Implantation dans le matériel de fonctionnalités temps-réel dans une caméra intelligente ultralégère spécialisée pour la prise de vue aérienne

Ahmad Audi

▶ **To cite this version:**

Ahmad Audi. Implantation dans le matériel de fonctionnalités temps-réel dans une caméra intelligente ultralégère spécialisée pour la prise de vue aérienne. Traitement du signal et de l'image [eess.SP]. Université Paris-Est, 2017. Français. NNT : 2017PESC1022 . tel-01670060v2

HAL Id: tel-01670060
https://hal.science/tel-01670060v2

Submitted on 13 Mar 2018

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

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

UNIVERSITÉ PARIS-EST

IGN
INSTITUT NATIONAL DE L'INFORMATION GÉOGRAPHIQUE ET FORESTIÈRE

École doctorale Paris-Est
Mathématiques & Sciences et Technologies de l'Information et de la Communication
Spécialité: Électronique, Optronique et Systèmes

THÈSE DE DOCTORAT

En vue de l'obtention du grade de :
Docteur de l'Université Paris-Est

présentée et soutenue le 14 Décembre 2017 par :

Ahmad Audi

Implantation dans le matériel de fonctionnalités temps-réel dans une caméra intelligente ultralégère spécialisée pour la prise de vue aérienne

Composition du jury :

Rapporteur	M. Kasser Michel	HEIG-VD
Rapporteur	M. Manzanera Antoine	ENSTA-ParisTech
Examinateur	M. Le Besnerais Guy	ONERA/DTIS
Examinateur	Mme. Gouet Valerie	IGN/LaSTIG
Directeur	M. Pierrot-Deseilligny Marc	IGN/LaSTIG
Co-directeur	M. Thom Christian	IGN/LaSTIG

THESE

Remerciements

Tout d'abord, je tiens à remercier Marc Pierrot-Deseilligny pour la confiance qu'il m'a accordée en acceptant de diriger cette thèse. Je le remercie aussi pour ses conseils qui m'ont permis de mener à bien ces travaux de recherche.

Je remercie également Christian Thom pour m'avoir proposé cette thèse au sein de son laboratoire du LaSTIG/LOEMI, ainsi que pour son encadrement et pour son soutien tout au long de cette thèse.

Merci également à Christophe Meynard pour sa participation à la réalisation de ce travail et pour ses multiples conseils qui ont été toujours efficaces. Grâce à son expérience, j'ai réussi à acquérir des compétences en programmation et en FPGA.

J'adresse aussi mes remerciements aux membres du jury qui ont accepté de juger mon travail.

Mes remerciements vont aussi aux membres de l'équipe du LOEMI qui ont participé au bon déroulement de la thèse, par l'ambiance qu'ils ont créée au sein du laboratoire. Merci à Olivier pour ses nombreux coups de main et pour son aide lors des tests réalisés sur le terrain. Merci à Jean-Michael pour ses conseils qui ont été toujours efficaces. Merci à Jean-Philippe qui aura donné de son temps pour relire et corriger ce manuscrit. Merci aussi à Yann, Amjad, Mehdi, Giang pour leurs soutiens et leurs encouragements. Merci à nouveau à Christian, à Marc et à Joe qui ont contribué activement à la réalisation de mes travaux et qui m'ont permis de construire une première formation de recherche. J'ai été extrêmement sensible à leurs qualités humaines tout au long de ma thèse.

Pendant les trois années de la thèse, j'ai eu l'occasion d'enseigner un peu l'informatique dans le cadre d'un contrat de monitorat à l'ENSG. Je tiens donc à remercier Emmanuel Bardière et Didier Richard pour m'avoir proposé de donner des cours de programmation orienté objet (C/C++ et Java) et de programmation parallèle. Cela a été pour moi une excellente première expérience d'enseignement supérieur.

Je tiens à remercier aussi Jean Luc Sorin de l'Ifsttar et Mehdi Daakir de Vinci pour les expérimentations réalisées avec les drones.

Enfin, merci aux membres de ma famille au Liban pour leur soutien même s'ils sont physiquement loin de moi et pour leurs encouragements.

Résumé

Au cours des dernières années, les drones civils sont devenus un outil intéressant dans la photographie aérienne et dans les travaux de photogrammétrie. Cela a poussé le LOEMI (Laboratoire d'Opto-Electronique, Métrologie et Instrumentation) de l'IGN (Institut National de l'Information Géographique et Forestière) à mettre au point une nouvelle caméra aérienne mieux adaptée pour l'exploitation photogrammétrique et métrologique des images que les caméras grand public. Cette caméra est composée essentiellement d'un capteur CMOS "global-shutter", d'une centrale inertielle IMU, et d'un système sur puce (FPGA + 2 CPUs) pour la gestion de l'acquisition des images. Ce SoC/FPGA ouvre la porte à l'implémentation temps-réel des algorithmes de traitement d'image.

Parmi les travaux futurs de l'IGN, on peut distinguer certaines applications qui nécessitent l'acquisition des images aériennes avec un temps d'exposition long, comme par exemple les prises de vue aériennes en bande spectrale étroite et les prises de vue aériennes de nuit. Ce type de prises de vue manifeste un flou de bougé dans les images dû aux mouvements erratiques du drone.

