est donnée par une variable aléatoire de moyenne $f(\bar{v})$ et de covariance $J\Sigma J^T$, où J est la Jacobienne de f.*

***Théorème 1.* 2.** *Théorème de la backpropagation pour les paramétrisation non-redondantes : Soit $\mathbb{R}^m \to \mathbb{R}^n$ différentielle et soit J sa Jacobienne évaluée en $\bar{P}$, telle que $rang(J) = M$. Dans ce cas f est injective dans un voisinage de $\bar{P}$. Cela implique que l'image de f, notée ici S_m, est une $m - variété$ différentielle. Soit X une variable aléatoire dans $\mathbb{R}^n$ de moyenne $\bar{X} = f(\bar{P})$ et de covariance Σ_X, et notons $\eta : \mathbb{R}^n \to S_M$ la fonction qui associe chaque point $X \in \mathbb{R}^n$ au point le plus proche (au sens d'une distance de mahalanobis ou Euclidienne) sur S_m. Dans ce cas $f^{-1} \circ \eta(X)$ est une variable aléatoire de moyenne $\bar{P}$ et de covariance $(J^T \Sigma_X J)^{-1}$.*

En appliquant le théorème 1.2 à l'équation 1.25, on remarque que la covariance du vecteur d'état (*i.e.* du vecteur concaténant les poses et les points $3d$) peut s'écrire $(J_e^T \Sigma J_e)^{-1}$. Comme en général il est impossible ou difficile en pratique de connaître Λ, on suppose ([34] que $\Lambda = \sigma Id$ où σ est un coefficient réel. Dans notre cas, nous poserons $\Lambda = \frac{1}{2} Id$ afin de simplifier les notations. En effet, on remarque que dans ce cas, la covariance est donnée par $(J_e^T J_e)^{-1}$, c'est à dire par l'inverse de l'Hessienne de la fonction B définie par l'équation 1.25. Comme nous l'avons vu, cette Hessienne possède une structure particulière (équation 1.28). Notons la covariance (qui est une matrice symmétrique et définie positive)

$$Cov(X_{cams}, X_{points}) = \begin{pmatrix} C_a & C_b \\ C_b^T & C_d \end{pmatrix} \tag{1.31}$$

Comme nous pouvons calculer cette matrice en inversant H_B via le complément de Schur (annexe A, 1), on peut facilement déduire que

$$C_a = (U - WV^{-1}W^T)^{-1}. \tag{1.32}$$

de la même manière, on peut écrire

$$C_d = (V - W^T U^{-1} W)^{-1}. \tag{1.33}$$

Ce résultat nous sera utile dans le chapitre 3.

Supposons maintenant que l'on marginalise (au sens probabiliste, équation 1.22) toutes les caméras. Dans ce cas, la covariance des variables restantes (*i.e.* les points) sera égale à C_d. En notant J_{points} le sous-vecteur de J_e correspondant aux dérivées par rapport aux points, la matrice Hessienne de ce nouveau problème sera donc $J_{points}^T J_{points} = C_d^{-1} = V - W^T U^{-1} W$ (d'après l'équation 1.33). Comme $W^T U^{-1} W$ n'est pas nécessairement éparse, on constate que $J_{points}^T J_{points}$ est le résultat d'une densification de V. Dans cet exemple, nous avons marginalisé l'ensemble des points, mais on peut employer les mêmes arguments en marginalisant tout autre sous-bloc de C_a. Dans ce cas, le complément de Schur permettra de montrer que c'est un bloc dont V est un sous-bloc qui peut se retrouver densifié. Pour plus d'informations, le lecteur intéressé est invité à consulter les travaux de [22].

1.3.4 SLAM basé image clefs et fusion

Comme nous l'avons vu à la section précédente, le k-BA n'est qu'un processus d'optimisation d'une reconstruction déjà existante. Bien que nos contributions soient en grande partie indépendantes du système de SLAM employé, nos travaux reposent sur un SLAM basé image clefs,

qui est essentiellement identique à celui de [83]. L'ensemble des processus étant aujourd'hui standards en vision par ordinateur et employés dans plusieurs systèmes connus ([51, 86, 87], nous nous contenterons d'une description succincte de chaque brique.

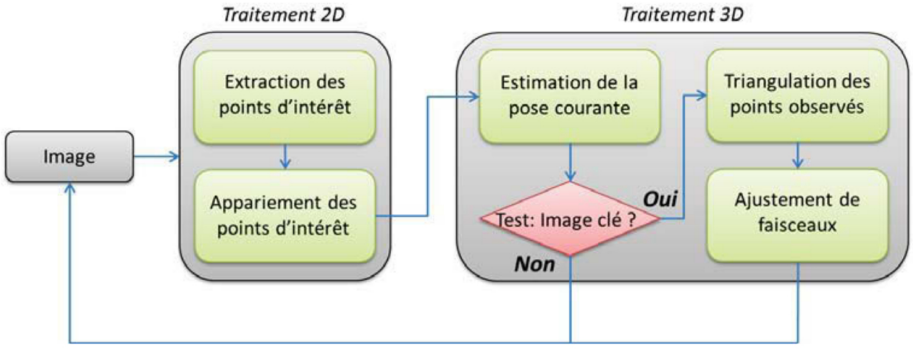


FIGURE 1.4 – Pipline du Slam basé images clefs de [83] (Figure tirée de la thèse de [56]).

1.3.4.1 Pipeline du SLAM basé images clefs

Extraction de points d'intérêt et de descripteurs Le k-BA est essentiellement une méthode qui repose sur l'optimisation d'un nuage de points éparse. Les points $3d$ sont le résultat de la triangulation d'un ensemble de points d'întérêt $2d$ détectés et appariés dans les images. Les points d'intérêt utilisés dans le cadre de cette thèse sont des points de Harris ([33]), et les descripteurs utilisés dans le but de permettre la mise en correspondance des points sont inspirés des descripteurs SURF ([7]). L'extraction de points d'intérêt se fait sur chaque nouvelle image.
Initialisation. La reconstruction initiale du nuage de points nécessite de connaître la transformation relative entre au moins un couple de caméras. Ceci peut être accompli par le calcul de la matrice Essentielle (ou Fondamentale) entre deux caméras. Plusieurs méthodes, telles que celles [74] et [93] ont été proposées. La première a l'avantage d'être plus rapide, mais ses résultats sont moins précis que ceux de [93] qui recherche les solutions de manière symbolique plutôt que numérique. C'est cette deuxième méthode que nous utilisons dans le cadre de cette thèse. Une fois la matrice Essentielle estimée, la transformation relative peut facilement être extraite ([34]), et les points $3d$ observés depuis les deux images peuvent être triangulés (§1.1.5). Notons que nous utilisons un triplet d'images pour l'initialisation afin d'en garantir la fiabilité : une matrice Essentielle est estimée entre deux d'entre elles, et la troisième permet de valider sa précision.

Détection des images clefs, triangulation et k-BA La suite du processus est constitué d'un ensemble de traitements que nous avons déjà décrits. La pose de chaque nouvelle image est calculées via la résolution d'un problème PnP, défini par la mise en correspondance des points $2d$ detectés dans celle-ci et les points $3d$ déjà triangulés. Si celle-ci est une image-clefs (voir le début de §1.3.2 pour une discussion sur les critères de sélection), les points $2d$ appariés sont triangulés et les points $3d$ résultants sont ajoutés au nuage de point. Finalement, un k-BA sur un certain nombre d'images clefs optimise l'estimation. Pour plus de détails, nous renvoyons le lecteur à la thèse de [83].

1.3.4.2 La fusion

La dérive du SLAM monoculaire en 7 degrés de liberté (6 pour la pose, 1 pour l'échelle) est un problème bien connu. La solution la plus efficace pour pallier ce problème est la fusion du SLAM avec d'autres capteurs (*e.g.* IMU, Odomètres, boussoles magnétiques, modèles $3d$ des bâtiments, etc). En outre, un SLAM monoculaire repose sur des mesures relatives, par conséquent le repère dans lequel il représente les données ne peut qu'être arbitraire. Ceci est problématique dans le cadre d'applications qui nécessitent une localisation géoréférencée. Cette dernière peut être atteinte via la fusion du VSLAM avec les données issues de capteurs tels que le GPS ou de bases de données tels que les modèles de bâtiments.

Plusieurs méthodes de fusion se concurrencent dans l'état de l'art, et nous reportons la discussion approfondie à ce sujet au chapitre 2 (section §2.2.1).

1.4 Groupes et Algèbres de Lie

Définition 1. **1.** *Un groupe de Lie G est un groupe qui est aussi une variété différentielle (annexe A, définition 4), et dont la loi de composition ainsi que l'opération d'inversion sont C^∞. L'espace tangent en l'identité I_G de ce groupe, défini par l'ensemble de vecteurs tangents en I_G aux chemins (i.e. fonctions continues) $\gamma : [0,1] \to G$ passant par I_G est l'algèbre de Lie de G et sera notée $\mathfrak{g}$ s'il est muni d'une application bilinéraire $[.]$ vérifiant :*

▷ $[x,x] = 0 \quad \forall x \in \mathfrak{g}$

▷ *(identité de Jacobi)* $[a,[b,c]] + [b,[c,a]] + [c,[a,b]] = 0 \quad \forall a,b,c \in \mathfrak{g}$

On appellera cet application le crochet de Lie.

Notre utilisation de ces concepts restera très élémentaire dans le cadre de cette thèse. C'est pourquoi nous ne nous attarderons pas sur la définition rigoureuse des vecteurs tangents. Nous nous contenterons de noter que cette définition se doit d'être indépendante de l'espace Euclidien dans lequel une variété peut être plongée. Le lecteur intéressé pourra consulter le deuxième chapitre de [127] pour une discussion plus avancée à ce sujet. On remarquera au passage que $\mathfrak{g}$ est un espace vectoriel.

Nous serons uniquement intéressés par certains sous-groupe fermés de $GL(n)$ et par leurs algèbres de Lie, auxquels nous consacrons la section suivante.

1.4.1 SO(3), SE(3) et SIM(3)

Dans nos travaux, nous sommes intéressés par les groupes des rotations, transformations Euclidiennes et similitudes dans $\mathbb{R}^3$, que nous notons respectivement $SO(3)$, $SE(3)$ et $SIM(3)$. Leurs algèbres de Lie seront notées $\mathfrak{so}(3)$, $\mathfrak{se}(3)$, $\mathfrak{sim}(3)$. Ces ensembles seront respectivement des variétés algébriques de dimensions 3, 6 et 7. Dans la suite, nous déduirons les bases $G_1^{\mathfrak{sim}}, ..., G_7^{\mathfrak{sim}}$ de $\mathfrak{sim}(3)$. Les bases de $\mathfrak{so}(3)$ et de $\mathfrak{se}(3)$ pouvant être obtenues de manière similaires.

Un élément $S \in SIM(3)$ peut s'écrire

$$\begin{pmatrix} sR & T \\ \mathbf{0}_{3\times 1} & 1 \end{pmatrix} \tag{1.34}$$

avec s l'échelle, R une matrice de rotation et T une translation. A l'identité $I_{4\times4} \in SIM(3)$, chaque perturbation d'un des 7 degrés de liberté permet d'obtenir un chemin $\gamma_i(t)$ dans $SIM(3)$ passant par $I_{4\times4}$. Les dérivées de ces chemins $\frac{\partial\gamma_i(t)}{dt}$ en l'identité (*i.e.* $t = 0$) seront des vecteurs tangents en $I_{4\times4}$, qui engendrent $\mathfrak{g}$.

$$\gamma_1(t) = \begin{pmatrix} cos(t) & -sin(t) & 0 & 0 \\ sin(t) & cos(t) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow \quad \frac{\partial\gamma_1(0)}{dt} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.35}$$

$$\gamma_2(t) = \begin{pmatrix} cos(t) & 0 & -sin(t) & 0 \\ 0 & 1 & 0 & 0 \\ sin(t) & 0 & cos(t) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow \quad \frac{\partial\gamma_2(0)}{dt} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.36}$$

$$\gamma_3(t) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(t) & -sin(t) & 0 \\ 0 & sin(t) & cos(t) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow \quad \frac{\partial\gamma_3(0)}{dt} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.37}$$

$$\gamma_4(t) = \begin{pmatrix} 0 & 0 & 0 & t \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \quad \frac{\partial\gamma_4(0)}{dt} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.38}$$

$$\gamma_5(t) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \quad \frac{\partial\gamma_5(0)}{dt} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.39}$$

$$\gamma_6(t) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t \\ 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \quad \frac{\partial\gamma_6(0)}{dt} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.40}$$

$$\gamma_7(t) = \begin{pmatrix} t & 0 & 0 & 0 \\ 0 & t & 0 & 0 \\ 0 & 0 & t & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \quad \frac{\partial\gamma_7(0)}{dt} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.41}$$

On peut constater d'après les équations 1.35 à 1.37 que l'espace engendré par $\frac{\partial\gamma_1(0)}{dt}$, $\frac{\partial\gamma_2(0)}{dt}$ et $\frac{\partial\gamma_3(0)}{dt}$ n'est autre que l'espace des matrices 3×3 anti-symmétriques. De la même manière (équations 1.38 à 1.40), l'espace engendré par $\frac{\partial\gamma_4(0)}{dt}$, $\frac{\partial\gamma_5(0)}{dt}$ et $\frac{\partial\gamma_6(0)}{dt}$ est équivalent à $\mathbb{R}^3$. Finalement, l'espace engendré par $\frac{\partial\gamma_7(0)}{dt}$ n'est autre que l'ensemble des multiples réels de l'identité (équations 1.41).

Nous utiliserons la base

$$\begin{cases} G_i^{\mathfrak{sim}(3)} = \frac{\partial\gamma_i(0)}{dt} & \text{si } i \neq 2 \\ G_2^{\mathfrak{sim}(3)} = -\frac{\partial\gamma_2(0)}{dt} & \text{si } i = 2 \end{cases} \tag{1.42}$$

Les équations 1.35 à 1.37 correspondant aux perturbations par des matrices de rotations, on peut observer qu'une base de $\mathfrak{so}(3)$ est donnée par

$$G_i^{\mathfrak{so}(3)} = G_i^{\mathfrak{sim}(3)}(1:3, 1:3) \quad i = 1, 2, 3 \tag{1.43}$$

où la notation $(1 : 3, 1 : 3)$ fait référence à la sélection du sous-bloc composé des lignes et colonnes 1 à 3 (incluses). Le choix de $-\frac{\partial\gamma_2(0)}{dt}$ plutôt que l'opposé de ce vecteur a pour motivation la simplification des notations. En effet, un élément de l'espace des matrices anti-symmétriques de taille 3 peut maintenant s'écrire

$$\sum_{i=1}^{3} \alpha_i G_i^{\mathfrak{so}(3)} = \begin{pmatrix} 0 & \alpha_1 & -\alpha_2 \\ -\alpha_1 & 0 & \alpha_3 \\ \alpha_2 & -\alpha_3 & 0 \end{pmatrix} = [\boldsymbol{\alpha}]_\wedge \tag{1.44}$$

où $[\boldsymbol{\alpha}]_\wedge$ est la notation classique pour la matrice de multiplication vectoriel par $\boldsymbol{\alpha} = [\alpha_1,\ \alpha_2,\ \alpha_3]$.

De la même manière, les vecteurs $G_i^{\mathfrak{sim}(3)}$ pour $i = 1, ..., 6$ (*i.e.* tous sauf 1.41) engendrent $\mathfrak{se}(3)$.

Ces deux fonctions s'avèreront utiles dans la suite :

$$\begin{aligned} &\vee : \mathfrak{g} \to \mathbb{R}^k \\ &\vee (\sum_i^k \alpha_i G_i) = (\alpha_1, ..., \alpha_k)^T \end{aligned} \tag{1.45}$$

$$\begin{aligned} &\wedge : \mathbb{R}^k \to \mathfrak{g} \\ &\wedge ((\alpha_1, ..., \alpha_k))^T = (\sum_i^k \alpha_i G_i) \end{aligned} \tag{1.46}$$

1.4.2 Exponentielle et logarithme

L'exponentielle matricielle est définie pour $A \in GL(n)$ par

$$e^A = \sum_{i=0}^{\infty} \frac{A^i}{i!}. \tag{1.47}$$

Il est important d'observer que

$$AB = BA \Rightarrow e^{A+B} = e^A e^B \tag{1.48}$$

mais que l'inverse n'est pas nécessairement vrai ([27, 112]).

Les fonctions obtenues en restreignant le domaine de l'exponentielle à $\mathfrak{so}(3)$, $\mathfrak{se}(3)$ et $\mathfrak{sim}(3)$ seront respectivement notées $\exp_{\mathfrak{so}(3)}$, $\exp_{\mathfrak{se}(3)}$ et $\exp_{\mathfrak{sim}(3)}$.

On peut démontrer ([27] que $\exp_{\mathfrak{so}(3)}$ est surjective, mais elle n'est pas bijective. Cependant, on peut définir l'inverse de l'exponentielle en restreignant son domaine de manière appropriée. Ses inverses seront notées $\log_{SO(3)}$, $\log_{SE(3)}$, $\log_{SIM(3)}$.

Dans le cadre de cette thèse, nous n'utiliserons que les fonctions $\exp_{\mathfrak{so}(3)}$, $\log_{SO(3)}$, $\log_{SE(3)}$, $\exp_{\mathfrak{se}(3)}$ [5]. Les deux premières seront utilisées pour la paramétrisation minimale des rotations et le calcul des dérivées de l'ajustement de faisceaux et d'une des fonctions de coût que nous présenterons au chapitre 3. L'exponentielle et le logarithme de $SE(3)$ seront utilisées pour évaluer la distance entre deux éléments de $SE(3)$, afin d'imposer de contraintes de graphe de pose.

5. Le facteur d'échelle fourni par le GPS ou les modèles $3d$ des bâtiments suffit pour nos travaux.

On peut démontrer que les exponentielles et le logarithme matriciel de $SO(3)$ peuvent s'écrire ([27])

$$\begin{aligned} &\exp_{\mathfrak{so}(3)} : \mathfrak{so}(3) \to SO(3) \\ &\exp_{\mathfrak{so}(3)}([\alpha]_\wedge) = \begin{cases} I & \text{si } \theta \to 0 \\ I + \frac{sin(\theta)}{\theta}[\alpha]_\wedge + \frac{1-cos(\theta)}{\theta^2}[\alpha]_\wedge^2 & \text{si } d \in (-1,1) \end{cases} \end{aligned} \tag{1.49}$$

$$\begin{aligned} &\log_{SO(3)} : SO(3) \to \mathfrak{so}(3) \\ &\log_{SO(3)}(R) = \begin{cases} \frac{1}{2}(R - R^T) = 0 & \text{si } d \to 0 \\ \frac{arccos(d)}{2\sqrt{1-d^2}}(R - R^T) & \text{si } d \in (-1,1) \end{cases} \end{aligned} \tag{1.50}$$

où $\theta = ||\alpha||_2$ et $d = 0.5(\mathtt{trace}(R) - 1)$. En notant $w = (\alpha_1, \alpha_2, \alpha_3)^T$ et $v = (\alpha_4, \alpha_5, \alpha_6)^T$, Les fonctions exponentielles et matricielles correspondants à $SE(3)$ peuvent être exprimées par ([112], page 127, en posant $\sigma = 0$ et $s = 1$ selon leurs notations) :

$$\exp_{\mathfrak{se}(3)}(v, w) = \begin{pmatrix} \exp_{\mathfrak{so}(3)}([w]_\wedge) & Wv \\ 0 & 1 \end{pmatrix} \tag{1.51}$$

$$\log_{SE(3)} \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \log_{SO(3)}(R) & W^{-1}T \\ 0 & 1 \end{pmatrix} \tag{1.52}$$

avec

$$\frac{(1 - cos(\theta))}{\theta^2}[w]_\wedge - \frac{sin(\theta)}{\theta^3}[w]_\wedge^2 \tag{1.53}$$

et $\theta = ||w||_2$ cette fois-ci.

1.4.3 Dérivées en un point quelconque

Soit $G \subset GL(n)$ un groupe de Lie dont l'algèbre est notée $\mathfrak{g}$, et dont la paramétrisation minimale est donnée par

$$\vee : \mathfrak{g} \to \mathbb{R}^k \tag{1.54}$$

Nous cherchons à dériver un élément $M = exp(\sum_{i=1}^{k} \alpha_i G_i)$ de G par rapport aux paramètres de $\mathbb{R}^k$, *i.e.* nous cherchons à trouver les éléments de l'espace tangent à M. Pour cela, nous pouvons choisir des chemins,

$$exp(\alpha_i G_i) M \tag{1.55}$$

passant par M. Les dérivées de ces chemins

$$\frac{\partial exp(\alpha_i G_i) M}{\partial \alpha_i} = G_i M \tag{1.56}$$

fournissent une base de $\mathfrak{g}$.

1.5 Réseaux de neurones profonds et réseaux convolutionnels

Les réseaux de neurones profonds sont aujourd'hui omniprésents dans l'état de l'art de la plupart des domaines de recherche appliquée, c'est pourquoi nous nous contenterons d'un rappel très succinct des idées de base d'un point de vue applicatif. Un réseau de neurones peut être appliqué à plusieurs problèmes : il peut s'agir d'un outil pour approximer une fonction non-triviale classifiant un signal (*e.g.* pour la classification d'images), ou afin de trouver une représentation compacte et efficace de celui-ci (*e.g.* autoencodeurs). Dans ce texte, nous serons intéressés par la première utilisation. En particulier, nous serons motivés par les besoins suivants :

- ▷ La représentation d'une fonction non-triviale (*e.g.* associant chaque pixel d'une image à un label) par une cascade de compositions, *e.g.* $f \approx f_0 \circ f_1 ... \circ f_n$.
- ▷ L'inférence et l'exploitation de représentations intermédiaires appropriées pour la tâche, plutôt que l'utilisation de représentations définies manuellement.
- ▷ La prise en compte de l'information contextuelle dans les images
- ▷ Des temps de calculs et une complexité en mémoire raisonnables

Nous définissons donc notre approximation de la fonction non-triviale par

$$\begin{aligned} &f : \mathcal{D} \times \mathcal{W} \to \mathbb{R} \\ &f = f_0 \circ f_1 \circ ... \circ f_n \end{aligned} \tag{1.57}$$

où

$$\begin{aligned} &f_0 : \mathcal{D} \times \mathcal{W}_1 \to \mathbb{R}^{k_0} \\ &f_i : \mathrm{Im}(f_{i-1}) \times \mathcal{W}_i \to \mathbb{R}^{k_i} \end{aligned} \tag{1.58}$$

Dans ces équations, $\mathcal{D}$ désigne l'ensemble des données (*e.g.* des images) et $\mathcal{W}$ est un ensemble de paramètres, que l'on souhaite apprendre. L'ensemble des $\mathcal{W}_i$ partitionne $\mathcal{W}$, *i.e.*

$$\begin{aligned} &\mathcal{W} = \bigcup_{i=1}^{n} \mathcal{W}_i \\ &i \neq j \Rightarrow \mathcal{W}_i \cap \mathcal{W}_j = \varnothing. \end{aligned} \tag{1.59}$$

Les f_i sont les *couches* du réseau, et les $\mathcal{W}_i$ sont les *poids* qui leur correspondent. Une fois l'apprentissage effectué, les sorties des f_i pourront être interprétées comme des représentations intermédiaires des données (elles pourront par exemple être exploitées pour la mise en correspondance d'images, *e.g.* [25]). Afin de répondre aux deux autres besoins mentionnés plus haut, c'est à dire la prise en compte de l'information contextuelle ainsi que les gains en temps de calcul et en mémoire, nous exploiterons en particulier des réseaux de neurones convolutionnels. Il s'agit de réseaux de neurones dont certaines couches retournent en sortie le résultat de la convolution entre leurs entrées et un noyau de petite taille (*e.g.* de taille 7×7) dont les valeurs

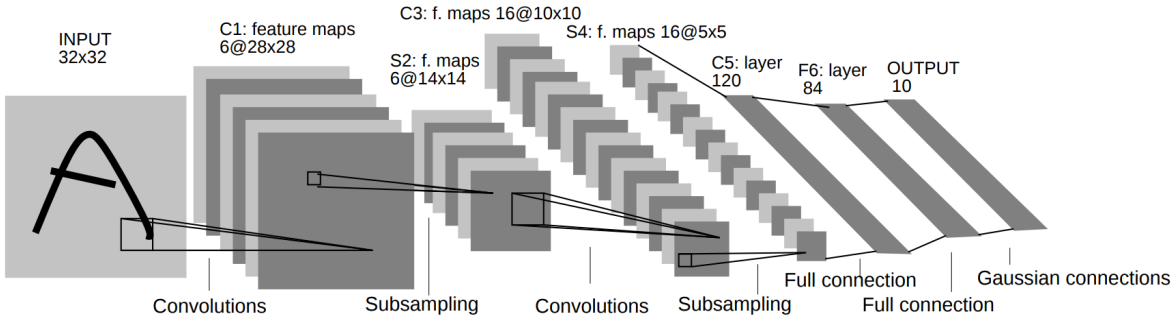


FIGURE 1.5 – Exemple d'architecture de base de réseau convolutionnel (LeNet, [63])

sont apprises. Ces sorties seront ensuite échantillonées par des couches de *Pooling* [6]. Comme ces deux actions peuvent s'écrire comme une succession de transformations linéaires, il s'ensuit que sans l'introduction de couches non linéaires, l'approximation entière f ne sera elle-même qu'une succession de produits matriciels et donc une transformation linéaire. C'est pour cela que les couches de pooling sont suivies de non-linéarités telles que des fonctions tanh, ReLU, PReLU, etc. La figure 1.5 montre un exemple d'architecture convolutionnelle de base (architecture LeNet, proposée dans les travaux de [63].). Notons que les convolutions sur des images sous-échantillonées permettent la prise en compte d'une information contextuelle, et que le fait que les couches de convolutions ne soient pas complètement connectées (contrairement aux couches linéaires) réduit le temps d'exécution et les exigences en terme de mémoire. Plusieurs architectures convolutionnelles ont été par la suite proposées, par exemple pour la classification d'images ([53]) ou la classification par pixel [72, 4, 95]. Comme notre objectif dans ce mémoire n'est pas d'améliorer les architectures existantes, mais plutôt d'explorer les possibilités d'intégration des techniques de deep learning dans un processus de SLAM afin de définir des contraintes géométriques robustes, nous ne nous attarderons pas d'avantage sur ce sujet. Pour plus d'information, le lecteur intéressé est invité à consulter à titre d'exemple [31, 1].

Dans ce qui suit, nous détaillerons premièrement l'algorithme de minimisation que nous utilisons pour optimiser les réseaux dans ce mémoire. Ensuite, nous décrirons de manière succincte certaines architectures, couches et stratégies permettant d'obtenir de meilleures propriétés de convergences, que nous avons utilisées dans nos travaux.

1.5.1 Algorithmes de minimisation pour réseaux de neurones

1.5.1.1 Descente de gradient stochastique

Les fonctions de coût des réseaux de neurones exploités dans le cadre d'apprentissages supervisés prennent souvent la forme d'une fonction

$$f : \mathcal{D}, \mathcal{W} \rightarrow \mathbb{R} \tag{1.60}$$

où $\mathcal{W}$ désigne l'ensemble des poids du réseau, et où $\mathcal{D}$ désigne l'ensemble des données d'apprentissage (qui seront constantes durant l'entraînement) :

$$\mathcal{D} = \{T_1, ..., T_n\} \tag{1.61}$$

6. Le *pooling* fait simplement référence au sous-échantillonage d'une matrice ou d'un tenseur (*e.g.* matrice/tenseur représentant l'image ou la sortie d'une couche intermédiaire du réseau.). Le lecteur intéressé trouvera aisément des information à ce sujet dans des textes d'introduction à l'apprentissage profond tels que [31]

avec chaque T_i un tenseur de dimension quelconque. La fonction f peut dans la majorité des cas se décomposer en une somme de plusieurs fonctions

$$\begin{aligned} f(\mathcal{D}) &= \sum_{i=1}^{k} f_i(\mathcal{D}_i) \\ f_i : \mathcal{D}_i, \mathcal{W} &\to \mathbb{R} \end{aligned} \tag{1.62}$$

où chaque $\mathcal{D}_i \subset \mathcal{D}$ est un mini-batch. La complexité en temps de calcul de l'évaluation de $\frac{\partial f}{\partial W}$ pour effectuer une étape de descente de gradient classique étant trop élevée pour être envisageable en pratique, il est indispensable de se tourner vers des approximations, dont la descente de gradient stochastique (DGS) est un exemple. On procédera dans ce cas à un échantillonnage pour obtenir des minibachs $\{\mathcal{D}_1, ..., \mathcal{D}_k\}$, et l'on mettra récursivement à jour les paramètres de la manière suivante :

$$\mathcal{W}_t = \mathcal{W}_{t-1} - \eta \frac{\partial f_i(\mathcal{D}_i)}{\partial \mathcal{W}} \qquad (\forall i, \text{de manière séquentielle}) \tag{1.63}$$

avec η le pas de la descente. Il est donc évident que la minimisation sera grandement influencée par les données d'apprentissage. Un paramètre $\mathcal{W}_j$ pourra influencer une grande majorité des f_i et donc être optimisé plusieurs fois, alors qu'un paramètre $\mathcal{W}_l$ n'intervenant que sur peu de mini-batchs ne sera que rarement mis à jour. La DGS n'est donc efficace que sur les ensembles équilibrés. En outre, la directions dans laquelle on se déplace dans l'espace des paramètres change avec chaque mini-batch, c'est pourquoi la DGS zigzague avant de converger.

Un certain nombre de travaux tentent de pallier ces difficultés en pondérant les paramètres en fonction de leurs fréquences et en approximant le gradient de f, ainsi que d'autres variables, par leurs espérances. Celle-ci est en général approximée par une moyenne sur une fenêtre temporelle, ce qui est un cas particulier d'optimisation par momentum ou inertie ([99]). Dans ce mémoire, nous ne nous pencherons que sur Adam ([50]). Nous expliciterons les raisons de ce choix dans la section suivante, qui lui est consacrée.

1.5.1.2 Adam

Cette méthode peut être vue comme une amélioration de AdaGrad ([16]) et RmsProp[7]. AdaGrad utilise comme pondération la racine carrée de l'inverse de la matrice de covariance diagonale des paramètres, mais effecture ce calcul sur l'ensemble des données, ce qui fait rapidement tendre la covariance vers l'infini, et par conséquent la pondération vers zéro. Ceci entraîne l'arrêt de l'opimisation avant la convergence. RmsProp pondère le gradient par sa variance sur une fenêtre temporelle, mais ne calcule pas l'espérance du gradient. Cela entraîne plus souvent une convergence plus lente en comparaison avec les algorithmes utilisant une inertie ([99]). En revanche, Adam ([50]) conserve une moyenne du gradient sur une fenêtre temporelle (*i.e.* elle conserve l'inertie du gradient précédent) et aussi une moyenne, sur la même fenêtre, de la variance centrée de celui-ci. En outre, cette méthode permet de résoudre le problème du biais de l'initialisation qui peut être très prononcé au début de la minimisation (cela est dû à l'effet de l'initialisation sur la moyenne fenêtrée). C'est pour ces raisons que nous avons choisi cette méthode, dont les performances en pratique ont été très satisfaisantes.

7. RmsProp est une méthode non-publiée, proposée par la première fois par Geoff Hinton : http ://www.cs.toronto.edu/ tijmen/csc321/slides/lecture_slides_lec6.pdf

Dans le reste de cette section, les mutliplications/divisions et additions/soustractions se font élément par élément, et H^2 désignera la multiplication élément par élément des entrées de H. Les moyennes du gradient et de sa variance centrée sont données par les tenseurs

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial f_{t-1}}{\mathcal{W}} \tag{1.64}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\frac{\partial f_{t-1}}{\mathcal{W}})^2 \tag{1.65}$$

de la même taille que les $\mathcal{W}_i$. Les coefficients β_1 et β_2 sont choisis dans $[0,1]$. L'initialisation a pour effet l'introduction d'un coefficient multiplicatif dans ces moyennes, il est donc important pour l'éliminer de mettre à jour ces variables comme suit :

$$m_t/ = 1-\beta_1^{t-1} \tag{1.66}$$

$$v_t/ = 1-\beta_2^{t-1} \tag{1.67}$$

Les paramètres à l'instant t (*i.e.* lors de l'évaluation correspondant au mini-batch $\mathcal{D}_i$) sont alors mis à jour par

$$\mathcal{W}_t = \mathcal{W}_{t-1} - \frac{\eta}{\sqrt{v_t}+\epsilon} m_t \tag{1.68}$$

Notons que AdaDelta ([134]) est une méthode proche de Adam, qui élimine la nécessité de définir le pas η, mais qui ne prend pas en compte le biais de l'initialisation.

1.5.2 ResNet

Cette architecture couramment utilisée et publiée par [37] est motivée par l'observation suivante. Supposons que l'on entraine un réseau profond dont les couches ont une architecture naïve, *i.e.* séquentielle et sans aucun raccourci d'une couche à l'autre. Ensuite, supposons que l'on augmente sa taille de manière significative et que l'on l'entraîne pour la même tâche. Dans ce cas, le deuxième réseau sera en général bien moins précis que le premier en pratique. Cela est contre-intuitif, car nous pouvons construire manuellement le deuxième réseau de manière à ce qu'il produise les mêmes résultats que le réseau moins profond : il suffit pour cela de concaténer le premier réseau et un nombre de couches identité jusqu'à atteindre la profondeur désirée. Compte tenu de cela on peut soupçonner que l'algorithme de minimisation et le réseau ont des difficultés à approximer la fonction identité.

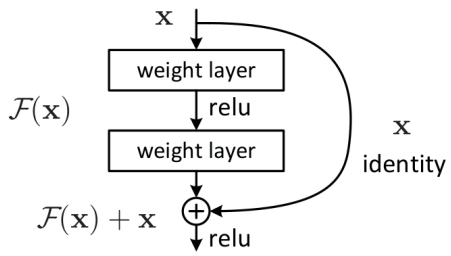


FIGURE 1.6 – Architecture ResNet.

Partant de ce constat, [37] ont proposé l'architecture présentée dans la figure 1.6. Au lieu d'apprendre la fonction $\mathcal{F}(x)$ que l'on désire idéalement approximer, la branche principale approxime $\mathcal{F}(x)-x$, et un raccourci entre les entrées/sorties de cette branche permet de retrouver

$\mathcal{F}(x)$ via une simple addition. Il a été expérimentalement vérifié que dans le cas où l'identité serait plus convenable, le réseau peut plus facilement faire tendre $\mathcal{F}(x)-x$ vers 0 que si il devait apprendre l'identité en la modélisant sans le raccourci.

1.5.3 Batch normalisation

La distribution des entrées de chaque couche affecte l'entraînement du réseau. En particulier, le changement de ces distributions à l'issue de chaque itération de descente de gradient, appelé *décalage covariant* (*covariate shift* en Anglais) ralentit l'apprentissage, et en particulier, peut avoir pour résultat l'annulation des gradients lorsque l'activation utilisée peut être saturée (*e.g.* sigmoïd ou tanh). La *Batch normalization*, proposée par [41], permet de normaliser les entrées des couches sensibles, en garantissant une moyenne nulle et une covariance unitaire pour chaque scalaire passé en entrée à celles-ci. Les statistiques sont calculées sur un mini-batch. Ces modules de normalisation, qui réduisent la sensibilité à l'initialisation des poids ainsi que le nombre d'itérations requises pour converger sont considérés comme des outils standards, et sont implémentées dans toutes les librarairies d'apprentissage profond de l'état de l'art (*e.g.* torch7, caffe, TensorFlow, theano, etc). Nous exploiterons ces modules de manière sporadique sans leur apporter de modifications, c'est pourquoi nous ne nous lancerons pas dans une digression sur les détails d'implémentation. Nous renvoyons le lecteur intéressé à la publication originale de [41].

1.5.4 Dropout

Cette technique, introduite par [111], empêche l'*overfitting*. Le principe est simple : il s'agit de sélectionner un certain nombre de neurones selon une loi de probabilité lors de l'apprentissage, et de mettre leur gradient à zéro. Cela simule l'entraînement de plusieurs réseaux indépendants dont on ferait la moyenne des résultats lors de l'évaluation. Notons que la mise à l'échelle des paramètres est importantes, et qu'il est préférable de l'effectuer à l'entraînement afin de réduire les temps de traitements. Pour plus de détails, le lecteur intéressé pourra consulter [111].

1.6 Quelques distributions et fonctions utilisées dans le mémoire

1.6.1 Distribution Bêta

La loi Bêta donnée par

$$\mathrm{Beta}(\alpha,\beta,x)=\frac{1}{\mathcal{B}(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1} \tag{1.69}$$

où $\mathcal{B}(\alpha,\beta)$ est la fonction Bêta décrit une famille de distributions continues sur $[0,1]$. Nous l'utiliserons au chapitre 4 afin de modéliser la vraisemblance de l'appartenance d'un point $3d$ à un plan, en fixant α et β de manière à ce que la distribution résultante corresponde à nos attentes.

1.6.2 Distribution de Dirichlet

La distribution de Dirichlet de paramètres $\alpha = \{\alpha_1, ..., \alpha_k\}$, souvent présentée comme une distribution continue de distributions discrètes, est une généralisation de la loi Bêta au cas multi-dimensionel :

$$\mathfrak{D}(x_1, ...x_k, \alpha_1, ...\alpha_k) = \frac{1}{\beta(\alpha)} \prod_{s=1}^{k} x_s^{\alpha_s - 1} \tag{1.70}$$

Dans cette équation, β désigne la fonction Bêta multinomiale, et les x_i doivent vérifier les deux conditions suivantes :

- $x_i \geq 0 \quad \forall i$
- $\sum_{i=1}^{k} x_i = 1$

On voit donc que le vecteur $(x_1, ..., x_k)^T$ peut être interprété comme une distribution discrète de probabilité d'une variable aléatoire $X = 1, ..., k$. Dans le chapitre 4, nous modéliserons la distributions continue d'un ensemble de distributions conditionnelles discrètes sous forme d'un processus de Dirichlet.

1.6.3 Fonction Softmax

La fonction softmax, donnée par

$$\text{Softmax}(\mathcal{D}, x) = \frac{e^x}{\sum_{y \in \mathcal{D}} e^y} \tag{1.71}$$

sera utilisée dans ce mémoire non seulement comme fonction de coût pour la classification, mais aussi dans le but de modéliser la probabilité de la correspondance entre un point et un plan sachant la distance qui les sépare. Dans l'équation 1.71, $\mathcal{D}$ est une ensemble fini de réels, et $x \in \mathcal{D}$.

1.6.4 Divergence de Kullback-Leibler

La divergence de Kullback-Leibler est définie par

$$KL(P||Q) = \sum_k P(k) \log \frac{P(k)}{Q(k)} \tag{1.72}$$

dans le cas discret et par

$$KL(P||Q) = \int_{-\infty}^{+\infty} P(k) \log \frac{P(k)}{Q(k)} dk \tag{1.73}$$

dans le cas continue.Intuitivement, elle mesure la quantité d'information perdue si l'on utilise la distribution Q afin de résoudre un problème qui est correctement modélisé par une distribution P. Bien que cette divergence puisse être vue comme une mesure de similarité, elle ne vérifie pas l'inégalité triangulaire et n'est pas symétrique. Ce n'est donc pas une distance.

1.7 Graphe de poses

1.7.1 Le formalisme

Un graphe de poses décrit la trajectoire parcourue par la caméra en faisant abstraction des éléments de la scène, via un formalisme emprunté à la théorie des graphes. On définit un graphe par $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ où $V_\mathcal{G}$ est un ensemble de "vertexes", et chaque élément de $E^{ij}_\mathcal{G} \in E_\mathcal{G}$ est une paire d'éléments de $V_\mathcal{G}$, *e.g.*

$$E^{ij}_\mathcal{G} = (V^i_\mathcal{G}, V^j_\mathcal{G}) \tag{1.74}$$

où les superscripts i et j correspondent aux $i-$ème et $j-$ème éléments de $V_\mathcal{G}$. On appellera chaque élément de $E_\mathcal{G}$ une arrête du graphe. Dans le cas où l'ordre du couple $(V^i_\mathcal{G}, V^j_\mathcal{G})$ est important, on parlera de graphe orienté. Nous sommes intéressés par des graphes non-orientés, par conséquent, dans la suite de ce mémoire, nous noterons de manière interchangeable

$$E^{ij}_\mathcal{G} = (V^i_\mathcal{G}, V^j_\mathcal{G}) = (V^j_\mathcal{G}, V^i_\mathcal{G}) = \{V^i_\mathcal{G}, V^j_\mathcal{G}\} \tag{1.75}$$

Dans le cadre du SLAM, un graphe de pose est généralement défini en posant :

$$\begin{aligned} V_\mathcal{G} &= \text{Ensemble de poses des caméras} \\ E_\mathcal{G} &= \text{un sous ensemble de } \{(V_i, V_j) | V_i, V_j \in V_\mathcal{G}\} \end{aligned} \tag{1.76}$$

et en définissant une fonction

$$\begin{aligned} &\mathfrak{E} : E_\mathcal{G} \to \mathbb{R}^{3\times 4} \\ &\mathfrak{E}(E_{\mathcal{G}^{ij}}) = S_{ij} \end{aligned} \tag{1.77}$$

associant chaque arrête du graphe à une pose relative. En notant (R_i, T_i) la matrice 3×4 de la pose de la caméra i dans un repère Euclidien, et (R_j, T_j) la pose de la caméra j dans le même repère, il est évident que le passage du repère j vers le repère i se fait via

$$X_i = R^T_i(R_j X_j) + R^T_i(T_j - T_i) \tag{1.78}$$

Donc, la pose relative entre les deux caméras est $(R^T_i R_j \ \ , \ \ R^T_i(T_j - T_i))$. Par métonymie, $E^{ij}_\mathcal{G}$ et $\mathfrak{E}(E^{ij}_\mathcal{G})$ sont souvent utilisés de manière interchangeable.

Concernant le choix de $E_\mathcal{G}$, il s'agit en général d'un graphe éparse, où les caméras voisines sont les caméras vérifiant un certain critère de covisibilité (*e.g.* suffisamment d'observations de points $3d$ en commun), ou simplement les caméras successives. Dans ce mémoire, nous travaillerons avec des graphes définis sur des caméras successives.