Cette thèse consiste en l'implémentation dans la caméra légère de l'IGN d'un algorithme qui permet de remédier à ce problème de flou de bougé.

La première partie de ce travail a été consacrée au développement de la méthode qui consiste à acquérir plusieurs images avec un temps de pose court, puis utiliser un algorithme de traitement d'image afin de générer une image empilée finale avec l'équivalent d'un temps de pose long. Les paramètres des orientations correctes pour le ré-échantillonnage des images sont obtenus par l'estimation de la transformation géométrique entre la première image et la nième image à partir des points d'intérêts détectés par FAST dans la première image et les points homologues obtenus par corrélation dans les autres images accélérées par les capteurs inertiels intégrés à la caméra.

Afin d'augmenter la vitesse de traitement de calcul de notre algorithme, certaines phases sont accélérées en les implémentant dans le matériel (SoC/FPGA).

Les résultats obtenus sur des jeux de tests acquis avec un drone type Copter 1B UAV et la caméra ultra-légère de l'IGN montrent que l'image finale empilée ne présente pas un flou de bougé. Les temps d'exécution des différentes phases montrent qu'il est utilisable en temps-réel pendant les prises de vue pour la plupart de nos applications.

Dans le futur, les paramètres de l'algorithme seront affinés afin d'optimiser les résultats et le temps d'exécution du processus. Il serait intéressant aussi d'exploiter la présence du récepteur GNSS et des accéléromètres pour améliorer la phase de prédiction de positions de points homologues. Par extension, ce travail sera utilisé dans le cadre du projet de réalisation d'un prototype de système d'imagerie superspectral aéroporté.

Mots clefs : UAVs, Traitement d'image, Temps-réel, Capteurs inertiels, FPGA, Photogrammetrie

Abstract

In the recent years, the civilian UAVs (Unmanned Aerial Vehicles) have become an interesting tool in aerial photography and in photogrammetry. This led the LOEMI (Laboratoire d'Opto-électronique, de Métrologie et d'Instrumentation) team of IGN (Institut National de l'Information Géographique) to design a light-weight digital camera better adapted for exploiting photogrammetry and metrology applications than consumer cameras. This camera consists essentially of a CMOS "global shutter" sensor, an inertial measurement unit IMU, and a system on chip (FPGA + 2 CPUs) used originally to acquire image data from the sensor. This SoC/FPGA-based camera opens the door to implement in hardware some real-time image processing algorithms.

Night-time surveys and narrow spectral bandwith imagery are one of the next applications targeted by IGN, this type of applications needs a long-exposure time imagery that usually leads to a motion blur due to erratic movements of the UAV.

This thesis consists in the implementation on the light-weight IGN camera of an alogirithm which makes it possible to remedy this problem of motion blur.

The first part of this work was devoted to the development of the method which consists in acquiring several images with a short exposure time and then using an image processing algorithm in order to generate a stacked image with the equivalent of a long-exposure time. To obtain the correct parameters for the resampling of images, the presented method accurately estimates the geometrical relation between the first and the nth image, taking into account the internal parameters and the distortion of the camera. Features are detected in the first image by the FAST detector, then homologous points on other images are obtained by template matching aided by the IMU sensors.

In order to speed up the processing of our algorithm, some phases are accelerated by implementing them in the hardware (SoC / FPGA).

The results obtained on real surveys show that the final stacked image does not present a motion blur. The time results of the different phases of the algorithm also show that it is usable in real time during shooting for most of our applications.

In the future, the parameters of the algorithm will be refined in order to optimize the results and the execution time of the process. It would also be advantageous to exploit the presence of the GNSS receiver and the accelerometers in order to improve the phase of prediction of homologous points positions. By extension, this work is intended to be used on an airborne prototype imaging system with different narrow spectral channels.

Keywords : UAVs, Image processing, Real-time, Inertial sensors, FPGA, Photogrammetry

Publications

Ce document s'inspire des publications scientifiques suivantes :

- Audi, A., Pierrot-Deseilligny, M., Meynard, C., and Thom, C. (2017). Implementation of an IMU Aided Image Stacking Algorithm in a Digital Camera for Unmanned Aerial Vehicles. Sensors, 17(7), 1646.
- IMPLEMENTATION OF A REAL-TIME STACKING ALGORITHM IN A PHOTOGRAMMETRIC DIGITAL CAMERA FOR UAVS. A. Audi, M. Pierrot-Deseilligny, C. Meynard, and C. Thom, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLII-2-W6, 13-19.
- Audi, A.; Pierrot Deseilligny, M.; Meynard, C.; Thom, C. Implementation of real-time functionalities in the hardware of an intelligent ultra-light camera specialized for aerial photography. In GeoUAV-ISPRS Geospatial Week; ISPRS: La Grande Motte, France, 2015.

Table des matières

Chapitre 1

Introduction

Sommaire

1.1 Introduction générale

Historiquement, les drones ou UAVs (Unmanned Aerial vehicles) ont été développés et utilisés dans des applications militaires. De nos jours, ce type de robots connaît un succès grandissant dans le domaine civil, spécialement les vecteurs de type quadricoptères. Après des années de recherche et de développement, les drones ont eu la capacité de réaliser des tâches complexes d'une manière autonome, en bénéficiant des différents capteurs qui sont embarqués dans le système et qui peuvent fournir sans aucune intervention humaine des informations pertinentes supplémentaires et nécessaires pour un accomplissement parfait des missions. Cependant, le résultat des différentes applications dépendra fortement de la fiabilité et de la précision des différents capteurs utilisés. On peut distinguer parmi les capteurs les plus nécessaires pour la navigation et pour l'exploitation de l'environnement : les capteurs inertiels tels que les gyromètres, les accéléromètres et les magnétomètres, et les capteurs de vision tels que les caméras. D'une part, le drone a appris à se localiser d'une façon autonome en utilisant des capteurs inertiels qui sont en général d'une qualité limitée du fait des contraintes de poids et de prix et d'autre part, il a appris à voir par l'intermédiaire des caméras intégrées au système. Le défi peut être défini ici par la contribution collective des données acquises par les différents capteurs de qualité limitée afin de réaliser des tâches plus complexes et par la fusion de données afin de compenser les erreurs de mesures pour améliorer la qualité d'une telle tâche. Les caméras peuvent servir par exemple à faire des mesures pour aider judicieusement d'autres capteurs utilisés dans le contrôle du drone, et d'un autre côté les capteurs inertiels par exemple peuvent contribuer à l'optimisation des tâches de traitement d'image comme on le verra dans cette thèse.
Cette fusion de données a été rendue possible grâce au progrès croissant de la microélectronique numérique, surtout les architectures à bases des circuits intégrés tels que les FPGAs (Field Programmable Gate Arrays) qui donne une grande opportunité de paralléliser des opérations de calcul prédéfinies et donc permet d'implémenter en temps-réel des algorithmes très complexes, en particulier ceux de traitement d'image qui requièrent une grande quantité de données et qui nécessitent une grande puissance

de calcul.
Les caméras ont ouvert de nouvelles perspectives d'utilisation des drones dans plusieurs secteurs de l'industrie et de la recherche tels que la photographie professionnelle et la cartographie, la surveillance en temps-réel, la photogrammétrie, la recherche et le sauvetage, etc. Toutefois, l'un des défauts majeurs des drones est leur incapacité à voler avec une stabilité suffisante, ce qui ne convient pas à certains types de missions. La photographie est une des applications qui est fortement affectée par la stabilité des drones, en particulier les missions photographiques qui nécessitent un temps de pose long. L'absence de tout système de stabilisation peut causer un phénomène indésirable, le "flou de bougé", ce qui peut rendre l'image non exploitable par la suite. La question qui peut être posée ici est : comment peut-on remédier à ce problème si on a besoin par la suite de conserver la qualité géométrique des images ? C'est essentiellement sur ce thème de recherche que se situe la contribution de cette thèse.

1.2 Plan du document

Ce manuscrit organisé principalement en sept chapitres, présente la description et l'implémentation d'un algorithme de "stacking" temps-réel destiné à compenser le flou de bougé. Le **chapitre 2** décrit le contexte scientifique dans lequel ce travail est fait. Le **chapitre 3** est consacré à la présentation des différentes thématiques existantes dans l'état de l'art qui sont attachées directement au sujet. Le **chapitre 4** présente la plate-forme matérielle dans laquelle l'algorithme est implémenté. Dans le **chapitre 5**, notre algorithme proposé est détaillé en profondeur, après la description de l'architecture de l'algorithme, les différents blocs sont détaillés, et des résultats intermédiaires seront présentés afin de qualifier la précision de la méthode proposée. Le **chapitre 6** s'attache à l'implémentation dans le FPGA des parties coûteuses en temps de calcul de l'algorithme, du fait que le but final de cette thèse est d'obtenir une performance temps-réel. Le **chapitre 7** présente des résultats obtenus à partir de vols test dans des conditions réelles. Le **chapitre 8** termine ce manuscrit par une conclusion et des perspectives pour le travail présenté.