1.7.2 Pourquoi utiliser un graphe de poses ?

Dans la littérature, un graphe de pose est un moyen efficace (en terme de temps de traitement) permettant d'apporter des corrections globales à l'estimation de la trajectoire sans perturber les estimations locales de manière significative. L'efficacité provient du fait que ce formalisme permet de remplacer les représentations de l'environnement par des contraintes inter-caméra (les poses relatives). Par exemple, supposons que l'on ai obtenu un ensemble de poses

$\{(R_1, T_1), ..., (R_l, T_l)\}$ correspondant à des caméras $1, ..., l$ à l'issue d'une odométrie visuelle. En général, l'estimation sera fiable localement, mais aura dérivé de manière progressive sur l'ensemble de la trajectoire. On cherchera donc à corriger les poses dans le repère global sans dégrader les relations locales entre elles. Plus précisément, on cherchera à obtenir, pour une caméra i, une pose corrigée $(\tilde{R}_i, \tilde{T}_i)$, de manière à ce que pour toute caméra k voisine de i dans le graphe, la pose relative entre k et i ne soit pas dégradée, *i.e.*

$$(R_i^T R_k \ , \ R_i^T(T_k - T_i))^{-1}(\tilde{R}_i^T \tilde{R}_k \ , \ \tilde{R}_i^T(\tilde{T}_k - \tilde{T}_i)) \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.79}$$

ce qui en utilisant la paramétrisation minimale donnée par $SE(3)$, peut s'écrire

$$\log_{SE(3)}((R_i^T R_k \ , \ R_i^T(T_k - T_i))^{-1}(\tilde{R}_i^T \tilde{R}_k \ , \ \tilde{R}_i^T(\tilde{T}_k - \tilde{T}_i)))^{\vee} \sim \mathbf{0} \tag{1.80}$$

Dans le cadre de cette thèse, on utilisera la paramétrisation par $SE(3)$. Les travaux de [112] indiquent que dans le cadre d'une fermeture de boucle monoculaire, il est plus avantageux de choisir $\mathcal{S} = SIM(3)$. Nous justifions notre choix par le fait que nous n'utilisons le formalisme du graphe de pose qu'en conjonction avec des données GPS, qui nous fournissent une information implicite sur l'échelle (3.3). De plus, notre utilisation est incrémentale, et non pas limitée à des fermetures de boucles ponctuelles (Voir le chapitre 3).

1.8 Visualisations

1.8.1 T-SNE

La méthode SNE (*distributed Stochastic Neighbour Embedding*, [38]) est une méthode permettant d'obtenir une représentation d'une varitété différentielle de haute dimensionalité dans un espace à dimensionalité réduite en conservant la structure locale de celle-ci. Une telle représentation s'avère particulièrement utilise lorsque l'on souhaite visualiser la variété en question en $2d$ ou $3d$ afin de développer une intuition concernant les problèmes associés. L'algorithme SNE repose sur la représentation de la structure locale sous forme de distributions conditionnelles : on cherche à définir un voisinage pour chaque point de la variété, en posant l'hypothèse que la probabilité du choix d'un voisin x_j d'un point x_i est proportionnelle à la probabilité suivant de x_j donnée par une Gaussienne centrée sur x_i. Plus formellement, on définit la probabilité conditionnelles $P_{j|i}$ par

$$P_{j|i} = \frac{exp(-||x_j - x_i||^2/2\sigma_i^2)}{\sum\limits_{k \neq j} exp(-||x_k - x_j||^2/2\sigma_i^2)}. \tag{1.81}$$

Notons l'importance de la normalisation : elle permet de définir les voisinage de manière relative, c'est à dire non seulement en fonction de la Gaussienne mais aussi en fonction des distances relatives entre les points. On note par exemple que s'il n'existe qu'un unique point j avec une probabilité infinitesimale selon la Gaussienne, celui-ci sera choisi comme voisin de i avec une probabilité de 1 grâce à la normalisation. On définit ensuite une distribution dans l'espace de dimensionalité réduite

$$P_{j|i} = \frac{exp(-||y_j - y_i||^2)}{\sum\limits_{k \neq j} exp(-||y_k - y_j||^2)}. \tag{1.82}$$

où y_i est le point de dimensionalité réduite correspondant à x_i. Finalement, la minimisation de la divergence de Kullback-Leibler (§1.6.4) permet d'obtenir les y_i.

Ainsi que le soulignent [76], cette méthode présente deux inconvénients : l'optimisation de sa fonction de coût est difficile, et la représentation obtenue à tendance à souffrir d'un problème d'agglomération des points similaires. La solution que propose les auteurs de ce papier, nommée T-SNE (pour *T-distributed Stochastic Neighbour Embedding*), que nous utiliserons dans nos travaux, permet de pallier ces problèmes. La solution repose sur l'utilisation d'une fonction de coût symétrique, qui minimise la divergence de Kullback-Leibler entre des distributions jointes, ainsi que sur l'utilisation d'une distribution $t - student$ afin de définir les probabilités conditionnelles dans l'espace à dimensionalité réduite. Nous ne nous attarderons pas d'avantage sur les détails de cette méthode, car nous ne l'exploitons que dans le but de visualiser les représentations intermédiaires produites par les couches de nos réseaux. Le lecteur intéressé pourra consulter la publication originale de [76].

1.8.2 graphe de facteurs

Formellement, le terme *graphe de facteurs* ne désigne pas une représentation, mais un graphe biparti représentant la factorisation d'une distribution. Soit une distribution

$$p(y_1, ..., y_n) = Z^{-1} \prod_a \Psi_a(Y_a) \tag{1.83}$$

où $Y_a \subseteq \{y_1, ..., y_n\}$, et $\Psi_a \geq 0 \ \ \forall a$. La variable $Z = \sum_y \prod_a \Psi_a(y)$ n'est qu'un facteur de normalisation. Dans ce cas, on peut définir le graphe de facteurs correspondant à cette factorisation de la distribution comme le graphe biparti dont l'ensemble de sommets est donné par l'union des Ψ et des y_i, et dont l'ensemble d'arrêtes est défini par la règle suivante : il existe une arrête d'un sommet Ψ_a vers un sommet y_i si et seulement si $y_i \in Y_a$ (*i.e.* si la fonction Ψ_a dépend de la variable aléatoire y_i. Afin de visualiser ce graphe, la convention suivante est souvent utilisée dans la littérature : chaque fonction Ψ_a est représentée par un rectangle et chaque variable y_i est représentée par une forme géométrique distincte, *e.g.* par des cercles. La figure 1.7 montre un exemple de graphe de facteurs. Dans la littérature, le terme graphe de facteurs désigne souvent par métonymie aussi bien la formalisation que la représentation visuelle. Pour plus d'informations à ce sujet, voir les travaux de [116]. Dans cette thèse, nous n'utiliserons ce concept qu'a fin de visualiser les graphes des problèmes que nous traitons.

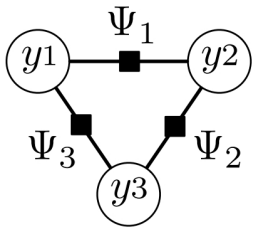


FIGURE 1.7 – Exemple de graphe de facteurs, représentant la distribution $Z^{-1}\Psi_1(y_1, y_2)\Psi_2(y_2, y_3)\Psi(y_1, y_3)$ où Z est le facteur de normalisation approprié (Figure tirée de [116]).

1.9 Données géoréférencées

Comme il a été indiqué à la section 1.3.4, le repère arbitraire dans lequel est exprimée la reconstruction d'un SLAM basé vision n'est que de peu d'utilité dans le cas d'applications de conduite autonome ou de réalité augmentée. Dans cette thèse, nous sommes donc intéressés par une localisation géoréférencée, que nous obtenons via la fusion entre le SLAM visuelle et différents types de données. Avant que nous puissions les détailler, une digression sur les différents systèmes de coordonnées employés dans les systèmes d'information géographique s'impose.

Comme nous le savons, la terre est souvent modélisée par une éllipsoïde applatie. Un modèle d'éllipsoïde standard utilisé est le WGS84 ([45]). Comme de telles approximations induisent des inexactitudes plus ou moins prononcées en fonction de l'emplacement sur terre, des modèles d'éllipsoÏde ont été developpés qui minimisent les erreurs de modélisation pour un pays ou une région spécifique. En France, le système géodésique standard est le RGF93 [8] depuis 2001. Comme la localisation et le calcul des distances entre deux points en coordonnées élliptiques se font de manière moins triviale que dans un repère Euclidien, il est de transformer ces coordonnées via une projection planaire. On notera que comme la projection d'un ellipsoïde vers un plan n'est pas possible sans distorsion, chaque projection planaire ne convient que pour le voisinage d'un certain point. En France, on utilisera la projection Lambert93 [9]. C'est dans le repère résultant de cette projection que nous souhaitons exprimer notre reconstruction.

Nous exploiterons plusieurs données géoréférencées, que nous présentons par la suite.

1.9.1 Modèles $3d$ non-texturés de bâtiments

Les modèles $3d$ de bâtiments que nous utilisons sont fournis par l'IGN (Institut national de l'information géographique et forestière), et facilement accessibles [10]. Leur caractéristiques les plus importantes sont les suivantes :

- ▷ Ils sont non-texturés
- ▷ Leur approximation de la géométrie des bâtiments est grossière : chaque façade est représentée par un plan, les détails liés à l'architecture ainsi que l'emplacement des portes et fenêtres sont perdus.
- ▷ Leur précision en hauteur n'est pas garantie. Il s'agit donc, en ce sens, plus de modèles 2.5 que nous désigons par le terme $3d$ par simple abus de langage.
- ▷ Leur géométrie est imprécise en plus d'être simplifiée : la création à grande échelle de ces modèles repose sur des algorithmes exploitant le plus souvent des images satellitaires (*e.g.* [17]) , qui peuvent produir des erreurs et des résultats aberrants. Dans le cas des données que nous utilisons, l'erreur maximum dans le plan horizontal est d'environ $2.5m$, mais il arrive que des problèmes liés à la mise à jour de la base (*e.g.* destruction ou construction d'un ensemble de bâtiments) introduisent des aberrations plus importantes de manière temporaire.
- ▷ Il sont exprimés dans le repère Lambert93.

Un exemple de ces modèles est donné par la figure 1.8.

8. http ://geodesie.ign.fr/index.php ?page=rgf93
9. http ://geodesie.ign.fr/contenu/fichiers/documentation/rgf93/Lambert-93.pdf
10. www.ign.fr

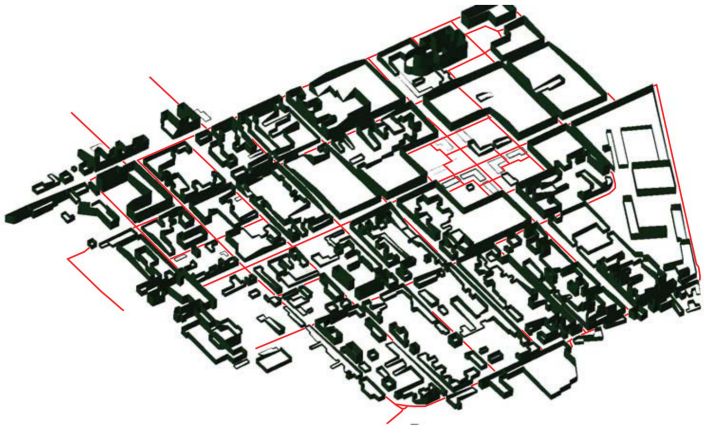


FIGURE 1.8 – Les modèles $3d$ des bâtiments non-texturés (en noir) et les modèles d'élévation de terrain (en rouge). Ces derniers, donnés sous forme de segments $2d$, ne fournissent pas d'autre information que l'altitude approximative de la route (Figure tirée de la thèse de [56]).

1.9.2 Modèles d'élévation de terrain

Les modèles d'élévation de terrain que nous exploitons correspondent à une estimation grossière de l'altitude des routes par des segments (figure 1.8) Ils ne fournissent donc pas d'information sur les variations d'altitude dans le sens de la largeur de la route, ni sur l'élévation des bâtiments ou terrains environnants. Ils sont aussi exprimés directement dans le repère Lambert93. L'erreur maximum de ces modèles est de $2.0m$ environ.

1.9.3 Le GPS

Le GPS (Global Positioning System) est un système de positionnement par satellite. D'autres systèmes de positionnement global tels que GLONASS, Galileo et Beidou-2 sont opérationnels ou en développement [11]. Le GPS a été developpé par le département de la défense des États-Unis à partir de 1973 et a été ouvert aux applications civiles en 2000 ([45]). Il repose sur au minimum 24 satellites dont les orbites sont définies de manière à ce qu'une constellation d'au moins quatre satellites soit à chaque instant visible à partir d'un point quelconque sur la surface de la terre. Dans le cadre de cette thèse, nous utilisons les données GPS par défaut, mais comme notre fusion VSLAM/GPS est relachée [12], nos travaux peuvent sans aucune modification être utilisés avec les résultats d'autres systèmes de localisation. Dans ce qui suit, nous décrirons de manière succincte le principe de la localisation satellitaire ainsi que le modèle de bruit et le capteur que nous utilisons.

11. GLONASS est un système Russe d'origine soviétique lancé en 1996, interrompu en vers la fin des années 1990 et restauré en 2010. Les systèmes Galileo et Beidou-2 sont respectivement développés par l'union Européenne et par la chine, et devront être opérationnels en 2020. D'autres systèmes non-globaux (*i.e.* assurant uniquement une couverture locale), tels que le système QZSS sont aussi en développement.

12. Une fusion relachée (*loose coupling* en Anglais) désigne une fusion de données effectuée après une estimation initiale de la pose ou de la position effectuée séparément pour chaque capteur. Par exemple, la fusion entre les résultats d'un VSLAM et les positions calculées par un GPS (*e.g.* [67]) est une fusion relachée. A l'opposé, l'estimation conjointe des pseudo-distances, le nuage de point et les poses ([18]) est une approche de fusion étroite (*tight fusion* en Anglais). Pour plus d'informations à ce sujet, nous invitons le lecteur à consulter [8].

1.9.3.1 Postionnement GPS

Comme nous connaissons la vitesse de la lumière, il suffit de connaître le temps de transfert du signal électromagnétique entre un satellite et le capteur afin de pouvoir calculer la distance qui les sépare. La distance entre un satellite et un capteur définit une sphère centrée à la position du satellite, sur la surface de laquelle se situe le capteur. Il nous faut donc en théorie trois satellites afin de pouvoir calculer la positions de capteur en trouvant l'intersection des trois sphères définies. Cependant, bien que chaque satellite soit équipé d'horloges atomiques, et que les horloges des satellites opérationnels soient synchronisées entre elles avec précision, ce n'est pas le cas du capteur GPS de l'utilisateur. Celui-ci possède bien sûr lui aussi une horloge, mais il s'agit le plus souvent d'une horloge bas coût (en comparaison avec les horloges atomiques) et son manque de précision est à l'origine d'un décalage. Ce décalage introduit une variable supplémentaire dans les calculs, ce qui augmente de un le nombre de degrés de liberté : on aura besoin d'un minimum de quatre satellites pour se localiser. Cette localisation peut se faire via de simples moindres carrés, dont les détails peuvent facilement être retrouvés sans la littérature (par exemple, voir le compendium de [125], ou le livre de [45]).

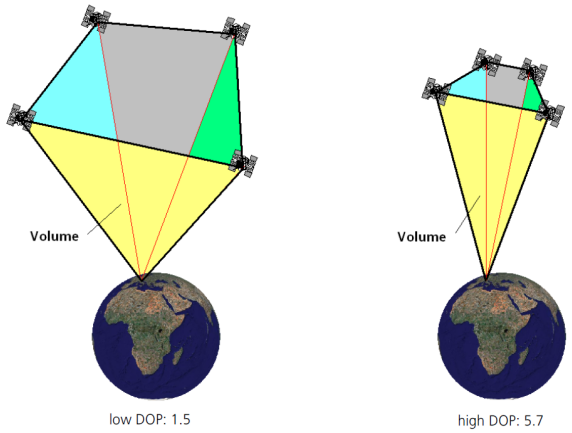


FIGURE 1.9 – La DOP est inversement proportionnelle au volume du polyèdre formé par le capteur les satellites (Figure tirée du compendium [125]).

1.9.3.2 Sources d'incertitudes et modèle de bruit

Plusieurs effets peuvent contribuer à l'erreur totale :

- ▷ Les effets ionosphériques et troposphériques.
- ▷ Les erreurs liées aux horloges des satellites. Une erreur de $10ns$ induit une erreur de positionnement d'environ $3.0m$.
- ▷ Les erreurs d'approximation des orbites des satellites.
- ▷ L'effet du multipath et du shadowing : cet effet est particulièrement présent dans les environnements urbains denses, bien qu'il soit observables lorsque les inégalités de terrain soient extrêmement prononcées. Dans un contexte urbain, les ondes électromagnétiques sont réfléchies par les façades des bâtiments. Lorsque les bâtiments on une hauteur considérable et que le signal ne peut atteindre le capteur en suivant une ligne droite, il devra parcourir une distance considérablement plus longue. L'estimation de la position se retrouve donc biaisée.

▷ Dilution de la précision (DOP) : la position est plus difficile à déterminer si les satellites utilisés sont proches les un des autres (voir la figure 1.9).

Pour plus d'information à ce sujet, nous invitons le lecteur à consulter le livre de [45] ou le compendium Ublox ([125]).

Aini qu'il a été démontré expérimentalement par [55], il est possible de modéliser la position calculée par le GPS par

$$x_{gps} = x_{true} + \mathcal{V}_{gps} + \epsilon_{gps} \tag{1.84}$$

où ϵ_{gps} est un bruit Gaussien de faible amplitude, et $\mathcal{V}_{gps}$ est un vecteur localement constant (en général sur quelques mètres).

1.9.3.3 Capteur utilisé

Le capteur utilisé dans le cadre de cette thèse est un GPS UBlox EVK-6PPP-0, caractérisé par :

▷ Une fréquence de localisation de 1 Hz.
▷ Une précision sur la position horizontale de 2.5m.
▷ Une précision sur la vitesse de 0.1m/s.
▷ Une précision sur le cap de 0.5 degré.
▷ Une altitude limite opérationnelle de 50km.
▷ Une vitesse limite opérationnelle de 500m/s.

Notons finalement que les données GPS du capteur sont exprimées dans le repère WGS84, que nous transformons en coordonnées RGF93 puis en Lambert93 dans une étape de pré-traitement.

CHAPITRE 2

État de l'art et positionnement

Dans ce chapitre, nous discuterons premièrement des différentes familles d'approches et de formulations du VSLAM (non-contraint et contraint) afin de positionner nos travaux, qui, bien que intégrés dans un système existant d'ajustement de faisceaux contraint basé images-clefs pour les besoins de ce mémoire, sont en partie indépendants des descripteurs et processus d'optimisation utilisés. Nos contributions sont axées autours de deux thèmes : l'amélioration du temps de calcul de la fusion vision/GPS par terme barrière, et l'exploration des possibilités d'exploitation des approches d'apprentissage profond en conjonction avec un SLAM, le plus souvent contraint aux modèles 3d des bâtiments. Nous préciserons donc l'apport de nos recherches par rapport aux travaux existants en lien avec ces domaines.

Par soucis de clarté, nous discuterons premièrement de l'état de l'art, et repoussons notre positionnement jusqu'à la section 2.5.

2.1 Les différentes formulations du SLAM

2.1.1 Filtrage et ajustement de faisceaux basé images-clefs

Les approches de SLAM visuel (contraints ou non contraints) de l'état de l'art, sauf exception (§2.1.2.1) reposent en général soit sur des variantes du filtre de Kalman (*e.g.* [69, 85, 75]), soit sur l'ajustement de faisceaux basé images clefs (*e.g.* [87, 86, 68, 66]) [1]. Il est donc pertinent qu'un état de l'art en lien avec le SLAM débute par une discussion sur ces méthodes et leur comparaison.

2.1.1.1 Les méthodes basées sur les filtre de type Kalman

Les approches basées sur les différentes version du filtre de Kalman considèrent le problème du SLAM comme l'estimation récursive de l'état optimal et de sa covariance à partir d'un en-

1. Les filtres à particules, que l'on pouvait encore sporadiquement rencontrer dans la littérature liée au SLAM visuel il y a quelques années (*e.g.* [109]), sont de plus en plus délaissés par les systèmes exploitant la vision à cause de leur exigence en terme de temps de calcul, et la difficulté de leur paramétrisation. Néanmoins, ils restent utilisés dans certains systèmes de localisation non-visuels, *e.g.* par [88].

semble de mesures (dans le cas du VSLAM, cela peut être les observations $2d$ d'un nuage de point). Notons au passage que contrairement à ce qui est quelquefois affirmé à tort dans la littérature, le filtre de Kalman ne requière aucune hypothèse sur la distribution de l'état, et qu'il est applicable dès lors que l'on considère que l'estimation optimale peut être exprimée comme un élément de la variété différentielle linéaire générée par l'ensemble des vecteurs d'observation ([44]). Le filtre de Kalman original ayant été developpé pour les systèmes linéaires, les variantes EKF/UKF ([121]) sont celles qui sont utilisées en pratique. Notons que ces approches conduisent à une estimation de l'état qui ne résulte que d'une unique linéarisation, ce qui peut conduire à une sensibilité excessive à l'incertitude ([121]). Il peut donc être pertinent d'utiliser un EKF itératif (*e.g.* [35]).

Les propriétés les plus importantes de ces filtres dans le cadre du SLAM peuvent être résumées ainsi :

▷ Quantification de l'incertitude (covariance globale)

▷ Possibilité de fusion de données venant de divers capteurs sans effort ou modifications

▷ Marginalisation (voir le dernier paragraphe de la section 1.3.4)

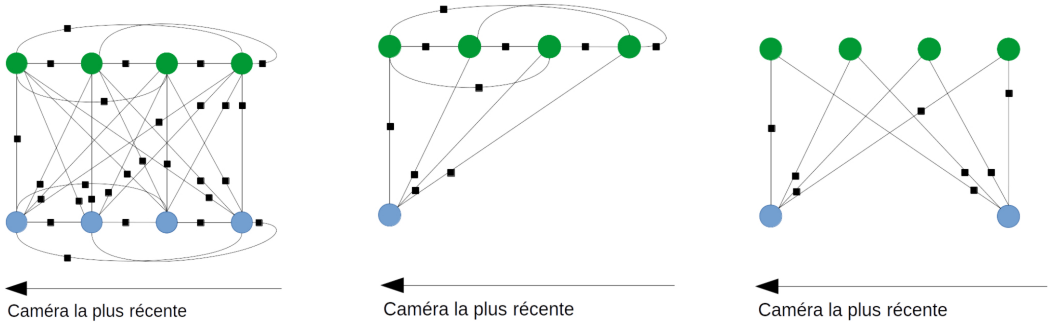
Le premier point découle de la définition du filtre. Le deuxième résulte du fait qu'il suffit de concaténer les vecteurs d'état de plusieurs capteurs/agents pour parvenir à faire la fusion entre les données en provenance de ceux-ci. Cette modularité est illustrée par [75]. On peut vérifier le dernier point en remarquant qu'un filtre de Kalman est un filtre de Bayes, et que l'étape de prédicion peut s'exprimer par

$$\begin{aligned} P(x_t|z_1,...,z_{t-1}) &= \int_{x_{t-1}} P(x_t, x_{t-1}|z_1,...,z_{t-1})dx_{t-1} \\ &= \int_{x_{t-1}} P(x_t|x_{t-1}, z_1, ...z_{t-1})bel(x_{t-1})dx_{t-1} \end{aligned} \tag{2.1}$$

Cette marginalisation a pour conséquence la densification du graphe de facteur (§1.8.2) du problème, car elle encode l'information des poses/points marginalisées dans de nouvelles dépendances entre les variables d'état (chapitre 1, section 1.3.3, ou encore [22]). Ceci conduit à une complexité en temps de calcul de $O(d^2)$ où d est le nombre de dimensions du vecteur d'état, ce qui réduit le nombre d'amers que l'on peut optimiser conjointement avec la pose. Notons cependant que le graphe de facteur est bien sûr moins dense que dans le cas du problème du SLAM complet (figure 2.1).

2.1.1.2 Ajustement de faisceaux basé images clefs

L'ajustement de faisceaux basé image-clef [84, 51], que l'on notera k-BA (acronyme Anglais de Key-frame Bundle Adjustment) dans la suite, optimise les poses des caméras et les positions des points 3d associés de façon à ce que la fonction 1.25 soit minimisée, le plus souvent en utilisant une variante de l'algorithme de Levenberg-Marquardt. Afin de limiter la redondance et de permettre une estimation en temps-réel, ces approches conservent uniquement une image I_k si celle-ci apporte suffisamment d'information par rapport à l'image clef précédente. Le choix de celles-ci peut se faire à partir de différents critères. Par exemple, on peut



(a) Graphe de facteurs du problème du SLAM non-simplifié. (b) Graphe de facteurs d'un EKF basique. (c) Graphe de facteurs du k-BA.

FIGURE 2.1 – Comparaison des graphes de facteurs du problème du SLAM, d'un EKF simple et d'un k-BA. Les points $3d$ sont représentés par les nœuds en verts, et les caméras sont représentées par les cercles bleus. Le graphe du k-BA est beaucoup plus éparse car contrairement au cas des approches de type Kalman, il ne contient pas de liens explicites entre les points ou les caméras.

considérer un seuil sur le nombre d'inliers dans le calcul de pose entre deux images consécutives, ou un seuil sur le nombre de points 3d en commun entre celles-ci. Afin de garantir le fonctionnement temps-réel, l'optimisation se fait uniquement sur un nombre réduit d'images clefs. L'ensemble de images clefs à optimiser peut être déterminé par un critère de covisibilité (*e.g.* [86]), ou de manière plus classique, correspondre à une fenêtre temporelle constituée uniquement de nombre N d'images clefs les plus récentes. On parle alors de k-BA local. Dans la suite de ce mémoire, par soucis de concision, nous omettrons le terme local. Sous la formulation k-BA, le graphe du problème du SLAM devient beaucoup plus parcimonieux (figure 2.1), et il est donc facilement possible d'utiliser des techniques d'optimisation non-linéaires sur de plus grandes fenêtres. Cependant, l'accès à la covariance globale n'est pas naturelle, car bien que la fenêtre d'optimisation soit liée aux anciennes caméras (fixes) par le biais des observations de certains points 3d, aucune propagation d'incertitude n'est effectuée entre la fenêtre optimisée au temps t et la fenêtre optimisée au temps $t+1$. On peut donc aisément obtenir une covariance locale via la backpropagation ([34]) de l'incertitude des observations utilisées dans la fonction de coût (1.3.3), mais la propagation d'une covariance globale requière de plus grands efforts ([21]). Notons que la complexité en terme de temps de calcul de l'ajustement de faisceaux (en notant respectivement p et c le nombre de points et le nombre de caméras optimisées) est de $O(c^3 + cp)$, ce qui rend le temps de calcul pour le k-BA approximativement linéaire en nombre de points $3d$.

2.1.1.3 Comparaison entre les deux familles d'approches

La comparaison entre ces deux catégories d'approches n'est certainement pas triviale, d'autant plus que les techniques utilisées par l'une ne sont pas incompatibles avec la méthodologie employée par l'autre. Par exemple, bien que les méthodes de k-BA aient initialement simplement ignoré l'information des anciennes poses, celles-ci sont le plus souvent marginalisées dans les systèmes récents, suivant le papier de [66]. D'autres travaux tels que C-KLAM ([90]) marginalisent en outre les informations en provenance des images non-clefs. De manière similaire,

[28] réduisent la complexité du temps de calcul du filtre de Kalmam, la rapprochant de celle, linéaire, du k-BA. De la même manière, le filtre de Kalman itératif se rapproche de la résolution des moindres carrés à l'aide d'un algorithme de Gauss-Newton itératif. A la connaissance de l'auteur au moment de la rédaction de ce mémoire, les seuls travaux publiés où les auteurs abordent le problème de la comparaison entre le k-BA et le filtrage avec rigueur sont ceux de [112, 114]. Bien que ceux-ci semblent indiquer une légère supériorité du k-BA par rapport à l'EKF (principalement en terme de compromis entre précision et temps de calculs), ces résultats doivent être interprétés avec circonspection. En effet, cette étude ne prend pas en compte les variantes itératives du filtre de Kalman, ni les problématiques liées à l'observabilité ([69]), et ne concernent que le VSLAM non-contraint. Ainsi que l'on peut le constater dans les travaux de [85, 69] sur la fusion vision/IMU, les contraintes supplémentaires fournies par la centrale inertielle permettent de ne pas inclure les points 3d dans le vecteur d'état, ce qui efface l'avantage du k-BA en terme de temps d'exécution. En outre, l'accès simple et rapide à la covariance que permet le filtre de Kalman en fait un choix naturel pour la fusion de données. Cependant, la distribution multi-modale des données d'un capteur peut rendre l'estimation de l'état optimal par une espérance et sa covariance inutile. De plus, ce type de fusion peut conduire à la perte de points $3d$ inliers, ce qui peut conduire à des difficultés de tracking. C'est le cas pour la fusion avec le GPS, dans ce cas, il est plus intéressant, d'un point de vue pratique, de choisir une approche non-probabiliste reposant sur une optimisation non-linéaire contrainte à l'aide d'un terme barrière ([67]).

2.1.2 Formulations directes, indirectes, denses, semi-denses et éparses

2.1.2.1 Méthodes directes/indirectes

Le terme "indirect" désigne l'ensemble d'approches qui calculent des représentations intermédiaires des données brutes (*e.g.* descripteurs SIFT, ORB, etc) qui sont ensuite utilisées comme paramètres dans des fonctions de coût le plus souvent géométriques (*e.g.* erreurs de reprojections, écart de flux optique, écart angulaire, etc). La plupart des méthodes de l'état de l'art (*e.g.*[66, 87, 85, 113]) font partie de cette catégorie. Par contraste, les approches directes s'alignent sur le point de vue (similaire à celui de l'apprentissage profond) qu'il est en général plus avantageux d'apprendre les descriptions/représentations utiles à la résolution du problème plutôt que de les définir manuellement. Par conséquent, ces méthodes définissent une fonction de coût prenant directement en compte les données des capteurs (*e.g.* temps d'exposition, irradiance, etc.) sans transformation intermédiaire. Parmi ces méthodes, on retrouve les travaux de [92, 20, 19]. En particulier, Direct Sparse Odometry (DSO) de [19] semble produire des résultats surpassant certaines méthodes indirectes/éparses telles que [86]. Notons cependant que les approches directes sont très dépendantes de la calibration photométrique, et que les imprécisions dans ce processus, outre les problèmes liés à une robustesse moins élevée face aux baselines de plus grande tailles (à cause de l'inconsistance photométrique), peuvent conduire à une sensibilité élevée au rolling shutter, à l'auto-gain et à l'auto-exposition ([86]). Les méthodes indirectes, de par leur exploitation de features par définition invariantes aux variations globales d'intensité et à plusieurs transformation géométriques, sont en principe plus robustes face au bruit ([122]).

2.1.2.2 Méthodes denses/parcimonieuses/semi-denses

Les méthodes denses sont celles qui représentent l'environnement sous forme d'ensembles de cartes de profondeurs (*e.g.* [92, 91, 130, 124]). En général, il s'agit de systèmes exploitant des capteurs supplémentaires (*e.g.* [91, 130, 124] prennent en entrée les données RGB-D d'une kinect.). Il existe cependant des méthodes telles que D-TAM ([92]) qui reconstruisent des cartes de profondeurs dense à partir d'images RGB monoculaire, en minimisant une erreur photométrique. Néanmoins, elles restent coûteuse et doivent être implémentées sur GPU. A l'opposé, on retrouve les méthodes éparses qui reposent sur des concepts plus classiques tels que l'extraction et la triangulation d'amers. La plupart des méthodes de l'état de l'art se rangent dans cette catégorie (*e.g.*[87, 113, 66]). Les méthodes semi-denses, quand à elles, représentent un compromis permettant l'exploitation d'une plus grande quantité d'information, notamment des zones où les variations d'intensité sont homogènes (murs, etc), sans augmenter la complexité en temps de traitement de manière considérable ([19, 20]).

2.2 VSLAM contraint

Le VSLAM monoculaire non-contraint souffre du problème bien connu de la dérive en sept degrés de liberté. Bien que certains travaux récents (*e.g.* [113, 108]) permettent quelque peu de réduire cette dérive, les améliorations apportées par ces méthodes ne suffisent pas dans le cadre du VSLAM à grande échelle où les contraintes inter-caméras sont rares et le plus souvent localisées sur une fenêtre temporelle (sauf lors des fermetures de boucles, qui elles aussi sont moins fréquentes qu'en navigation intérieure). Il est donc courant d'avoir recours à des capteurs dont les erreurs sont conditionnellement indépendantes. On peut par exemple envisager l'utilisation de capteurs tels que des accéléromètres et gyroscopes, généralement encapsulés dans une IMU ([69, 85, 66]), l'odométrie du véhicule ([94]), ou encore les boussoles magnétiques ([49]). Ainsi qu'il a été démontré par [118, 58, 59], il est aussi avantageux d'utiliser d'autres sources d'informations telles que les modèles $3d$ de bâtiments et les modèles d'élévation de terrain pour contraindre un SLAM à grande échelle en milieu urbain. D'autres travaux, tels que ceux de [71, 97] exploitent le modèle CAO des objects dans le cadre d'un SLAM en intérieur ayant pour objectif le tracking d'objet. Notons au passage que certaines applications nécessitent une localisation géo-référencée. Dans ce cas, les données GPS peuvent s'avérer utiles. Il est bien sûr possible d'utiliser des modèles 3d de bâtiments géo-référencés, ou encore de se relocaliser dans une base de points $3d$ apprise lors d'un passage antérieur ([60]).

2.2.1 Méthodes de fusion

Deux méthodes se concurrencent lorsqu'il s'agit de fusion entre la vision et d'autres capteur. La première, qui est aussi la plus répandue, est la fusion via les variantes du filtre de Kalman, en particulier de l'EKF. La deuxième est celle qui repose sur la fusion entre l'ajustement de faisceaux et différentes autres sources de données. Nous ne nous intéresserons pas à la fusion par filtre de Kalman dans ce tapuscrit, justifiant notre décision en reprenant les arguments présentés à la section 2.1.1.3. Nous invitons le lecteur intéressé à consulter, à titre d'exemple, les travaux de [26, 85, 69, 128]. En reprenant les notations du premier chapitre, et en notant $\zeta(\mathbf{X})$ une fonction exprimant la qualité des contraintes multivues issues de la vision (*e.g.* la somme des carrés des erreurs de reprojection), soient $C_1, ..., C_h$ les contraintes à respecter (*e.g.* contraintes GPS/INS ou encore modèles de bâtiments ou d'élévation de terrain).

La fusion entre k-BA et autres capteurs se fait en général soit

▷ Via la minimisation d'une fonction de coût linéaire (*e.g.* [118])

$$\underset{\mathbf{X}}{argmin} \quad \zeta(\mathbf{x}) + \lambda_1 C_1 + ... + \lambda_h C_h \quad \lambda_i \in \mathbb{R} \quad (\forall i) \tag{2.2}$$

▷ Via une optimisation par terme barrière ([6, 123, 67])

$$\underset{\mathbf{X}}{argmin} \quad (\frac{\gamma}{t - \zeta(\mathbf{x})} + C_1(\mathbf{x}) + ... + C_h(\mathbf{x})) \tag{2.3}$$

où $t, \gamma \in \mathbb{R}$. L'optimisation de l'équation 2.3 est initialisée avec le paramètre **X*** qui minimise ζ.

Les λ_i de l'expression 2.2 sont des coefficients de pondération. Notons au passage que sous hypothèse d'erreurs Gaussiennes, la minimisation de cette fonction de coût est équivalente à la maximisation de la vraisemblance de l'état sachant les observations. Par conséquent, la formulation du problème est la même que pour l'étape de mise à jour d'un EKF Gaussien. Le choix des λ_i est particulièrement difficile, car elle peut démultiplier le risque de perte de points $3d$ inliers que l'on encourt en choisissant ce type de fonction de coût (section 3.1 de [67]). En effet, dans le cas d'une contrainte C_i uniquement imposée sur les poses des caméras, il suffit d'un λ_i légèrement trop grand pour que les caméras soient déplacées sans que les points $3d$ ne le soient. Ceci aura pour conséquence l'augmentation de l'erreur de reprojection, et entraînera des difficultés dans le tracking. Pour ces raisons, nous choisissons d'utiliser l'optimisation par terme barrière.

L'optimisation par terme barrière (équation 2.3) n'a pas pour ambition de fournir un cadre probabiliste à la fusion. On pose l'hypothèse que la reconstruction dont la qualité est exprimée par ζ est plus fiable (de par son coté incrémental) que les contraintes (le plus souvent ponctuelles) imposées par les autres capteurs, et on la minimise séparément. Ensuite, la minimisation de la fonction de coût 2.3 aura pour effet, intuitivement, d'imposer les contraintes C_i tant que ceci ne détériorera pas ζ plus que le seuil imposé par t, fixé à un pourcentage de $\zeta(\mathbf{X}*)$. Suivant [67], nous définirons $t = 1.05\zeta(\mathbf{X}*)$ dans ce mémoire. Nous noterons cependant que cette fusion peut être problématique lors d'un déploiement sur des plate-formes de faible capacité. En effet, alors que 3 à 10 caméras suffisent dans un k-BA pur ou dans la fusion linéaire, il est nécessaire d'utiliser en pratique ~ 40 caméras lors de la fusion par terme barrière ([67]). Notons que pour ce type de fusion nécessite la résolution d'un système dense, et que par conséquent la méthode classique basée sur le complément de Schur, telle qu'elle a été présentée en section 1.3.2 ne peut être utilisée directement. Cependant, on peut facilement exploiter la formule de Sherman-Morrison (9) afin de retomber sur deux systèmes creux. Les détails de ces calculs, tout comme le calcul de la Jacobienne de l'expression 2.3 sont regroupés dans l'annexe B (B).

2.2.2 Fusion avec les modèles 3d des bâtiments et modèles d'élévations de terrain

Les modèles 3d des bâtiments ont été utilisés dans un certain nombre de travaux récents. Ils sont par exemple exploités (sans textures) par [3] afin d'initialiser le SLAM avec une pose géoréférencée. Leur approche consiste en trois étapes : i) La matrice de rotation permettant

l'alignement entre le point de fuite ([34]) dominant vertical et l'axe de la gravité est calculée. Ceci corrige le roulis et le tanguage ; ii) La correction du lacet est calculée à partir de l'observation que la projection d'une droite horizontale dans le plan focal de la caméra dont l'orientation correcte est connue doit passer par le point de fuite horizontal correspondant à la droite $3d$, qui est calculée en utilisant les modèles $3d$ des bâtiments. En effet, le point de fuite horizontal de chaque plan d'un modèle dans le repère correct de la caméra serait donné par le produit vectoriel entre la normale de ce plan et l'axe vertical de la caméra ; iii) Finalement, la translation est estimée via la recherche du meilleur alignement entre les projections des arrêtes des modèles $3d$ un ensemble de droites verticales calculées à partir du gradient horizontal et une segmentation sémantique de l'image. [77] reprojettent des images de Google street view sur des modèles $3d$ des bâtiments, obtenant ainsi des modèles $3d$ texturés. La base de données ainsi générée est ensuite utilisée en ligne afin de résoudre le problème PnP résultant de la mise en correspondance $2d-3d$ entre des descripteurs SIFT de points d'intérêt de l'image et ceux de la base.

Dans ce mémoire, nous nous intéresserons principalement à l'utilisation de modèles $3d$ non texturés mais géoréférencés de bâtiments dans le but de contraindre l'estimation de la structure géométrique de l'environnement de manière explicite, et de pouvoir fournir une pose dans un repère absolu, même en l'absence de GPS. [118] et [58] ont respectivement démontré l'intérêt d'une telle contrainte dans le cadre d'une fusion linéaire et dans le contexte d'une fusion par terme barrière. Notre choix portera sur cette dernière compte tenu des avantages qui l'accompagnent (§2.2.1). Soit $\mathcal{Q}$ le sous-ensemble des points $3d$ de la scène qui appartiennent à des bâtiments, et notons p_q l'index du plan du maillage (des bâtiments) auquel $q \in \mathcal{Q}$ appartient vraisemblablement. Nous reprenons la définition de la contrainte aux bâtiment proposée par [118] :

$$C_i = \sum_{q \in \mathcal{Q}} \rho(d(q, p_q)) \tag{2.4}$$

Il s'agit donc de minimiser la distance Euclidienne (notée $d(.)$) entre un point $q \in \mathcal{Q}$ et le plan qui lui est associé. Afin de limiter l'impact des outliers, un noyau robuste de type Geman-McClure ([118]), que nous notons ρ est utilisé.

Il faut bien évidemment définir quelques heuristiques afin de déterminer l'ensemble $\mathcal{Q}$, puis d'associer chaque point de cet ensemble à un plan du modèle $3d$. Dans leur travaux, [118, 58, 59, 60], la segmentation du nuage de point $3d$ est effectué via un simple lancé de rayon : un point est considéré comme appartenant à un modèle $3d$ de bâtiment si le rayon passant par le centre optique de la caméra et le point en question intersecte un plan du maillage. L'association point/plan se fait simplement par un critère de proximité. En d'autre terme, chaque point $q \in \mathcal{Q}$ est associé au plan du modèle le plus proche.

Cette ségmentation rapide ne fait pas obstacle au déploiement en temps-réel, mais, sans surprise, elle ne présente aucune robustesse vis à vis des occultations. Dans un context urbain, cela conduit fréquemment à la dérive de l'estimation lorsque les façades des bâtiments sont occultées par une végétation dense (ou plus rarement par les véhicules ou encore des structures telles que colonnes Morris, etc). C'est principalement dans le but de pallier ce problème que nous tentons, dans nos travaux, d'exploiter des réseaux convolutionnels profonds permettant de segmenter le nuage de points 3d.