Chapitre 2

Contexte et objectifs

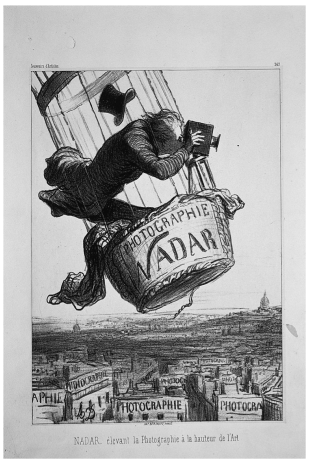
Sommaire

2.1 Contexte global

2.1.1 Photographie aérienne

La photographie aérienne est une technique qui consiste à prendre des images depuis un véhicule aérien verticalement ou d'une manière oblique afin d'enregistrer la surface visible de la terre vue d'en haut. Elle constitue depuis très longtemps une source d'informations extrêmement riches tant dans le domaine militaire que dans le domaine civil.
La première photographie aérienne a eu lieu en 1858 grâce au photographe et ballonniste français Gaspard-Félix Tournachon qui a pris à basse altitude (80m) une prise de vue aérienne oblique de Paris depuis un ballon captif (figure 2.1).
Cependant, la photographie aérienne n'a pris réellement son essor que pendant la première guerre mondiale où le développement des ballons d'observation et des aéronefs a permis d'utiliser les caméras pour répondre à des besoins de renseignement militaire au-dessus du territoire "ennemi" (figure 2.2).
De plus, la technique des pigeons photographes a été inventée par l'apothicaire allemand "Julius Neubronner" pendant cette période de guerre pour faire aussi de la reconnaissance aérienne photographique de façon plus simple (figure 2.3). La deuxième guerre mondiale a occasionné un fort développement de la photographie aérienne surtout avec la naissance des avions militaires spécialisés et des fusées type V2.
De nos jours, il existe une grande variété de vecteurs utilisés pour la prise de vues aériennes. On peut citer les avions, les hélicoptères, les ballons captifs et dirigeables, les ULMs, les cerf-volants et les drones. En pratique, le choix d'utilisation d'un de ces vecteurs pour la prise de vue aérienne dépend des performances et des caractéristiques techniques du vecteur ainsi que du sujet à photographier. Les moyens légers tels que les

(a) La première photo aérienne prise de Paris en 1858.



(b) Lithographie d'Honoré Daumier parue dans Le Boulevard, le 25 mai 1863

FIGURE 2.1 – Première prise de vue aérienne effectuée par Félix Tournachon.

(a) Un ballon d'observation mis en service pendant la première guerre mondiale. Source : www.carnetdevol.org.

(b) Un photographe aérien britannique pendant la première guerre mondiale. Source : www.theatrum-belli.com.

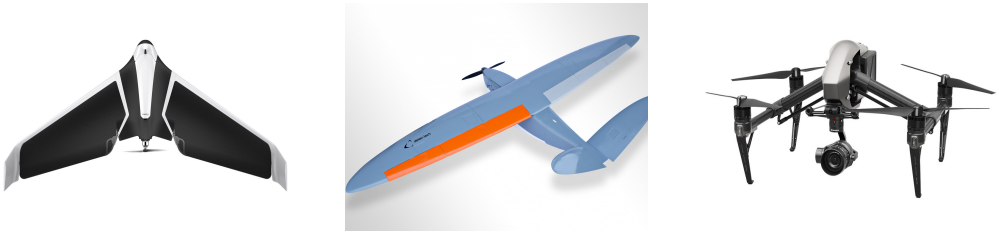
FIGURE 2.2 – Photographie aérienne à l'aide des ballons et des aéronefs.

FIGURE 2.3 – Le pigeon photographe. Source : www.buclermont.hypotheses.org.

drones restent aujourd'hui les moins coûteux et les mieux adaptés pour certains types de vols (vols à basse altitude, sous couvert nuageux,...).

Les drones

Historiquement destinés à faire de la reconnaissance et de la surveillance aérienne par les militaires, les drones utilisés dans le secteur civil sont en plein croissance. Au cours des dernières années, ils sont devenus un outil intéressant en photographie aérienne professionnelle et en travaux photogrammétriques de haute qualité. Aujourd'hui, il existe dans le marché mondial un grand nombre d'entreprises qui conçoivent des drones civils de différentes tailles, pour usage grand public ou professionnel, telles Parrot (figure 2.4a), Delair-Tech (figure 2.4b), DJI (figure 2.4c), Amazon et 3DR...



(a) Drone Parrot. (b) Drone Delair-Tech. (c) Drone DJI.

FIGURE 2.4 – Trois acteurs qui dominent le marché des drones civils.

Les médias ont été les premiers utilisateurs de drones civils avec caméras embarquées (figure 2.5), mais la situation a changé fortement depuis quelques années avec la multiplication de nouvelles applications.

(a) Une manifestation au centre-ville Washington. Source :www.Airphotoslive.com.

(b) Un match de football. Source : www.youtube.com.

FIGURE 2.5 – Deux exemples de prises de vue par drone dans le domaine des médias.

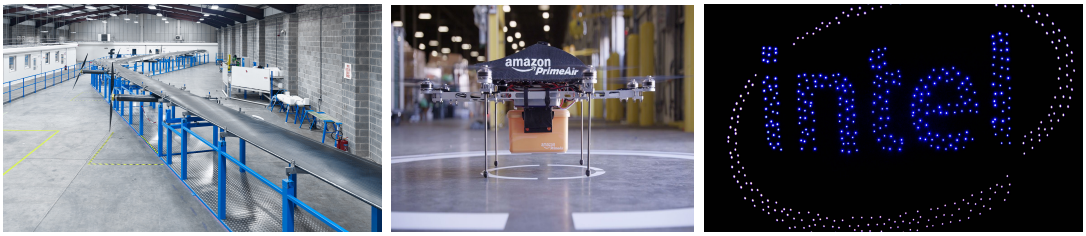
Les utilisations de drones professionnels équipés de capteurs, en particulier les caméras sont très multiples : on peut citer la cartographie pour les agriculteurs (figure 2.6a) et les ouvrages d'arts, l'inspection d'infrastructures et des bâtiments (figure 2.6b), la sécurité civile et la photogrammétrie de haute précision (figure 2.6c)...
Certaines entreprises sont allées plus loin, en concevant des drones pour des usages très spécifiques telles que Facebook (connexion internet : figure 2.7a), Amazon (livraison : figure 2.7b), Intel (spectacles de drones lumineux : figure 2.7c)...