2.3 Optimisation de l'ajustement de faisceaux

Ainsi que nous l'avons indiqué à la fin de la section 2.2.1, l'ajustement de faisceaux par terme barrière nécessite l'optimisation d'un plus grand nombre de caméras, ce qui a pour effet l'augmentation de la consommation de ressources. Dans cette thèse, nous nous intéresserons (chapitre 3) à la réduction de la complexité en temps d'exécution de la fusion par terme barrière. Ce problème est intimement lié à l'optimisation en terme de temps de calcul du k-BA non-contraint (*i.e.* vision pure). Ceci est principalement dû à la formulation du problème sous forme de la minimisation d'une fonction de coût, et à l'utilisation d'algorithmes d'optimisation exploitant l'Hessienne. En effet, les problèmes de temps de calcul découlent en général de la taille de cette matrice, et des difficultés lié au calcul/approximation de son inverse. Donc, les différentes philosophies motivant une approche d'optimisation du k-BA sont transférables à l'optimisation de la fusion par terme barrière et vice versa.

Nous distinguons deux familles d'approches :

- ▷ Les approches améliorant l'optimisation elle-même.
- ▷ Celles qui réduisent la dimensionalité du problème.

En général, les approches de la première catégorie exploitent soit un algorithme d'optimisation développé de manière spécifique pour le k-BA (*e.g.* [42, 10, 73]), soit la structure éparse inhérente à l'Hessienne et le ré-ordonnement de variables ([42, 52]).

La deuxième famille d'approches repose sur des suppressions de variables et sur la marginalisation ([42, 90, 66]). Par métonymie, nous avons employé le terme "marginalisation" aussi bien pour faire référence à la marginalisation probabiliste (§1.3.3) que pour désigner le remplacement de l'estimation de la structure de l'environnement (*e.g.* les points $3d$) par des contraintes relatives entre caméras. Ce dernier cas conduit aux approches de graphes de pose (*e.g.* [43, 113, 52]). Notons que les travaux où les erreurs de reprojection sont remplacées par des contraintes épipolaires peuvent êtres inclues dans cette deuxième famille.

Ainsi que nous le verrons dans §3, nos travaux de réduction de complexité sont liés à la deuxième des catégories mentionnées ci-dessus.

2.4 Deep learning et SLAM

On peut distinguer, grossièrement, quatre catégories de méthodes où le SLAM et le DL intéragissent :

- ▷ Celles qui tentent de proposer des méthodes de SLAM (contraint ou non contraint) entièrement basées sur l'apprentissage
- ▷ Celles qui fusionnent les résultats d'un SLAM sans apprentissage (ou de ses composants) avec la sortie d'une ou plusieurs méthodes basées sur de l'apprentissage
- ▷ Celles qui peuvent se substituer à une ou plusieurs composantes d'un SLAM plus classique, ou dont la sortie peut potentiellement être utilisée dans une fusion avec un SLAM

▷ Celles qui n'utilisent le SLAM qu'afin d'augmenter les performances d'un système de DL

Nous présentons quelques approches remarquables de chaque catégories de manière succincte dans ce qui suit.

2.4.0.1 Première famille.

L'intérêt pour la première famille est très récente [2]. Les travaux effectués dans cette direction, dont les plus notables à la connaissance de l'auteur sont ceux de [136, 126], restent peu nombreux, mais ambitieux.

La solution de [136], qui estime de manière conjointe la pose de la caméra et une carte de profondeur dense à partir d'un flux mononculaire est particulièrement intéressante car elle formule ce problème de manière à pouvoir utiliser un apprentissage non-supervisé. L'idée principale est d'exploiter la synthèse de point de vue : en notant I_k une image "cible", et $I_1, ..., I_{k-1}$ un ensemble d'images dont les poses et profondeurs ont déjà été estimées, il est possible de synthétiser une image $\hat{I}_k$ à partir d'une estimation initiale de la pose et de la carte de profondeur qui correspondent à I_k. Le problème peut donc être formulé comme la recherche de la pose et de la carte de profondeur qui minimisent la distance photométrique entre I_k et $\hat{I}_k$. Ce critère permettant d'évaluer la qualité d'une estimation de pose/profondeur, il permet de s'affranchir de tout supervision dans l'apprentissage. Suivant ce constat, les auteurs entraînent deux réseaux de manière conjointe. L'un prédit la profondeur de chaque pixel, le second prédit la pose d'une caméra "cible" en fonction d'un nombre de caméras "sources" (*i.e.* dont les champs de vue intersectent suffisamment celui de la cible), ainsi qu'une carte "d'explicabilité". Cette dernière cherche à pallier les problèmes liés aux occultations, à la non-rigidité de la scène et aux inconsistances photométriques. Elle produit, pour chaque pixel, un poids qui indique sa contribution à la fonction de coût. Les résultats qu'obtiennent les auteurs sont encourageants : l'approche semble produire des résultats de précision comparable à celle d'ORB-SLAM ([87]) lorsque le nombre d'images consécutives fournies à ORB-SLAM est le même que celui sur lequel le réseau peut se baser pour une prédiction. Mais les résultats obtenus restent moins précis que ceux d'ORB-SLAM sur une trajectoire plus longue, où un système de SLAM géométrique peut bénéficier, entre autre optimisations, des fermetures de boucles.

La solution proposée par [126] cherche à prédire la pose et les cartes de profondeur de manière conjointe à partir d'un flux stéréo. L'apprentissage se fait de manière supervisée. Les auteurs affirment qu'une architecture naïve de type CNN a tendance à ignorer l'une des images du couple stéréo lors de l'apprentissage de la prédiction de profondeur. Ayant identifié cet écueil, ils proposent une architecture itérative sous forme d'une cascade d'encodeurs et de décodeurs, où la prédiction du flux optique entre les deux images est ré-injectée dans le réseau. Les résultats obtenus sont une fois de plus encourageant, mais les auteurs ne comparent leurs travaux qu'avec des systèmes d'odométrie visuelle (*e.g.* il n'est pas question d'ajustement de faisceaux).

2. Jusqu'au milieu de l'année 2016, la plupart des approches se concentraient sur la localisation (*e.g.* [48]) ou sur des problèmes tels que la mise en correspondance de points d'intérêt (*e.g.* [32]). De ce fait, elles peuvent être considérées comme appartenant à la troisième famille.

2.4.0.2 Deuxième famille

[100] proposent une solution remarquable par son originalité au problème de fusion vision-IMU. Un module de calcul de pose relative, basé sur une approche traditionnelle de correspondances entre descripteurs 3d et 2d estime la pose de chaque nouvelle image. En parallèle, un réseau récursif de type LSTM, pour Long Short Term Memory [29], prenant en entrée l'historique des mesures IMU, des poses de caméras estimées et des points 3d prédit une pose de manière indépendante. Les auteurs soulignent que les prédictions du réseau, contrairement à celles du calcul de pose basé vision qui peuvent être aberrantes sur certaines images, ne dérivent que très progressivement. Par conséquent, dans les rares cas où l'écart entre la pose prédite par le calcul de pose géométrique et la pose prédite par le LSTM est jugé trop grand, seul le résultat de ce dernier est utilisé. Dans le cas contraire, un filtre de Kalman Linéaire est utilisé afin de déduire la pose finale et sa covariance, qui sont bien sûr réinjectées dans le LSTM. Les résultats expérimentaux démontrent surtout l'applicabilité de la méthode, et l'avantage qu'elle représente par rapport à une approche purement visuelle. Malheureusement, aucune comparaison avec les approches plus classique de fusion (*e.g.* EKF) n'est fournie.

[120] proposent eux aussi une solution basée sur un apprentissage supervisé. L'algorithme repose essentiellement sur la fusion entre des cartes de profondeur/incertitudes fournies par un CNN avec celles produites par un SLAM semi-dense basé images-clefs reposant sur LSD-SLAM ([20]). L'approche améliore les résultats de LSD-SLAM, et est plus précise que ORB-SLAM.

On retrouve aussi dans cette famille, les premières recherches sur les avantages que présente l'utilisation des features issues de couches intermédiaires de CNNs entrainés à des tâches telles que la segmentation dans la comparaison d'images, par exemple les travaux de [25]. Par la suite, d'autres études ce sont penchées sur l'utilisation des représentations intermédiaires fournies par des réseaux. Par exemple, [5] améliorent de cette manière la méthode SeqSlam [81], qui est une méthode de relocalisation basée sur des appariements entre des séquences d'images.

2.4.0.3 Troisième famille

Nous plaçons dans cette catégorie les méthodes pouvant être employées comme une composante d'un SLAM classique, ou dont la sortie peut être utilisée dans une fusion avec un SLAM. A ce titre, les approches d'extraction et mise en correspondance de features, les méthodes de régression de pose (relative ou absolue) ainsi que les méthodes d'odométrie visuelle font partie de celle-ci.

Plusieurs études ce sont concentrées sur les problématiques liées aux extractions de descripteurs et à l'appariement d'images ([15, 98])

Les méthodes de régression de pose (*e.g.* [48, 46, 78]) peuvent aussi être inclues dans cette troisième famille, car, elles peuvent, à terme, se substituer au calcul de pose relatif entre caméras pour l'initialisation du SLAM ou la fermeture de boucle, ou même au calcul PnP classique utilisé dans les méthodes de BA. L'approche intiale de poseNet ([48]) était basée sur un apprentissage basique d'un CNN sur un grand nombres d'images monoculaires afin de produire la pose absolue. Le réseau entraîné produisait des résultats plutôt imprécis. L'itération la plus récente de cette méthode ([46]) améliore grandement la performance du premier système, notamment via la présentation de fonctions de coût basées sur *l'incertitude homoscédastique* [3] et les erreurs

3. L'incertitude des données d'entrée est en général divisée en deux catégories : l'incertitude homoscédastique et l'incertitude hétéroscédastique. Ces incertitudes, contrairement à celle du modèle, ne peuvent pas être réduites par l'utilisation de plus de données pour l'apprentissage. L'incertitude homoscédastique correspond à une incer-

de reprojections. Un résultat intéressant est qu'un réseau pré-entrainé avec la fonction homoscedastique et entrainé par la suite avec la fonction utilisant des termes de reprojection produit les meilleurs résultats. Les expériences des auteurs montrent que cette itération de poseNet présente plusieurs avantages par rapport aux approches traditionnelles utilisant les SIFT, notamment en terme des tailles d'images requises et de temps de traitements. [78] proposent une solution de régression de pose relative elle aussi basée sur de l'apprentissage supervisé, intéressante par sa simplicité, et aussi par son utilisation de layers de Spatial Pyramid Pooling ([36]), ce qui permet d'entrainer/tester le réseau sur des images de tailles variables. L'approche repose sur l'entraînement d'un CNN siamois en utilisant comme fonction de coût une simple somme pondérée des erreurs aux carré des orientations et des translations. Les résultats présentés par les auteurs indiquent une certaines robustesse par rapport aux changements de points vue.

De la même manière, les méthodes d'odométrie visuelles basées sur l'apprentissage ([96, 129]) peuvent être considérées comme appartenant à cette troisième famille, car bien qu'elles ne reconstruisent pas l'environnement de manière explicite, elle peuvent potentiellement être utilisées afin de contraindre la pose des caméras dans un SLAM. [129] présentent un réseau composé d'un CNN qui extrait des features, et d'un réseau récurrent de type LSTM chargé de la regression de la pose. Leurs résultats démontrent clairement l'apport de la prise en compte des liens séquentiels par le LSTM. Dans leur travaux, [96] modélisent la densité conditionnelle de la pose sous forme d'une somme pondérée de Gaussiennes (Gaussian Mixture Model en Anglais), implémentée par un MDN (Mixture Density Network). Le conditionnement se fait par rapport au flux optique. En plus de prendre en compte un terme de régression local, ils introduisent dans l'apprentissage un terme global garantissant la cohérence de la trajectoire. Le système qu'ils présentent possède la particularité de pouvoir se superviser automatiquement, en produisant une vérité terrain basée sur une fusion GPS/IMU/Odométrie. Finalement, la reconstruction du flux optique d'entrée par un décodeur permet de filtrer le flux qui n'est pas issu du déplacement du véhicule, mais des différentes sources d'erreurs ainsi que des objets non-dynamiques de la scène. La publication faisant état de ces travaux étant encore au stade de pré-publication au moment de la rédaction de ce mémoire, les résultats expérimentaux ne semblent pas encore suffisants et surtout axés sur des simulations, sur lesquels ils sont convaincants.

Le calcul d'homographie par DL peut lui aussi s'avérer utile dans le cadre de certaines méthodes d'appariement de patchs, par exemple, il serait intéressant de coupler le réseau de régression de [14] avec le système de [11].

2.4.0.4 Quatrième famille

Nous ne nous attarderons pas sur ces approches car elles n'ont que peu de liens avec les objectifs de cette thèse. Nous noterons simplement qu'il s'agit le plus souvent d'utiliser les résultat d'un SLAM ou d'une méthode SFM afin de constituer une base d'apprentissage supervisé (*e.g.* l'utilisation des résultats de [131] par [78]), ou de l'utilisation de l'information fournie par un SLAM afin de guider un agent dans le cadre d'un apprentissage par renforcement ([9, 82]).

titude constante pour des entrées différentes, alors que l'incertitude hétéroscédastique peut varier d'une donnée à l'autre. Pour plus d'informations à ce sujet, nous invitons le lecteur à consulter [47].

2.5 Positionnement

L'objectif de cette thèse étant, comme nous l'avons vu dans l'introduction (§), d'améliorer les performances de la fusion par terme barrière et d'étudier la possibilité et l'apport de l'utilisation du deep learning avec la contrainte aux bâtiments. Pour les besoins de cette thèse, nous nous somme basés sur un k-BA, et c'est le terme d'erreur correspondant à celui-ci (1.25) que nous avons utilisé comme terme barrière dans 2.3. Cependant, il est important de remarquer qu'aucune de nos contributions ne dépend du k-BA. En effet, l'erreur ζ utilisée dans le terme barrière doit simplement exprimer la qualité d'une reconstruction. Il peut par conséquent s'agir d'une erreur de graphe de pose comme d'une erreur photométrique (*e.g.* celle de [19]). Les résultats de [67] énoncés dans le cas d'un ζ général, et nos discussion ainsi que le théorème que nous démontrons au chapitre §3 sont vérifiés sous réserve que la fonction de coût choisie vérifie certaines conditions (§3). De manière similaire, le couplage SLAM/deep learning effectué dans cette thèse exploite le deep learning dans une brique d'association de données, indépendante du SLAM, dont l'objectif principal reste l'appariement correct des points $3d$ avec les plans du maillage. Il est donc possible, avec un minimum d'effort, d'adapter l'ensemble des approches proposées à d'autres types de SLAM.

Nos travaux sur les interactions entre le deep learning et le SLAM peuvent être regroupés dans la troisième famille d'approches présentée dans la §2.4. Plus précisément, nous exploitons les sorties de réseaux profonds afin de réduire le bruit dans l'association de données, qui, même en présence de M-estimateurs, peut conduire à des résultats de fusion aberrants si les occultations sont omniprésentes.

Concernant l'optimisation de la fusion par terme barrière, notre approche réduit la dimensionalité du problème en encodant les relations implicites entre les caméras (imposées par une partie des point $3d$ de la scène) sous forme de contrainte de pose relative. Elle appartient donc à la deuxième famille d'approches (§2.3).

2.6 Résumé

Dans ce chapitre, nous avons présenté les différentes formulations du SLAM visuel non-contraint et contraint et de la fusion. Nous nous sommes en particulier penchés sur la fusion par terme barrière et sur la contrainte aux bâtiments. Nous avons ensuite fait un tour d'horizon des approches de deep learning de l'état de l'art en lien avec le SLAM. En particulier, nous retiendrons les points suivants :

- ▷ La définition du terme barrière employée dans l'état de l'art ([67, 59, 58] rend la fusion exigeante en temps de calcul, ce qui rend l'algorithme difficilement déployable sur les plateformes de faible capacité.
- ▷ La contraînte aux bâtiments souffre de l'introduction de bruit lors de l'étape de l'association de données, en particulier en présence d'occultation (*e.g.* arbres en milieu urbain). Ceci est dû à l'utilisation d'un critère géométrique.
- ▷ Nos travaux sur la minisation par terme barrière font partie des approches de réduction de dimensionnalité.
- ▷ Nos recherche concernant les interactions entre le SLAM et le deep learning ce basent sur l'exploitation de réseaux entrant en composition avec des modules d'un SLAM k-BA.
- ▷ Les approches proposées sont en grande partie indépendantes de l'approche de SLAM utilisée. Nos travaux sur les interactions entre le SLAM et le deep learning sont en

intégralité adaptables pour d'autre types de SLAM, et la fusion par terme barrière ne dépend pas du type de SLAM.

CHAPITRE 3

Optimisation algorithmique de la fusion VSLAM/GPS par terme barrière

Alors que les fonctions de coût linéraires (i.e. approches par maximum de vraisemblance) requièrent en général, afin de converger, l'hypothèse de bruits Gaussiens pour l'ensemble des erreurs issues des capteurs impliqués dans la fusion, la fusion par terme barrière permet d'apporter des corrections globales à une estimation initiale sans autre a priori sur la distribution des erreurs que la nullité de leurs espérances. Cependant, elle s'avère être plus exigeante en terme de temps de calcul, ce qui rend son implémentation sur des plateformes de faible capacité délicate, voir impossible. Dans ce chapitre, nous nous concentrerons sur l'optimisation, en termes de temps de traitements, de cette approche. Nous exposerons premièrement nos contributions théoriques, avant de présenter des résultats expérimentaux mettant en exergue les avantages de notre solution.

3.1 Introduction

Dans ce chapitre, nous nous concentrerons sur la fusion VSLAM/GPS par terme barrière ([67]) :

$$\underset{X}{argmin} \quad \frac{\gamma}{t - \zeta(X)} + ||y - y_{gps}||^2. \tag{3.1}$$

Ainsi que nous l'avions souligné lors de l'introduction de la fusion par terme barrière (équation 2.3), ζ est une fonction quantifiant la qualité de la reconstruction du VSLAM. Il peut s'agir, par exemple, des erreurs de reprojections ([67, 58]). L'optimisation que nous proposons dans ce chapitre repose sur une nouvelle définition de ζ, basée sur la prise en compte simultanée d'erreurs de reprojections et d'erreurs de graphe de pose. Dans l'équation que l'on vient de présenter, y désigne notre estimation de la position dans le plan horizontal de la caméra la plus récente, et y_{gps} la position GPS associée. Le choix de n'utiliser qu'une seule contrainte est motivé par deux observations :

- ▷ D'un point de vue pratique et expérimental, le choix d'une seule caméra est suffisant ([68], section 6.1).

▷ Ce choix nous permet d'obtenir des propriété théoriques intéressantes, permettant une démonstration formelle de la validité de notre approche (dont l'approche de [68] devient un cas particulier).

On notera cependant que bien que la contrainte imposée ne porte que sur une unique caméra (la plus récente), l'optimisation s'effectue sur les $l > 1$ dernières caméras, afin de conserver la cohérence de la solution. Ainsi que nous l'avons indiqué à la section §2.2.1, l'optimisation de 3.1 est initialisée avec la solution de $\underset{X}{argmin}\,\zeta(X)$, que nous noterons X^*.

Le calcul des dérivées de l'équation 3.1 est trivial et est rapporté dans l'annexe B (§B). On voit cependant que la taille de l'Hessienne de cette fonction de coût, que nous noterons H dans la suite de ce chapitre, est identique à celle H_ζ. Donc, la résolution du problème dépend de la taille de cette Hessienne. Les approches de l'état de l'art (*e.g.* [68, 58, 59] définissent ζ comme étant la somme des erreurs de reprojections au carré. Pour des raisons que nous expliciterons dans la section suivante (§3.2), ceci entraîne l'augmentation de la taille de l'Hessienne. Notre objectif dans ce chapitre est donc de proposer une définition de ζ qui réduise le nombre de paramètres et par conséquence la taille de H sans perte significative de précision de localisation ou d'inliers.

Afin d'éviter des notations encombrantes et en utilisant les paramétrisations minimales introduites au chapitre 1, nous définirons $S : SE(3) \times SE(3) \rightarrow SE(3)$ comme suit :

$$S(R_i, T_i, R_j, T_j) = \begin{pmatrix} R_i^T R_j & R_i^T(T_j - T_i) \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.2}$$

Par un même soucis de concision, et étant donné la matrice bloc-diagonale $U = diag\{U_1, ..., U_c\}$, nous introduisons la notation $U_{i:j} = diag\{U_i, ..., U_j\}$. Nous utiliserons aussi le prédicat $Id(.)$ qui retourne l'entier 1 si l'élément qui lui est fourni en argument est vrai, et qui retourne 0 dans le cas contraire.

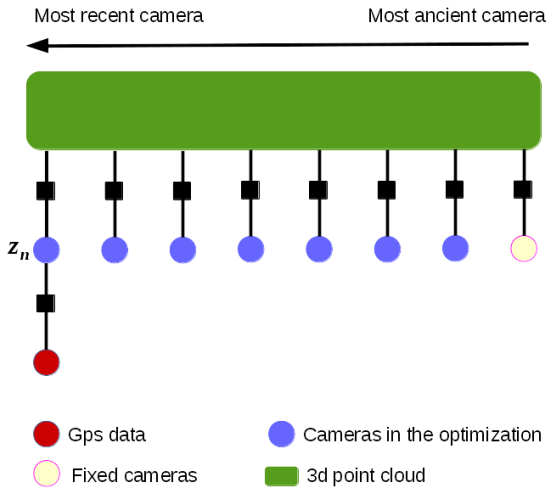
3.2 Motivations

Soit Δ l'incrément optimal devant être appliqué à X^* afin d'effectuer la fusion VSLAM/GPS. Ainsi qu'il a été affirmé à la section précédente, la contrainte GPS n'est imposée que sur la caméra la plus récente. On peut donc démontrer ([68]) que

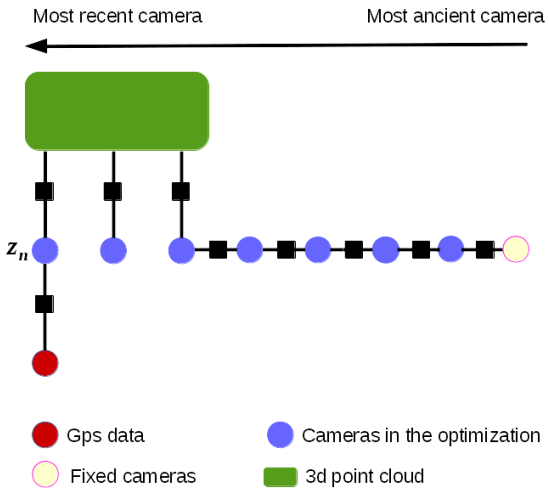
$$\zeta(X + \Delta) - \zeta(X) \approx \frac{1}{2}(\Delta_{c_1})^T C_1^{-1}(\Delta_{c_1}) \tag{3.3}$$

où C_1 et Δ_{c_1} sont, respectivement, la matrice de covariance et la partie du vecteur d'incréments Δ qui correspond à la caméra la plus récente. Ainsi que le fait remarquer [67], $\Delta\zeta \triangleq \zeta(X + \Delta) - \zeta(X)$ doit être suffisamment petit pour éviter que la fusion par terme barrière n'échoue. Cela est dû au fait que la solution doit être dans la région de confiance définie de manière implicite par le seuil t ([68]), mais aussi à cause du taux d'augmentation très élevé de l'erreur quand $\zeta(X)$ tend vers t. De l'équation 3.3, il découle que $\Delta\zeta$ est toujours positif et que sa valeur dépend de l'évolution de la covariance : $\Delta\zeta$ décroit lorsque C_1 augmente (au sens de l'augmentation des valeurs propres, et donc de l'ellipse d'incertitude). Ainsi qu'il a été démontré de manière expérimentale dans les travaux de [68, 67], la covariance croît avec l'augmentation du nombre de caméras (et des points $3d$ observés depuis celles-ci) incluses dans

l'optimisation. [68, 59, 58] choisissent cette solution afin de réduire la taille de $\Delta\zeta$. Cependant, cette approche nécessite l'optimisation de 30 à 40 caméras et les points qui les accompagnent. Ceci rend la procédure de fusion trop exigeante en terme de temps de calcul pour les plateformes embarquées ou le matériel de faible capacité en général.



(a) Le graphe de facteurs (§1.8.2) de la fusion BA/GPS d'origine.



(b) Le graphe de facteurs résultant de notre approche de fusion hybride (BA/pose-graph).

FIGURE 3.1 – Comparaison des graphes de facteurs de l'approche de fusion BA/GPS d'origine et de notre solution.

3.3 Esquisse de l'approche

Afin de réduire le nombre de paramètres, nous proposons de définir ζ comme suit. En posant l'hypothèse que la minimisation est effectuée sur les l images clefs les plus récentes $z_1, ..., z_l$, notons $\mathcal{C}$ l'ensemble des caméras clefs précédent z_1 (*i.e.* $\mathcal{C}$ est l'ensemble de caméras qui ne sont

pas incluses dans l'optimisation). Si $\mathcal{C} \neq \varnothing$ alors on note z_0 l'image clef qui précède z_1. Nous proposons de retirer les points observés depuis $z_1, ..., z_n$ (avec $n < l$) de la minimisation, et de les remplacer en revanche par des contraintes relatives entre les caméras $z_1, ..., z_n$. En d'autres termes, nous définissons, de manière similaire à ce qui a été présenté pour le BA non contraint dans [113], une fenêtre de "graphe de pose" $z_1, ..., z_n$ et une fenêtre de BA $z_{n+1}, ..., z_l$. Seul les points qui sont observés depuis cette deuxième fenêtre sont considérés dans la minimisation.

$$\begin{aligned}\zeta(X) = &\sum_{i=n+1}^{l} \sum_{j \in \mathcal{O}_i} ||\pi(K(R_i X_{ij} + T_i)) - \tilde{x}_{ij}||^2 \\ &+ \sum_{i=Id(\mathcal{C}=\varnothing)}^{n} \xi(R_i, T_i, R_{i+1}, T_{i+1})\end{aligned} \tag{3.4}$$

où $\mathcal{O}_i$ est l'ensemble de tout les points qui sont observés depuis z_i. Les variables R_i, T_i indiquent respectivement la rotation et la translation de la caméra z_i, et $\tilde{x}_{ij}$ est l'observation dans la caméra i du point X_{ij}. La matrice de calibration est notées K et la fonction réelle ξ exprime les contraintes du graphe de poses. Nous remarquerons que les méthodes de BA contraint de [68, 58] qui utilisent un terme barrière sont un cas particulier de notre formulation (*i.e* lorsque $n = 0$). Le graphe de facteurs de ces BA contraints et celui de notre solution sont comparés dans la figure 3.1.

Dans ce texte, nous poserons l'hypothèse que le nombre de caméras dans la fenêtre de k-BA, $l - n$, vérifie $l - n \geq l_b \geq 3$ où le minorant l_b est un entier positif. Ce choix concernant le minorant présente deux avantages principaux. Premièrement, il garantit qu'il existera toujours en pratique un ensemble de points $3d$ dans l'optimisation, ce qui permet une estimation précise des poses relatives des nouvelles caméras sans avoir besoin de recourir à la re-triangulation ou à d'autres efforts supplémentaires. En deuxième lieu, ce choix facilite grandement les notations et les développements théoriques présentés dans la section suivante. En outre, conserver un nombre minimum de points $3d$ peut s'avérer utile, car ils peuvent être exploités afin d'imposer de nouvelles contraintes ([118], ou afin d'estimer le biais du GPS ([61]).

Nous considérerons deux définitions différentes de ξ. Cependant, avant de les présenter (§3.5), Nous devons au préalable expliquer pourquoi nous attendons de cette approche qu'elle soit efficace.

3.4 Contributions théoriques

Partitionnons H_ζ, la matrice Hessienne de ζ comme suit :

$$\begin{pmatrix} U & W^T \\ W & V \end{pmatrix} \tag{3.5}$$

où les matrices bloc-diagonales $U = diag\{U_1, ..., U_c\}$ et $V = diag\{V_1, ..., V_p\}$ correspondent respectivement aux paramètres des caméras et des points, avec $U_i \in \mathbb{R}^{6\times6}$ et $V_i \in \mathbb{R}^{3\times3}$ pour chaque i défini. La densité de W^T dépend de la structure de la scène. On peut facilement démontrer ([123]) que le seul effet du rajout de contraintes relative (*i.e.* de graphe de poses) sur l'Hessienne est la densification du bloc U. Nous démontrons que l'ajout de caméras, que ce soit en conjonction avec des contraintes de graphe de pose ou avec des erreurs de reprojections traditionnelles, aura pour effet la décroissance du réel positif $\Delta\zeta$. Donc, le théorème suivant démontre, en plus de la plausibilité de notre approche, la propriété de croissance de la covariance

avec l'ajout de caméras plus anciennes, uniquement vérifiée de manière expérimentale ([68]) avant nos travaux.

$$\begin{bmatrix} U_1 & W_1^T \\ W_1 & V \end{bmatrix}$$

(a) Hessienne d'origine avec une seule caméra.

$$\left[\begin{array}{c|cc} U_1 & 0 & W_1^T \\ \hline 0 & U_2 & Q^T \\ W_1 & Q & V \end{array}\right]$$

(b) L'Hessienne lorsqu'une caméra est rajoutée.

$$\left[\begin{array}{c|cc} U_1 & W_1^T & 0 \\ \hline W_1 & V & \Omega^T \\ 0 & \Omega & U_{2:k} \end{array}\right]$$

(c) L'Hessienne réordonnée avec K caméras.

$$\left[\begin{array}{c|cc|c} U_1 & W_1^T & 0 & 0 \\ \hline W_1 & V & \Omega^T & S^T \\ 0 & \Omega & U_{2:k} & \\ \hline 0 & \multicolumn{2}{c|}{S} & U_{k+1} \end{array}\right]$$

(d) L'Hessienne réordonnée avec $K+1$ caméras.

FIGURE 3.2 – Les matrices Hessiennes utilisées dans la preuve du théorème 1

***Theorème 3.* 1.** *Soit C_t la covariance de la caméra la plus récente lorsque t caméras et un nombre de points $3d$ sont considérés dans la minimisation. Rajoutons maintenant une caméra à l'optimisation, mais sans ajouter de points $3d$, et notons la nouvelle covariance C_{t+1}. Dans ce cas,*

$$x^T C_{t+1}^{-1} x \leq x^T C_t^{-1} x \tag{3.6}$$

pour tout $x \in \mathbb{R}^6$, indépendamment de la fenêtre (i.e. fenêtre BA ou graphe de pose) à laquelle la nouvelle caméra est associée.

Démonstration. Considérons d'abord le cas où il n'existe qu'une seule caméra dans l'optimisation ($t = 1$). Dans ce cas, l'Hessienne de $\zeta(X)$ peut être partitionnée comme dans la figure 3.2 (a), et la covariance de cette caméra est donnée par $C_1 = (U_1 - W_1^T V^{-1} W_1)^{-1}$. Notons $a(x) \triangleq x^T U_1 x$. L'Hessienne est symmétrique et définie positive (SDP), par conséquent U_1 est

SDP et $a(x)$ est à valeurs réelles et positives. Rajoutons maintenant une nouvelle caméra à l'optimisation. Alors, l'Hessienne est celle qui est donnée dans la figure figure 3.2 (b), et l'on a $C_2 = (U_1 - [0 \;\; W_1^T]F^{-1}[0; W_1])^{-1}$ où $F = [U_2 \; Q^T; Q \; V]$. Nous avions posé l'hypothèse plus tôt (§3.3) que $l_b \geq 3$, *i.e.* que nous conservons au moins 3 cameras dans la fenêtre du BA. Les blocs nuls dans la figure 3.2 (b) sont dûs au fait que pour ces caméras, aucune contrainte de graphe de pose n'est imposée. Écrivons maintenant

$$\eta_1 \triangleq x^T C_1^{-1} x = a(x) - x^T W_1^T V^{-1} W_1 x \tag{3.7}$$

$$\eta_2 \triangleq x^T C_2^{-1} x = a(x) - x^T [0 \;\; W_1^T] F^{-1} [0; W_1] x \tag{3.8}$$

L'inverse de la matrice F peut aisément être obtenue via le complément de Schur de son bloc V, *i.e.* $F/V = U_2 - Q^T V^{-1} Q$. En substituant l'inverse dans l'équation 3.8, et suite à quelques manipulations algébriques simples, il s'ensuit que

$$\eta_2 = \eta_1 - (W_1^T V^{-1} W_1) G (W_1^T V^{-1} W_1) \tag{3.9}$$

où $G = (U_2 - Q^T V^{-1} Q)^{-1}$. Donc, G est la covariance de U_2, obtenue via la marginalisation des variables du bloc V. Elle est par conséquent SDP, et il en résulte que le troisième terme de l'équation 3.9 est positif. Il s'ensuit que $\eta_2 \leq \eta_1$.

Nous devons maintenant généraliser ce résultat. Soit $t = k$. Une fois de plus, nous définissons $\eta_k = x^T C_k^{-1} x$ et $\eta_{k+1} = x^T C_{k+1}^{-1} x$. Nous recherchons l'expression qui relie η_k et η_{k+1}. Cet objectif peut être atteint plus facilement si l'on réordonne les Hessiennes pour $t = k$ et $t = k + 1$, ainsi que les figures 3.2 (c) et 3.2 (d) l'illustrent. Nous invitons le lecteur à remarquer que comme précédemment, il n'existe aucune contrainte entre les caméras 1 et les cameras $2, ..., k + 1$ car $l_b \geq 3$. Les blocs correspondants sont donc nuls dans les deux figures mentionnées. Cependant, des contraintes de type graphe de poses peuvent exister entre les caméras restantes. C'est pour cela que les blocs S, S^T, Ω, Ω^T sont donnés.

Posons, afin de simplifier les notations, $N \triangleq [V \;\; \Omega^T; \Omega \;\; U_{2:k}]$ et $M \triangleq [N \;\; S^T; S \;\; U_{k+1}]$. Nous pouvons maintenant écrire

$$\eta_k = a(x) - x^T \begin{pmatrix} W_1^T & 0 \end{pmatrix} N^{-1} \begin{pmatrix} W_1 \\ 0 \end{pmatrix} x \tag{3.10}$$

$$\eta_{k+1} = a(x) - x^T \begin{pmatrix} W_1^T & 0 & 0 \end{pmatrix} M^{-1} \begin{pmatrix} W_1 \\ 0 \\ 0 \end{pmatrix} x \tag{3.11}$$

de la substitution de l'inverse de M, obtenue via le complément de Schur M/U_{k+1}, il découle que le second terme de l'équation 3.11 peut être exprimé ainsi :

$$\begin{aligned} & x^T \begin{pmatrix} W_1^T & 0 & 0 \end{pmatrix} M^{-1} \begin{pmatrix} W_1 \\ 0 \\ 0 \end{pmatrix} x \\ & = x^T \begin{pmatrix} W_1^T & 0 \end{pmatrix} (N - S^T U_{k+1}^{-1} S)^{-1} \begin{pmatrix} W_1 \\ 0 \end{pmatrix} x \end{aligned} \tag{3.12}$$

Ensuite, nous développons $(N - S^T U_{k+1} S)^{-1}$ en utilisant l'identité de Woodbury ([40]). Après quelques manipulations algébriques triviales, on obtient

$$\begin{aligned} \eta_{k+1} &= \eta_k \\ &- x^T \begin{pmatrix} W_1 \\ 0 \end{pmatrix}^T (N^{-1} S^T (U_{k+1} - S N^{-1} S^T)^{-1} S N^{-1}) \begin{pmatrix} W_1 \\ 0 \end{pmatrix} x \end{aligned} \tag{3.13}$$

La matrice $(U_{k+1} - S N^{-1} S^T)$ est SDP, et N l'est aussi. On en déduit que $\eta_{k+1} \leq \eta_k$. □

Ceci démontre que rajouter une caméra, qu'elle appartienne à la fenêtre du graphe de pose ou à la fenêtre du BA, fait décroitre $\Delta^T C^{-1} \Delta$ et par conséquent $\Delta\zeta$, alors que le nombre de points $3d$ demeure fixe. Cependant, il est important de remarquer que la quantité dont décroit $\Delta\zeta$ n'est pas la même en fonction de l'appartenance de la nouvelle caméra à la fenêtre de BA ou à la fenêtre du graphe de pose. Ceci est dû au fait que le bloc correspondant aux caméras $k, k+1$ est non-nul si la caméra appartient à la fenêtre du graphe de pose, alors qu'il s'annule dans le cas contraire. Intuitivement, on attend des blocs camera/camera correspondants aux contraintes du graphe de pose qu'ils encodent l'information de covariance qui était contenue dans les blocs camera/point avant que les points $3d$ ne soient supprimés.

De manière similaire, on peut démontrer l'énoncé suivant.

Theorème 3. 2. *Rajouter un point $3d$ qui n'est pas observé par la caméra la plus récente fera décroître $\Delta\zeta$.*

Nous omettons la démonstration, car elle est essentiellement la même que celle de du théorème 3.1 (il n'y a que les noms des blocs qui changent). Il est intéressant de noter que si le point $3d$ rajouté est observé par la caméra la plus récente, alors $\Delta\zeta$ peut augmenter en fonction de la structure de la scène. Notons cependant que comme $l_b \geq 3$, nous avons déjà une certain nombre de caméras et de points observés depuis celles-ci dans la minimisation. Par conséquent, les points observés depuis la caméra la plus récente sont déjà dans l'optimisation et l'inclusion de points plus anciens dans la minimisation ne peut que diminuer $\Delta\zeta$.

Ces deux points principaux ressortent des théorèmes précédents :

- ▷ Notre solution, qui consiste à rajouter des caméras avec des contraintes de type graphe de poses mais pas de points $3d$ fait décroître $\Delta\zeta$ d'une certaine valeur positive a_1. Par conséquent, elle fonctionnera si un nombre suffisant de caméras n_1 est rajouté.

- ▷ La solution de [68, 67] pour réduire $\Delta\zeta$, qui consiste à rajouter des caméras et leurs points $3d$ (mais sans contraintes de graphe de pose), peut réduire $\Delta\zeta$ d'une valeur positive qu'on notera $a_2 + a_3$, où a_2 résulte de l'ajout de la caméra, et a_3 du rajout des points $3d$. Par conséquent cette approche fonctionne si un nombre suffisant de caméras n_2 est rajouté.

- ▷ En général, on a $a_1 \neq a_2$, car la structure des blocs camera/camera de l'Hessienne diffèrent.

Cependant, la relation entre les deux formes définies positives a_1 et a_2 n'est pas claire. Il est évident qu'elle dépend de la structure de la scène et que dans le cas où $n > 0$ (*i.e.* au moins une caméra en graphe de pose). Elle dépent aussi de la définition choisie pour $\xi(X)$ (*i.e.* du terme évaluant les erreurs liées à la fenêtre du graphe de pose, introduit dans l'équation 3.4).

Dans [68], il est démontré de manière expérimentale que n_2 se situe entre 30 et 40. De manière similaire, nous montrerons que pour les deux définitions de ξ que nous présenterons (section suivante) une valeur $n_1 \sim 30$ est suffisante pour que l'optimisation par terme barrière produise de bons résultats. Cela semble être une indication (assez vague, cependant), que $a_1 >> a_2$ ou que a_3 est négligeable (en fonction de la structure de la scène). Mais quel que soit le cas, notre méthode diminue le nombre de paramètres de manière importante, ce qui était l'objectif recherché.

3.5 Choix de la fonction de coût

Nous avons démontré que l'approche hybride proposée fonctionnera si un nombre suffisant de caméras ainsi que les contraintes relatives les liant sont ajoutées dans l'optimisation. Nous somme donc prêt à définir ξ (*i.e.* La fonction mesurant à quelle point la pose relative est modifiée par rapport à l'estimation initiale, présentée dans l'équation 3.4). Nous proposons deux formulations différentes.

3.5.0.1 Contraintes de poses directes (CPD)

Notons $\hat{R}_i$ et $\hat{T}_i$ de manière respective la rotation et la translation de z_i avant la fusion (*i.e.* après la minimisation de ζ.). Dans ce cas, la matrice de changement de coordonnées de z_{i+1} à z_i est donnée par (la notation $\hat{S}$ a été définie dans l'introduction, §3.1) :

$$\hat{S} = S(\hat{R}_i, \hat{T}_i, \hat{R}_{i+1}, \hat{T}_{i+1}) \tag{3.14}$$

En posant

$$q_i = (log_{SE(3)}(\hat{S}^{-1} S(R_i, T_i, R_{i+1}, T_{i+1})))^{\vee} \tag{3.15}$$

,

nous définissons

$$\xi(R_i, T_i, R_{i+1}, T_{i+1}) = q_i^T D_\xi q_i \tag{3.16}$$

où la matrice bloc-diagonal $D_\xi = diag\{A_{3\times3}, B_{3\times3}\}$ sert de pondération entre les termes d'erreur. Actuellement, nous initialisons chaque bloc par l'identité.