Bien que l'on puisse définir un drone comme étant tout système aérien piloté à distance, on peut regrouper les drones en deux catégories, chacune ayant son usage :

- Les drones à voilure fixe : ces drones sont utiles pour couvrir de longues distances ou atteindre de hautes altitudes, ils sont plus dédiés pour la recherche et le sauvetage, et les applications cartographiques sur de grandes surfaces.

(a) Cartographie des mauvaises herbes. Source : CSIC.
(b) Inspection des pipelines. Source : www.perthdrone.com.au.
(c) Modèle 3D du château de Chambord. Source : ENSG.

FIGURE 2.6 – Usages professionnels de drones.

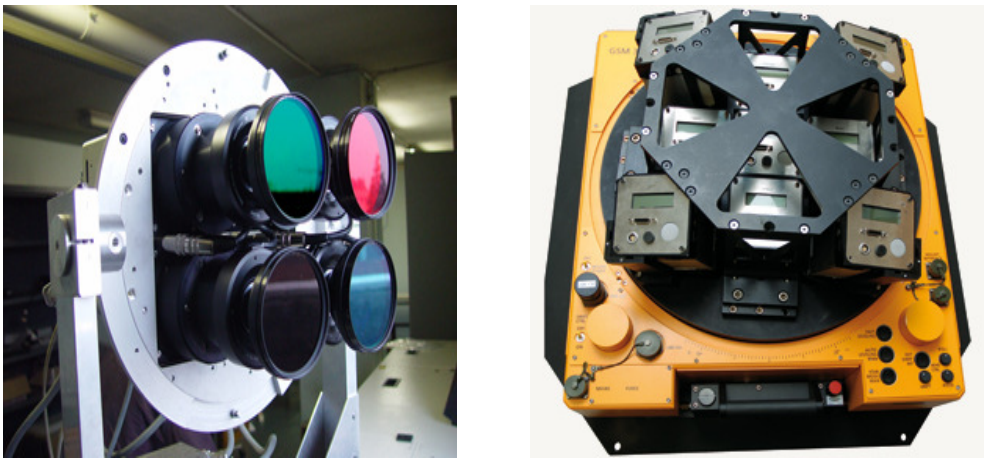


(a) Drone de Facebook.
(b) Drone d'Amazon.
(c) Drones d'Intel.

FIGURE 2.7 – Usages très spécifiques de drones.

- Les drones à voilure tournante : on peut distinguer les hélicoptères traditionnels et les multirotors à pas fixe. Ils sont généralement plus faciles à manipuler et en particulier en vol stationnaire, comme par exemple pendant l'inspection des gros ouvrages d'art.

2.2 Contexte IGN



(a)
(b)

FIGURE 2.8 – Caméras aériennes développées par le LOEMI.

Depuis des décennies, l'IGN (Institut National de l'Information Géographique et Forestière) a réalisé des travaux de photogrammétrie aérienne pour photographier et

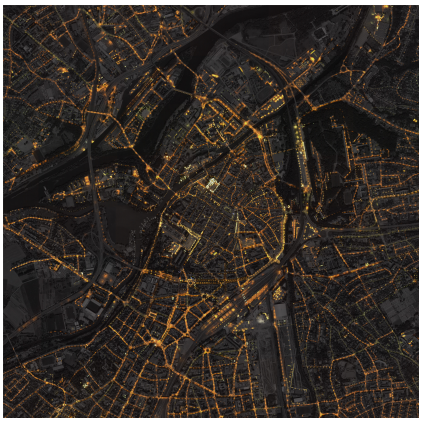
cartographier le territoire national français. Et depuis deux décennies, ces prises de vue sont faites par des caméras qui ont étés développées par le LOEMI (laboratoire d'Optique, Électronique, Métrologie et Instrumentation) (figure 2.8).
Avec l'explosion du marché des drones civils ces dernières années, le LOEMI s'est lancé en 2012 dans la conception d'une nouvelle caméra ultra-légère de très haute résolution destinée principalement à la prise de vue sur drone et dédiée en même temps aux travaux photogrammétriques et métrologiques de haute qualité. Le LOEMI a pris en main le développement de ce nouvel imageur et il a abouti à un premier prototype fonctionnel nommé initialement "Camlight" (figure 2.9).

FIGURE 2.9 – La caméra légère de l'IGN.

2.3 Présentation du sujet de thèse

L'initiation de cette thèse a eu lieu lorsque le développement de l'électronique et des logiciels internes de la caméra a suffisamment avancé pour permettre d'une part d'acquérir et de sauvegarder des images et d'autre part d'exploiter les capteurs inertiels intégrés dans la caméra. Pour ouvrir la voie à des applications innovantes, la caméra a été dotée d'une intelligence embarquée permettant d'y implémenter des algorithmes temps-réel de traitement d'image.

(a) Prise de vue en cas de crue (IGN).



(b) Prise de vue de nuit sur la ville Metz (IGN).