3.5.0.2 Contraintes de poses multivues (CPM)

La définition présentée ci-dessus pour ξ a l'inconvénient de ne pas quantifier l'erreur de graphe de pose en terme de pixels. La fonction ζ est donc définie comme une somme d'erreurs de différentes natures : l'une est exprimée en pixels (erreurs de reprojections), l'autre est donnée par une norme pondérée de vecteurs $\mathbb{R}^3$ (l'erreur du graphe de pose). Cela rend la pondération des termes (via l'initialisation de Q) difficile à appréhender de manière intuitive, et dans certain cas, elle devient indispensable. Afin de nous affranchir de la nécessité de recourir à la définition de poids entre les termes d'erreur, nous exprimons ξ en terme de pixels, en exploitant les contraintes multi-vues entre caméras. Plus précisément, étant donnée une estimation initiale de

la matrice Fondamentale F_i entre chaque couple de caméras i et $i+1$, nous cherchons à minimiser la distance entre un point x_j de l'image i et la droite épipolaire qui lui correspond, que nous noterons $F_i x'_j$. En notant la droite $F_i x'_j = (a, b, c)^T$, il est clair que celle-ci est une droite dans le plan image dont la normale unitaire (*i.e.* le gradient normalisé) est $(\frac{a}{\sqrt{a^2+b^2}}, \frac{b}{\sqrt{a^2+b^2}})^T$, et qui intersecte l'axe vertical en $\frac{-c}{a}$. La distance signée entre le point $x_j = (x_1^j, x_2^j, 1)^T$ et cette droite est alors donnée par

$$\begin{pmatrix} x_1^j & x_2^j + \frac{c}{a} \end{pmatrix} \begin{pmatrix} \frac{a}{\sqrt{a^2+b^2}} \\ \frac{b}{\sqrt{a^2+b^2}} \end{pmatrix} = \frac{1}{\sqrt{a^2+b^2}} x_j^T \begin{pmatrix} a \\ b \\ c \end{pmatrix} = x_j^T F_i x'_j \tag{3.17}$$

On peut par conséquent exprimer la contrainte ξ de cette manière :

$$\xi(R_i, T_i, R_{i+1}, T_{i+1}) = \sum_{j=0}^{s} || \frac{x_j^T F_i x'_j}{||P F_i x'_j||} ||^2 \tag{3.18}$$

où $\{(x_0, x'_0), ..., (x_s, x'_s)\}$ est l'ensemble des correspondances $2d$ entre z_i et z_{i+1}. La matrice P est définie par $P = diag\{1, 1, 0\}$. Nous invitons le lecteur à remarquer que bien que les paramètres résultants de la minimisation d'une distance $x_j F_i x'_j$ sont les mêmes que si l'on minimise $\frac{1}{||P F_i x'_j||^2} x_j F_i x'_j$, l'objectif de cette contrainte était d'exprimer l'erreur en pixels afin de rendre la pondération de ζ plus intuitive. C'est pour cela que les coefficients de normalisation $\frac{1}{||P F_i x'_j||^2}$ sont inclus dans l'expression 3.18.

3.6 Discussion sur les Jacobiennes et la structure de l'Hessienne

On peut maintenant écrire

$$\zeta = r^T r \tag{3.19}$$

où r est un vecteur regroupant les erreurs du graphe de pose et les erreurs de reprojection, *i.e.*

$$r = \begin{pmatrix} \xi_1 \\ \dots \\ \xi_\varrho \\ e_{(n+1)1} \\ \dots \\ e_{(n+1)\Phi(1)} \\ e_{(n+2)1} \\ \dots \\ e_{(n+2)\Phi(2)} \\ \dots \\ e_{l\Phi(l)} \end{pmatrix}. \tag{3.20}$$

Comme dans la section §1.3.1, e_{ij} est l'erreur de reprojection du point j dans la caméra i, et $\Phi(i)$ est le nombre de points $3d$ inliers visibles depuis cette dernière. Les variables $\xi_1, ..., \xi_\varrho$ désignent les erreurs du graphe de pose.

Dans le cas de l'erreur CPD (équation 3.16), on aura une contrainte par couple de caméras successives, on aura donc $\varrho = n + 1$ si z_0 existe (voir les notations introduites dans §3.3) , et $\varrho = n$ dans le cas contraire. Notons maintenant $\Upsilon(i)$ le nombre de couples de points $2d$ correspondants dans les caméras i et $i+1$. On peut alors, lorsque l'erreur CPM (équation 3.18) est utilisée, constater que $\varrho = \sum_{i=0}^{n} \sum \Upsilon(i)$ si z_0 existe et $\varrho = \sum_{i=0}^{n} \sum \Upsilon(i)$ si aucune caméra ne précède z_1. Afin de pouvoir résoudre le système découlant de la fusion par terme barrière (B), il est nécessaire pour nous de connaître la Jacobienne J_r et l'Hessienne de la fonction de coût 3.1.

3.6.1 Jacobienne de la contrainte CPD

Généralement, les dérivées analytiques sont plus précises que les dérivées numériques (en générale basées sur des différences finies). Cependant, ainsi que le soulignent [112] (section 5.3.3), le calcul d'une expression symbolique de type $\frac{\partial log(X)^{\vee}}{\partial X_{i,j}}$ (nécessaire pour calculer les dérivées de 3.15) n'est possible que pour certains groupe de Lie pour lesquels on connaît l'expression analytique exacte du logarithme matriciel. Bien que ce soit le cas pour les groupe tels que $SO(3)$ qui nous intéressent, l'expression symbolique est extrêmement longue et ne peut donc pas être calculée efficacement (en terme de temps de traitement). [112] proposent une approximation basée sur la formule de Campebell-Baker-Hausdorff (voir le chapitre 2 de [112]), cependant, celle-ci reste inférieure par rapport aux dérivées numériques en terme de précision. Compte tenu de cela, et du fait que le nombre de paramètres de la minimisations se retrouve réduit par notre approche, nous avons choisi d'utiliser les dérivées numériques (différences finies) pour calculer la Jacobienne de r lorsque les ξ prennent la forme des contraintes de l'équation 3.16.

3.6.2 Jacobiennne de la contrainte CPM

Les détails du calcul de la Jacobienne de la contrainte CPM sont rapportés dans l'annexe C. Cependant, les expressions symboliques obtenues sont longues et en termes de temps de calcul, significativement plus lentes ($\sim 10ms$) que les dérivées numériques via les différences finies. Nous utilisons donc les dérivées analytiques uniquement afin de bénéficier de leur précision supérieure, et afin d'éviter les problèmes numériques potentiels liés aux choix de l'incrément des différences finies.

3.6.3 Structure de l'Hessienne

D'après la définition de l'Hessienne, chaque terme de la contrainte ξ, quelle que soit le type de fonction de coût (CPM ou CPD), ne concernera que deux caméras successives i et $i+1$. Par conséquent, seuls les blocks non-diagonnaux liant les caméras successives seront densifiés, ainsi que l'illustre la figure 3.3.

3.7 Évaluation expérimentale

Il est important de souligner que notre approche de fusion entre GPS et VSLAM n'a pas pour ambition de remplacer un algorithme de SLAM complet, mais de fournir une brique pouvant être exploitée dans le cadre d'un système de localisation plus général. Dans ce qui suit, nous

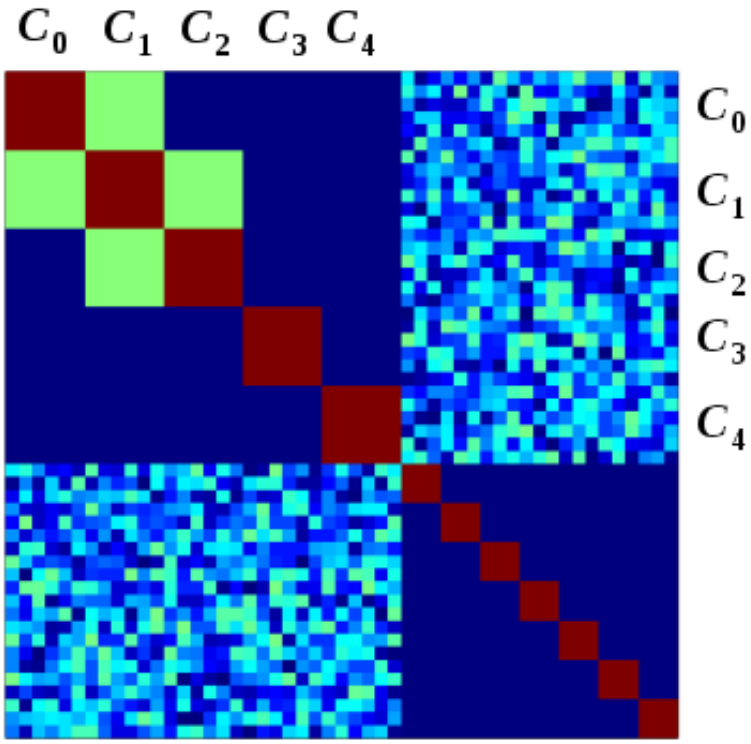


FIGURE 3.3 – Illustration de l'Hessienne d'un exemple avec 2 caméras en graphe de pose et 3 caméras dans la fenêtre du BA.

montrerons que cette fusion correspond à nos attentes : nous montrerons que notre solution, en plus d'être rapide, fait converger les positions des caméras vers les positions GPS sans dégrader les contraintes multivues. Nous montrerons la convergence en comparant nos reconstructions à

Méthode	σ_b	μ_R	σ_R	m_R	μ_T	σ_T	m_T	#Inliers
Fusion BA/GPS	0	0.15	0.01	0.16	0.15	0.02	0.16	10427
	0.75	0.09	0.04	0.09	0.68	0.45	0.80	10422
	1.5	0.40	0.24	0.09	0.45	0.23	0.40	10462
Hypbride avec DPC	0	0.09	0.17	0.04	0.15	0.02	0.16	10415
	0.75	0.12	0.18	0.06	0.63	0.43	0.54	10282
	1.5	0.42	0.26	0.41	0.59	0.9	0.49	10409
Hybride avec MPC	0	0.05	0.04	0.04	0.15	0.02	0.16	10369
	0.75	0.29	0.19	0.27	0.78	0.50	0.69	10360
	1.5	0.15	0.19	0.08	1.26	0.87	1.02	10439

TABLE 3.1 – Comparaison entre les précisions des contraintes CPD et CPM, dans le cas où l'on fait varier l'écart type du bruit du GPS σ_b (en mètres) sur la séquence de synthèse. Le nombre d'inliers restants, calculé sur la séquence entière est aussi indiqué dans la dernière colonne. Notons que les erreurs de translations sont métriques, et que les erreurs de rotation sont exprimées en Radians.

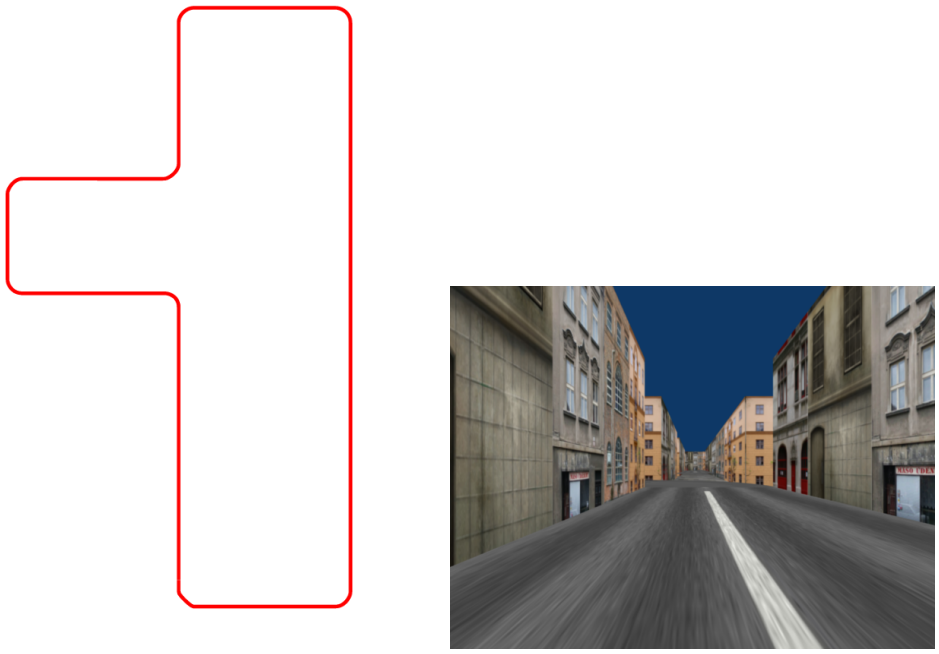


FIGURE 3.4 – (a) Vérité terrain de la séquence de synthèse. (b) Un exemple d'image de cette séquence.

celles que nous avons obtenues en utilisant l'algorithme de [68], et nous tiendrons compte du nombre de points inliers restants après la fusion afin de montrer que les contraintes multivues sont respectées. Nous dirons d'un point qu'il est inlier si toutes ses erreurs de reprojection dans toutes les caméras depuis lesquelles il est observé sont inférieurs à 2 pixels.

Nos expérience ont été conduites sur trois séquences différentes. Le code a été exécuté sur un unique cœur (d'un CPU @ 2.50GHz Intel(R) Core(TM) i7-4710HQ) tourant à 1.6GHz, sous linux (Ubuntu 14.04). Le nombre ce caméras à optimiser à été fixé à $l = 40$ Pour toutes ces expériences. Pour DPC et MPC, nous avons fixé $n = 30$. Notons que l'approche de [68, 67], à laquelle nous nous comparons est un cas particulier de notre formulation (*i.e.* $n = 0$). Malheureusement, nous n'avions pas accès à la vérité terrain des séquences réelles. Nous avons donc comparé les approches mentionnées aux résultats d'un k-BA contraint global (*i.e.* prenant en compte l'ensemble des caméras de la séquence). Ce dernier a été contraint aux modèles $3d$ des bâtiments et aux modèles d'élévation de terrain ([58]).

Méthode	Perte d'inliers (%) par rapport à la fusion BA/GPS
Hybride avec CPD (Séquence de synthèse, sans biais)	0.11
Hybride avec CPD (Séquence de synthèse, biais de $0.75m$)	1.34
Hybride avec CPD (Séquence de synthèse, biais de $1.5m$)	0.50
Hybride avec CPM (Séquence de synthèse, sans biais)	0.55
Hybride avec CPM (Séquence de synthèse, biais de $0.75m$)	0.59
Hybride avec CPM (Séquence de synthèse, biais de $1.5m$)	0.21
Hybride avec CPD (Séquence réelle I)	0.61
Hybride avec CPM (Séquence réelle I)	0.83
Hybride avec CPD (Séquence réelle II)	0.68
Hybride avec CPM (Séquence réelle II)	0.57

TABLE 3.2 – Perte d'inliers (%) par méthode/séquence.

	ratio des moyennes des temps d'exécution
Simulations avec CPD	0.40
Simulations avec CPM	0.66
Séquence réelle I avec CPD	0.42
Séquence réelle I avec CPM	0.66
Séquence réelle II avec CPD	0.45
Séquence réelle II avec CPM	0.72

TABLE 3.3 – Ratios des temps d'exécution. Nous avons calculé $\frac{r}{r_{BA}}$, avec r_{BA} le temps d'exécution pour une itération de la fusion par BA/GPS contraint avec $n = 0$ (*i.e.* la méthode de [68, 67]) et r le temps d'exécution d'une itération de notre méthode. En pratique, $r_{BA} \sim 300ms$. On voit clairement que notre approche permet des gains considérables en terme de coût de calcul.

Dans ce qui suit, la moyenne, l'écart type et la médiane des erreurs de rotation sont respectivement notées μ_R,σ_R et m_R. De manière similaire, μ_T,σ_T et m_T désignent de manière respective la moyenne, l'écart type et la médiane de l'erreur de translation.

Séquence réelle I							
Method	μ_R	σ_R	m_R	μ_T	σ_T	m_T	#inliers
fusion BA/GPS	0.20	0.02	0.29	1.48	0.93	1.43	18670
Hypbride avec CPD	0.24	0.09	0.29	1.58	1.40	1.10	18555
Hybride avec CPM	0.15	0.05	0.15	1.27	1.26	0.88	18515
Séquence réelle II							
Method	μ_R	σ_R	m_R	μ_T	σ_T	m_T	#inliers
fusion BA/GPS	0.18	0.02	0.19	0.18	0.79	1.62	21785
Hypbride avec CPD	0.15	0.02	0.16	0.19	0.80	2.01	21642
Hybride avec CPM	0.21	0.02	0.20	0.24	0.88	1.67	21656

TABLE 3.4 – Comparaison de la précision et du nombre d'inliers sur les deux séquences réelles. Les erreurs de translation sont données en mètres, et les erreurs de rotation en Radians. Notons que les erreurs de translations sont élevées dans tout les cas à cause d'un biais très important des positions GPS (jusqu'à $4m$ de biais).

3.7.1 Temps d'éxécution et précision

3.7.1.1 Séquence synthétique

La vérité terrain de cette séquence ainsi qu'un exemple d'image sont illustrés par la figure 3.4. La longueur de la trajectoire est de 640m. Nous évaluons les trois approches mentionnées précédemment (CPD, CPM et celle de [68]) sur des données non-bruitées afin de valider notre méthode, et sur des données bruitées afin d'évaluer la robustesse des algorithmes en question. Les données bruitées ont été obtenues en rajoutant du bruit Gaussien (tiré d'une distribution

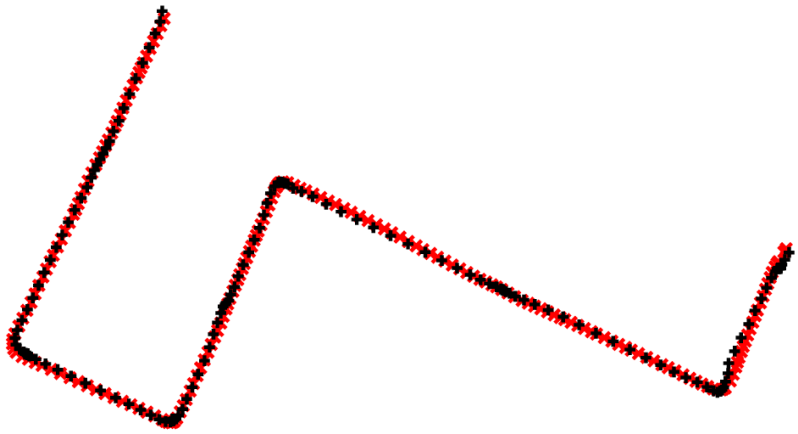


FIGURE 3.5 – (Séquence réelle I) Comparaison les donnés GPS et la trajectoire reconstruite par notre méthode. Les croix rouges et noires indiquent les positions des camera et les positions GPS (respectivement).

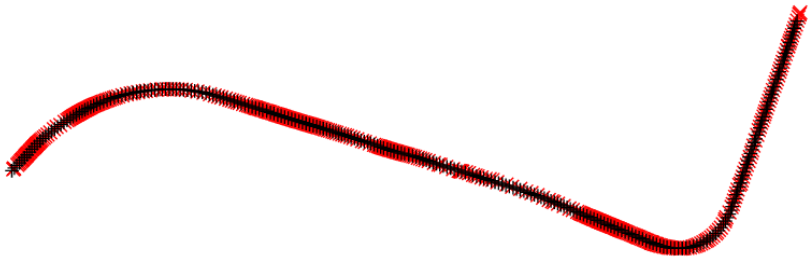


FIGURE 3.6 – (Séquence réelle II) Comparaison les donnés GPS et la trajectoire reconstruite par notre méthode. Les croix rouges et noires indiquent les positions des camera et les positions GPS (respectivement).

$\mathcal{N}(0, \sigma_b)$, où σ_b varie d'une expérience à l'autre) à la vérité terrain. Les résultats peuvent être consultés dans la table 3.1, et la perte d'inlier est donnée dans la table 3.2. Comme nous nous y attendions, les trois approches empêchent la fusion avec des données bruitées grâce à la formulation du terme barrière qui permet d'éviter les dégradations importantes du terme de vision. Toutes trois semblent être équivalentes dans tout les cas. Le nombre de points inliers perdu par CPD et CPM est de l'ordre de $\sim 1\%$. Pour $\sigma = 0.75$, CPD semble être plus précise que les autres méthodes : son erreur de translation et la médiane de son erreur d'orientation sont moins importantes. Notons que CPM semble faire preuve de moins de robustesse par rapport au bruit Gaussien d'amplitude $\sigma_b = 1.5m$. Quoi qu'il en soit, il est très invraisemblable de rencontrer un bruit Gaussien aussi important en pratique (contrairement au biais qui peut atteindre plusieurs dizaines de mètres). Par conséquent, et compte tenu des gains considérables en temps de calcul (table 3.3) obtenus avec CPD et CPM, nous pouvons considérer que ces pertes de précision sont insignifiantes.

3.7.1.2 Séquence réelle I

Cette séquence vidéo a été enregistrée à Versailles sur une longueur de $718m$. Il s'agit donc d'un environnement urbain typique où plusieurs occultations sont présentes, et où le biais du GPS peut prendre des valeurs assez élevées (jusqu'à $4.0m$ sur cette séquence). Le ratio entre les temps d'éxectution est donné dans la table 3.3. On voit que la réduction en temps de calcul offerte par CPD est d'environ 58%. CPM permet une réduction considérable, mais moins importante, de 34%. Les données GPS et la trajectoire reconstruite par notre méthode sont illustrées par la figure 3.5. Les résultats quantitatifs sont regroupés dans les tables 3.4 et 3.2. Il est important de souligner que les erreurs sont calculées par rapport à la pseudo vérité terrain fournie par un k-BA *global* contraint aux modèles $3d$ des bâtiments et aux modèles d'élévation de terrain [58]. D'après les erreurs de translation, il est clair que les trois méthodes sont affectées par le biais élevé du GPS. En général, concernant la translation, nos approches (CPM et CPD) présentent une médiane d'erreur moins élevée, mais la moyenne et la dispersion de leurs erreurs sont plus importantes. On remarque que ces résultats sont cohérents avec la perte d'inliers ($\sim 0.6\%$). Les erreurs d'orientation sont plus ou moins équivalentes.

3.7.1.3 Séquence réelle II

Cette séquence a été enregistrée sur une longueur de $647m$. Il s'agit d'une caméra placée en hauteur sur l'un des cotés d'un tramway, et son axe optique est approximativement orthogonal à la trajectoire. Une fois de plus, nous substituons le résultat d'un k-BA *global* contraint (aux modèles 3d et aux modèles d'élévation) à la vérité terrain. Les résultats quantitatifs peuvent être consultés dans les tableaux 3.4, 3.2, et la figure 3.6 permet de comparer les données GPS et la trajectoire reconstruite par notre méthode de manière visuelle. Le biais du GPS est beaucoup moins élevé que pour la séquence précédente, et par conséquent, les erreurs de translation des trois méthodes sont beaucoup moins importantes. Les erreurs de translation et d'orientation semblent être équivalentes pour tout les cas, et la perte d'inliers est insignifiante ($< 1\%$). Compte tenu des temps d'exécution (table 3.3), nous pouvons conclure de CPD est la méhode la plus appropriée pour les déploiements sur les plateformes de faible capacité.

3.7.2 Empreinte mémoire

Il est important de noter que notre approche réduit les exigences en terme de mémoire de manière significative (figure 3.7). Bien que de telles économies en mémoire ne soient d'aucun intérêt pour un ordinateur portable ou personnel moderne, les implémentations visant les plateformes embarquées ne peuvent en général pas se permettre d'allouer plus de quelques mega-octets pour la fusion VSLAM/GPS.

3.8 Résumé, limites et perspectives

Nous avons proposé dans ce chapitre une méthode de fusion entre les données GPS et le VSLAM pour les plate-formes embarquées de faible capacité en terme de mémoire et de capacité de calcul. Notre solution repose sur une nouvelle définition de l'erreur employée dans le terme barrière. En particulier, nous avons :

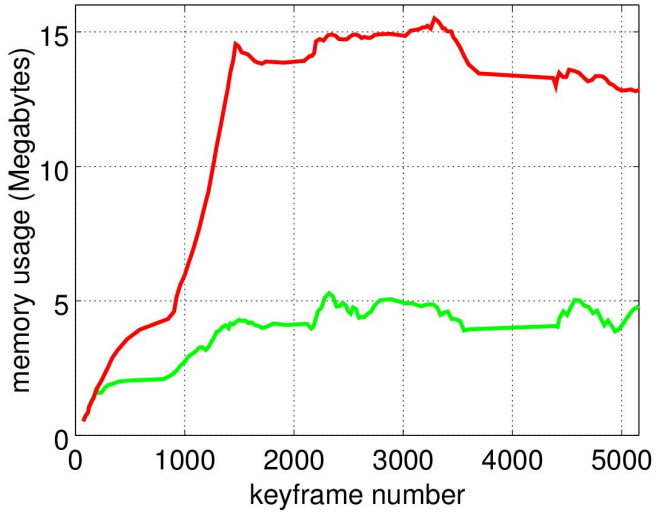


FIGURE 3.7 – Comparaison de l'empreinte mémoire de l'Hessienne de DPC sur la deuxième séquence réelle. Nous omettons les courbes pour les autres séquences, car elles sont très proches de celles-ci. En vert : mémoire utilisée par notre approche avec $n = 30$. En rouge : mémoire utilisée par la fusion BA/GPS [68] (*i.e.* $n = 0$). Notre approche réduit la consommation en mémoire d'un facteur 3.

- ▷ démontré de manière théorique que notre approche possède certaines propriétés garantissant son fonctionnement avec le terme barrière.
- ▷ présenté deux formulations différentes, la première ayant pour but l'optimisation maximale, et la deuxième une pondération intuitive des termes d'erreur.
- ▷ montré de manière expérimentale que nos deux formulations permettent à l'estimation de converger vers les données GPS sans perte significative d'inliers (*i.e.* sans dégradation importante des contraintes multivues).
- ▷ montré de manière expérimentale que notre approche est beaucoup plus rapide,et qu'en particulier, CPD permet de réduire le temps de calcul d'environ 60%.
- ▷ montré que l'empreinte mémoire de notre solution est plus faible (d'une facteur 3).

.

Bien que ces résultats soient encourageants, il existe plusieurs voies d'amélioration. Premièrement, bien que nous n'ayons pas rencontré de difficultés liées à l'initialisation de la matrice de pondération de la contrainte DPC (équation 3.16) et que dans nos expériences, l'initilisation par la matrice identité produise de bons résultats, nous pensons que cela est principalement dû aux propriétés des séquences en question, et qu'il est probable en pratique que cette matrice puisse jouer un rôle important dans le succès ou l'échec de la fusion dans certains contextes. La suite logique de ces travaux serait donc d'établir un lien précis entre l'incertitude des paramètres de rotation et de translation afin de pouvoir définir cette matrice de manière dynamique. Cependant, l'approche naïve qui consisterait en une propagation (vers l'avant, voir §1.3.3) ne semble pas pertinente, car le biais du GPS n'étant pas à ce stade de nos recherches correctement

modélisé (le biais reste inconnu), l'aspect incrémental revêt une importance primordiale. Nous pensons donc qu'il serait raisonnable, dans nos travaux futurs, de tenter de coupler cette approche à une approche inspirée de la propagation d'incertitue dans le k-BA proposée par [21]. En outre, nous pensons qu'il serait avantageux de rajouter des M-estimateurs aux fonctions de coût définies par les équations 3.18 et 3.16.

Comme nous l'avons souligné au début de ce chapitre, nos conclusions théoriques ne sont valides que dans le cas où la contrainte n'est imposée que sur la caméra la plus récente, et les résultats expérimentaux que nous avons présentés respectent cette condition. Nos investigations préliminaires ne nous permettent pas de conclure de manière théorique que la covariance de la caméra plus récente augmente (*i.e.* que l'incrément retourné par l'algorithme de Levenberg Marquardt décroit) lorsque l'on contraint plus d'une caméra. Cependant, nous observons en pratique que la fusion ne semble pas souffrir de l'augmentation du nombre de caméras. Intuitivement, il semblerait que seule la contribution d'un nombre minoritaire de caméras à l'incrément soit positive. Il existe donc, d'un point de vue théorique, un certain intérêt à formaliser ces problèmes et d'arriver à une conclusion définitive. Notons pour finir que bien que nous ayons pu déterminer expérimentalement qu'un nombre de 30 caméras en graphe de pose et 10 caméras dans la fenêtre de l'ajustement de faisceaux produisait des résultats comparables à ceux d'une fusion sans graphe de pose avec approximativement 40 caméras, nous n'avons pas déduit ce lien entre ces nombres de manière théorique. Il s'agit donc à ce stade de méta-paramètres dont la conversion automatique pourrait rendre l'approche plus attrayante. Il serait par conséquent aussi intéressant de mener des travaux théoriques dans cette direction.

Notons qu'un système de SLAM exploitant les informations de positionnement du GPS doit être robuste face aux pertes temporaires du signal GPS. C'est en partie la raison pour laquelle la fusion avec les modèles 3d des bâtiments est un thème central de nos recherches. Nous consacrons le chapitre suivant à nos effort de robustification (face aux occultations) des contraintes que définissent ces derniers.

CHAPITRE 4

Robustification de la contrainte aux bâtiments pour un SLAM à grande échelle et sans dérive

La contrainte aux bâtiments, telle qu'elle a été présentée au chapitre 2, souffre du problème de l'association de données erronée, notamment à cause de l'utilisation de critères géométriques simples. En effet, les façades des bâtiments sont souvent occultées (e.g. par des arbres, panneaux publicitaire, véhicules, etc) dans les environnements urbains. Dans ces conditions, un critère ne prenant en compte que l'aspect géométrique en ignorant toute information photométrique produira en pratique un nombre d'associations aberrantes trop élevé pour permettre d'imposer une contrainte correcte sur la reconstruction, même lorsque les résultats sont filtrés par un M-estimateur ou un RANSAC. C'est à ce problème que nous apportons une solution dans ce chapitre.

4.1 Motivations

Nous avons vu précédemment (chapitre 2) que la formulation de la contrainte aux bâtiments de [118, 58, 60], donnée par l'équation 2.4, et que nous reprendrons ici, nécessite la segmentation du nuage de points $3d$ afin de déterminer le sous-ensemble de celui-ci qui correspond aux bâtiments. Une fois l'appartenance à la classe bâtiment établie, une étape supplémentaire d'association est nécessaire afin de mettre en correspondance chaque points $3d$ avec le plan du maillage qui lui correspond. Dans [118, 60, 58, 59], la segmentation est effectuée via un simple lancé de rayon, et l'association point/plan se fait par l'intermédiaire d'un critère de proximité : chaque point est associé au plan le plus proche. Effectivement, l'utilisation de noyaux robustes ([118]), que nous avons noté ρ dans l'équation 2.4 atténue l'effet des outliers, et permet d'établir des contraintes robustes, qui cependant peuvent être perturbées lorsque les occultations dans l'image dépassent un certain seuil (voir la figure 4.1). Quelques exemples illustratifs de séquences réelles non-problématiques et problématiques sont présentés dans la figure 4.2. Les images (b) et (c) de cette figure sont tirées d'une séquence où ce type d'occultation est omniprésent, et la segmentation géométrique suivie de l'association au plan le plus proche conduit à de fausses associations faisant dériver le SLAM (figure 4.3).

Nos objectifs dans ce chapitres sont

▷ De réduire le bruit dans la segmentation en exploitant un critère prenant en compte la géométrie et l'information photométrique des images de manière conjointe.

▷ D'éliminer les fausses contraintes en présentant une solution bayésienne simple mais efficace.

Concernant le premier point, nous exploiterons des réseaux de neurones afin de segmenter chaque image, et nous propagerons les résultats au nuage de points. Notons cependant que notre objectif n'est pas d'améliorer l'état de l'art en terme de segmentation par pixel, mais d'exploiter les architectures existantes afin d'améliorer la contrainte aux bâtiments. En ce qui concerne le deuxième point, c'est à dire la solution bayésienne, elle repose simplement sur le calcul de la probabilité de l'appartenance de chaque point $3d$ classifié comme étant de la classe bâtiments à un plan du modèle. Un a priori nous est donné par la distance entre chaque point $3d$ et le plan le plus proche. Nous le mettrons à jour en exploitant une vraisemblance sur la structure locale de l'environnement en ce point.

Il est important de noter que la fusion entre le SLAM visuel et la contrainte aux bâtiments peut avoir comme résultat la perte d'un grand nombre d'inliers. En effet, la situation est semblable à celle à laquelle nous devions faire face lors de la fusion VSLAM/GPS : les incertitudes peuvent être grandes, et leur modélisation par une distribution unimodale ne serait pas raisonnable. Comme nous l'avons souligné au chapitre 1 (§1.9.1), ces erreurs découlent non seulement des problématiques inhérentes aux algorithmes ayant servi à la construction des modèles, mais aussi de retards dans la mise à jour de la base (destruction ou construction d'un ensemble de bâtiments depuis la construction de la base). Afin d'éviter la dégradation trop importante des contraintes multivues et l'échec des appariements 3d/2d en présence de telles aberrations, nous exploiterons les contraintes au bâtiment dans la cadre d'une fusion par terme barrière. La fusion de données dans ce chapitre prendra donc la forme

$$\underset{\mathbf{x}}{argmin} \quad (\frac{\gamma}{t-\zeta(\mathbf{x})} + C_1(\mathbf{x}) + ... + \sum_{q \in \mathcal{Q}} \rho(d(q, p_q)) + ... + C_h(\mathbf{x})) \tag{4.1}$$

où la contrainte aux bâtiments présentée précédemment au chapitre 1 (§2.4) remplace l'un des C_i de l'équation 2.3.

Ainsi que nous l'avons indiqué aux chapitre 3 (§3.2), ce type de fusion nécessite la considération des erreurs d'un plus grand nombre de caméras dans la définition de ζ, ce qui implique un coût supplémentaire. Compte tenu du nombre de paramètres élevé de la plupart des réseaux, le choix d'une architecture simplifiée était d'une importance critique afin de pouvoir effectuer les estimations en un temps raisonnable (proche du temps réel ou temps réel). Les réseaux de l'état de l'art au début de ces travaux de thèse étaient en général très exigeants en termes de temps de calcul. Nos premiers efforts ($\sim$ début 2015 à 2016) ont donc dû être orientés vers l'obtention d'un compromis entre temps de traitement et précision via la simplification d'une architecture dont nous discuterons dans la section suivante. Les résultats obtenus étant extrêmement bruités, nous avons présenté une approche de filtrage des faux positifs reposant sur la modélisation de la distribution continue des distributions discrètes conditionnelles d'appartenance d'un pixel à une classe. L'émergence en 2016 d'architectures plus légères et plus performantes (*e.g.* [4, 95]) nous ont permis ensuite de nous affranchir de ce filtrage, et de modifier très légèrement l'un de ces réseaux dans le but d'inférer de manière conjointe des informations sémantiques et géométriques, tout en respectant les contraintes temps-réel grâce à une parallélisation GPU/CPU.

4.2 Première tentative (2015) : intégration d'une segmentation sémantique bruitée dans le BA contraint aux bâtiments

Dans cette section, nous rajouterons un poids pour chaque point intervenant dans la contrainte aux bâtiments présentée dans les équations 2.4 et 4.1. Nous l'exprimerons donc par

$$C_i = \sum_{q \in \mathcal{Q}} W_q \rho(d(q, p_q)) \tag{4.2}$$

où W_q est un poids.

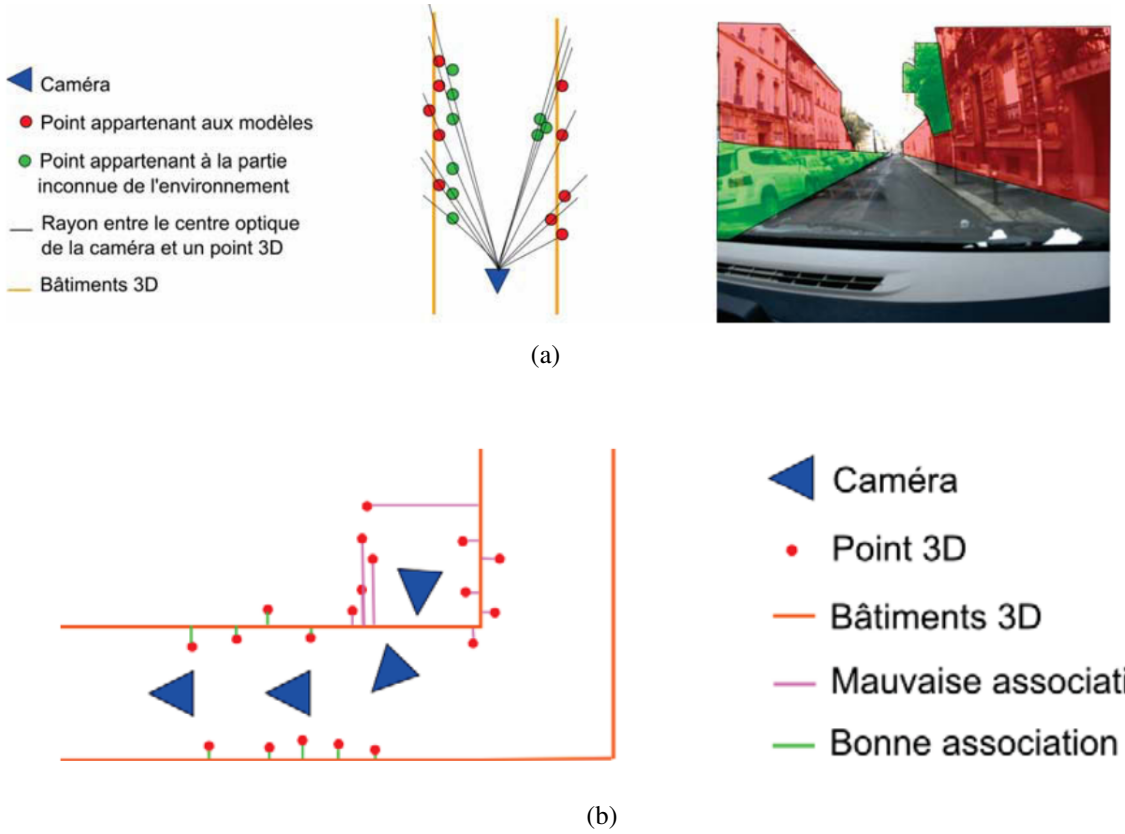


FIGURE 4.1 – Illustration des limites des méthode de segmentation/association basées sur le lancé de rayon et un critère de proximité, tirées de [56]. (a) On voit clairement que le lancer de rayon classifie les points $3d$ appartenant aux véhicules comme des points appartenant aux façades des bâtiments. (b) L'association au plan le plus proche entraîne de fausses contraintes, qui font immédiatement dériver l'estimation.

4.2.1 Architecure du réseau

Ainsi qu'il a été souligné à la section précédente, les approches de segmentation sémantique par pixel de l'état de l'art au début de cette thèse (∼2015) étaient très exigeantes en terme de mémoire GPU et en temps de calcul. Il y avait d'une part les méthodes graphiques, par exemple basées sur des CRFs ([54, 103]) ; et d'autres part, les méthodes d'apprentissage profond

(a) (b)

(c) (d)

(e) (f)

FIGURE 4.2 – Première ligne : exemple d'images où la segmentation et l'association de données basées sur des critères géométriques produisent de bons résultats lorsque les résultats sont filtrés par un M-estimateur ou un RANSAC. Deuxième et troisième lignes : exemples tirés de séquence où cette approche échoue. Notre objectif est de pouvoir imposer une contrainte aux bâtiment fiable même dans ces conditions.

FIGURE 4.3 – La segmentation par lancer de rayon suivie de l'association au plan le plus proche peut conduire à de fausses contraintes. Les points ayant été classifiés comme des points de la classe bâtiments sont de couleur dorée, tout les autres points sont en vert.

reposant sur des architectures lourdes de type AlexNet ([53]) ou VGG ([110]). Il est notoire que les approches d'optimisation par graphe sont inadaptées au temps-réel de par leur aspect combinatoire. De la même manière, la plupart des architectures profondes étaient définies par un trop grand nombre de paramètre. A titre d'exemple, le nombre de paramètre de l'architecture VGG-16 est d'environ $\sim 144e6$ (figure 4.4). Un tel nombre de paramètres rend non seulement la backpropagation, mais aussi la pass forward trop coûteuse à effectuer dans le cadre d'un SLAM.

Nous nous sommes donc inspirés de l'architecture présentée dans les travaux de [23]. Dans le papier d'origine, il s'agit d'un réseau convolutionnel constitué de trois branches, chacune opérant en parallèle sur des échelles différentes. Les architectures de celles-ci sont identiques, et ne présentent aucune particularité par rapport à une cascade naïve de convolution/maxpooling/non-linéarité que la specification des connexions entre les plans d'entrée et sortie des modules de convolution. Cette dernière caractéristique permet une réduction importante du nombre des paramètres. Cependant, les résultats de la segmentation par ce réseau sont extrêmement bruités et spatialement incohérentes. Les auteurs ont recours à des régularisations basées sur des représentations en sous forme de graphe qui sont trop coûteuses pour être déployées en temps réel. Afin de garantir un fonctionnement proche du temps-réel, nous avons fait le choix d'exploiter l'architecture simple de leurs travaux, mais en ignorant l'aspect multi-échelle. Celle-ci est donnée dans la figure 4.5.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

FIGURE 4.4 – L'architecture VGG.

4.2.2 Filtrage des faux positifs

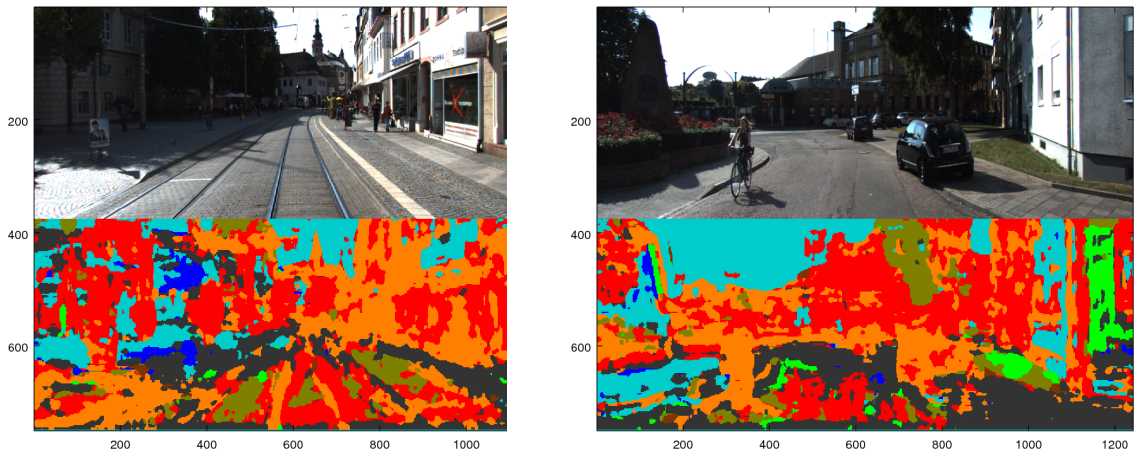
L'architecture minimaliste choisie avait pour objectif de fournir un a priori sur le contenu sémantique de la scène en privilégiant la rapidité d'exécution. Il n'est donc pas surprenant que la segmentation obtenue soit extrêmement bruitée (figure 4.6).

Comme nous le savons, la dernière couche FC (Fully connected, aussi dite linéaire) d'un réseau convolutionnel classique a pour rôle de calculer des scores d'appartenances à chaque classe. Le SoftMax rajouté permet une interprétation probabiliste : la somme des score est normalisée, et ils deviennent tous positifs. On peut donc dire qu'un tel réseau attribue une distribution discrète P_x à chaque pixel x de l'image, tel que $P_x(i)$ est la probabilité d'appartenance du pixel x à la classe i. Par conséquent il est en général raisonnable de classifier le pixel x comme appartenant à la classe indexée par $\underset{i}{argmax}\, P_x(i)$. Cependant, ceci ne prend pas en compte la forme de la distribution. Comme nous l'avons vu dans les figures 4.6 (a) et (b), beaucoup de faux positifs sont détectés. Par exemple, un grand nombre de pixels correspondants à la chaussée sont labélisés comme étant du bâtiment dans ces images. Afin de les filtrer, nous allons tenter de prendre la forme de la distribution attribuée à chaque pixel par le réseau, ainsi que l'illustre la figure 4.7. En d'autre termes, nous attendons de la distribution $P(x|x$ appartient à la classe bâtiment$)$ qu'elle prenne le plus souvent un aspect particulier, par exemple les mêmes modes dans la plupart des cas. Nous pourrons ensuite filtrer les distributions trop distantes de cette espérance.

Afin d'atteindre cet objectif, nous modélisons la distribution (continue) des $P(x|x$ appartient à la classe bâtiment$)$ par une loi de Dirichlet. Nous soulignons qu'il ne

```
(1): SpatialConvolutionMap
(2): ReLU
(3): SpatialMaxPooling(2x2, 2,2)
(4): SpatialConvolutionMap
(5): ReLU
(6): SpatialMaxPooling(2x2, 2,2)
(7): SpatialConvolutionMap
(8): Reshape(256)
(9): Linear(256 -> 512)
(10): ReLU
(11): Linear(512 -> 8)
(12): LogSoftMax
```

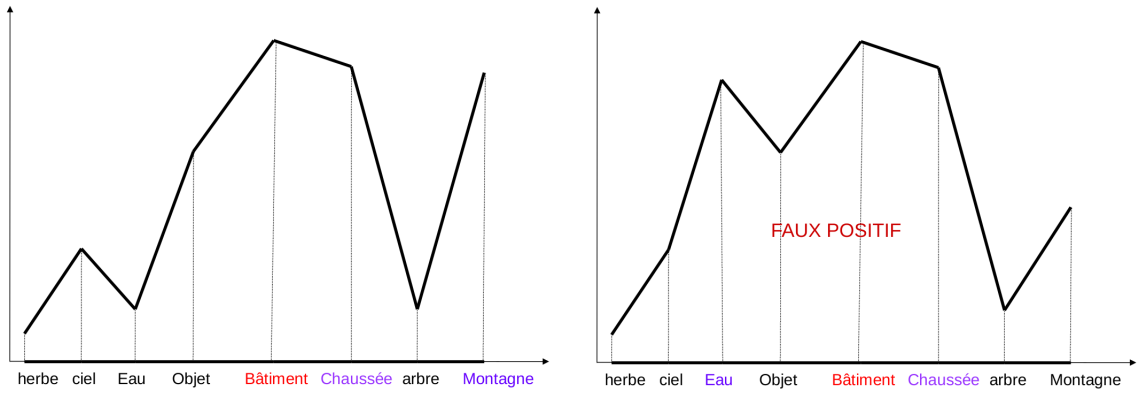
FIGURE 4.5 – Architecture du réseau convolutionnel séquentiel utilisé. Comme on peut le constater, il s'agit de la forme la plus simple que peut prendre un réseau de ce type. Le principal intérêt est le nombre de paramètres réduit. La terminologie employée étant empruntée à la librairie Torch7, nous précisons que "SpatialConvolutionMap" désigne une couche du réseau permettant de spécifier les connexions entre les couches d'entrée et de sortie. Le module "Reshape", ainsi que son nom l'indique, ne fait que réordonner le tenseur de sortie.



(a) (b)

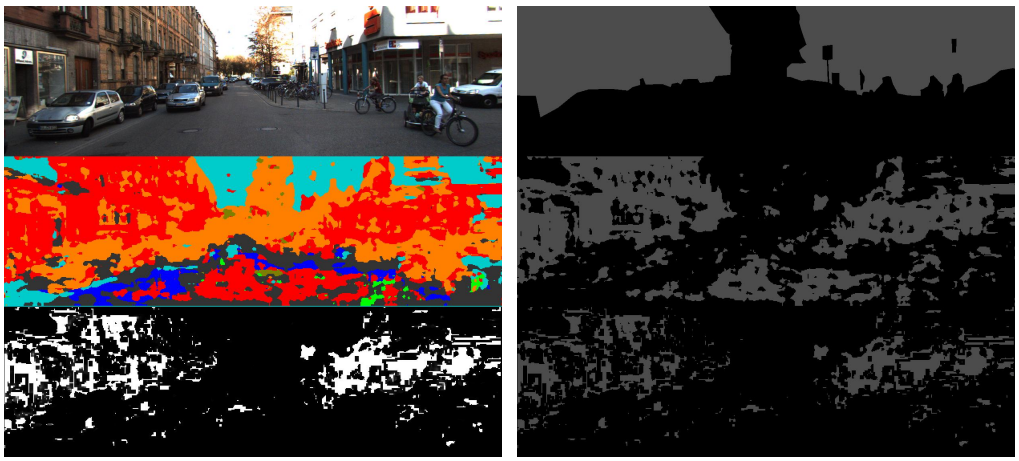
FIGURE 4.6 – Deux exemples de segmentation obtenue avec l'architecture minimaliste présentée plus haut. Première ligne : image d'entrée. Seconde ligne : résultats de la segmentation. Les pixels ayant été classifiés comme appartenant aux bâtiments sont en rouge. La classe objet est en orange. Outre la confusion entre ces deux classes, on remarque plusieurs incohérences spatiales, notamment la détection de pixels appartenant à la classe bâtiment sur le sol. La régularisation proposée par [23] était trop coûteuse pour être applicable en temps réel.

s'agit *pas* de remplacer le réseau par cette distribution (ce qui n'aurait aucun sens), mais de définir une fonction à valeurs réelles sur l'espace des distributions discrètes prédites par le



(a) (b)

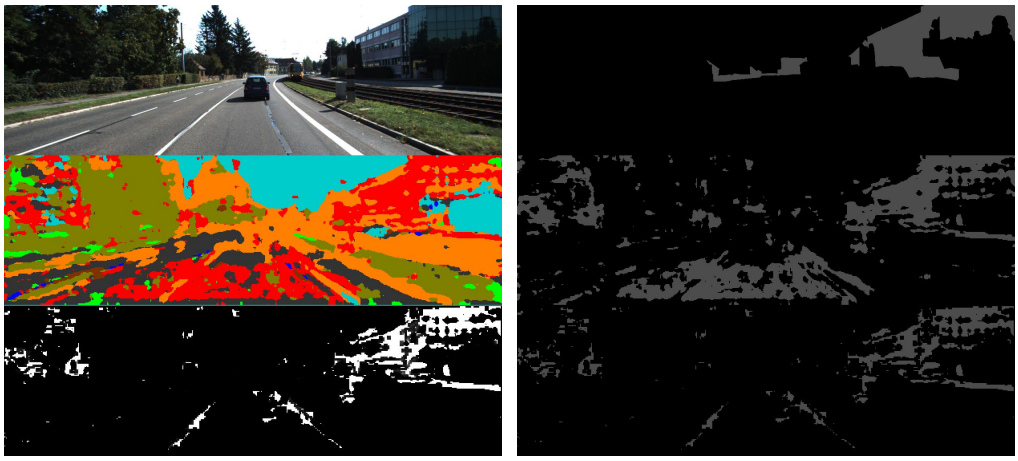
FIGURE 4.7 – Motivation de notre méthode de filtrage. Nous cherchons à apprendre la forme générale que prend la distribution $P(x|x$ appartient à la classe bâtiment$)$ afin de pouvoir filtrer les faux positifs. Par exemple, supposons que (a) soit la forme typique de cette distribution, où la classe "bâtiment" est souvent confondue avec la classe "montagne". Dans ce cas il n'est pas vraisemblable qu'une distribution de type (b) confondant "eaux" et "bâtiments" soit issue d'un pixel appartenant véritablement à la classe bâtiment. Exploiter l'ensemble des valeurs de la distribution permettrait de détecter et de filtrer ce type de faux positifs, alors que prendre simplement l'*argmax* produirait une classification erronée.



(a) (b)

FIGURE 4.8 – Exemple de notre filtrage sur une image. (a, première ligne) : image d'origine. (a, deuxième ligne) : Résultat de la segmentation du réseau lorsque la classification se fait en prenant l'*argmax*. Les pixels classifiés comme appartenant à du bâtiment sont en rouge. (a, dernière ligne) Les pixels retenus comme appartenant à la classe bâtiment à l'issue de notre filtrage. (b, première ligne) La vérité terrain pour la classe bâtiment. (b, deuxième ligne) Les pixels classifiés comme bâtiment sans notre traitement. (b, dernière ligne) Les pixels retenus après notre filtrage.

réseau. En d'autres termes, on considère chacune des ces distributions comme une variable aléatoire, et l'on apprend la distribution du sous ensemble d'entre elles qui correspondent



(a) (b)

FIGURE 4.9 – Autre exemple de notre filtrage. (a, première ligne) : image d'origine. (a, deuxième ligne) : Résultat de la segmentation du réseau lorsque la classification se fait en prenant l'*argmax*. Les pixels classifiés comme appartenant à du bâtiment sont en rouge. (a, dernière ligne) Les pixels retenus comme appartenant à la classe bâtiment à l'issue de notre filtrage. (b, première ligne) La vérité terrain pour la classe bâtiment. (b, deuxième ligne) Les pixels classifiés comme bâtiment sans notre traitement. (b, dernière ligne) Les pixels retenus après notre filtrage.

à des pixels véritablement appartenant à du bâtiment. Etant donné un ensemble de données d'apprentissage labéllisé, on peut définir l'échantillon

$$D_{\text{build}} = \{P_i \mid i \text{ appartient à du bâtiment dans l'image}\} \tag{4.3}$$

Le problème est maintenant d'estimer les paramètres $\alpha = \{\alpha_1, ..., \alpha_k\}$ (voir la définition de la distribution au chapitre 1, §1.70) permettant la meilleure explication des données. Nous pouvons exprimer cela sous forme d'un problème de maximum de vraisemblance :

$$\underset{\alpha}{argmax} \prod_{X \in D_{\text{build}}} \mathfrak{D}(X|\alpha) \tag{4.4}$$

où $\mathfrak{D}$ est la densité de Dirichlet (1.70) de paramètres α. Afin d'éviter les problèmes liés à l'underflow, et aussi pas soucis de concision, nous résolvons le problème équivalent

$$\underset{\alpha}{argmin}\{-\ln(\prod_{X \in D_{\text{build}}} \mathfrak{D}(X|\alpha))\} \tag{4.5}$$

Quelques manipulations algébriques de base nous permettent de réécrire cette équation comme

$$\underset{\alpha}{argmin}\{m \ln(\beta(\alpha)) + \sum_{i=1}^{k}(1-\alpha_i)\ln(t_i)\} \tag{4.6}$$

où $t_i = \prod_{q \in D_{\text{build}}} q(s)$ et où k est le nombre de classes. Comme $\alpha_i \geq 0$ pour tout i d'après la définition de la densité de Dirichlet, nous rajoutons k termes agissant comme un terme barrière afin d'empêcher les valeur de α_i de devenir négatives. Nous avons remarqué dans nos

expériences que la fonction de coût convergeait plus facilement si ces termes étaient de nature exponentielle, c'est pourquoi nous définissons la fonction de coût finale par :

$$C(\alpha) = m \ln(\beta(\alpha)) + \sum_{i=1}^{k}(1-\alpha_i)\ln(t_i) + \lambda \sum_{i=1}^{k} e^{-\alpha_i} \tag{4.7}$$

avec $\lambda \in \mathbb{R}^+$ un poids permettant d'influencer l'impact des exponentielles. La Jacobienne est donnée par

$$\frac{\partial C}{\partial \alpha_l} = m(\psi_0(\alpha_l) - \psi_0(\sum_{i=1}^{k}\alpha_i)) - \ln(t_l) - \lambda e^{-\alpha_l} \tag{4.8}$$

où ψ_0 est la fonction digamma. Afin d'éviter les expressions symboliques excessivement longues lors de calcul de l'Hessienne, nous avons fait le choix d'apprendre les paramètres α en minimisant la fonction de coût 4.7 avec l'algorithme l-BFGS (1.2.3), qui ne calcule pas l'Hessienne mais utilise une représentation parcimonieuse (donc efficace en terme de mémoire) permettant de l'approximer. Cet algorithme est sensible à l'initialisation, mais dans notre cas quelques essais avec une initialisation aléatoire on suffit pour converger.

Nous avons utilisé environ $2e6$ pixels labélisés (tirés uniformément) de 70 images de la base kitti annotées par [132]. Notons que cet apprentissage est rapide (compte tenu du nombre insignifiant de paramètres) : il peut se faire en quelques secondes.

4.2.3 Propagation de la segmentation 2d aux points 3d

Chaque point Z de la carte a été observé plusieurs fois (sinon, il n'aurait pas pu être triangulé). Notons $\{I_0, I_1, .., I_n\}$ Les key-frames où Z est observé, et $\{z_0, ..., z_n\}$ ses observations $2d$. Nous cherchons à déterminer la classe à laquelle Z appartient. Le réseau de neurone et le filtrage présentés sont exécutés pour chaque image clefs. Cela produit une distribution par observation de Z. Nous noterons l'ensemble de ces distributions $M = \{P_0, P_1, ..., P_n\}$ dans ce paragraphe. En pratique, ces distributions peuvent être différentes. Formellement, cette différence peut s'exprimer par une divergence de Kullback-Leibler (§1.6.4)importante entre celles-ci. Nous les combinons de la manière la plus simple, c'est à dire en calculant une distribution moyenne

$$P_Z = \frac{1}{N_c}\sum_{i=1}^{n} P_i \tag{4.9}$$

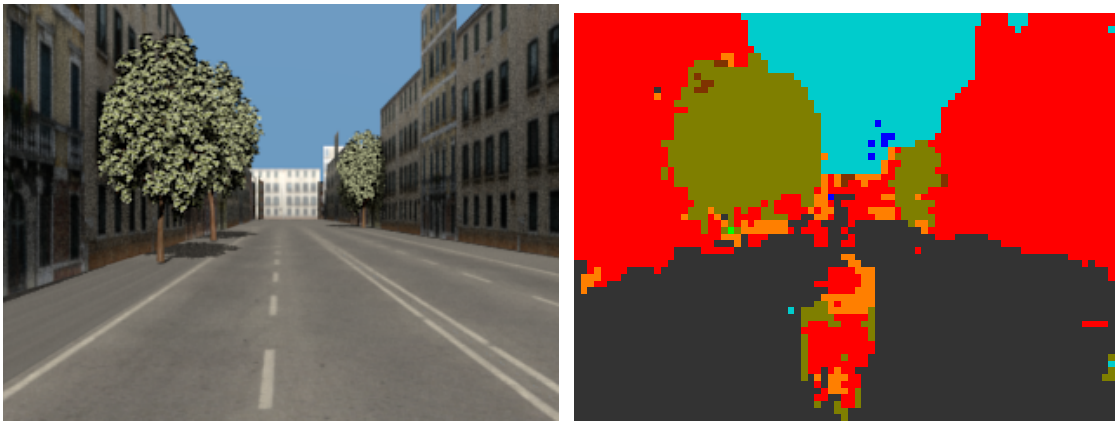
Ensuite, nous classifions Z comme appartenant à l'ensemble des points $3d$ appartenant aux bâtiments si et seulement si $\mathfrak{D}(P_Z) > t_{\mathfrak{D}}$, où $t_{\mathfrak{D}}$ est un seuil. Finalement, on pose $W_q = \mathfrak{D}(P_Z)$ dans l'équation 4.2.

4.2.4 Évaluation expérimentale de la première approche

4.2.4.1 Évaluation de l'approche de filtrage

Nous évaluerons premièrement notre approche de filtrage. Comme celle-ci n'est pas très performante et qu'elle n'a plus d'intérêt pratique compte tenu de l'état de l'art actuel, notre évaluation sera brève, et nous nous contenterons de démontrer son applicabilité sur un sous-ensemble des images de kitti, annotées par [132], qui n'ont pas été utilisées dans l'apprentissage

(50 images). Nous nous somme servis de la vérité terrain correspondant à chaque image afin de déterminer si la prise en compte de la forme de la distribution plutôt que l'*argmax* avait un effet sur le nombre de faux positifs et de faux négatifs. Deux exemples illustratifs sont donnés dans les figure 4.8, 4.9. Dans les figures 4.8 (a) et 4.9 (a), la première ligne est l'image requête, la deuxième est l'image segmentée par le réseau. Les pixels classifiés comme appartenant à la classe bâtiment sont en rouge. On constate qu'il existe un nombre considérable de faux positifs, notamment sur la chaussée. La troisième ligne des figures 4.8 et 4.9 illustre l'ensemble de pixels retenu comme appartenant à du bâtiment à l'issue de notre post-traitement. Bien qu'un certain nombre de classifications correctes aient été éliminées, un nombre considérable de faux positifs on été filtrés. Les figures 4.8 (b) et 4.9 permettent de comparer la vérité terrain (première ligne) avec les résultats de la segmentation sans et avec notre filtrage (deuxième et troisièmes lignes respectivement). Dans l'ensemble, notre méthode permet d'éliminer 70% des faux positifs, en augmentant le nombre de rejets erronés d'environ 15% sur l'ensemble des 50 images utilisées.



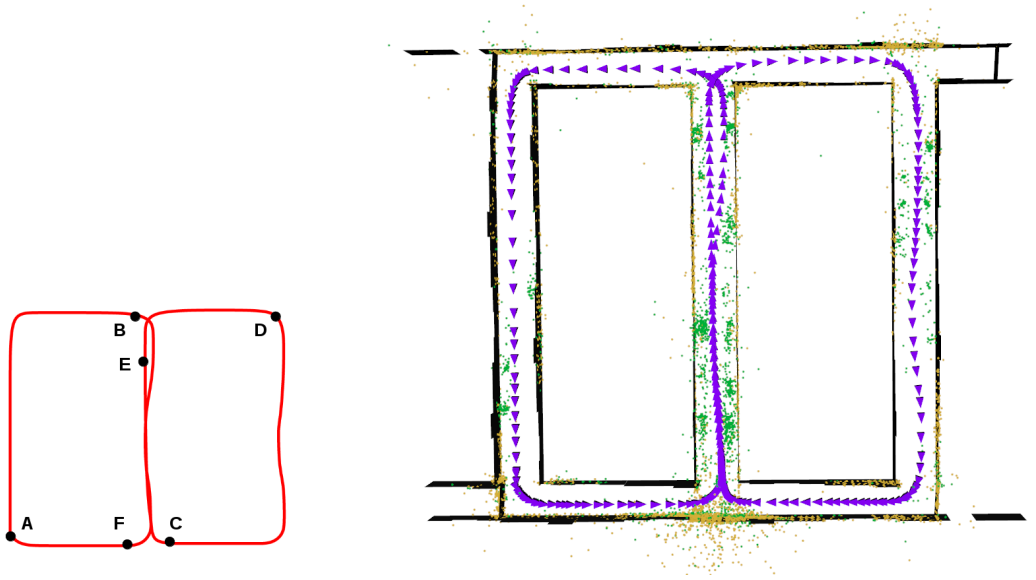
(a) (b)

FIGURE 4.10 – (a) Exemple d'image tiré de notre séquence de synthèse. Les occultations par les arbres et les véhicules sont omniprésentes, et la contrainte aux bâtiment basée sur une segmentation géométrique conduit rapidement à la dérive. Cependant, notre approche permet de filtrer les points 3d occultants afin d'obtenir des contraintes correctes. (b) La segmentation de l'image par le réseau.

4.2.4.2 Évaluation de l'intégration de la segmentation dans le Bundle

Nous présentons les résultats de notre approche sur une séquence synthétique et sur une séquence réelle. Comme les modèles $3d$ des bâtiments ne contraignent pas tout les axes, il est en général nécessaire de coupler les contraintes aux bâtiments avec les données issues d'un autre capteur, par exemple avec l'odométrie, ou avec le GPS si le biais de celui-ci est acceptable. Comme il a été démontré expérimentalement par [58], il est aussi avantageux de rajouter une contrainte d'élévation de terrain (B). Malheureusement, nous ne connaissons à l'heure de la rédaction de ces lignes aucun dataset/benchmark disponible qui fournisse les modèles $3d$ des bâtiments, la vérité terrain, les modèles d'élévation de terrain et les données issues de l'odométrie ou d'un GPS bas coût de manière conjointe. C'est pourquoi nous ne présentons que des résultats qualitatifs obtenus sur nos propres acquisitions.

La séquence de synthèse que nous avons générée représente un environnement urbain où les occultations sont omniprésentes, ce qui entraîne l'échec des méthodes d'association de don-



(a) (b)

FIGURE 4.11 – (a) La vérité terrain de la séquence de synthèse. (b) La trajectoire et le nuage de points reconstruit par notre méthode. Les points appartenant à la classe bâtiment sont de couleur dorée. Tout les autres points sont en vert.

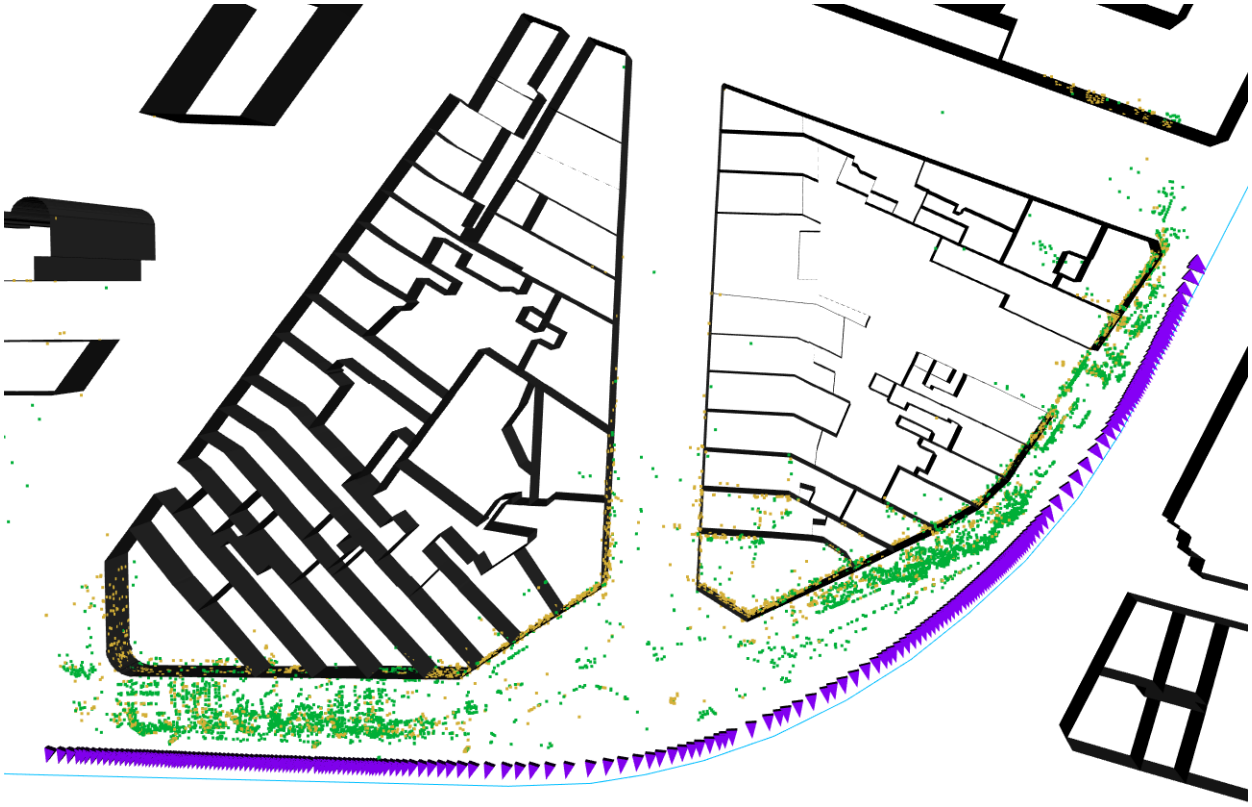


FIGURE 4.12 – Notre reconstruction à partir de la séquence réelle détaillées dans la section 4.2.4.2.

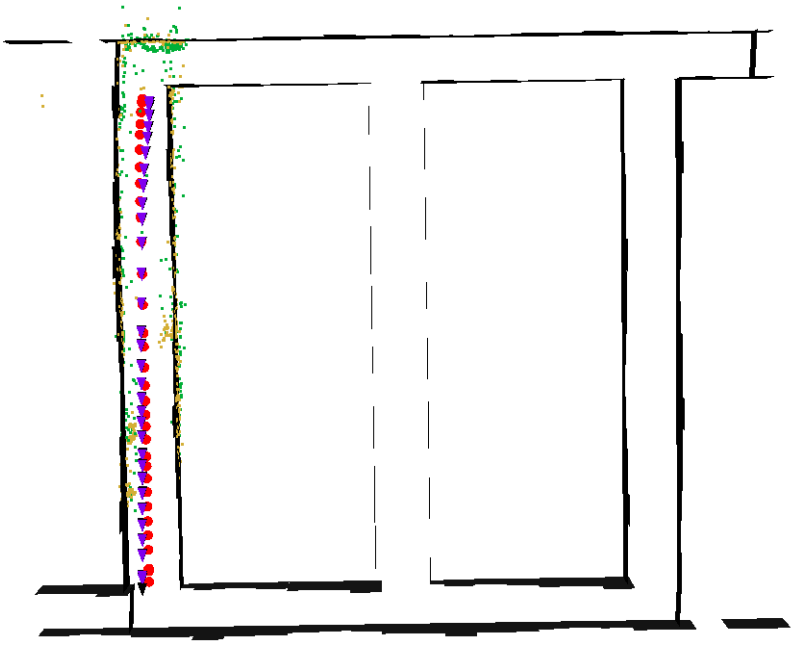


FIGURE 4.13 – L'estimation dérive très rapidement lorsque nous utilisons le lancer de rayon de [118, 58] pour segmenter le nuage de points. Les cercles rouges représentent la position donnée par la vérité terrain. On remarquera que les arbres, correctement identifiés par notre méthode (figure 4.11, (b)) sont ici associés aux modèles $3d$ des bâtiments, et ce sont ces fausses contraintes qui sont à l'origine de la dérive.

nées géométriques telles que le lancer de rayon et l'association par proximité. La longueur de la séquence est approximativement de $1200m$, et la trajectoire effectue plusieurs boucles. Un exemple de cette séquence ainsi que sa segmentation par le réseau sont donnés par les figures 4.10, (a) et (b). La vérité terrain de la trajectoire est illustrée dans la figure 4.11 (a), et la trajectoire reconstruite et raffinée par notre approche est présentée dans la figure 4.11 (b). L'erreur moyenne en translation est de 1.3 centimètres, et l'erreur de rotation est d'environ 0.05 radians. Nous soulignons que le SLAM contraint aux bâtiments basé sur une segmentation/association géométrique dérive très rapidement (figure 4.13), et que le tracking échoue après une cinquantaine d'images clefs.

La séquence réelle est assez courte : il s'agit d'une trajectoire d'environ $400m$ effectuée par un tramway. La caméra est montée à environ $2.0m$ du sol et son axe optique est approximativement orthogonal à la trajectoire. Malheureusement, nous n'avons pas accès à la vérité

FIGURE 4.14 – Reprojection des modèles $3d$ des bâtiments dans les images clefs de la séquence réelle avec notre méthode de segmentation et de filtrage.

terrain. Nous ne pouvons donc que présenter une comparaison qualitative. La reconstruction obtenue avec la contrainte au bâtiment basée sur le lancer de rayon et le critère de proximité avait déjà servi d'exemple illustratif dans la figure 4.3. La figure 4.12 permet de constater une amélioration considérable dans les résultats. En effet, il n'existe aucune dérive, et l'on constate clairement la structure de plusieurs objets qui avaient été, comme nous l'avons vu dans la figure 4.3, faussement associés à des plans du maillage par la segmentation géométrique. Afin de valider visuellement la qualité de nos estimations d'orientations, nous reprojetons les modèles $3d$ dans les images clefs, afin de vérifier si ces reprojections s'alignent correctement avec les bâtiments. Quelques exemples sont rapportés dans la figure 4.14. Notons que pour cette expérience, nous avons utilisé l'odométrie du tramway et la contrainte d'élévation de terrain (annexe B) en plus de la contrainte aux bâtiments.

4.2.5 Discussion

4.2.5.1 Pertinence de l'architecture

Ainsi qu'il a été indiqué précédemment, la méthode que nous avons proposée dans cette section répondait à un problème qui se posait en 2015. En effet, les architectures de l'état de l'art étaient à l'époque soit trop lourdes, soit trop simples pour permettre une segmentation sans post-traitement. Comme nous étions motivés par le besoin de développer une solution pouvant être aussi proche du temps réel que possible, nous avons utilisé une architecture minimaliste et mis en place une solution dont les besoins en temps étaient insignifiants afin de filtrer les résultats du réseau. Cependant, le domaine du machine learning évoluant très rapidement, des architectures pouvant représenter un juste milieu telle que [4, 95] ont été publiées au cours de l'année 2016. Nous avons donc basé la suite de nos travaux sur l'architecture Enet ([95]). Bien que celui-ci soit beaucoup plus complexe que le réseau que nous avions présenté dans la figure 4.5, est $18\times$ plus rapide que les architectures de type VGG et contient $80\times$ moins de paramètres que ces dernières, tout en produisant en pratique des résultats précis et cohérents. Nous noterons cependant que Enet ne doit pas ses performances qu'a son architecture encodeur/décodeur, mais aussi à l'utilisation de plusieurs stratégies telles que l'utilisation de réseaux résiduels (resNet, [37], voir le chapitre 1, §1.5.2) et de convolutions dilatées. On peut donc si cela s'avérait nécessaire, ignorer le décodeur et faire une interpolation des résultats de l'encodeur pour obtenir une estimation certes bruitée, mais beaucoup plus fiable que celle du réseau minimaliste de la figure 4.5, pouvant être utilisée sans post-traitement. Dans ce qui suit, nous exploiterons Enet dans le but non seulement de classifier chaque point $3d$, mais aussi afin de déterminer une information sur l'orientation de la surface à laquelle il appartient. Nous atteindrons des performances temps-réel à l'aide d'une parallélisation GPU/CPU.

4.2.5.2 Problème d'association de données

Il est pertinent ici d'introduire le problème illustré par la figure 4.15, que l'on rencontre après l'étape de segmentation du nuage de points $3d$. Celui-ci a motivé la solution d'association de données bayésienne que nous avions brièvement mentionnée dans la section (§4.1).Comme nous l'avons indiqué précédemment, une fois que le nuage de points a été segmenté, nous devons associer chaque point à un plan du maillage afin de définir les contraintes. En pratique, il est possible que les modèles de bâtiments ne contraignent le nuage de points que sur un axe. C'est à cette situation que nous faisons face dans la figure 4.15 : l'estimation dérive le long de l'axe du mouvement, car les associations points/plan n'imposent que des contraintes le long

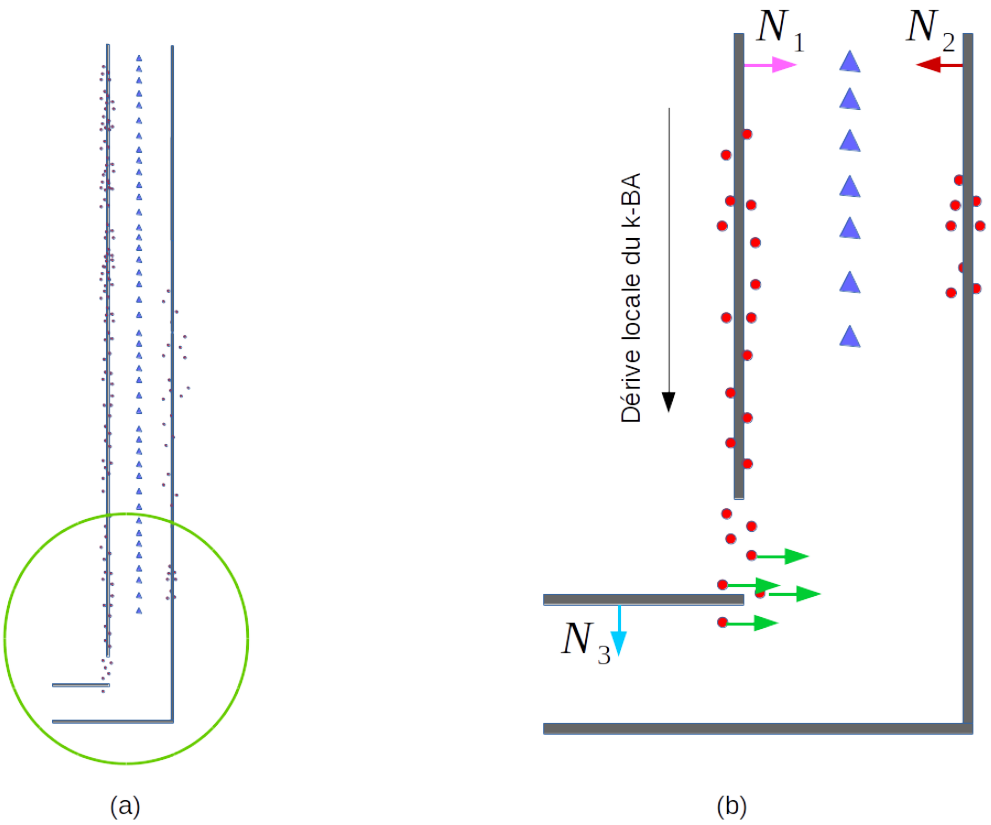
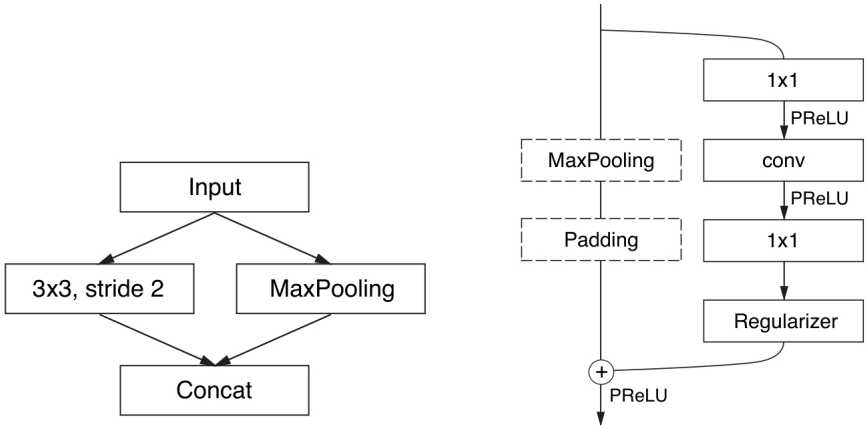


FIGURE 4.15 – Les points $3d$ de cette figure sont supposés appartenir aux bâtiments, et l'on recherche à associer chacune d'eux à un plan. La dérive du SLAM le long de l'axe non-contraint par les bâtiments peut conduire à une estimation où certains points sont placés à proximité de plans auquels ils n'appartiennent pas. Les vecteurs N_1, N_2 et N_3 représentent la normales des plans associés, et les vecteurs verts représentent la normale de surface associée à un certain nombre de points $3d$. Il est clair que les points dont la normale de surface est indiquée devraient être associés au plan de normale N_1. Cependant, un critère d'association basé uniquement sur la proximité les associerait au plan de normale N_3. C'est pourquoi il est important de prendre en compte un information structurelle, telle que les normales de surface dans cette étape.

de la direction orthogonale à celui-ci. En conséquence, lorsque le véhicule approche du virage, l'estimation place certains points à proximité d'un plan auquel ils n'appartiennent pas, ce qui fait qu'un critère de proximité conduirait inévitablement à de mauvaises associations et à terme à l'échec du tracking. C'est pourquoi nous utiliserons le transfert d'apprentissage dans la suite afin d'inférer les normales de surface en plus de classifier chaque pixel de l'image.



(a) couche initiale (b) Structure générale d'un "bottleneck".

FIGURE 4.16 – La structure des couches de Enet.

4.3 Deuxième solution (2016) : exploitation de l'inférence simultanée d'information structurelle et sémantique

4.3.1 Choix de l'architecture

Ainsi que nous l'avons indiqué précédemment, ce n'est qu'en 2016 que des architectures pouvant représenter un compromis acceptable entre précision et temps d'exécution ont été publiées. Nous avons choisi de baser la suite de nos travaux sur l'architecture Enet ([95]), qui, ainsi qu'il a été mentionné précédemment, est $\sim 18\times$ plus rapide que les architectures de type VGG et contient $\sim 80\times$ moins de paramètres que celles-ci. Le réseau est constitué d'une couche initiale (figure 4.16 (a)) et d'une succession de "bottlenecks" [1]. L'objectif du module initial est de réduire la dimensionalité des donnée d'entrée. Ainsi que le soulignent les concepteurs du réseau, la réduction de dimensionalité dans les architectures de type VGG se fait habituellement via un MaxPooling suivi d'une convolution, ce qui conduit à des limitations dans la capacité de représentation du modèle. A l'inverse, une convolution suivie d'un pooling augmente le nombre de featuremaps, mais aussi le temps de calcul. Ceci motive donc l'architecture du bloc initial où une opération de pooling et une convolution d'un pas de 2 sont effectuées en parallèle. En ce qui concerne les modules "bottleneck" (figure 4.16 (b)), ils sont sous forme de modules résiduels (Resnets, voir le chapitre 1, section 1.5.2 pour une explication des avantages de cette architecture), où une branche apprend le résidu entre la fonction qu'il est désirable pour le module d'approximer et la sortie de l'autre branche qui, elle, opère comme un raccourci (le plus souvent l'identité ou l'identité suivie d'échantillonages) entre les entrées et les sorties de la première branche. Ainsi que l'on peut le percevoir d'après la figure 4.17, l'architecture est de type encodeur/décodeur, mais contrairement à un grand nombre d'architectures de ce type (*e.g.* [4]), elle est asymétrique, au sens où l'encodeur est considérablement plus profond que l'encodeur. Ce choix d'architecture est motivé par l'intuition que l'encodeur ne doit avoir pour objectif que de remettre les résultats à la même dimension que les données d'entrée, et que son rôle est plutôt de raffiner légèrement une segmentation plutôt cohérente. Cette hypothèse se trouve validée par les résultats expérimentaux des auteurs, notamment par l'observation que les couches PReLU

1. Nous conservons le terme Anglophone par soucis de concision.

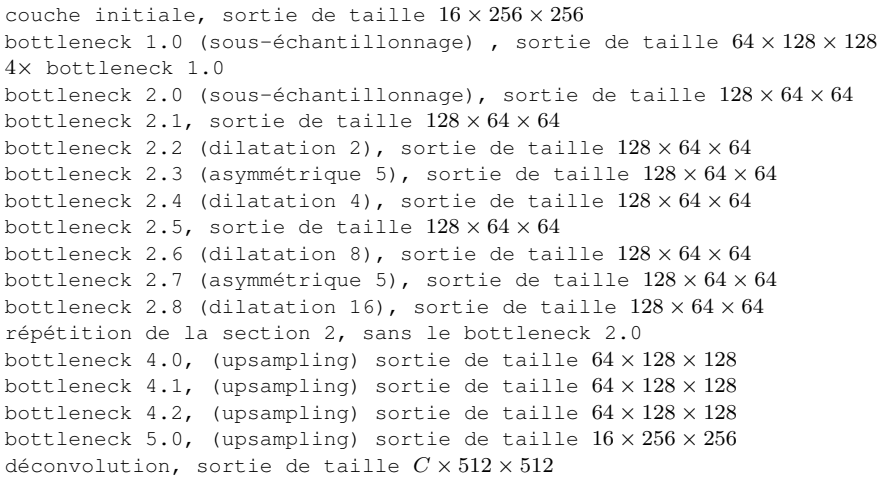


FIGURE 4.17 – Architecture du réseau Enet. Le nombre suivant la fonctionnalité d'un module indique la taille du noyau, *e.g.* "dilatation 16" indique que la convolution dilatée se fait sur un largeur de 16 pixels.

du décodeur on tendance à converger vers un paramètre de 1, ce qui indique que la fonction identité est plus désirable qu'une non-linéarité de type ReLU dans ces couches. De manière similaire à SegNet ([4]), le décodeur utilise les indexes sauvegardés lors des poolings de l'encodeur afin d'avoir un a priori sur l'image qu'il doit reconstruire. Cette approche permet un gain très important en terme de temps de traitement. On notera aussi l'utilisation de convolutions dilatées plûtot que standards afin d'incorporer le plus d'information spatiale possible dans les couches ou la résolution des featuremaps est faible (notamment les dernièrs couches de l'encodeur). Finalement, l'utilisation de couches de BatchNorm (chapitre 1, section 1.5.3) et une régularisation de type Dropout (chapitre 1, §1.5.4) permet d'obtenir une meilleure convergence et une meilleure généralisation.

Les caractéristique du réseau présentées plus haut mettent en évidence que cette architecture est un choix raisonnable pour nos travaux. L'architecture est suffisamment légère pour pouvoir opérer en un délai proche du temps-réel dans le cadre d'un SLAM séquentiel, et les nombreux ajouts par rapport à une architecture naïve telle que celle de la figure 4.5 contribuent à l'obtention de résultats suffisamment précis pour être utilisés sans post-traitements pour déterminer les contraintes à imposer sur le nuage de points. Comme notre objectif n'est pas d'améliorer l'architecture Enet, nous renvoyons le lecteur intéressé à la publication de [95] pour plus de détails sur l'architecture.