FIGURE 2.10 – Deux exemples de prises de vue aériennes qui nécessitent un temps de pose long.

Parmi les applications visées, on peut distinguer les prises de vue aériennes en bande spectrale étroite et les prises de vue aériennes de nuit (figure 2.10b). Ces prises de vue nécessitent un temps de pose long, ce qui peut provoquer un flou de bougé dans l'image dû aux mouvements du vecteur qui peut la rendre inexploitable. Le même problème peut se trouver à moindre échelle dans des applications plus classiques telles que les prises de vue sous couvert nuageux, par exemple pour la surveillance des crues (figure 2.10a).

La résolution de ce problème fonde le sujet de cette thèse.

Dans le prochain chapitre, nous allons montrer que les techniques existantes ne sont pas adaptées à notre cas. Nous avons donc été amenés à développer une nouvelle méthode de compensation de bougé dont l'originalité consiste à acquérir en rafale plusieurs images à temps d'exposition assez court, puis superposer et empiler ces images en effectuant un recalage photogrammétrique en s'aidant de données inertielles. L'image finale sera l'équivalent d'une image à temps de pose long. Un des défis est d'obtenir le résultat en un temps compatible à la fois avec les contraintes techniques de la caméra et les impératifs de la prise de vue aérienne. En effet, compte tenue du débit du système de sauvegarde interne de la caméra, il est impossible de sauvegarder la séquence complète d'images entre deux sommets de prises de vue successifs, il est donc nécessaire de réaliser tout le processus en temps-réel et de ne sauvegarder que le résultat. Le critère du temps-réel ici est lié fortement aux contraintes imposées par la mission ou l'application visée, comme par exemple : le type de drone, la vitesse, le type de la mission, le terrain...

Chapitre 3

État de l'art

Sommaire

3.1 Introduction

La thématique de notre sujet est liée à plusieurs disciplines. L'idée globale se concentre sur l'obtention d'une image bien exposée dans des conditions de faible luminosité à partir d'une caméra embarquée sur un drone par l'utilisation d'une exposition suffisamment longue.
Deux grandes méthodes sont utilisées pour régler ce problème, une permet d'empêcher le flou de bougé dans l'image, on parlera donc ici de différentes techniques de stabilisation qui sont déjà développées. Une autre consiste à éliminer le bougé en utilisant des méthodes mathématiques de dé-convolution, on parlera aussi de ces méthodes.

Notre méthode utilisant le recalage géométrique de l'image, on parlera donc des techniques de recalages existantes, particulièrement en vision par ordinateur.

L'intégration de données des capteurs inertiels dans la chaîne du traitement faisant partie des points clés de notre travail, on décrira les techniques de fusion des données inertielles et visuelles qui font l'objet de recherches dans différents domaines.

3.2 Stabilisation des images

Il existe depuis longtemps des systèmes qui stabilisent l'image pour les caméras et les appareils photographiques : stabilisation mécanique extérieure de la caméra entière, stabilisation optique dans l'objectif. De nos jours, de nombreuses caméras numériques commerciales les ont repris avec en plus la stabilisation mécanique à l'intérieur de la caméra au niveau du capteur, et la stabilisation électronique de l'image.
Dans le domaine des drones, la stabilisation mécanique extérieure se présente souvent sous la forme d'une nacelle gyro-stabilisée, sinon la stabilisation de l'APN lorsqu'il en dispose est utilisée.

3.2.1 Stabilisation mécanique extérieure

Ce type de stabilisation consiste à utiliser essentiellement un système composé d'une nacelle 3 axes actionnée par des moteurs standards (type servomoteurs ou brushless) et un système de capteurs trois axes gyroscope-accéléromètre. Les capteurs inertiels permettent de détecter les mouvements de la caméra et donc de donner les changements de position de la nacelle en passant par une carte de contrôle. Cette carte reçoit les mesures des capteurs, et transmet après traitement les ordres aux servomoteurs qui font bouger à leur tour la nacelle afin de stabiliser la caméra. De cette façon, les mouvements de la caméra seront compensés.

(a) Nacelle DYS 3-axes. Source : www.altigator.com

(b) Drone type 3DR solo avec une nacelle et une caméra Héro4. Source : www.macway.com

FIGURE 3.1 – Exemples de nacelles gyro-stabilisées sur 3 axes très utilisées sur les drones type quadricopter.

Il en existe sur le marché un grand nombre de modèles de qualité très variable. Cette qualité dépend d'une part du type de moteurs utilisés et d'autre part du type des capteurs inertiels utilisés. Par conséquent, si on veut une stabilisation ultra-fluide de la caméra, on a besoin d'acheter un modèle coûteux et lourd, ce qui n'est pas souhaitable pour nous et pour beaucoup d'autres particuliers et d'entreprises.

3.2.2 Stabilisation mécanique intérieure

La stabilisation mécanique effectuée dans le boîtier de la caméra agit directement sur le capteur. Pendant l'exposition, le capteur lui-même se déplace de manière à compenser les mouvements de la caméra. Ce type de stabilisation nécessite la présence d'un système complet de stabilisation composé essentiellement des capteurs gyroscopiques, des moteurs (généralement de type piézoélectrique) et d'un microprocesseur. Les capteurs permettent au micro-processeur de calculer le mouvement de correction qui sera appliqué par les moteurs au capteur pour que les rayons lumineux issus d'un même point frappent toujours la même position sur la surface du capteur.