4.3.2 Inférence simultanée d'information sémantique et structurelle

Ainsi que nous l'avons souligné dans la discussion à la fin de la dernière section (§4.2.5.2), il ne suffit pas de segmenter correctement le nuage de points 3d. En effet, la dérive du SLAM sur les axes non contraints par les bâtiments peut avoir comme résultat de mauvaises associations point/plan (figure 4.15). Afin de pallier ce problème, nous avons fait le choix de rajouter une branche au réseau afin d'estimer les normales de surfaces associées à chaque point $3d$. Cette décision se trouve justifiée par les observations suivantes :

- ▷ La normale de surface d'un point $3d$ de la classe bâtiment est en général proche de la normale du plan auquel il appartient. Un a priori sur cette normale peut donc s'avérer utile pour guider la recherche du plan correspondant.
- ▷ Nous exploitons déjà un réseau profond afin de labéliser chaque pixel. Le coût de l'inférence des normales via l'ajout d'une branche serait négligeable.

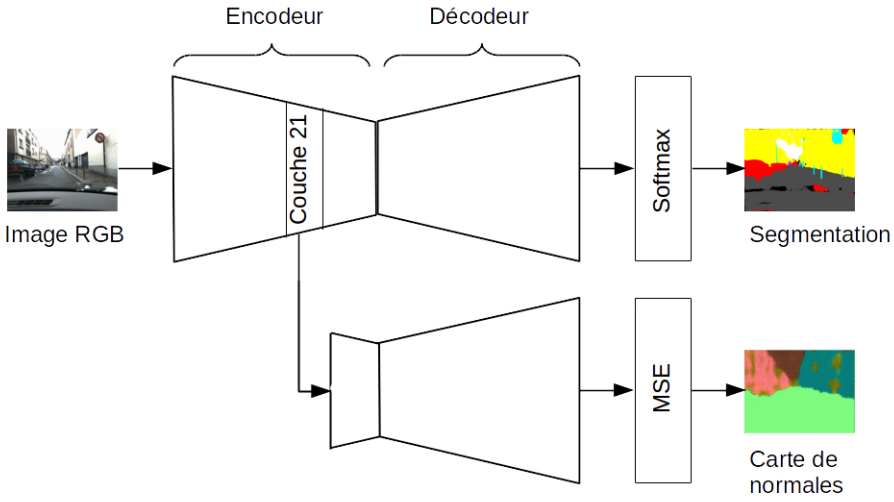


FIGURE 4.18 – Réseau entraîné afin d'obtenir une carte de normales et une segmentation de l'image de manière conjointe. L'architecture des encodeurs et décodeur est identique à celle de ENet. Afin de faire l'apprentissage des normales de surfaces, nous avons utilisé une erreur MSE (Mean squared error) entre les normales prédites et la vérité terrain.

Le réseau que nous exploitons pour l'inférence conjointe est illustré par la figure 4.18. Lors de la forward pass, l'image d'entrée (en RGB) est passée à l'encodeur de ENet afin d'obtenir la représentation intermédiaire fournie par la 21-ème couche de celui-ci. Cette représentation est ensuite donnée en entrée à deux branches à l'architecture identique (basée sur les couches restantes de ENet). En ce qui concerne l'apprentissage, il n'existe malheureusement, à notre connaissance, aucun dataset fournissant simultanément la vérité terrain de la segmentation et des cartes de normales. Nous sommes donc dans l'obligation d'entraîner chaque branche du réseau séparément, ce qui revient à faire un transfert d'apprentissage. Dans un contexte urbain habituel, le problème d'association point/plan de la figure 4.15 se manifeste principalement aux intersections, carrefours et espaces à l'horizon moins chargés, où les normales des façades distinctes sont très rarement proche les une des autres. La discrimination entre les normales s'avère donc plus simple dans ces situations, et l'on peut alors tolérer un bruit plus important dans l'estimation des normales que dans la labellisation des pixels. Il est donc raisonnable d'apprendre l'ensemble de l'encodeur et décodeur de l'architecture Enet pour la segmentation, et d'entraîner ensuite une deuxième branche, constituée des couches suivant la couche 21 de Enet afin de déduire les normales. Notons que le choix de la couche 21 a été fait car elle représente un bon compromis entre temps d'exécution et précision des estimations. En effet, il est évident que le choix d'une couche située au niveau les plus bas de l'architecture (*e.g.* couche 5) permet d'obtenir une très bonne précision, mais comme cela duplique une grande partie de l'architecture, les exigences en temps de traitement augmentent aussi de manière considérable. Par ailleurs, la couche choisie doit se situer dans le décodeur, car ainsi que nous l'avons vu dans la partie 4.3.1,

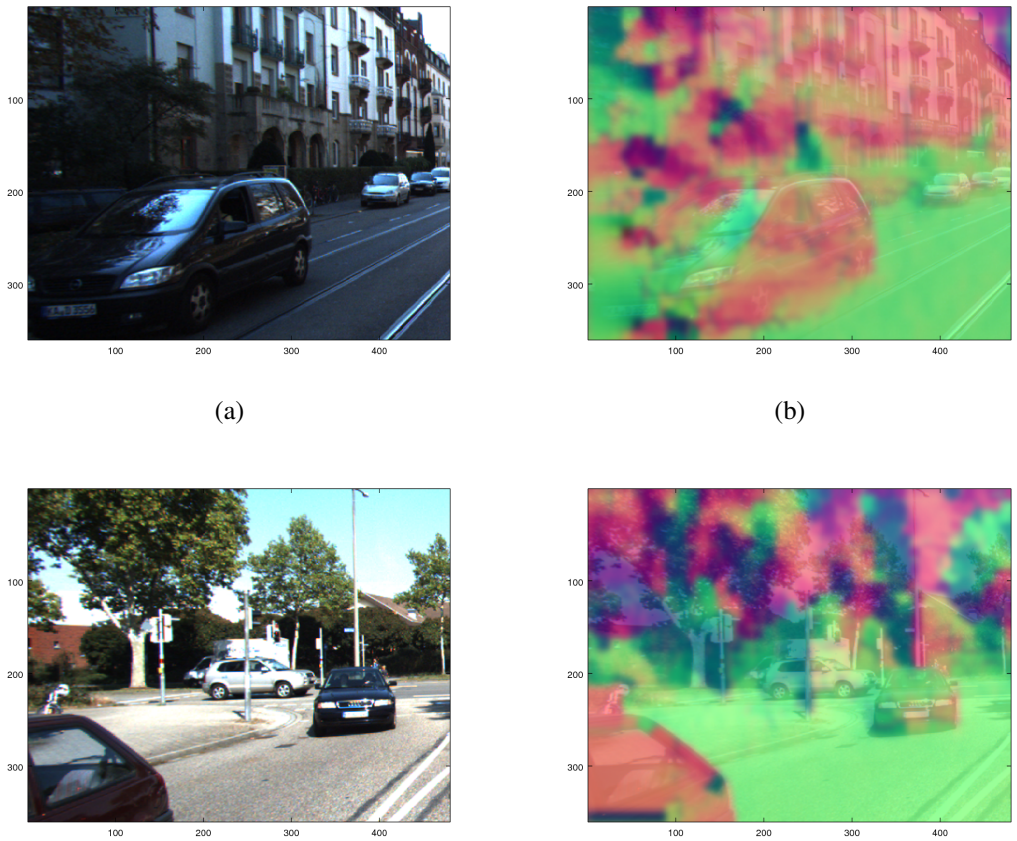


FIGURE 4.19 – Exemple de vérité terrain obtenue à partir des cartes de disparité fournies par la méthode de[133]. (a) et (c) : Exemples d'images d'entrée. (b) et (d) : pseudo vérité terrain obtenue pour les normales.

le décodeur n'a pour rôle que de remettre la carte obtenue aux mêmes dimensions que l'image d'entrée, en ajustant que des détails mineurs. En outre, comme l'encodeur a été appris pour la segmentation, la branche effectuant l'estimation de normales doit être suffisamment profonde afin de pouvoir transformer la représentation intermédiaire utile à l'estimation des labels à une représentation exploitable pour le calcul des normales. En pratique, nous avons observé que le choix des layers entre 18 et 22 produisait en général de bons résultats. Notons que le choix de la couche 21 n'augmente le nombre de paramètres que de $\sim 10\%$, ce qui a un effet négligeable sur le temps d'exécution.

Nous avons entraîné la branche principale (*i.e.* la branche effectuant la segmentation) sur le dataset Cityscapes ([13]). L'unique dataset fournissant la vérité terrain correspondant aux normales des surfaces, c'est à dire le dataset "Nyu depth dataset" ([89]) était insuffisant pour nos besoins car toutes les images de celui-ci sont des images de scènes d'intérieur. Nous avons donc utilisé la méthode open-source de [133] afin d'estimer des cartes de disparité stéréo sur le dataset "stereo/flow 2015" de la base kitti ([79]). Ceci nous a permis de construire des cartes de profondeur à partir desquelles nous avons calculé les normales de surfaces afin d'élargir notre dataset. Deux exemples de cette pseudo vérité terrain sont présentés sur la figure 4.19.

La fonction de coût utilisée pour l'apprentissage du réseau dédié à la segmentation est une

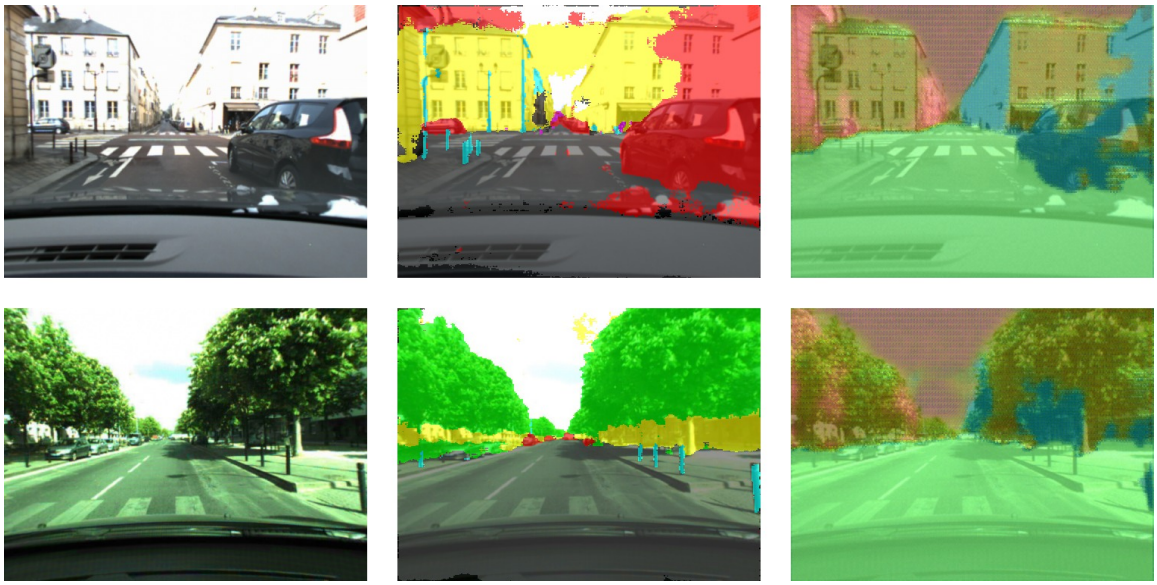


FIGURE 4.20 – Exemples d'entrées/sorties de notre réseau entraîné. De gauche à droite : images RGB, images segmentée, normales prédites par le réseau.

fonction Softmax standard (§1.6.3). Pour entraîner la branche chargée de la prédiction des normales, nous utilisons simplement un critère MSE (pour Means Squared Error), qui n'est autre que l'erreur suivante :

$$MSE(\mathcal{W}) = \sum_{\hat{l}} ||\hat{l} - l_{gt}|| \tag{4.10}$$

où $\hat{l}$ est une normale de surface prédite et l_{gt} est la vérité terrain qui lui correspond. Quelques exemples d'entrées/sorties du réseau sont présentées par la figure 4.20. L'algorithme Adam ([50], voir le chapitre 1 ce ce mémoire, §1.5.1.2 pour une explication) est utilisée pour minimiser la fonction de coût.

Notons que dans le cadre du SLAM, nous utiliserons la méthode de transfert d'intensité de [107] sur certaines séquences afin de rendre la luminosité de la séquence d'entrée aussi proche que possible de celle de la base Cityscapes ([13]).

4.3.3 Intégration du réseau dans le SLAM contraint

4.3.3.1 Intéractions avec l'ensemble du système

L'architecture globale de notre système est illustré par la figure 4.21. Les contraintes autres que celles des bâtiments sont données à titre d'exemple, et peuvent être ou ne pas être utilisées. Elles peuvent aussi être remplacées par d'autres types de contraintes comme des contraintes IMU. Chaque image clef sélectionnée est fournie en entrée au réseau (encadré par les pointillés rouges), et une BA contraint suivi d'un BA non-contraint sont effectués en parallèle sur cette image. Ceci est dû au fait que le BA contraint, qui est basé sur le terme barrière, doit être initialisé avec le résultat d'un BA non-contraint (§2.2.1). Les contraintes aux bâtiments sont définies en fonction des résultats du réseau, et ainsi que nous le verrons dans la section 4.3.4, peuvent être introduites avec un léger délai. Avant de nous intéresser au détails d'implémentation, il est pertinent de préciser notre méthode d'association de données.

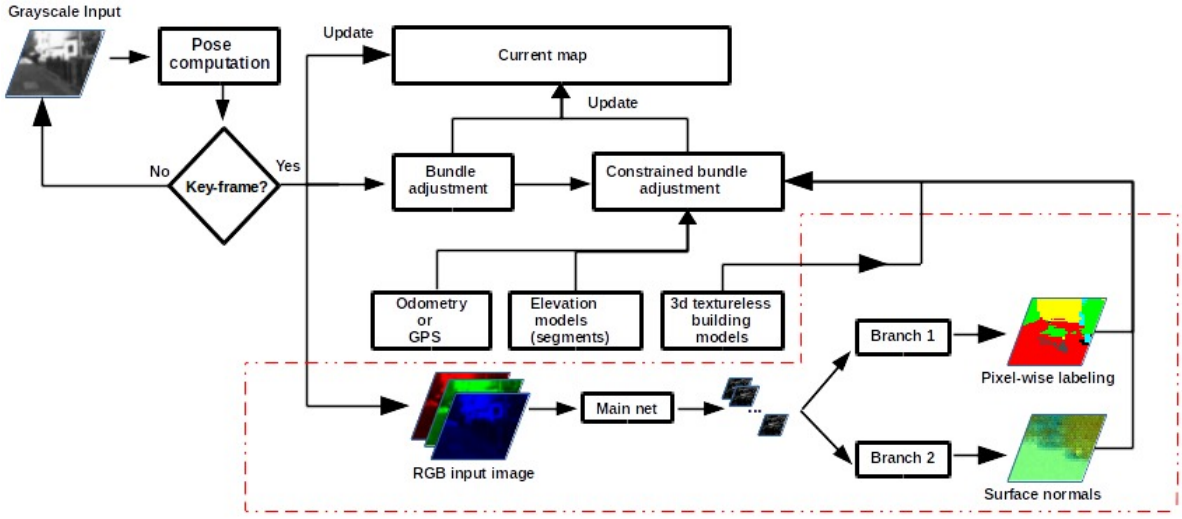


FIGURE 4.21 – La solution proposée. Le réseau chargé de l'inférence conjointe des labels et des normales est encadré par les pointillés rouges.

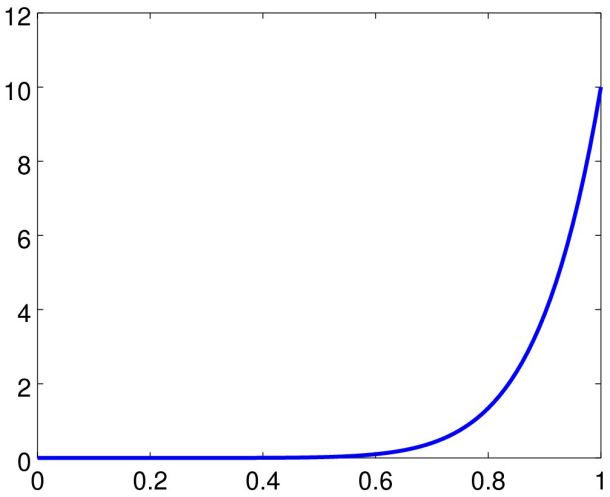


FIGURE 4.22 – Nous modélisons $f(N_Q|p_Q)$ par une distribution Beta de paramètres $\alpha = 10$ et $\beta = 1$. Cette modélisation est motivée par notre souhait de voir f prendre des valeurs élevées lorsque la normale de surface N_Q d'un point est suffisamment proche de la normale p_Q du plan représentant la façade, et de la voir tendre vers zéro dans le cas contraire.

4.3.3.2 Associations point/plan

Soit Z un point $3d$ de la carte. Comme il a été triangulé, on peut déduire qu'il a au moins 3 observations. Soient $\{z_0, ..., z_n\}$ ses observations 2d dans n images-clefs différentes pour lesquelles la forward pass du réseau a été exécutée. Ces passes ont produit n cartes de labels/normales (une pour chaque observation de Z). Nous propageons les labels des pixels aux points 3d via un simple vote majoritaire : un point $3d$ est classifié comme appartenant aux bâtiments si la plupart de ses observations appartiennent à la classe bâtiment. Afin d'obtenir une estimation de la normale de ce point $3d$, nous calculons simplement la moyenne normalisée des n normales de surfaces prédites par les réseau pour $\{z_0, ..., z_n\}$. La normale ainsi obtenue sera

notée $\mathbf{N}_Z$ dans la suite de cette section.

Soit maintenant Q un point $3d$ ayant été classifié comme appartenant aux bâtiments, et soit $\mathbf{N}_Q$ notre estimation de sa normale. Nous recherchons le plan du maillage auquel Q appartient. Nous considérerons ce plan comme identique à la variable aléatoire discrète p_Q dont les valeurs possibles sont les indexes des plans du maillage $\{1, ..., k\}$. Les normales unitaires des plans du modèle $3d$ seront notées $\mathbf{l}_1, \mathbf{l}_2, ..., \mathbf{l}_k$. Le problème que nous cherchons à résoudre est donc

$$\underset{i}{argmax}\, P(p_Q = i|\mathbf{N}_Q). \tag{4.11}$$

En notant f la densité de probabilité continue de $\mathbf{N}_Q$ et $P(p_Q = i)$ la probabilité que le plan indexé par i soit le plan recherché (*i.e.* le plan correspondant au point $3d$), nous pouvons écrire suivant le théorème de Bayes que

$$P(p_Q = i|\mathbf{N}_Q) \propto f(\mathbf{N}_Q|p_Q = i)P(p_Q = i). \tag{4.12}$$

Maintenant, soit $s_i = \mathbf{N}_Q^T\mathbf{l}_i$. Il est raisonnable de poser l'hypothèse suivante :

$$f(\mathbf{N}_Q|p_Q) \propto f(\mathbf{N}_Q|s_i). \tag{4.13}$$

Nous fixons $P(p_Q = i) = 0$ si $s_i < 0$. Pour $s_i \geq 0$, souhaitons que la probabilité de $\mathbf{N}_Q$ soit élevée si s_i est proche de 1, et qu'elle soit basse dans le cas contraire. Par conséquent, une distribution Bêta avec un paramètre α grand et un paramètre β suffisamment petit (voir la figure 4.22) est un choix naturel pour la modélisation de f. Dans nos travaux, nous avons fixé $\alpha \approx 10$ et $\beta \approx 1$.

Nous modélisons l'a priori P en utilisant un critère de proximité. Notons d_i la distance Euclidienne entre Q et le plan d'indexe i. Nous désirons que $P(p_Q)$ soit plus élevé si d_i est de valeur faible. Ceci peut être modélisé par une fonction Softmax (1.6.3) définie sur les $\{-d_1, ..., -d_k\}$. Notons que nous cherchons à ce que l'a priori P soit maximisé pour le d_i ayant la plus petite valeur absolue, c'est pourquoi nous avons pris l'ensemble des opposées de d_i comme domaine de la fonction Softmax. Nous pouvons maintenant écrire :

$$P(p_Q = i|\mathbf{N}_Q) \propto \frac{1}{\mathcal{B}(\alpha, \beta)} s_i^{\alpha-1}(1 - s_i)^{\beta-1} \frac{e^{-d_i}}{\sum\limits_j e^{-d_j}} \tag{4.14}$$

avec $\alpha >> \beta \approx 1$. Nous maximisons le logarithme naturel de cette fonction en faisant une recherche exhaustive sur les P_Q dans un voisinnage de Q de rayon r.

4.3.4 Détails d'implémentation et parallélisation GPU/CPU

L'architecture profonde et le transfert d'intensité sont implémentés en Torch7 [12], et la forward pass est exécutée sur le GPU. L'ajout de la deuxième branche pour estimer les normales de surfaces conduit à une augmentation de $\sim 10\%$ du nombre de paramètres, ce qui a pour résultat des temps de calcul légèrement plus élevés par rapport au temps d'exécution indiqué dans les travaux de [95] : en moyenne, l'inférence conjointe des normales et des labels prend $\sim 76ms$ pour une image de taille 640×480 sur un GPU Geforce GTX 860M (4G de mémoire). Cependant, le code développé en Torch est appelé depuis la librairie du SLAM (implémentée

en C++) via l'API C luaT. Comme nos transferts de données entre l'environnement lua-jit et la framework de SLAM ne sont pas encore optimisées, le coût de l'exécution augmente d'environ $120ms$ lorsque les résultats du réseau sont intégrés dans le k-BA. Par conséquent, suivant les travaux de [71], nous séparons le thread d'optimisation du thread chargé de l'interfaçage entre Torch et de l'association entre points 3d et plans. Plus précisément, la triangulation, l'ajustement de faisceaux non-contraint et contraint se font de manière séquentielle sur le thread principal, et un deuxième thread est dédié aux appels vers l'environnement Lua, à la récupération des estimations de normales et de labels, et à l'association point/plan bayésienne. Dans un environnement urbain habituel, et en supposant que la vitesse maximum du véhicule ne dépasse pas les $\sim 30km$, la moyenne de la distance temporelle entre deux images-clefs est d'environ $300ms$. Cela indique que toutes les tâches seront en général complétées à temps. Cependant, dans les rares cas (*virages serrés, environnement non-texturé, etc*), le nombre d'images clefs augmente. Dans ces situations, comme le k-BA contraint n'attend pas la fin de la forward pass et de l'association des données, le SLAM continuera à fonctionner et les contraintes correspondants aux points nouvellement triangulés seront incluses dans l'optimisation avec un délai négligeable. Le nombre d'images clefs problématiques n'est en général pas élevé, le second thread rattrape le premier assez rapidement.

4.4 Évaluation expérimentale

Comme nous l'avions précisé dans la section 4.2.4.2, les modèles $3d$ des bâtiments ne contraignent pas tous les axes du mouvement, c'est pourquoi il est plus avantageux d'utiliser aussi d'autres capteurs et bases de données. Dans nos évaluations, nous utiliserons les modèles d'élévation de terrain (sous forme de segments, voir 1.9.2) et soit les données GPS, soit l'odométrie en plus de la contraînte aux bâtiments. Ainsi que nous nous l'avions aussi souligné en §4.2.4.2, il n'existe à notre connaissance aucune base de données qui fournisse simultanément les modèles $3d$ des bâtiments et les modèles d'élévation de terrain, la vérité terrain, les images et les données issues de capteurs GPS bas-coût ou d'odométrie. C'est pourquoi une fois de plus, nous ne pouvons que présenter des résultats qualitatifs sur nos propres acquisitions. Il s'agit de trois séquences enregistrées dans des conditions très différentes : *les séquences Versailles, Créteil et Bordeaux*. Quelques exemples d'images-clefs ainsi qu'une représentation de chaque trajectoire sous forme de segments sont présentés dans la figure 4.23. Comme nous n'avons pas de vérité terrain, nous présenterons la projection des modèles $3d$ des bâtiments dans les images-clefs en plus des visualisations des trajectoires reconstruites par notre méthode. Ceci permet une évaluation qualitative des estimations des orientation. Quelques exemples sont donnés dans la figure 4.24.

Les modèles $3d$ des bâtiments ne sont que légèrement occultés dans la première séquence (La séquence Versailles), alors que les occultations (par les arbres, les panneaux publicitaires, les toits de stations de tramway, etc) sont omniprésentes dans les deux autres.

L'avantage de l'utilisation des modèles $3d$ ayant été démontré auparavant [118, 57, 104], nous nous contentons de comparer notre approche, dont la principale contribution est l'amélioration de la segmentation et l'association point/plan, à celle de [57]. Cette dernière contraint la reconstruction aux modèles $3d$ des bâtiments, mais, ainsi que nous l'avions souligné précédemment, se base sur un lancer de rayon et un critère de proximité pour segmenter le nuage de points et associer les points $3d$ classifiés comme appartenant à la classe bâtiment aux plans du maillage. Nous soulignons que les modèles $3d$ géoréférencés mais non texturés que nous

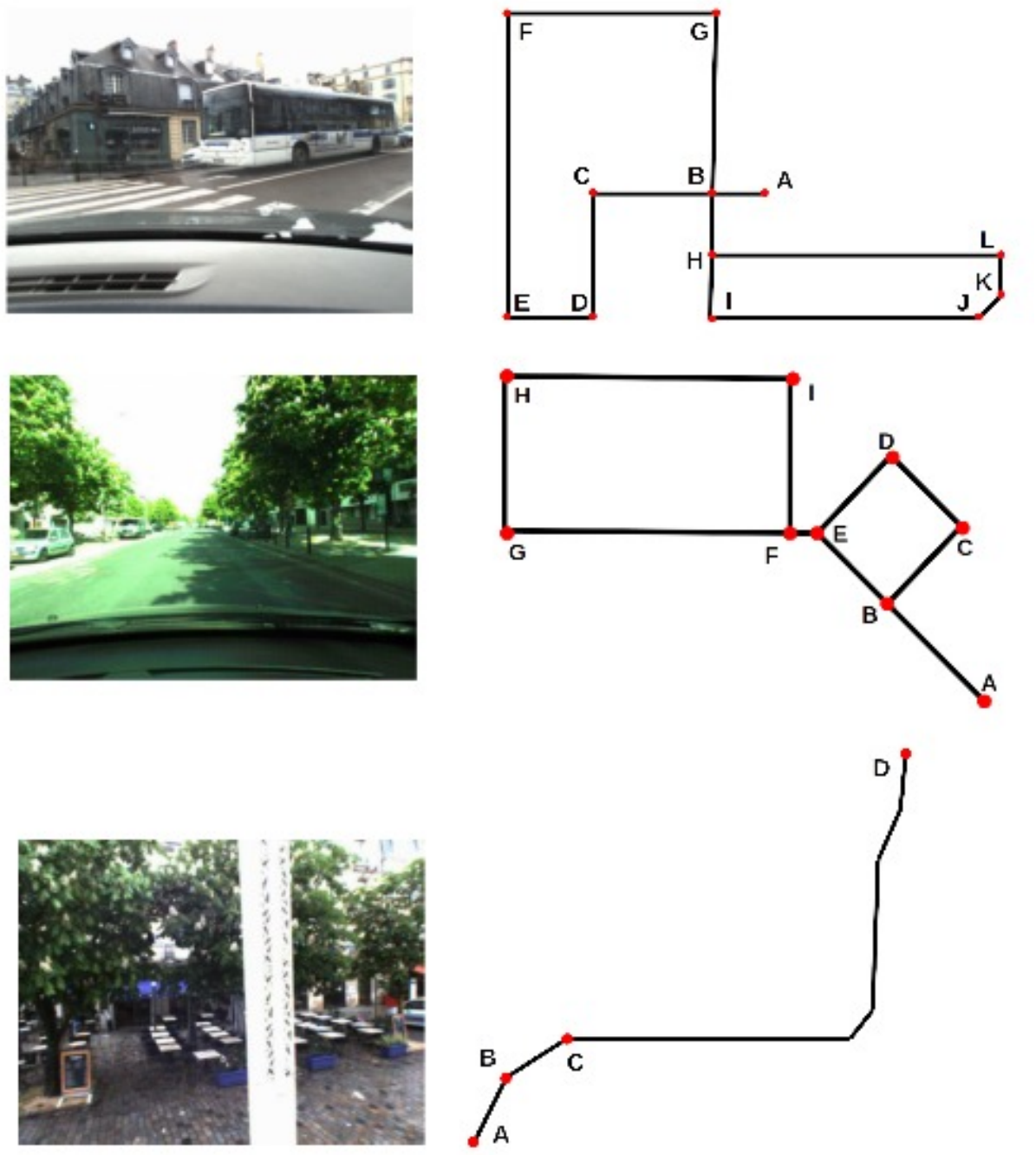


FIGURE 4.23 – **(à gauche.)** Exemples d'images-clefs. **(à droite.)** Les trajectoires. Les trois lignes correspondent (respectivement) aux séquences Versailles, Créteil et Bordeaux. La première trajectoire est donnée par $A-B-C-D-E-F-G-H-I-J-K-L-H$. La trajectoire correspondant à la séquence Créteil est donnée par $A-B-C-D-E-F-G-H-I-F-E-B-C-D-E-B$. La dernière trajectoire ne contient pas de boucle et relie simplement A à D.

FIGURE 4.24 – Exemples de projections de modèles $3d$ (en rouge), en utilisant la pose des caméras estimées par notre méthode. Nous soulignons le fait que les hauteurs des modèles $3d$ des bâtiment sont imprécises, et que leur erreur horizontale maximale est d'environ $2.5m$.

utilisons ont une erreur horizontale maximum d'environ $2.5m$, mais que leurs hauteurs sont très imprécise (aucune borne n'est donnée par l'IGN, voir la section 1.9.1). Pour la première et la troisième expériences (Versailles et Bordeaux), les contraintes d'élévation de terrain et les contraintes GPS sont utilisées en plus de la contrainte aux bâtiments. Le capteur GPS utilisé pour ces deux expériences est le uBlox EVK-6PPP-0, qui fournit des positions à une fréquence de $1Hz$. Nous n'utilisons pas l'information d'altitude que celui-ci fournit car elle n'est pas fiable. Pour les expériences un et deux (Versailles et Créteil), la caméra (une F046C Guppy) est montée sur le tableau de bord du véhicule. Dans notre deuxième expérience (Créteil), les données d'odometrie sont utilisées en l'absence de données GPS fiable. Cette information d'odométrie n'inclue que la vitesse fournie par l'odomètre standard du véhicule (une BMW X5). La caméra utilisée pour acquérir la troisième séquence (Bordeaux) est une IDS 3240.

4.4.1 La séquence Versailles

Il s'agit d'un environnement urbain typique, où les occultations sont rares et principalement causées par des véhicules en stationnement. Nous utilisons donc cette séquence afin de démontrer que notre approche produit des résultats corrects.

FIGURE 4.25 – Comparaison entre les projections des modèles de bâtiment dans les images-clefs. Notons que les images clefs ne sont pas nécessairement identiques d'une exécution à l'autre, et que la sélection des images-clefs dépend du type de SLAM utilisé. C'est pour cela que les image-clefs présentées à gauche sont proches mais pas identiques à celles qui sont présentées à droite. **(à gauche).** Les résultats avec la méthode de [57]. **(à droite).** Les résultats basés sur notre estimation. On constate que les orientations sont estimées de manière beaucoup plus précise avec notre approche.

La longueur de cette séquence est de $2.2km$. Un exemple d'image clef est présenté dans la figure 4.23 (à gauche), et une représentation (sous forme de segments connectés) de la trajectoire peut être observée dans la même figure (première ligne à droite). La trajectoire et le nuage de points estimés par notre méthodes peuvent être visualisés dans la figure 4.26. Nous ne présentons pas de visualisation de la reconstruction obtenue avec la méthode de [57] car les

FIGURE 4.26 – La trajectoire correspondant à la séquence Versailles, obtenue avec notre méthode. Les points dorés sont ceux qui ont été associés à un plan du modèle avec succès. Tout les autres points sont en vert.

deux estimations sont suffisamment proches pour qu'il soit impossible de les distinguer visuellement l'une de l'autre. Cependant, les reprojections des modèles $3d$ des bâtiments (figure 4.25, première ligne) montrent que notre estimation est plus précise.

4.4.2 La séquence Créteil

Cette séquence est d'une longueur d'approximativement $1.3km$. Les occultations sont abondantes et les façades des bâtiments se retrouvent très souvent presque entièrement recouvertes (4.23, deuxième ligne à gauche). De plus, les images sont pour la plupart saturées, c'est pourquoi nous utilisons le transfert de couleur de [107] pour améliorer les performances du réseau. Une représentation de la trajectoire sous forme de segments est donnée par la figure 4.23 (deuxième ligne à droite) afin de permettre de visualiser les nombreuses boucles. La méthode de [57] qui se base sur le lancer de rayon et un critère de proximité introduit trop de fausses associations entre les points $3d$ et les plans dans la contrainte aux bâtiments (*i.e.* la plupart des objets occultants sont associés aux modèles des bâtiments). Il en résulte que la reconstruction dérive vers les modèles $3d$. Ainsi que l'on peut l'observer dans la figure 4.27 (a, à gauche), ces contraintes erronées conduisent à des incohérences entre le nuage de points 3d estimé et le maillage. Un grand nombre de points qui se situent loin des façades sont classifiés comme appartenant aux bâtiments, et un nombre important de points se retrouvent dans des clusters à

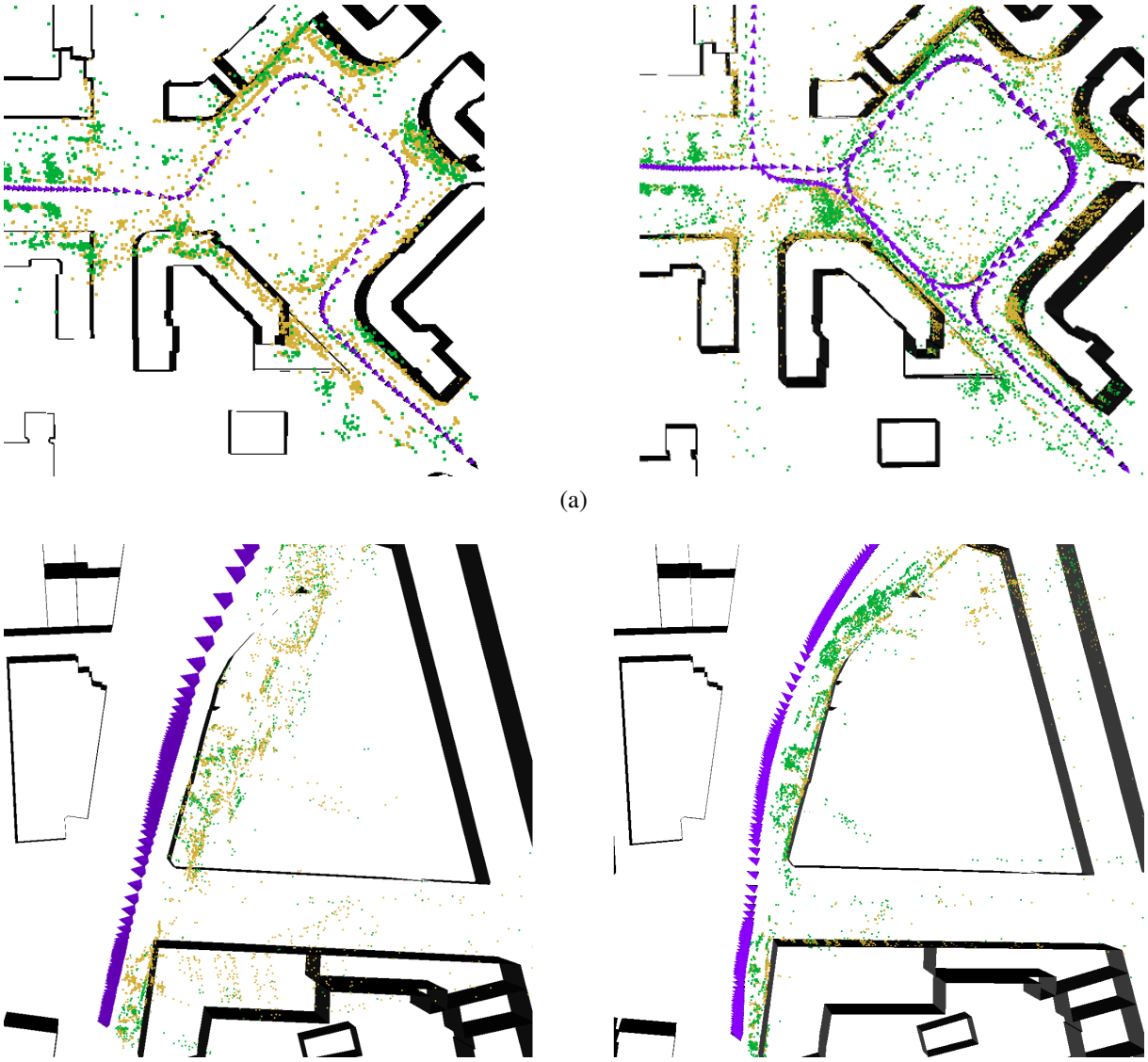


FIGURE 4.27 – Illustration des différences entre la reconstruction obtenue avec l'approche de [57] et notre estimation. Les points dorés sont ceux qui sont classifiés comme appartenant aux bâtiments, et tout les autres points sont en vert. Il est important de remarquer qu'une segmentation suivie d'une classification correcte doit conduire à l'alignement des points dorés avec les façades des bâtiments ainsi qu'à une distribution spatiale cohérente des points verts (*e.g.* compte tenue de l'erreur horizontale faible des modèles, les points verts ne doivent que très rarement se retrouver à l'intérieur des bâtiments). **(a, à gauche).** Les résultats de [57] sur un sous-ensemble de la séquence Créteil. **(a, à droite).** Les résultats de notre approche sur la séquence mentionnée. **(b, à gauche).** Les résultats de [57] sur le segment $A - B$ de la séquence Bordeaux. **(b, à droite).** L'estimation obtenue avec notre méthode.

l'intérieur des modèles.

En revanche, notre méthode gère ces occultations (figure 4.27 (a, à droite)). La structure des éléments occultants tels que les arbres est clairement visible, et les points $3d$ des bâtiments sont correctement alignées avec les plans des modèles. La trajectoire entière reconstruite par notre

FIGURE 4.28 – La trajectoire correspondant à la séquence Créteil, obtenue avec notre méthode. Les points dorés sont ceux qui ont été associés à un plan du modèle avec succès. Tout les autres points sont en vert.

méthode est présenté dans la figure 4.28. Une comparaison entre les projections des modèles est donnée par la figure 4.25.

4.4.3 La séquence Bordeaux

Cette séquence de $1.2km$ est particulièrement difficile. D'une part, la caméra est fixée à une distance d'environ $2.0m$ du sol en haut d'un tramway et l'axe optique de la caméra est orthogonal à la direction du mouvement, ce qui réduit grandement le nombre d'observations de chaque point $3d$. D'autre part, l'omniprésence des arbres et de structures peu texturées introduit un bruit très important dans l'appariement, ce qui perturbe la segmentation et l'association via des critères géométriques. Une exemple d'image clef est présenté dans la figure 4.23 (dernière ligne à gauche). La trajectoire ne contient pas de boucle (figure 4.23 dernière ligne à droite). Comme pour la séquence précédente, l'estimation de [57] dérive vers les modèles $3d$ des bâtiments à cause du nombre élevé d'associations point/plan erronées. Ceci peut être observé dans la figure 4.27 (b, à gauche). Cependant, notre méthode (figure 4.27 b, à droite) permet une fois de plus d'aligner correctement le nuage de points et le maillage, et les structures occultantes se détachent clairement de l'ensemble de points. La figure 4.29 montre la trajectoire et le nuage de points estimés par notre approche, et la dernière ligne de la figure 4.25 montre une comparaison entre les reprojections de modèles $3d$.

FIGURE 4.29 – La trajectoire correspondant à la séquence Bordeaux, obtenue avec notre méthode. Les points dorés sont ceux qui ont été associés à un plan du modèle avec succès. Tout les autres points sont en vert. On notera que l'axe focal de la caméra est orthogonal à la direction du mouvement du tramway, et c'est à cause de cela que les points $3d$ n'apparaissent que d'un coté de la trajectoire.

4.4.4 Résumé, limites et perspectives

Ce chapitre avait pour objectif de réduire le bruit dans l'association point/plan afin de mieux contraindre le nuage de points estimé par le SLAM aux modèles $3d$ géoréférencé et non-texturés des bâtiments. Nous avons vu que les méthodes exploitant des critères de segmentation et d'association purement géométriques sont vouées à l'échec lorsque l'occultation des façades par différents objets (végétation, panneaux publicitaires, etc) n'est plus négligeable. Nous avons présenté une solution basée sur l'intégration d'un réseau de neurone dans le processus du k-BA contraint. Nous avons proposé une première solution en 2015, dont on peut résumer ainsi les caractéristiques principales :

- ▷ Utilisation d'un réseau d'une architecture très simple mais légère, ce qui avait pour résultat une segmentation extrêmement bruitée.

- ▷ Filtrage des faux positifs des résultats du réseau : La densité continue des distributions discrètes retournées par le réseau, conditionnellement à l'appartenance de pixel à la classe bâtiment a été modélisée par une loi de Dirichlet $\mathcal{D}$ dont les paramètres peuvent être appris en quelques millisecondes. Les pixels dont l'*argmax* correspondait à la classe bâtiment mais dont la probabilité $\mathcal{D}$ est inférieure à un seuil sont considérés comme des faux positifs.

▷ Les résultats expérimentaux on démontré l'apport de cette méthode par rapport à l'approche de [58].

Nous avons ensuite indiqué que les architectures de type [4, 95], publiées en 2016, représentaient un compromis idéal entre temps d'exécution et précision, et pouvaient directement être exploitées sans post-traitement. Nous avons aussi remarqué le problème d'association point/plan dû à la dérive du SLAM visuel sur les axes non contraints par les bâtiments, et l'avons pris en compte dans la deuxième solution que nous avons proposée, dont les particularités sont les suivantes :

▷ Estimation conjointe de l'information sémantique (labélisation par pixel) et structurelle (carte de normales) par une réseau à deux branches, dont les poids ont dû être appris séparément car aucun dataset ne fournissait les cartes de normales et la segmentation sémantique de manière simultanée.

▷ L'exploitation de l'architecture Enet [95].

▷ Association point/plan via une approche bayésienne simple mais efficace, utilisant la distance point/plan comme un a priori, et la similarité entre les normales afin d'obtenir la probabilité postérieure.

▷ Résultats temps-réel grâce à une parallélisation GPU/CPU.

▷ Estimation précise de la position et de l'orientation sur des séquences sur lesquelles les méthodes employant des critères de segmentation/association uniquement géométriques dérivent très rapidement.

Le problème principal de notre solution reste le coût de l'exécution. Comme nous l'avons indiqué, la segmentation se fait en parallèle de l'optimisation non-contrainte et contrainte, mais cette dernière n'attend pas la fin de la "forward pass" du réseau. Les contraintes sont donc intégrées dès leur disponibilités, ce qui peut impliquer un léger délais dans leur inclusion. Nous n'avons pas observé de difficulté lié à ce délais dans nos expériences, cependant, nous pensons qu'il serait plus avantageux de réduire ce délai autant que possible. De plus, les architectures employées pour la segmentation ne se basent que sur une seule image d'entrée et ignorent entièrement les dépendances temporelles. Nous pensons donc qu'il est essentiel dans nos travaux futurs d'orienter nos recherches vers la conception d'un réseau plus léger pouvant propager les résultats précédents, par exemple via l'utilisation d'unités LSTM.

Une difficulté que nous rencontrons dans nos travaux est liées au couplage des modèles 3d des bâtiments au GPS. Le plus souvent, nous ne pouvons employer ces deux sources d'informations simultanément qu'avec difficulté, car il faut alors pondérer les termes manuellement afin d'éviter que les contraintes définissent des minima locaux différents ([56]. Nous préférons donc, initialement, ne prendre en compte que l'une d'entre elles dans un SLAM, à laquelle nous substituerons la seconde en cas d'insuffisance de la première. Il serait pour cela intéressant une fois de plus, d'explorer les possibilités d'élimination du biais du GPS via une fusion rapprochée [8, 119].

CHAPITRE 5

Conclusion, travaux en cours et perspectives

Nous résumerons en premier lieu les problématiques étudiées et les solutions proposées pendant cette thèse. La majeure partie de ce chapitre sera cependant dédiée à l'ensemble des travaux en cours n'ayant pas été finalisés au moment de la rédaction de cette thèse. Ceux-ci ont pour sujet l'amélioration de la robustesse de la relocalisation par rapport à une base d'images géoréférencées afin d'apporter des corrections ponctuelles aux estimations d'un SLAM à grande échelle. Nous conclurons ce chapitre par une discussion sur les perspectives et travaux futurs sur le plus long terme.

Nous nous sommes penchés dans cette thèse sur l'amélioration d'un système de SLAM à grande échelle exploitant des capteurs bas-coûts (caméra monoculaire, GPS ou odémétrie) et des bases de données gratuites (modèles 3d non-texturés géoréférencés de bâtiments, aisément accessibles en ligne). Bien que ce système soit basé sur l'ajustement de faisceaux contraint, nos contributions sont en grande partie indépendantes des algorithmes de SLAM utilisés, car elles avaient comme objectif

- ▷ L'amélioration des performances (en terme de temps de calcul) de la fusion GPS/VSLAM par terme barrière. Cette méthode optimise une estimation initiale en la faisant tendre vers les contraintes imposées. On peut donc, en pratique, l'utiliser avec des fonctions de coût denses ou même avec le résultat d'une approche de filtrage.

- ▷ La définition des contraintes robustes aux occultations à partir de modèles $3d$ de bâtiments, afin de contraindre l'estimation géométrique du SLAM. Nous nous sommes efforcés de ne prendre en compte que les primitives de la carte, les modèles $3d$ est les images-clefs afin de garantir l'indépendance de nos contributions au système de SLAM sous-jacent.

- ▷ L'exploration de la possibilité d'une exploitation des techniques de Deep learning dans des modules pouvant être intégrés ou pouvant remplacer certaines composantes d'un systèmes de SLAM géométrique.

La complexité en temps de la fusion VSLAM/GPS par terme barrière semble découler, d'après les arguments théoriques et les résultats empiriques de [67], des liens entre la covariance de la caméra la plus récente et l'incrément retourné par l'algorithme d'optimisation non-linéaire (en l'occurrence l'algorithme de Levenberg-Marquardt), dont la réduction de la magnitude facilite la fusion. Cette relation peut intuitivement être vue comme une relation inverse entre la taille de l'ellipse de cette covariance et l'incrément en question, une solution naturelle est d'inclure un plus grand nombre de caméras dans la fusion, ce qui est à l'origine du coût plus élevé de la méthode. Nous avons donc tenté d'explorer la possibilité de diminuer cet incrément par d'autres moyens. Nous avons pour cela présenté un terme barrière reposant sur une fonction de coût hybride composée d'une terme d'ajustement de faisceaux et d'un terme de graphe de poses, et nous avons démontré de manière théorique que notre solution pouvait avoir l'impact recherché sur la fusion. Nos résultats expérimentaux ont démontré des gains importants en temps d'exécutions (près de 60%), sans que les pertes de précisions soient significatives.

Concernant la contrainte aux bâtiments et l'intégration des méthodes d'apprentissage profond au SLAM géométrique, nous avons proposé deux approches. La première, formulée en 2015, se basait sur une architecture convolutionnelle naïve, et avait pour objectif de propager de la segmentation sémantique des images-clefs par le réseau aux points $3d$. Comme le réseau ne pouvait produire de résultats spatialement cohérents sans post-traitements coûteux, nous avons présenté une méthode de filtrage basée sur la prise en compte de la forme de la distribution associée à chaque pixel par le réseau afin de réduire le nombre de faux positifs. Compte tenu de l'évolution des architectures des CNNs visant la segmentation, celle-ci ne présente plus de véritable utilité.

La deuxième approche que nous avons proposée exploite l'architecture récente ENet ([95]), et a pour objectif additionnel de résoudre le problème des mauvaises associations points/plans dues aux dérives locales du SLAM lorsque les modèles de bâtiments ne contraignent pas tous les axes. Cette méthode repose sur l'utilisation de l'information structurelle (normales de surfaces) servant à guider la recherche du plan correspondant en chaque point. La parallélisation GPU/CPU de l'optimisation et de la forward pass permet d'obtenir des résultats temps-réel. Nos évaluations sur des séquence synthétiques et réelles montrent que notre méthode améliore considérablement la robustesse des contraintes face aux occultations en comparaison avec les méthodes exploitant des critères purement géométriques.

Comme nous l'avons souligné aux chapitres précédents (sections 4.4.4 et 4.4.4), il existe plusieurs voies d'amélioration. Nous repoussons cependant la discussion à ce sujet à la fin de ce chapitre, et présentons auparavant les travaux en cours, qui n'ont pas encore été finalisés au moment de la rédaction de ces lignes.

5.1 Travaux en cours

Le système de SLAM basé sur la fusion par terme barrière des différentes informations que nous avons exploitées jusqu'ici (flux vidéo monoculaire, modèles 3d non-texturés de bâtiments ou GPS, modèles d'élévation de terrain, odométrie) produit des résultats plus précis et robustes que ceux d'un k-BA basé uniquement sur la vision. Cependant, il est clair que la qualité de l'estimation obtenue dépend de la précision des données d'entrée et des hyperparamètres (*e.g.* seuil du terme barrière) qui ne sont pas estimés de manière dynamique. De plus, l'absence d'interprétation probabiliste de ce processus rend l'évaluation de l'incertitude des prédictions plus délicate. De plus, la perte du signal GPS dans des zones où la base des modèles 3d n'est pas à

jour, incomplète ou indisponible peut avoir pour résultat une dérive du SLAM et en conséquence de mauvaise associations point/plan qui conduiront à l'échec du suivi.

Afin de pallier ces problèmes, il est possible d'effectuer une reconstruction hors-ligne d'une base parcimonieuse d'images géoréférencées accompagnée d'un nuage de points éparse de l'environnement, pour ensuite les exploiter afin d'apporter des corrections à un SLAM temps-réel évoluant dans celui-ci. Le principe est simple : il s'agit de trouver dans la base une ou plusieurs images suffisamment proches de l'image clef courante du SLAM. On peut ensuite, en fonction du nombre d'images correspondantes et de la qualité la mise en correspondance, utiliser ces information ponctuellement ou sur des portions complètes de la trajectoire. Cependant, les systèmes (*e.g.* [30] ou le chapitre 8 de [61]) exploitant cette stratégie reposent sur des points saillants et des descripteurs définis manuellement (*e.g.* SIFT, SURF, ORB, etc). Ceci réduit l'applicabilité de ces systèmes lorsque les conditions dans lesquelles la base a été acquise diffèrent des conditions dans lesquelles le SLAM opérera en temps réel. Par exemple, il est très difficile de mettre en correspondance deux images lorsque celles-ci ont été acquises lors de saisons différentes, (figures 5.1) ou lorsqu'il existe des changements d'illumination extrêmes et des différences de conditions météorologiques (figure 5.2) entre elles.

FIGURE 5.1 – Les changements de saisons sont des obstacles difficiles à surmonter pour les systèmes de relocalisation basés sur des descripteurs définis manuellement (*e.g.* SIFT, SURF, etc), car non seulement les conditions d'illumination mais aussi la géométrie et les textures varient d'une saison à l'autre (figure tirée de [2])

Notons que certaines solutions (*e.g.* [2, 81]) permettent d'obtenir des résultats fiables en faisant la mise en correspondance entre des séquences d'images plutôt qu'entre des images indi-

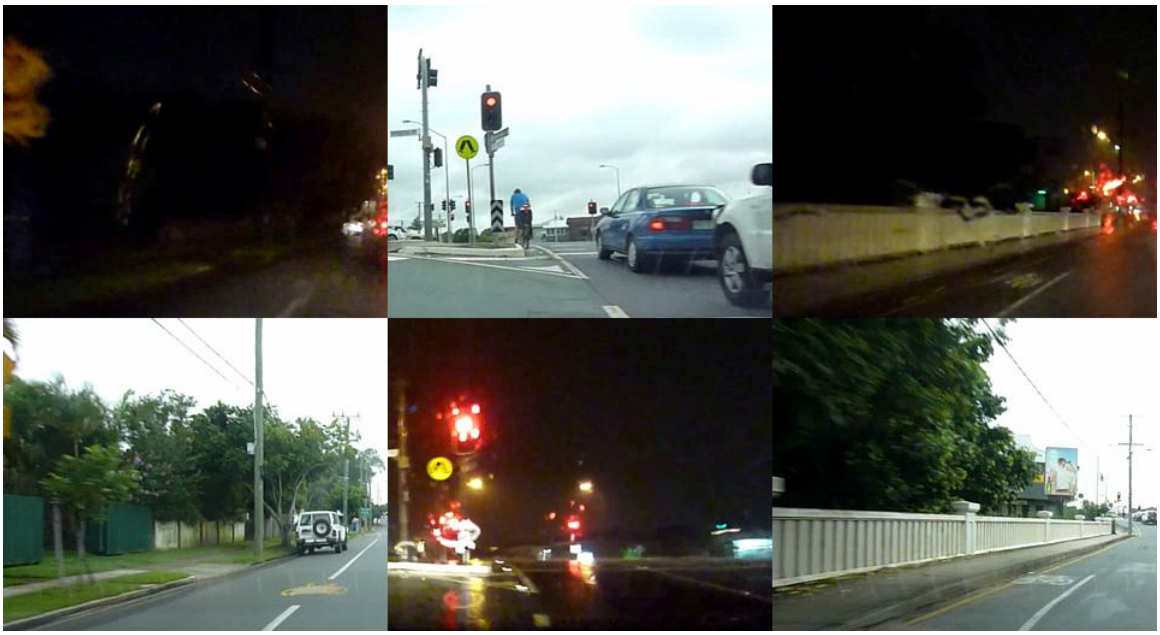


FIGURE 5.2 – Les variations d'illumination représentent elles aussi un obstacle à la mise en correspondance basée sur des descripteurs définis manuellement. Dans les exemples ci-dessus, tirés de la base d'image Alderly ([81]), chaque colonne est un couple d'images acquises au même endroit, mais dans des conditions météoriques et d'illumination très différentes.

viduelles. Néanmoins, [5] montrent que ces types de méthodes souffrent toujours d'un manque de robustesse lié aux descripteurs utilisés, et proposent une amélioration de [81] qui exploite un vecteur composé des résultats de plusieurs couches intermédiaires d'une réseau convolutionnel afin d'effectuer la mise en correspondance. Nous remarquons cependant que ces méthodes imposent des contraintes fortes sur la base utilisée : celle-ci doit contenir des sous-séquence qui puissent être mises en correspondance avec des sous-séquence du flux vidéo que traite le SLAM exécuté en temps réel. Cela implique que nous somme soit dans un contexte de fermeture de boucle, soit que nous avons un a priori très fort sur la trajectoire.

L'étape subséquente à la mise en correspondance peut être un calcul de pose relative entre les images (*e.g.* pour une correction de type fermeture de boucle) entre les images ou la définition de contraintes basées sur la reprojections dans les images de la base ([80]). Cependant, quelle que soit notre choix, l'exploitation de descripteurs classiques de type SIFT ou SURF s'accompagnera de problèmes de robustesses. Même si l'on arrive à mettre, correctement, deux images acquises à des saisons différentes en correspondance via la comparaison de leurs descripteurs, il est vraisemblable que le nombre de matchs inliers ne seront pas suffisants pour un calcul de pose fiable, ou qu'il ne permettra pas d'imposer des contraintes suffisamment fortes lors de la reprojection. Nous nous pencherons donc naturellement sur le calcul de poses relatives entre les images plutôt que sur les contraintes de reprojections (car les points $3d$ reconstruits pas le SLAM sont le résultat d'une triangulation de points d'intérêts classiques).

Suivant cette discussion, il est donc naturel que nous cherchions à nous baser sur les valeurs des couches intermédiaires d'un réseau profond. Nous recherchons une solution plus générale et robuste de mise en correspondance et de calcul de pose *absolue*, sans imposer de contraintes supplémentaires sur la base et la trajectoire. Nous faisons donc face aux difficultés suivantes :

▷ Nous nous intéressons principalement à la mise en correspondance entre les images-

clefs d'un SLAM en cours d'exécution et les images d'une base ne contenant pas nécessairement de données sur la même trajectoire. Il peut s'agir par exemple d'une base où les images sont issues d'acquisitions indépendantes, en provenance de plusieurs utilisateurs et à différents moments de l'année. Par conséquent, les approches basées sur la mise en correspondance de séquences entières, très utilisées pour la fermeture de boucle, ne nous sont pas d'un grand secours.

▷ Nous utilisons déjà une réseau afin de segmenter les images et de prédire les normales de surfaces associées à chaque point de l'image. Nous n'avons d'autre choix, afin de préserver la capacité d'exécution en temps-réel, que d'exploiter ce réseau pou la mise en correspondance.

Nous décrirons nos solutions ainsi que quelques résultats expérimentaux préliminaires dans les sections qui suivent.

5.1.1 Appariement d'images

Comme nous exploitons déjà un réseau dans notre système de SLAM, nous sommes contraints d'exploiter les résultats des couches intermédiaires de celui-ci afin de pouvoir préserver l'aspect temps-réel de nos solutions. Comme nous l'avons précisé au chapitre précédent (§4.3), l'architecture que nous exploitons est Enet ([95]). Nous avons réalisé de manière empirique que l'utilisation des sorties des couches intermédiaires du réseau, c'est à dire les couches finales de l'encodeur, permettent d'atteindre un compromis raisonnable entre robustesse aux changements d'illumination et aux changements géométriques. Dans nos expériences, nous utilisons la couche 21, mais nous n'avons pas observé de différences notables entre l'utilisation de ces couches et ses voisines proches. Étant donné un triplet d'images I_1, I_2, I_3, et en notant h_i l'expression sous forme d'un vecteur des sorties de la couche 21 pour l'image I_i, nous considérons simplement que I_1 est plus proche de I_2 que de I_3 si et seulement si la distance Euclidienne entre h_1 et h_2 est inférieure à la distance Euclidienne entre h_1 et h_3.

Cette stratégie, certes très simple, est cependant très efficace. Nous l'avons testée pour l'appariement entre deux séquences acquises depuis un tramway, l'une au printemps et l'autre en hiver (figure 5.3). La base que l'on considère n'est constituée que d'images espacées, ce qui résulte en deux bases de 428 images sur une distance d'environ $5km$. Nous utilisons chaque image de la séquence enregistrée en été comme image requête et recherchons un appariement dans la deuxième séquence. L'utilisation de la méthode décrite pour la mise en correspondance produit des erreurs assez faibles :

▷ Lorsque l'image requête est comparée à toutes les images de la base, le taux de faux appariement est de 7.2%. Cependant, l'image correspondante correcte ce situe parmi les trois premiers matchs pour toutes les requêtes.

▷ L'erreur d'appariement se retrouve réduite à 0.4% lorsque l'on limite la fenêtre de recherche à un voisinage proche de l'image requête (on peut en pratique se baser sur les coordonnées GPS, puisque la base sera géoréférencée).

Ces résultats sont bien plus encourageants que ce que nous avons obtenus avec des appariement basés sur des descripteurs SIFT, qui étaient en grande majorité complètement faux.

FIGURE 5.3 – Les séquences filmées depuis un tramway à des saisons différentes. Chaque ligne représente deux images correspondantes.

Une question cependant se pose : les représentations utilisées (*i.e.* les sorties de la couche 21) sont-elles optimales ? La visualisation du plongement de celles-ci dans $\mathbb{R}^2$ peut permettre de développer une intuition sur le problème. Nous utilisons l'algorithme standard t-sne ([76]), introduit au chapitre 1. Afin de déterminer les relations entre les représentations résultant de la couche 21 pour les deux séquences, leur concaténation a été fournie à t-sne. Le résultat est donné par la figure 5.4. Ainsi qu'on peut le remarquer, les représentations utilisées forment deux clusters distincts. Cependant, nous avons précédemment constaté que ces représentations peuvent être exploitées afin d'obtenir une mise en correspondance très robuste. Intuitivement, ceci semble donc indiquer que la transformation τ_h entre les deux clusters dans l'espace à haute dimensionalité où résident les features est une transformation qui préserve les distances. En effet, il est raisonnable de conjecturer que le fait que l'on puisse effectuer un appariement en utilisant ces vecteurs est dû au fait que pour $\tau_h(h_i)$ (pour une image I_i) reste plus proche de h_i que des autres images I_j (avec $i \neq j$) du premier cluster. Il est donc vraisemblable que τ soit

très sensible, *i.e.* qu'il ne s'agisse que d'une translation particulière (sur un nombre à déterminer de dimensions).

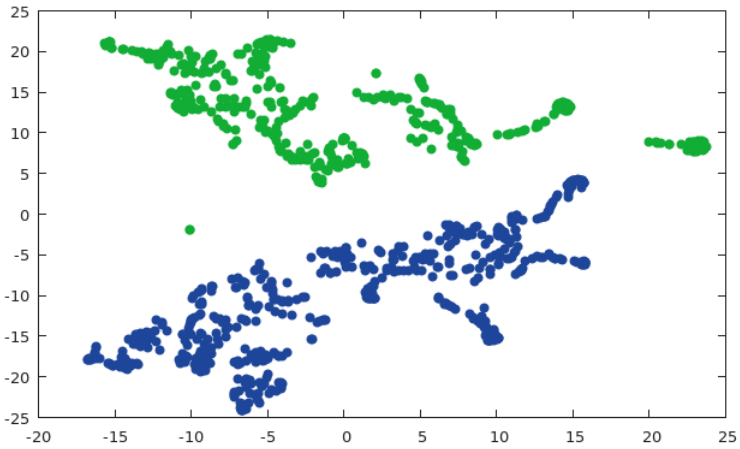


FIGURE 5.4 – Visualisation t-sne du plongement des representations intermédiaires utilisées dans $\mathbb{R}^2$ pour nos séquences été/hiver acquises depuis un tramway. Le nuage de point vert correspond à la séquence d'été, et le nuage bleu à la séquence acquise en hiver.

Afin de vérifier cette hypothèse, nous avons rajouté quelques couches supplémentaires après la couche 21 (5.5). Il s'agit simplement de réduire la dimensionalité et permettre quelques transformations simples afin de rapprocher les clusters. Concernant le choix de la fonction de coût, le point le plus important à prendre en compte est qu'il ne suffit pas que le réseau apprenne à produire un plongement tel que les représentations des images similaires soient proches les unes des autres. Il est tout aussi important que les représentations d'images qui ne correspondent pas soient distantes dans ce plongement. Il serait donc raisonnable d'utiliser une fonction de type Hinge loss classique (*e.g.* [102]), qui minimise la distance entre les représentations pour les données d'entrée similaires, et qui assure une distance minimale (*i.e.* une marge) entre les représentations des données d'entrées différentes. Cependant, nous avons observé qu'en pratique, l'utilisation de triplets d'images accompagnés de la fonction de coût proposée par [39] facilite la convergence. Le principe est simple : chaque batch de taille N sera composé de N triplets d'images $I_{ref}, I_{match}, I_{other}$, où I_ref est une image de référence et où les deux autres images sont choisies de manière à ce que (I_{ref}, I_{match}) soit un couple d'images correspondantes et que (I_{ref}, I_{other}) ne le soit pas. Plus formellement, on minimisera

$$-log(\frac{exp(-d(\mathcal{D}_{ref}, \mathcal{D}_{match}))}{exp(-d(\mathcal{D}_{ref}, \mathcal{D}_{match})) + exp(-d(\mathcal{D}_{ref}, \mathcal{D}_{other}))}) \tag{5.1}$$

où $d(.)$ retourne la distance (dans notre cas L2) entre ses arguments. Dans cette équation, $\mathcal{D}_i$ est le plongement que retourne la dernière couche avant la fonction de coût pour l'image I_i. Le réseau utilisé est illustré par la figure 5.5.

Nous avons utilisé la base *Alderly* ([81]) afin d'entraîner ces parties du réseau (le gradient du sous-réseau précédent la couche 22 est mis à zéro). Il s'agit des deux séquences jours/nuits acquises de surcroît dans des conditions météorologiques différentes, que nous avions précédemment mentionnées et dont quelques exemples illustratifs avaient été rapportés dans la figure 5.2. Les premières 8000 images de la base sont utilisées pour l'apprentissage et 2000 images sont utilisés pour les tests. Nos premiers résultats sont encourageants : le réseau prédit correctement l'image correspondante pour 94% de tous les triplets de l'ensemble de tests.

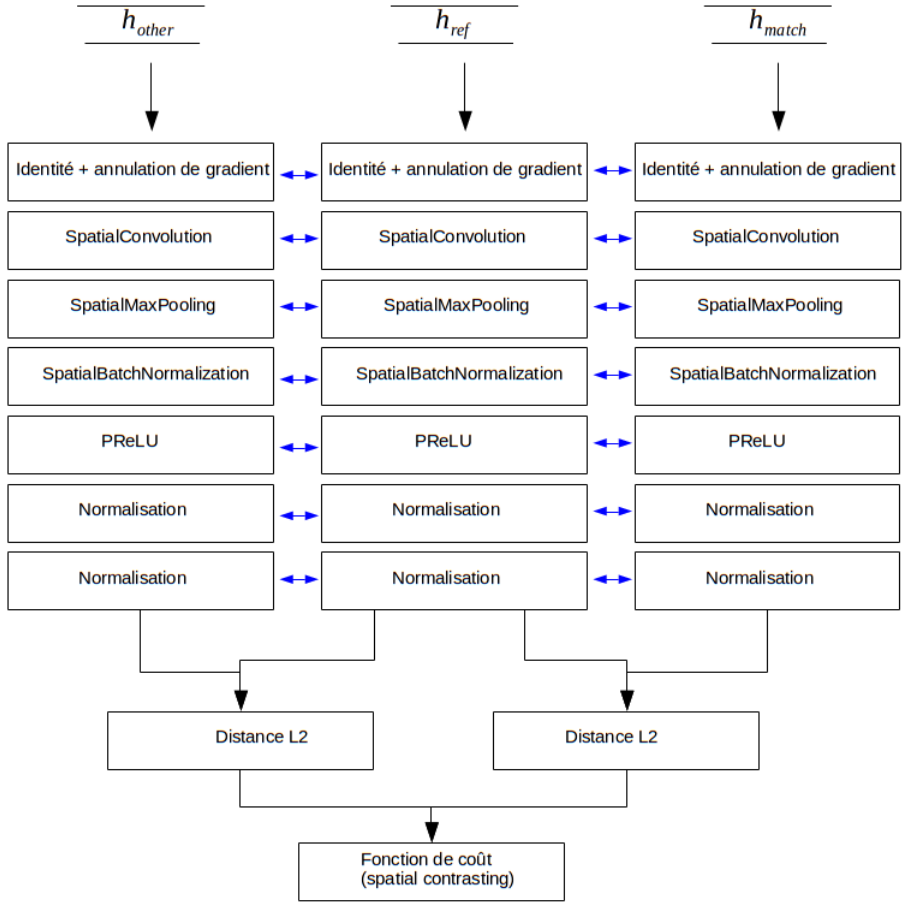


FIGURE 5.5 – Le réseau utilisé afin d'obtenir des représentations intermédiaires optimales. Les flèches bleues indiquent que les poids sont partagés entre les modules. La variable h_i représente le tenseur produite en sortie de la 21ème couche de Enet, lorsque celui-ci est entraîné pour la segmentation.

Bien que ces résultats soient encourageants, il nous sera nécessaire de conduire un plus grand nombre de tests pour évaluer la pertinence de la prise en compte des représentations issues de Enet de telle manière.

5.1.2 Calcul de pose absolue

Comme nous l'avons indiqué précédemment (§5.1), nous recherchons à robustifier notre système de VSLAM via l'addition d'une composante chargée d'imposer les contraintes issues de la mise en correspondance des images clefs du SLAM avec les images géoréférencées de la base. Comme les points $3d$ reconstruits par le SLAM sont le résultat de la triangulation de points d'intérêt classiques, nous choisissons de calculer la pose entre les images appariées via un réseau profond plutôt que d'imposer des contraintes définies par des reprojections de point.

Le calcul de pose à partir d'images monoculaires ne peut se faire en général qu'à une échelle près. Cependant, certains auteurs ([48]) ont exploré le problème mal posé de l'estimation des 7 degrés de liberté de la caméra. Comme on peut s'y attendre, l'entraînement nécessite un très grand nombre d'images, et le réseau ne parvient pas à généraliser le calcul de pose. Nous

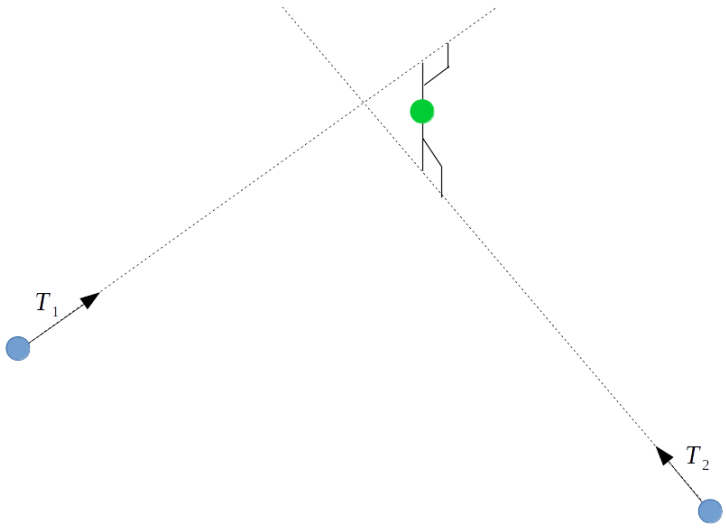


FIGURE 5.6 – Estimation de la position absolue d'une image requête à partir des positions relatives estimées à une échelle près par le réseau. Le principe reste celui de la triangulation (§1.1.5). Les cercles bleus représentent les caméras de la base de relocalisation, et nous connaissons leurs poses dans un repère géo-référencé. Le réseau estime uniquement la direction des translations entre chacune de ces caméras et l'image requête, translations qui sont notées T_1 et T_2 dans la figure. Il est clair que la position absolue de cette dernière se situe dans le cas idéal à l'intersection des rayons définis par T_1 et T_2, et en pratique au point minimisant la somme des distances à ceux-ci. Cette position est indiquée par le cercle vert.

pensons donc qu'il est plus raisonnable de calculer la pose relative à une échelle près entre une image clef du SLAM et $n > 1$ images de la base géoréférencée. Une fois que ces poses sont estimées, nous pouvons utiliser la triangulation afin d'obtenir la position de l'image dans le repère absolu (*i.e.* dans le repère géoréférencé de la base). Ce processus est illustré par la figure 5.6. Notons qu'il est plus avantageux d'effectuer ce calcul dans le cadre d'un RANSAC ou en incluant un noyau robuste.

Le réseau que nous utilisons prend en entrée les représentation h_i et h_j (produite par la 21ème couche de Enet) des deux images I_i et I_i et retourne, à une échelle près, la pose relative entre celles-ci. Son architecture globale est illustré par la figure 5.7.

Nous proposons d'utiliser la fonction de coût suivante :

$$\lambda(1 - T_{est}^T T_{gt}) + \begin{pmatrix}\alpha_1 & \alpha_2 & \alpha_3\end{pmatrix}^T \begin{pmatrix}\alpha_1 & \alpha_2 & \alpha_3\end{pmatrix} \tag{5.2}$$

Les variables α_i sont les coefficients des générateurs de l'algèbre de lie exprimant l'orientation. T_{est} et T_{gt} sont les vecteurs unitaires des directions des translations donnés par (respectivement) le réseau et la vérité terrain. Nous espérons que l'absence de contraintes sur l'amplitude des translations facilitera la généralisation du réseau à des exemples différents de la base d'apprentissage.

Nous noterons [78] proposent dans une publication récente d'estimer la pose relative entre les caméras de manière assez similaire à ce que nous proposons. Cependant, ces différences sont à noter :

- ▷ Leur fonction de coût est différent. Alors que celle que nous proposons cherche uniquement à ce que les directions de la translation données par la vérité terrain et celle

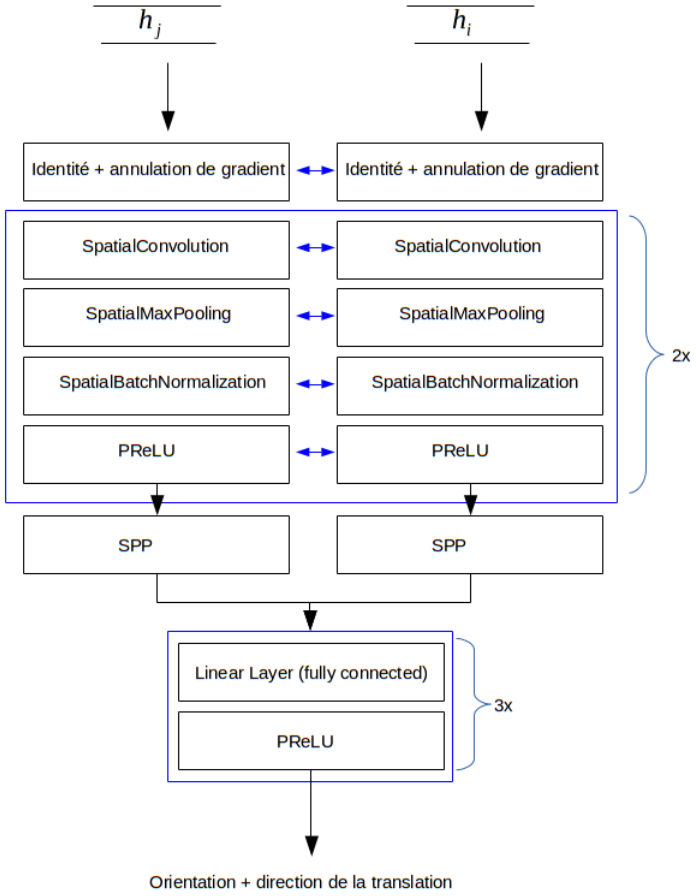


FIGURE 5.7 – Le réseau utilisé pour le calcul de la pose relative. Les flèches bleues indiquent que les poids sont partagés entre les modules.

de la translation estimée par le réseau coïncident, [78] imposent des contraintes dures sur sur chaque composantes de la translation, *i.e* ils remplacent le premier terme de l'équation 5.2 par la distance Euclidienne entre les translations à l'échelle exacte.

▷ L'architecture sur laquelle leurs travaux sont basés est différente. En effet, le réseau pré-entrainé qu'ils utilisent est un Hybrid-CNN ([135]).

▷ Nous cherchons à calculer la pose absolue dans le repère du SLAM, c'est pourquoi nous exploitons une étape de triangulation supplémentaire.

Notons que bien que les couches qu'ils rajoutent au CNN hybride soient légèrement différentes du notre, le principe reste semblable, c'est pourquoi nous pensons qu'il s'agit d'une différence mineure.

Bien entendu, il nous faudra conduire plusieurs expériences afin de pouvoir évaluer l'impact de ces différences. Nous faisons cependant face à un problème d'overfitting au moment de la rédaction de ces lignes. Nous soupçonnons le problème d'avoir pour origine l'insuffisance de la taille de notre base d'apprentissage, qui n'est autre que l'ensemble des résultats expérimentaux (~ 4000 images) présentés aux chapitre précédent (4.4). Bien qu'une telle base d'images

suffise à un apprentissage (chaque pixel est un exemple, donc on dispose en réalité de millions d'exemples labélisés), il est vraisemblable que ce nombre d'exemple soit dérisoire pour la tâche que nous souhaitons accomplir. Notre objectif à court terme est donc de rassembler une base d'image de taille suffisante.

5.2 Perspectives et travaux futurs

Nous avons déjà discuté des améliorations possibles aux sections 4.4.4 et 3.8 des deux chapitres précédents. Néanmoins, nous résumons ici les points importants :

Concernant l'amélioration de la fusion GPS/VSLAM (3), nos travaux futurs seront principalement orientés vers

- ▷ l'établissement de liens précis entre l'incertitude des paramètres de rotation et translation afin de pouvoir définir la matrice de pondération de l'équation 3.16.

- ▷ l'investigation théorique de l'effet d'un plus grand nombre de caméras contraintes sur la covariance de la caméra la plus récente (nous n'avons, suivant [67], imposer la contrainte GPS que sur la dernière caméra.

- ▷ des recherches plus théoriques visant à déterminer le liens précis entre le nombre de caméras nécessaire à la fusion sans graphe de pose et le nombre de caméras en graphe de pose nécessaire pour le succès de la fusion hybride.

L'amélioration de la fusion avec les bâtiments nécessitera des efforts dans le but de

- ▷ prendre en compte l'aspect temporel des séquences (*e.g.* via l'utilisation d'unités LSTM).

- ▷ concevoir une architecture plus légère afin de minimiser les décalages possibles entre la reconstruction initiale de chaque point et leur première association à un plan.

Comme nous l'avons souligné, la modélisation ou la correction du biais du GPS permettrait de résoudre plusieurs problèmes. D'une part, il serait possible de l'utiliser afin de définir la matrice de pondération de l'équation 3.16, et d'autre part, cela nous permettrait d'exploiter plus aisément le GPS et les modèles 3D des bâtiments simultanément. Dans nos travaux futur, nous tenterons donc d'explorer les possibilités de fusion rapprochée ([8, 119]).

ANNEXE A

Annexe A

Cette annexe regroupe quelques définitions, lemmes, théorèmes et formules utiles pour ce texte.

Lemme A. 1. *Soient $v, s \in \mathbb{R}^m$, tels que $\forall i$, les éléments v_i et s_i des deux vecteurs soient des fonctions réelles définies sur $\mathbb{R}^n$. Dans ce cas*

$$\frac{\partial v^T s}{\partial X} = v^T \frac{\partial s}{\partial X} + s^T \frac{\partial v}{\partial X} \tag{A.1}$$

Démonstration.

$$\begin{aligned} \frac{\partial v^T s}{\partial X} &= \begin{pmatrix} \frac{\partial v^T s}{\partial X_1} & \dots & \frac{\partial v^T s}{\partial X_n} \end{pmatrix} = \begin{pmatrix} \sum_{l=1}^{n} \frac{\partial v_i^T s_i}{\partial X_1} & \dots & \sum_{l=1}^{n} \frac{\partial v_i^T s_i}{\partial X_n} \end{pmatrix} \\ &= \begin{pmatrix} \sum_{l=1}^{n} \frac{\partial v_i}{\partial X_1} s_i & \dots & \sum_{l=1}^{n} \frac{\partial v_i}{\partial X_n} s_i \end{pmatrix} + \begin{pmatrix} \sum_{l=1}^{n} \frac{\partial s_i}{\partial X_1} v_i & \dots & \sum_{l=1}^{n} \frac{\partial s_i}{\partial X_n} v_i \end{pmatrix} \\ &= s^T \frac{\partial v}{\partial X} + v^T \frac{\partial s}{\partial X} \end{aligned} \tag{A.2}$$

□

Lemme A. 2. *Soient la matrice symétrique $A \in \mathbb{R}^{m \times k}$ et le vecteur $x \in \mathbb{R}^k$. Dance ce cas $\frac{\partial x^T A x}{\partial x} = 2x^T A$*

Démonstration. Triviale en appliquant le lemme précédent. □

Nous utiliserons la notation $g_f \triangleq J_f^T$ (pour une fonction f quelconque) dans le reste de l'annexe.

Lemme A. 3. *Soit $\mathcal{B}(x) = e^T e$ avec e est une fonction réelle, localement linéaire définie sur $\mathbb{R}^n$. Dans ce cas,*

$$J_{\mathcal{B}} = 2e^T J_e \tag{A.3}$$

et

$$H_{\mathcal{B}} = 2J_e^T J_e \tag{A.4}$$

Démonstration. L'équation A.3 découle naturellement du lemme 1. On déduit de A.3 que

$$H_{\mathcal{B}} = J_{g_{\mathcal{B}}} = 2(J_e^T J_e + e^T J_{J_e}) \tag{A.5}$$

Comme e est localement linéaire, J_{J_e} est nulle. □

Lemme A. 4. *Soient $U(x)\mathbb{R}^{n\times m}$ et $V(x)\mathbb{R}^{m\times k}$ deux matrices réelles, fonctions de $x \in \mathbb{R}$. Dans ce cas,*

$$\frac{\partial UV}{\partial x} = U\frac{\partial V}{\partial x} + \frac{\partial U}{\partial x}V \tag{A.6}$$

Démonstration. La démonstration est simple et peut être retrouvée par exemple dans [40]. □

Lemme A. 5. *Soient $f : \mathbb{R}^m \to \mathbb{R}^n$ et la fonction scalaire $\alpha : \mathbb{R}^k \to \mathbb{R}$. Dans ce cas, on a*

$$\frac{\partial \alpha(X) f(X)}{\partial X} = \alpha(X)\frac{\partial f(X)}{\partial X} + f(X)\frac{\partial \alpha(X)}{\partial X} \tag{A.7}$$

Démonstration. La démonstration découle naturellement de la définition de la Jacobienne. □

Lemme A. 6. *(Règle de la chaîne) Soient $f : \mathbb{R}^l \to \mathbb{R}^k$ et $g : \mathbb{R}^s \to \mathbb{R}^l$ deux fonctions différentiables en un point x. Dans ce cas les jacobiennes de f, g et $f \circ g$ vérifient*

$$J_{f\circ g} = J_f(g(x))J_g(x) \tag{A.8}$$

Démonstration. La démonstration est simple et est donnée dans la plupart des textes d'introduction au calcul matriciel. □

Définition A. 1. *(Complément de Schur) Soit*

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \tag{A.9}$$

une matrice. Soit D inversible, dans ce cas le complément de Schur par rapport à D est noté M/D et défini par

$$A - BD^{-1}C. \tag{A.10}$$

De manière similaire, on peut définir le complément de Schur de $M/A = D - CA^{-1}B$ si la matrice A est inversible.

Lemme A. 7. *Soit M une matrice partitionnée comme dans la définition précédente, et soit le bloc D inversible. Dans ce cas, le complément de Schur M/D nous permet d'écrire*

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} (M/D)^{-1} & -(M/D)^{-1}BD^{-1} \\ -D^{-1}C(M/D)^{-1} & D^{-1}c(M/D)^{-1}BD^{-1} + D^{-1} \end{pmatrix} \tag{A.11}$$

Démonstration. Il suffit de remarquer que la matrice M peut être réécrite sous forme bloc-diagonale comme suit :

$$\begin{pmatrix} I & -BD^{-1} \\ 0 & I \end{pmatrix}\begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} I & 0 \\ -D^{-1}C & I \end{pmatrix} = \begin{pmatrix} A/D & 0 \\ 0 & D \end{pmatrix} \tag{A.12}$$

Il suffit d'inverser les deux cotés de l'équation pour atteindre le résultat désiré. □

Lemme A. 8. Identité de Woodbury*([40]) : Soit A, U, C et V des matrices (respectivement) de tailles $n \times n$, $n \times m$,$m \times m$, $m \times n$ avec A et C inversibles. Dans ce cas*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \tag{A.13}$$

Démonstration. La démonstration est triviale, et peut se faire à partir du complément de Schur. Voir par exemple ([40]). □

Lemme A. 9. Identité de Sherman-Morrison *: Cette identité est un cas particulier de l'identité de Woodbury (lemme précédent). Lorsque la matrice C est de taille 1×1, i.e. lorsque UCV est une matrice de rang 1, alors*

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u} \tag{A.14}$$

Démonstration. Il suffit d'une substitutions dans l'équation A.13. □

Définition A. 2. *(Espace topologique) Notons 2^X l'ensemble des parties d'un ensemble X. Une topologie τ sur cet espace est un ensemble $\tau \subset 2^X$, dont chaque élément est appelé un ouvert, et qui vérifie*

- ▷ *$X \subset \tau$ et $\varnothing \subset \tau$.*
- ▷ *L'union d'un ensemble quelconque (fini ou infini) d'ouverts est un ouvert.*
- ▷ *L'intersection d'un ensemble fini d'ouvert est un ouvert.*

Définition A. 3. *(homéomorphisme) Une application f d'un espace topologique à un autre est dite homéomorphisme si et seulement si elle est bijective et continue, et que f^{-1} est elle aussi continue.*

Définition A. 4. *(Définition d'une variété différentielle) Soit M un espace topologique. Intuitivement, une variété différentielle de dimension m est un espace qui est localement Euclidien, de manière à ce que l'on puisse définir un système de coordonnées dans le voisinage de chaque point $p \in M$, et exploiter les outils standard de calcul différentiel dans ce repère. Ceci peut être formalisé comme suit : L'espace topologique M est une variété si et seulement si*

- ▷ *Pour un $p \in M$ quelconque, il existe au moins un ouvert O_i tel que $p \in O_i$.*
- ▷ *Pour chaque O_i , il existe un homéomorphisme $\psi_i : O_i \to V_i$ où V_i est un ouvert de $\mathbb{R}^m$.*

Soient O_i et O_j tels que $O_j \cap O_i \neq \varnothing$. Si la fonction $\psi_i \circ \psi_j^{-1}$, qui associe les points de $\psi_j(O_j \cap O_i) \subset V_j \subset \mathbb{R}^m$ à un sous-ensemble de $V_i \subset \mathbb{R}^m$ est de classe C^s et que $s >= 1$ alors la variété est différentiable. Dans le cadre de cette thèse, on supposera que ces fonctions sont de classe C^∞.