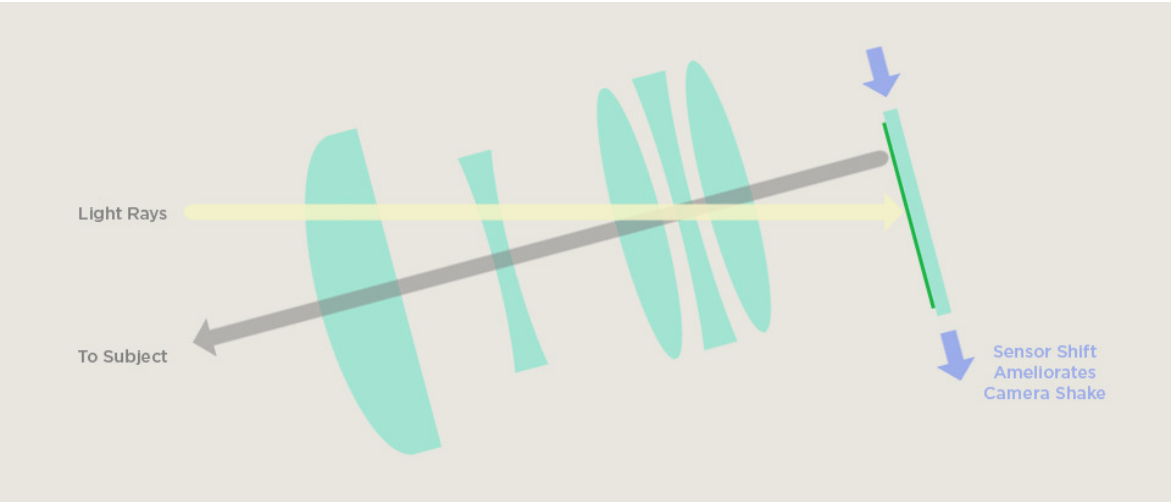


FIGURE 3.2 – Stabilisation mécanique dans le boîtier. Source : www.bhphotovideo.com

L'avantage de cette stabilisation mécanique est qu'elle peut être appliquée en présence de n'importe quelle optique. Ses défauts viennent de la complexité du système de stabilisation au niveau du capteur, ce qui rend le boîtier plus volumineux, plus lourd et aussi plus cher. Ce type de stabilisateur est destiné à ne compenser que des petits mouvements effectués par une caméra opérée par un être humain pendant l'exposition. Alors que dans le cas de drones, on ne sait pas à quel point cette stabilisation peut aboutir à des bons résultats.

3.2.3 Stabilisation optique

La stabilisation optique ressemble beaucoup à la stabilisation mécanique mais elle est implémentée au niveau de l'objectif. Elle manipule l'image avant qu'elle n'arrive au niveau du capteur en mettant en place dans l'objectif une lentille supplémentaire qui peut se déplacer afin de compenser le mouvement de la caméra. La lentille se déplace grâce à des moteurs liés à des capteurs gyroscopiques permettant au micro-processeur de calculer le mouvement de correction qui sera appliqué par les moteurs au capteur pour que les rayons lumineux issus d'un même point frappent toujours la même position sur la surface du capteur.
Ce stabilisateur est le plus utilisé aujourd'hui dans les caméras commerciales bien qu'il doit être intégré dans chaque objectif. Il implique l'augmentation du poids, du volume et du prix d'objectifs du fait qu'il utilise une (ou plusieurs) lentille(s) avec un système mécanique pour déplacer celle(s)-ci. L'autonomie de la caméra est aussi réduite car

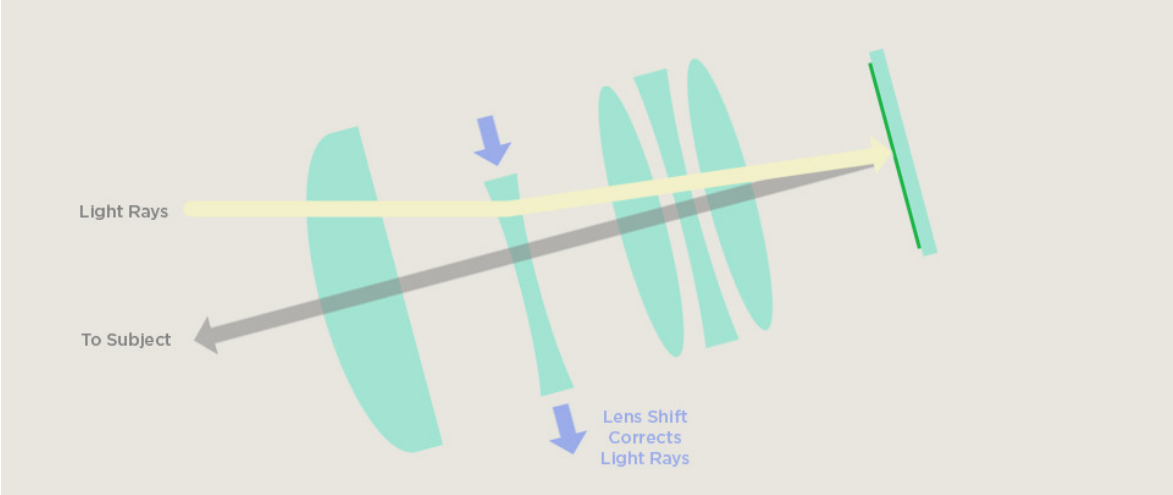


FIGURE 3.3 – Stabilisation optique dans l'objectif. Source : www.bhphotovideo.com