ANNEXE B

Annexe B

Les détails des contraintes GPS, d'élévation de terrain et d'odométrie, ainsi que les calculs de Jacobiennes et d'Hessiennes liées à la fusion par terme barrière sont regroupés dans cette annexe. On y retrouvera aussi les détails de la résolution du système Dense qui découle de cette dernière.

B.1 Jacobienne et Hessienne du terme barrière

Dans cette section, nous noterons le terme barrière $barr(X) = \frac{\gamma}{(t-\zeta(X))}$. Sa Jacobienne, son gradient et son Hessienne seront respectivement notées J_{barr}, g_{barr} et H_{barr}. La fonction ζ est une somme d'erreurs au carré

$$\zeta = r^T r. \tag{B.1}$$

Notons que le vecteur r n'est pas nécessairement homogène. Il peut par exemple s'agir simplement du vecteur des erreurs de reprojections (noté $e(X)$ au chapitre 1 , equation 1.25), ou de la concaténation d'erreurs de graphe de pose et de reprojections (chapitre 3).

Selon la règle de la chaîne (annexe B, 6), on a

$$J_{barr} = \gamma \frac{J_\zeta(X)}{(t-\zeta(X))^2} \tag{B.2}$$

et donc

$$g_{barr} = \gamma \frac{J_\zeta^T(X)}{(t-\zeta(X))^2} \tag{B.3}$$

On a naturellement $H_{barr} = \frac{\partial g_{barr}}{\partial X}$ d'après la définition de l'Hessienne. En utilisant le lemme 5, et le fait que $\frac{\partial J_\zeta^T}{\partial X} \approx 2 J_{err}^T J_{err}$, on obtient

$$H_{barr} = \frac{2\gamma}{(t-\zeta(X))^2} a (J_r^T J_r + \frac{1}{(t-\zeta)} J_\zeta^T J_\zeta) \tag{B.4}$$

Démonstration. □

Nous remarquons que $J_r^T J_r$ possède naturellement la structure creuse présentée à la section 1.3.2, mais que $J_\zeta^T J_\zeta$ est dense. Il n'est donc pas possible de résoudre le système $H_{bar}\Delta X = -g_{bar}$ via les méthodes classiques (*e.g.* complément de Schur, chapitre 1) de manière directe. Nous expliquerons dans la section § B.4, comment résoudre ce système de manière efficace.

B.2 Jacobienne et Hessienne de la fusion par terme barrière

Soient $C_1, ..., C_h$ des contraintes ajoutées dans le cadre de la fusion par terme barrière (*i.e.* équation 2.3). Le calcul des dérivées étant un opérateur linéaire, la Jacobienne et l'Hessienne de l'expression sont données par

$$J_{barr} + J_{C_1} + ... + J_{C_h} \tag{B.5}$$

$$H_{barr} + H_{C_1} + ... + H_{C_h} \tag{B.6}$$

Comme nous l'avons vu à la section précédente, H_barr est la somme d'une matrice creuse et d'une matrice dense de rang 1. Comme nous le verrons à la section §B.4, nous pouvons résoudre le système en temps-réel tant que le terme dense demeure de rang 1. Il faut donc accorder une attention particulière à la formulation des contraintes afint que leurs Hessiennes H_{C_i} ne contribuent pas au terme dense de manière à augmenter son rang. Nous détaillerons dans la section suivante les formulations utilisées pour nos contraintes, et nous vérifierons que ces dernières ont le comportement que nous désirons, c'est à dire qu'elle ne densifient pas l'Hessienne de manière à augmenter le rang du terme dense.

B.3 Jacobiennes et Hessiennes des différentes contraintes utilisées

B.3.1 GPS

La contrainte GPS que nous imposons est de la forme

$$\mathcal{C}_{gps} = ||P_{gps} X_{cam} - x_{gps}||^2 \tag{B.7}$$

où x_{gps} est la position GPS dans le plan horizontal (nous ignorons l'information d'altitude du GPS car celle que fournissent les capteurs bas-coût sur lesquels nous nous basons n'es pas fiable). Notons au passage que celle-ci est exprimée, comme les coordonnées des modèles $3d$ de bâtiments, dans le répère Lambert 93 (§1.9.3). Le vecteur X_cam de l'équation B.7 est le vecteur concaténant tout les paramètres extrinsèques des caméras optimisées (§1.3.2), *i.e.*

$$x_{cam} = \begin{pmatrix} \alpha_1^1 & \alpha_2^1 & \alpha_3^1 & T_1^1 & T_2^2 & T_3^3 & ... & \alpha_1^l & \alpha_2^l & \alpha_3^l & T_1^l & T_2^l & T_3^l \end{pmatrix}^T \tag{B.8}$$

où les α_i^j correspondent aux paramètres d'orientation de la caméra j. Les variables T_i^j indiquent la translation de cette caméra, et T_2 est son altitude. La matrice P_{gps} de l'équation B.7 est donc celle qui selectionne l'information de translation dans le plan. Plus formellement,

$$P_{gps} = \begin{pmatrix} \mathbf{0}_{3\times3} & B_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \dots & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & B_{3\times3} & \dots & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ & & \dots & & & & \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \dots & B_{3\times3} & \mathbf{0}_{3\times3} \end{pmatrix} \tag{B.9}$$

où chaque $B_{3\times3}$ est donnée par

$$B_{3\times3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{B.10}$$

Notons que dans le chapitre 3, $P_{gps} = (\mathbf{0}_{3\times3} \ , \ B_{3\times3})$ car seule la caméra la plus récente était contrainte. Le calcul de la Jacobienne et de l'Hessienne de cette contrainte est simple, il suffit d'utiliser le lemme 1 :

$$\frac{\partial \mathcal{C}_{gps}}{\partial X} = \frac{\partial (P_{gps}X_{cam} - x_{gps})^T (P_{gps}X_{cam} - x_{gps})}{\partial X} = 2(P_{gps}X_{cam} - x_{gps})^T P_{gps} \tag{B.11}$$

$$H_{gps} = 2P_{gps}^T P_{gps} \tag{B.12}$$

On voit donc clairement que cette matrice est diagonale, et qu'elle est non-nulle uniquement pour certains blocs correspondants à la translation. Par conséquent, l'ajout de la contrainte via le terme barrière ne densifiera pas l'Hessienne.

B.3.2 Contraîntes aux bâtiments, Contraintes d'élévation de terrain et d'odométrie

La contrainte aux bâtiments a été présentée au chapitre 2 (équation 2.4). La formulation des contraintes d'élévation de terrain est similaire : il s'agit de minimiser la distance Euclidienne entre chaque caméra de la fenêtre d'optimisation et l'élévation donnée par la somme entre la hauteur de la caméra par rapport à la chaussée et l'altitude fournie par les modèles d'élévation de terrain en ce point. La formulation de la contrainte d'odométrie est essentiellement la même que celle du GPS : il s'agit d'exploiter le vecteur de vitesse que retourne l'odomètre du véhicule afin de calculer un nouvelle position dans le repère monde, et de contraindre la position de la caméra à celle-ci.

L'expression sous forme d'équation de ces contraintes peut se faire de plusieurs manières mais reste triviale. De la même manière, le calcul des Jacobiennes se fait via des manipluations algébriques simples, similaires dans le principe à ce que nous avons vu pour la contrainte GPS. En outre, ces équations ne seront pas référencées dans ce mémoire. Nous renvoyons donc le lecteur intéressé à [56] (chapitres 4 et 5, et annexes A.2, A.3).

B.4 Résolution du système lié à la fusion par terme barrière

La méthode de résolution présentée est tirée de la section 5.1 de [68]. Nous reprendrons ici la notation introduite dans la section 1.3.2, mais pour le cas d'un vecteur d'erreurs r plus général (voir B.1). C'est à dire que nous noterons

$$H_\zeta = \begin{pmatrix} U & W \\ W^T & V \end{pmatrix} \approx 2J_r^T J_r \tag{B.13}$$

Comme nous l'avons vu, les contraintes imposées ne modifient pas la structure de l'Hessienne : les contraintes GPS, d'élévation de terrain ou d'odométrie ne rajoutent que des termes diagonaux à la matrice Uet les contraintes points/plan ne modifient que W. Par conséquent, on arrive, lors de la résolution de $H_{barr}\Delta X = -g_{barr}$ (ou lors de la résolution de $(H_{barr} + \lambda Id)\Delta X = -g_{barr}$ dans le cadre d'un algorithme de Levenberg-Marquardt), à un système semblable à

$$(\tilde{H} + \tilde{g}\tilde{g}^T)\Delta X = -g_{barr} \tag{B.14}$$

où $\tilde{H}$ possède la même structure que H_ζ, et $\tilde{g}\tilde{g}^T$ est un terme dense semblable au $J_{zeta}^T J_{zeta}$ de l'équation B.4. En utilisant l'identité de Sherman-Morrison (annexe A, équation A.14), on obtient

$$\Delta X = -(\tilde{H} + \tilde{g}\tilde{g}^T)^{-1} g_{barr} = (Id - \frac{\tilde{H}^{-1}\tilde{g}\tilde{g}^T}{1 + \tilde{g}^T\tilde{H}^{-1}\tilde{g}})\tilde{H}^{-1} g_{barr} \tag{B.15}$$

Puisque $\tilde{H}$ possède la structure creuse de B.13, on peut aisément et en temps réel résoudre les deux systèmes

$$\tilde{H}a = -g_{barr} \tag{B.16}$$

$$\tilde{H}b = \tilde{g} \tag{B.17}$$

en utilisant le complément de Schur (section 1.3.2). Ensuite, il suffit de substituer a à $-\tilde{H}^{-1}g_{bar}$, et b à $\tilde{H}^{-1}\tilde{g}$ dans B.15.

ANNEXE C

Annexe C

Dans cette annexe, nous détaillons le calcul de la Jacobienne de l'erreur du graphe de pose basée sur des contraintes épiploaires que nous avons présentée par l'équation 3.18.

C.1 Quelques notations et conventions

La matrice de calibration des paramètres intrinsèques sera notée K. Pour une matrice quelconque A, Nous utiliserons les notations d'indexation suivantes (empruntées à la syntaxe de MATLAB/Gnu Octave) :

$A(i,j) =$ élément de A se situant à l'intersection de la ligne i et de la colonne j.
$A(i:j,:) =$ La sous-matrice de A composée des lignes de i à j et de toutes les colonnes.
$A(:,i:j) =$ La sous-matrice de A composée des colonnes de i à j et de toutes les lignes.
$A(i:j,k:l) =$ La sous-matrice de A composée des lignes de i à j et de toutes les colonnes de k à l. (C.1)

On conservera les notations présentées à la section §3.5.0.2, mais en introduisant, par soucis de clarté, la notation

$$\Theta^i \in \mathbb{R}^{12} \tag{C.2}$$

telle que $\Theta^i(1:3,1)$ et $\Theta^i(7:9,1)$ sont, respectivement, les paramètres d'orientation [1] des caméras i et $i+1$, et telle que les paramètres de translation de i et $i+1$ soient respectivement donnés par $\Theta^i(4:6,1)$ et $\Theta^i(10:12,1)$. Nous noterons θ^i_k le $k-$ème élément de Θ^i.

Soient $R = R_i^T R_{i+1}$ et $T = R_i^T(T_{i+1} - T_i)$ la rotation et la translation permettant le passage du repère de la caméra $i+1$ à celui de la caméra i. Dans la suite, d'après la définition de la matrice Fondamentale, on développera certaines expressions en posant

1. *i.e.* les coefficients des générateurs de l'algèbre de Lie.

$$F_i = [T]_\wedge R = [R_i^T(T_{i+1} - T_i)]_\wedge R_i^T R_{i+1} \tag{C.3}$$

où $[.]_\wedge$ est l'opérateur du produit vectoriel, que nous avons introduit dans l'annexe A (????). En reprenant les notations introduites dans 1.4, on notera G_l le $l-$ème générateur de l'algèbre de Lie de $SO(3)$.

C.2 Le calcul des dérivées

Comme nous l'avons vu à la section 3.6, l'erreur ζ utilisée par le terme barrière (équation 2.3) peut s'écrire

$$\zeta = r^T r \tag{C.4}$$

où r est un vecteur regroupant les erreurs du graphe de pose et les erreurs de reprojection, *i.e.*

$$e = \begin{pmatrix} \psi_{11} \\ \psi_{1\Upsilon 1} \\ \dots \\ \psi_{n\Upsilon n} \\ \psi_{n\Upsilon n} \\ e_{11} \\ \dots \\ e_{1\Phi(1)} \\ e_{21} \\ \dots \\ e_{2\Phi(2)} \\ \dots \\ e_{l\Phi(l)} \end{pmatrix}. \tag{C.5}$$

Dans cette expression, e_{ij} est l'erreur de reprojection du point j dans la caméra i, et $\Phi(i)$ est le nombre de points $3d$ inliers visibles depuis cette dernière. La fonction $\Upsilon(i)$ désigne le nombre de couple de points $2d$ correspondants dans les images i et $i+1$. La variable ψ_{ij} désigne l'erreur de graphe de pose de type épipolaire pour le couple de caméra i,$i+1$ et le couple de points $2d$ correspondants (x_j, x'_j) (équation 3.18). Les points x_j et x'_j sont respectivement exprimés dans le repère du plan image des caméras i et $i+1$, mais en coordonnées homogènes (il s'agit donc de vecteurs de taille 3). Nous connaissons déjà les dérivées des erreurs e_{ij} [2], nous nous concentrons donc sur le calcul de

$$\frac{\partial \psi_{ij}}{\partial \Theta^i} = \frac{\partial}{\partial \Theta^i}\left(\frac{x_j^T F_i x'_j}{||PF_i x'_j||}\right) \triangleq \frac{u_{ij}(\Theta^i)}{v_{ij}(\Theta^i)} \tag{C.6}$$

Comme le contexte est clair, *i.e.* nous savons que l'équation C.6 correspond aux caméras i,$i+1$ et aux couple de points (x_j, x'_j), on omettra dans la suite les indexes et les superscripts i, j et ij. On réécrit par conséquent l'équation C.6 comme suit :

$$\frac{\partial}{\partial \Theta}\left(\frac{u(\Theta)}{v(\Theta)}\right) \in \mathbb{R}^{12} \tag{C.7}$$

2. Le calcul de ces dérivées est trivial, et a été détaillé de nombreuses fois dans la littérature, *e.g.* [83, 56].

Afin d'éviter des notations tensorielles encombrantes, nous calculerons chaque élément de ce vecteur de $\mathbb{R}^{12}$ séparément. Pour un élément quelconque de θ_l de ce Θ, nous restreignons le domaine des fonctions réelles u et v à $\mathbb{R}$. Nous pouvons donc poser l'expression suivante (d'après les règles de base de différentiation sur les fonctions de $\mathbb{R}$ vers $\mathbb{R}$) :

$$\frac{\partial}{\partial\theta_l}(\frac{u}{v}) = \frac{1}{v^2(\theta_l)}(\frac{\partial u}{\partial\theta_l}v(\theta_l) - \frac{\partial v}{\partial\theta_l}u(\theta_l)) \tag{C.8}$$

Dans les sous-sections qui suivent, nous détaillons le calcul de $\frac{\partial u}{\partial\theta_l}$ et $\frac{\partial v}{\partial\theta_l}$ pour chaque θ_l. Une fois ce calcul effectué, il suffit de les substituer dans l'équation C.8.

C.2.1 Calcul des $\frac{\partial u}{\partial\theta_l}$

On cherche à obtenir

$$\frac{\partial u(\theta_l)}{\partial\theta_l} = \frac{\partial}{\partial\theta_l}(x_j^T F_i x'_j) \tag{C.9}$$

Les expressions que nous obtiendrons pour cette dérivées sont très différentes en fonction de l'appartenance de θ_l aux paramètres de translation ou aux paramètres de rotation des caméras. C'est pourquoi nous séparons ce calcul pour les deux cas dans ce qui suit.

C.2.2 Si θ_l est un paramètre d'orientation

Dans ce cas, on a

$$\begin{aligned}\frac{\partial}{\partial\theta_l}(x_j^T F_i x' j) &= \frac{\partial}{\partial\theta_l}(x_j^T K^{-T}[R_i^T(T_{i+1}-T_i)]_\wedge R_i^T R_{i+1}k^{-1}x'_j) \\ &= x_j^T K^{-T}\frac{\partial}{\partial\theta_l}([R_i^T(T_{i+1}-T_i)]_\wedge R_i^T R_{i+1}k^{-1}x'_j)\end{aligned} \tag{C.10}$$

Notons que nous avons utilisé le lemme 4 pour obtenir la dernière expression. En utilisant une fois de plus ce théorème, l'équation C.10 peut être développée comme suit :

$$\begin{aligned}\frac{\partial}{\partial\theta_l}(x_j^T F_i x'_j) = x_j^T K^{-T}\Big([R_i^T(T_{i+1}-T_i)]_\wedge \frac{\partial}{\partial\theta_l}(R_i^T R_{i+1}K^{-1}x'_j) \\ + \frac{\partial}{\partial\theta_l}[R_i^T(T_{i+1}-T_i)]_\wedge R_i^T R_{i+1}K^{-1}x'_j\Big)\end{aligned} \tag{C.11}$$

Il nous faut donc calculer les deux termes $\frac{\partial}{\partial\theta_l}(R_i^T R_{i+1}K^{-1}x'_j)$ et $\frac{\partial}{\partial\theta_l}[R_i^T(T_{i+1}-T_i)]_\wedge$. Cependant, la forme que prend ces terme dépend de la caméra à laquelle θ_l appartient. Nous séparons donc ce calcul dans ce qui suit.

C.2.2.1 θ_l appartient à la caméra i.

Dans ce cas, on utilise une fois de plus le lemme 4 et on obtient

$$\begin{aligned}\frac{\partial}{\partial\theta_l}(R_i^T R_{i+1}K^{-1}x'_j) &= \frac{\partial R_i^T}{\partial\theta_l}R_{i+1}K^{-1}x'_j \\ &= \frac{\partial R_i^T exp(\sum\limits_k \theta_k G_k)}{\partial\theta_l}R_{i+1}K^{-1}x'_j = -R_i^T G_l R_{i+1}K^{-1}x'_j\end{aligned} \tag{C.12}$$

En ce qui concerne le deuxième terme que l'on recherchait, c'est à dire $\frac{\partial}{\partial\theta_l}[R_i^T(T_{i+1}-T_i)]_\wedge$, notons

$$\mathcal{L} = R_i^T(T_{i+1} - T_i) \tag{C.13}$$

On cherche donc à trouver

$$\frac{\partial[\mathcal{L}]_\wedge}{\partial\theta_l} = \frac{\partial}{\partial\theta_l}\begin{pmatrix}0 & -\mathcal{L}_3 & \mathcal{L}_2\\ \mathcal{L}_3 & 0 & -\mathcal{L}_1\\ -\mathcal{L}_2 & \mathcal{L}_1 & 0\end{pmatrix} = \begin{pmatrix}0 & -\frac{\partial\mathcal{L}_3}{\partial\theta_l} & \frac{\partial\mathcal{L}_2}{\partial\theta_l}\\ \frac{\partial\mathcal{L}_3}{\partial\theta_l} & 0 & -\frac{\partial\mathcal{L}_1}{\partial\theta_l}\\ -\frac{\partial\mathcal{L}_2}{\partial\theta_l} & \frac{\partial\mathcal{L}_1}{\partial\theta_l} & 0\end{pmatrix} \tag{C.14}$$

Nous nous permettons de rappeler au lecteur que θ_l est dans cette section un paramètre d'orientation. Il nous suffit pour trouver cette matrice, de connaître $\frac{\partial\mathcal{L}_k}{\partial\theta_l}$ pour chaque k et chaque θ_l. On utilise une fois de plus le lemme 4, et on obtient

$$\frac{\partial\mathcal{L}_k}{\partial\theta_l} = \frac{\partial R_i^T}{\partial\theta_l}(T_{i+1} - T_i) = -R_i^T G_l(T_{i+1} - T_i) \tag{C.15}$$

C.2.2.2 θ_l appartient à la caméra $i+1$.

Dans ce cas, il est évident que le terme $\frac{\partial}{\partial\theta_l}[R_i^T(T_{i+1} - T_i)]_\wedge$ s'annule car il ne dépend pas de la caméra $i+1$. In nous reste donc qu'à déterminer $\frac{\partial}{\partial\theta_l}(R_i^T R_{i+1}K^{-1}x'_j)$. Une fois de plus, le lemme 4 nous permet d'obtenir

$$\begin{aligned}&\frac{\partial}{\partial\theta_l}(R_i^T R_{i+1}K^{-1}x'_j)\\ &= R_i^T\frac{\partial}{\partial\theta_l}(R_{i+1}K^{-1}x'_j)\\ &= R_i^T\frac{\partial R_{i+1}}{\partial\theta_l}K^{-1}x'_j\\ &= R_i^T exp(\sum_k \theta_k G_k)R_{i+1}K^{-1}x'_j\\ &= R_i^T G_l R_{i+1}K^{-1}x'_j\end{aligned} \tag{C.16}$$

Nous pouvons donc maintenant calculer les dérivées par rapport à tous les paramètres d'orientation.

C.2.3 Si θ_l est un paramètre de translation

Une fois de plus, grâce au lemme 4, nous pouvons écrire

$$\begin{aligned}\frac{\partial}{\partial\theta_l}(x_j^T F_i x_j') &= \frac{\partial}{\partial\theta_l}(x_j^T K^{-T}[R_i^T(T_{i+1}-T_i)]_\wedge R_i^T R_{i+1}K^{-1}x_j') \\ &= x_j^T K^{-T}(\frac{\partial}{\partial\theta_l}[R_i^T(T_{i+1}-T_i)]_\wedge R_i^T R_{i+1}K^{-1}x_j') \\ &= x_j^T K^{-T}\frac{\partial}{\partial\theta_l}[R_i^T(T_{i+1}-T_i)]_\wedge R_i^T R_{i+1}K^{-1}x_j'\end{aligned} \tag{C.17}$$

Il nous suffit donc de connaître $\frac{\partial}{\partial\theta_l}[R_i^T(T_{i+1}-T_i)]_\wedge$. Reprenons la notation introduite plus haut, *i.e.*

$$\mathcal{L} = R_i^T(T_{i+1}-T_i) \tag{C.18}$$

Donc, la situation est à un détail prêt la même que pour l'équation C.14. En effet, il nous faut maintenant connaître les $\frac{\partial\mathcal{L}_k}{\partial\theta_l}$, sauf que cette fois, θ_l est un paramètre de translation. On a donc

$$\begin{aligned}\frac{\partial\mathcal{L}_k}{\partial\theta_l} &= R_i^T\frac{\partial}{\partial\theta_l}(T_{i+1}-T_i) \\ &= \begin{cases} -R_i^T L_l, & \theta_l \text{ appartient à la caméra } i \\ R_i^T L_l, & \theta_l \text{ appartient à la caméra } i+1 \end{cases}\end{aligned} \tag{C.19}$$

où le vecteur $L_l \in \mathbb{R}^3$ est tel que le $L_l(i,1) = 0$ pour tout $i \neq l$ et $L_l(l,1) = 1$.

C.2.4 Calcul des $\frac{\partial v}{\partial\theta_l}$

On cherche à trouver la dérivée de

$$v(\theta_l) = \Big((PF_i x_j')^T(PF_i x_j')\Big)^{\frac{1}{2}}, \tag{C.20}$$

par rapport à θ_l. L'application de la règle de la chaîne (6) permet d'obtenir

$$\frac{\partial v}{\partial\theta_l} = \frac{1}{2}((PF_i x_j')^T(PF_i x_j'))^{-\frac{1}{2}}\frac{\partial}{\partial\theta_l}\Big((PF_i x_j')^T(PF_i x_j)\Big) \tag{C.21}$$

En utilisant le lemme 1, on obtient

$$\frac{\partial}{\partial\theta_l}(PF_i x_j')^T(PF_i x_j') = 2(PF_i x_j')^T\frac{\partial}{\partial\theta_l}(PF_i x'j) \tag{C.22}$$

On remarque ensuite que le développent de $\frac{\partial}{\partial\theta_l}(PF_i x'j)$ est essentiellement le même que celui de C.10, sauf que le terme $x_j^T K^{-T}$ à gauche de la dernière expression est remplacé par PK^{-T}.

Bibliographie

[1] S. Yeung Fei-Fei Li A. Karpathy, J. Johnson. cs231n, stanford lecture. "https ://www.u-blox.com/en/tutorials-and-links".

[2] Roberto Arroyo, Pablo F Alcantarilla, Luis M Bergasa, and Eduardo Romera. Towards life-long visual localization using an efficient matching of binary sequences from images. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 6328–6335. IEEE, 2015.

[3] Clemens Arth, Christian Pirchheim, Jonathan Ventura, Dieter Schmalstieg, and Vincent Lepetit. Instant outdoor localization and slam initialization from 2.5 d maps. *IEEE transactions on visualization and computer graphics*, 21(11) :1309–1318, 2015.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet : A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv :1511.00561*, 2015.

[5] Dongdong Bai, Chaoqun Wang, Bo Zhang, Xiaodong Yi, and Xuejun Yang. Cnn feature boosted seqslam for real-time loop closure detection. *arXiv preprint arXiv :1704.05016*, 2017.

[6] Y Bard. *Nonlinear parameter estimation*. Academic pres, New york, 1974.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.

[8] Amani Ben-Afia, Lina Deambrogio, Daniel Salós, Anne-Christine Escher, Christophe Macabiau, Laurent Soulier, and Vincent Gay-Bellile. Review and classification of vision-based localisation techniques in unknown environments. *IET Radar, Sonar & Navigation*, 8(9) :1059–1072, 2014.

[9] Shehroze Bhatti, Alban Desmaison, Ondrej Miksik, Nantas Nardelli, N Siddharth, and Philip HS Torr. Playing doom with slam-augmented deep reinforcement learning. *arXiv preprint arXiv :1612.00380*, 2016.

[10] Martin Byröd and Karl Åström. Bundle adjustment using conjugate gradients with multiscale preconditioning. In *British Machine Vision Conference*, 2009.

[11] Baptiste Charmette, Eric Royer, and Frédéric Chausse. Efficient planar features matching for robot localization using gpu. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 16–23. IEEE, 2010.

[12] Ronan Collobert, Koray Kavukcuoglu, and ClÃ©ment Farabet. Torch7 : A matlab-like environment for machine learning.

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv preprint arXiv :1606.03798*, 2016.

[15] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Toward geometric deep slam. *arXiv preprint arXiv :1707.07410*, 2017.

[16] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul) :2121–2159, 2011.

[17] A Elaksher, J Bethel, and E Mikhail. Reconstructing 3d building wireframes from multiple images. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A) :91–96, 2002.

[18] Cameron Ellum. Integration of raw gps measurements into a bundle adjustment. *IAPRS series vol. XXXV*, 3025, 2006.

[19] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[20] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam : Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[21] Alexandre Eudes and Maxime Lhuillier. Error propagations for local bundle adjustment. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2411–2418. IEEE, 2009.

[22] Ryan M Eustice, Hanumant Singh, and John J Leonard. Exactly sparse delayed-state filters. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2417–2424. IEEE, 2005.

[23] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8) :1915–1929, 2013.

[24] Luis Ferraz, Xavier Binefa, and Francesc Moreno-Noguer. Very fast solution to the pnp problem with algebraic outlier rejection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–508, 2014.

[25] Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. Descriptor matching with convolutional neural networks : a comparison to sift. *arXiv preprint arXiv :1405.5769*, 2014.

[26] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4557–4564. IEEE, 2012.

[27] Jean Gallier. *Geometric methods and applications : for computer science and engineering*, volume 38. Springer Science & Business Media, 2011.

[28] Dinesh Gamage and Tom Drummond. Reduced dimensionality extended kalman filter for slam. In *BMVC*, 2013.

[29] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget : Continual prediction with lstm. 1999.

[30] Olivier Gomez, Achkan Salehi, Vincent Gay-bellile, and Mathieu Carrier. Accurate indoor localization through constrained visual slam. In *Proceedings of the 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[32] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet : Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015.

[33] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.

[34] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[35] Jindřich Havlík and Ondřej Straka. Performance evaluation of iterated extended kalman filter with variable step-length. In *Journal of Physics : Conference Series*, volume 659, page 012022. IOP Publishing, 2015.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.

[39] Elad Hoffer, Itay Hubara, and Nir Ailon. Deep unsupervised learning through spatial contrasting. *arXiv preprint arXiv :1610.00243*, 2016.

[40] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[41] Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[42] Yekeun Jeong, David Nister, Drew Steedly, Richard Szeliski, and In-So Kweon. Pushing the envelope of modern methods for bundle adjustment. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(8) :1605–1617, 2012.

[43] Hordur Johannsson, Michael Kaess, Maurice Fallon, and John J Leonard. Temporally scalable visual slam using a reduced pose graph. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 54–61. IEEE, 2013.

[44] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D) :35–45, 1960.

[45] Elliott Kaplan and Christopher Hegarty. *Understanding GPS : principles and applications*. Artech house, 2005.

[46] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. *arXiv preprint arXiv :1704.00390*, 2017.

[47] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision ? *arXiv preprint arXiv :1703.04977*, 2017.

[48] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet : A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2938–2946, 2015.

[49] Ho-Duck Kim, Sang-Wook Seo, In-hun Jang, and Kwee-Bo Sim. Slam of mobile robot in the indoor environment with digital magnetic compass and ultrasonic sensors. In *Control, Automation and Systems, 2007. ICCAS'07. International Conference on*, pages 87–90. IEEE, 2007.

[50] Diederik Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

[51] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 83–86. IEEE, 2009.

[52] Kurt Konolige and Willow Garage. Sparse sparse bundle adjustment. In *BMVC*, pages 1–11. Citeseer, 2010.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[54] Abhijit Kundu, Yin Li, Frank Dellaert, Fuxin Li, and James M Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *European Conference on Computer Vision*, pages 703–718. Springer, 2014.

[55] Jean Laneurit, Roland Chapuis, and Frédéric Chausse. Accurate vehicle positioning on a numerical map. *International Journal of Control, Automation, and Systems*, 3(1) :15–31, 2005.

[56] Dorra Larnaout. *Localisation d'un véhicule à l'aide d'un SLAM visuel contraint*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2014.

[57] Dorra Larnaout, Steve Bourgeois, Vincent Gay-Bellile, and Michel Dhome. Towards bundle adjustment with gis constraints for online geo-localization of a vehicle in urban center. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 348–355. IEEE, 2012.

[58] Dorra Larnaout, Vincent Gay-Bellile, Steve Bourgeois, and Michel Dhome. Vehicle 6-dof localization based on slam constrained by gps and digital elevation model information. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 2504–2508. IEEE, 2013.

[59] Dorra Larnaout, Vincent Gay-Bellile, Steve Bourgeois, Bastien Labbe, and Michel Dhome. Fast and automatic city-scale environment modeling for an accurate 6dof vehicle localization. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 265–266. IEEE, 2013.

[60] Dorra Larnaout, Vincent Gay-Bellile, Steve Bourgeois, Benjamin Labbe, and Michel Dhome. Driving in an augmented-city : from fast and automatic large scale environment modeling to on-line 6dof vehicle localization. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, pages 9–16. ACM, 2013.

[61] Dorra Larnaout, Vincent Gay-Belllile, Steve Bourgeois, and Michel Dhome. Vision-based differential gps : Improving vslam/gps fusion in urban environment with 3d building models. In *3D Vision (3DV), 2014 2nd International Conference on*, volume 1, pages 432–439. IEEE, 2014.

[62] Jean-Marc Lavest, Marc Viala, and Michel Dhome. Do we really need an accurate calibration pattern to achieve a reliable camera calibration ? In *European Conference on Computer Vision*, pages 158–174. Springer, 1998.

[63] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[64] Gim Hee Lee, Bo Li, Marc Pollefeys, and Friedrich Fraundorfer. Minimal solutions for pose estimation of a multi-camera system. In *Robotics Research*, pages 521–538. Springer, 2016.

[65] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp : An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2) :155–166, 2009.

[66] Stefan Leutenegger, Paul Timothy Furgale, Vincent Rabaud, Margarita Chli, Kurt Konolige, and Roland Siegwart. Keyframe-based visual-inertial slam using nonlinear optimization. In *Robotics : Science and Systems*, 2013.

[67] Maxime Lhuillier. Fusion of gps and structure-from-motion using constrained bundle adjustments. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3025–3032. IEEE, 2011.

[68] Maxime Lhuillier. Incremental fusion of structure-from-motion and gps using constrained bundle adjustments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(12) :2489–2495, 2012.

[69] M. Li and A. I. Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *International Journal of Robotics Research*, 32(6) :690–711, May 2013.

[70] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1) :503–528, 1989.

[71] Angelique Loesch, Steve Bourgeois, Vincent Gay-Bellile, and Michel Dhome. Generic edgelet-based tracking of 3d objects in real-time. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 6059–6066. IEEE, 2015.

[72] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[73] Manolis IA Lourakis and Antonis A Argyros. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment ? In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1526–1531. IEEE, 2005.

[74] Vincent Lui and Tom Drummond. An iterative 5-pt algorithm for fast and robust essential matrix estimation. In *BMVC*, 2013.

[75] Simon Lynen, Markus W Achtelik, Stephan Weiss, Margarita Chli, and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3923–3929. IEEE, 2013.

[76] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov) :2579–2605, 2008.

[77] András L Majdik, Damiano Verda, Yves Albers-Schoenberg, and Davide Scaramuzza. Micro air vehicle localization and position tracking from textured 3d cadastral models. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 920–927. IEEE, 2014.

[78] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Relative camera pose estimation using convolutional neural networks. *arXiv preprint arXiv :1702.01381*, 2017.

[79] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[80] Sven Middelberg, Torsten Sattler, Ole Untzelmann, and Leif Kobbelt. Scalable 6-dof localization on mobile devices. In *Computer Vision–ECCV 2014*, pages 268–283. Springer, 2014.

[81] Michael J Milford and Gordon F Wyeth. Seqslam : Visual route-based navigation for sunny summer days and stormy winter nights. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1643–1649. IEEE, 2012.

[82] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv :1611.03673*, 2016.

[83] Etienne Mouragnon. *Reconstruction 3D et localisation simultanée de caméras mobiles : une approche temps-réel par ajustement de faisceaux local*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2007.

[84] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Real time localization and 3d reconstruction. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 363–370. IEEE, 2006.

[85] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and automation, 2007 IEEE international conference on*, pages 3565–3572. IEEE, 2007.

[86] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam : a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5) :1147–1163, 2015.

[87] Raúl Mur-Artal and Juan D Tardós. Orb-slam2 : An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 2017.

[88] Achille Murangira, Christian Musso, and Karim Dahia. A mixture regularized rao-blackwellized particle filter for terrain positioning. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4) :1967–1985, 2016.

[89] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[90] Esha D Nerurkar, Kejian J Wu, and Stergios I Roumeliotis. C-klam : Constrained keyframe-based localization and mapping. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3638–3643. IEEE, 2014.

[91] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion : Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[92] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam : Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[93] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6) :756–770, 2004.

[94] Jason M O'Kane. Global localization using odometry. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 37–42. IEEE, 2006.

[95] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet : A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv :1606.02147*, 2016.

[96] Leonard John J Pillai, Sudeep. Towards visual ego-motion learning in robots. *arXiv preprint arXiv :1705.10279*, 2017.

[97] Victor A Prisacariu and Ian D Reid. Pwp3d : Real-time segmentation and tracking of 3d objects. *International journal of computer vision*, 98(3) :335–354, 2012.

[98] Timo Pylvanainen, Kimmo Roimela, Ramakrishna Vedantham, Joonas Itaranta, and Radek Grzeszczuk. Automatic alignment and multi-view segmentation of street view data using 3d shape prior. In *Fifth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, Paris, France, 2010.

[99] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1) :145–151, 1999.

[100] Jason R Rambach, Aditya Tewari, Alain Pagani, and Didier Stricker. Learning to fuse : A deep learning approach to visual-inertial camera pose estimation. In *Mixed and Augmented Reality (ISMAR), 2016 IEEE International Symposium on*, pages 71–76. IEEE, 2016.

[101] Fletcher Roger. Practical methods of optimization. *Constrained Optimization*, 2, 1987.

[102] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same ? *Neural Computation*, 16(5) :1063–1076, 2004.

[103] Chris Russell, Pushmeet Kohli, Philip HS Torr, et al. Associative hierarchical crfs for object class image segmentation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 739–746. IEEE, 2009.

[104] Achkan Salehi, Vincent Gay-Bellile, Steve Bourgeois, and Frédéric Chausse. Improving constrained bundle adjustment through semantic scene labeling. In *Computer Vision–ECCV 2016 Workshops*, pages 133–142. Springer, 2016.

[105] Achkan Salehi, Vincent Gay-bellile, Steve Bourgeois, and Frédéric Chausse. A hybrid bundle adjustment/pose-graph approach to vslam/gps fusion for low-capacity platforms. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 1728–1735. IEEE, 2017.

[106] Achkan Salehi, Vincent Gay-bellile, Steve Bourgeois, FrÃ©dÃ©ric Chausse, and Nicolas Allezard. Large-scale, drift-free slam using highly robustified building model constraints. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017.

[107] Peter Shirley. Color transfer between images. *IEEE Corn*, 21 :34–41, 2001.

[108] Dieter Sibley, Christopher Mei, Ian D Reid, and Paul Newman. Adaptive relative bundle adjustment. In *Robotics : science and systems*, volume 32, page 33, 2009.

[109] Robert Sim, Pantelis Elinas, Matt Griffin, James J Little, et al. Vision-based slam using the rao-blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, volume 14, pages 9–16, 2005.

[110] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.

[111] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1) :1929–1958, 2014.

[112] Hauke Strasdat. *Local accuracy and global consistency for efficient visual slam*. PhD thesis, Citeseer, 2012.

[113] Hauke Strasdat, Andrew J Davison, JMM Montiel, and Kurt Konolig. Double window optimisation for constant time visual slam. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2352–2359. IEEE, 2011.

[114] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Real-time monocular slam : Why filter ? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664. IEEE, 2010.

[115] Peter Sturm, Srikumar Ramalingam, Jean-Philippe Tardif, Simone Gasparini, and Joao Barreto. Camera models and fundamental concepts used in geometric computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(1–2) :1–183, 2011.

[116] Charles Sutton, Andrew McCallum, et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4) :267–373, 2012.

[117] Richard Szeliski. *Computer vision : algorithms and applications*. Springer Science & Business Media, 2010.

[118] Mohamed Tamaazousti, Vincent Gay-Bellile, Sylvie Naudet Collette, Steve Bourgeois, and Michel Dhome. Nonlinear refinement of structure from motion reconstruction by taking advantage of a partial knowledge of the environment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3073–3080. IEEE, 2011.

[119] Zui Tao and Philippe Bonnifait. Sequential data fusion of gnss pseudoranges and dopplers with map-based vision systems. *IEEE Transactions on Intelligent Vehicles*, 1(3) :254–265, 2016.

[120] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam : Real-time dense monocular slam with learned depth prediction. *arXiv preprint arXiv :1704.03489*, 2017.

[121] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. 2005.

[122] Philip HS Torr and Andrew Zisserman. Feature based methods for structure and motion estimation. In *Workshop on vision algorithms*, volume 1883, pages 278–294. Springer, 1999.

[123] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment-a modern synthesis. In *Vision algorithms : theory and practice*, pages 298–372. Springer, 1999.

[124] Tommi Tykkälä, Andrew I Comport, and Joni-Kristian Kämäräinen. Photorealistic 3d mapping of indoors by rgb-d scanning process. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1050–1055. IEEE, 2013.

[125] UBlox. Ublox gps compendium. "https ://www.u-blox.com/en/tutorials-and-links".

[126] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon : Depth and motion network for learning monocular stereo. *arXiv preprint arXiv :1612.02401*, 2016.

[127] Robert M Wald. *General relativity*. University of Chicago press, 2010.

[128] Chaolei Wang, Tianmiao Wang, Jianhong Liang, Yang Chen, Yicheng Zhang, and Cong Wang. Monocular visual slam for small uavs in gps-denied environments. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 896–901. IEEE, 2012.

[129] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo : towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2043–2050. IEEE, 2017.

[130] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion : Dense slam without a pose graph. Robotics : Science and Systems, 2015.

[131] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. In *European Conference on Computer Vision*, pages 61–75. Springer, 2014.

[132] Philippe Xu. Kitti semantic segmentation. "https ://www.hds.utc.fr/ xuphilip/dokuwiki/en/data".

[133] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *European Conference on Computer Vision*, pages 756–771. Springer, 2014.

[134] Matthew D Zeiler. Adadelta : an adaptive learning rate method. *arXiv preprint arXiv :1212.5701*, 2012.

[135] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[136] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. *arXiv preprint arXiv :1704.07813*, 2017.

Table des figures

Liste des tableaux

Table des matières