le stabilisateur a besoin d'énergie pour fonctionner. Les différents fabricants de caméras ont montré par des tests que ce stabilisateur est plus efficace que le stabilisateur mécanique, parce qu'il est optimisé pour chaque objectif. Cependant, avec ce type de stabilisation, l'état de l'optique change pendant l'acquisition du point de vue photogrammétrique, ceci change la géométrie de façon inconnue.

3.2.4 Stabilisation électronique

La stabilisation électronique, également connue sous le nom de stabilisation numérique fonctionne sur un principe complètement différent. Dans cette technique, le problème est résolu par des logiciels internes à la caméra sur les images déjà acquises, cela transforme le problème de stabilisation de l'image en un problème algorithmique d'analyse des images successives. Il est important ici de souligner la distinction entre la stabilisation vidéo numérique et la stabilisation d'image numérique :

- Stabilisation vidéo : elle fait référence au processus d'élimination des effets du mouvement indésirable de caméra à partir de données vidéo. Dans la littérature, on trouve un grand nombre d'algorithmes qui ont été développés pour faire le traitement a posteriori (Morimoto and Chellappa [1998], Chang et al. [2002], Juanjuan and Baolong [2008]) ou en temps-réel (Morimoto and Chellappa [1996], Duric and Rosenfeld [1996], Chang et al. [2006] Ertürk [2002], Wang et al. [2011]). Généralement, le problème de la stabilisation vidéo est résolu en se basant sur l'information de mouvement de la caméra. En fait, beaucoup d'algorithmes prédisent le mouvement de la caméra directement à partir des images, alors que d'autres utilisent des gyroscopes et des accéléromètres synchronisés avec le capteur d'image afin d'améliorer la précision. Une fois que le mouvement est estimé, des techniques de recalage sont utilisées pour recadrer en conséquence la vidéo stable. Dans notre contexte, on ne s'intéresse pas à la stabilisation vidéo.

- Stabilisation image : elle concerne la correction des effets du mouvement indésirable qui se déroule pendant le temps d'exposition de l'image. Certains algorithmes ne cherchent qu'à détecter après une étape de traitement une meilleure image parmi plusieurs images successives acquises (Han et al. [2011]). D'autres algorithmes

utilisent une technique de fusion de plusieurs images, comme par exemple (Lim and Silverstein [2008]) et (Choi et al. [2008]) : ils estiment et éliminent le flou de bougé d'une image acquise avec un temps d'exposition long en utilisant des informations issues d'une deuxième (ou de plusieurs) image(s) de la même scène acquise(s) avec un temps d'exposition court.

L'avantage de ce type de stabilisation est qu'elle n'a besoin d'aucun matériel supplémentaire, cela veut dire qu'on n'a pas besoin ni de poids ni de volume supplémentaire. Au niveau du coût, une fois que les algorithmes sont implémentés, le coût devient nul. Par contre au niveau efficacité, la stabilisation électronique est moins bonne que les autres techniques standard parce que la qualité de l'image finale stabilisée est basée sur des images déjà capturées et donc qui peuvent être déjà dégradées.

3.2.5 Conclusion

A la vue de ces différentes techniques, on remarque que certains systèmes de stabilisation d'image ne sont efficaces pour notre application et qu'aucun parmi eux n'est optimal pour tout type d'application. En effet, le choix de la méthode de stabilisation dépend fortement de plusieurs facteurs, principalement le budget disponible, la caméra utilisée et le type d'application visée. Il est clair alors que pour les temps de pose longs, un trépied bien robuste est plus performant que les systèmes de stabilisation intérieurs à la caméra néanmoins qu'il est plus cher, plus lourd et plus difficile à manipuler.

3.3 Suppression du flou de bougé dans les images

L'élimination des dégradations dans les images est aussi une partie importante de la recherche dans le domaine du traitement d'image (Katsaggelos [2012], Schuler et al. [2016]). Il existe des méthodes de restauration qui dépendent du type de dégradation observée : dégradations dues aux conditions de prise de vue (essentiellement le flou de bougé) (Yitzhaky et al. [1998], Nayar and Ben-Ezra [2004], Shan et al. [2008]), les dégradations dues aux défauts du capteur (essentiellement le flou causé par l'optique et le bruit électronique) (Sondhi [1972])...
Dans le cas d'un bougé de la caméra, la dégradation est modélisée par un opérateur de convolution où la fonction qui représente le flou est usuellement appelée fonction d'étalement d'un point ou PSF (Point Spread Function). L'opération de défloutage nommée "déconvolution non-aveugle" suppose que la PSF est connue (Samarasinghe et al. [2009]). Cependant, une autre approche appelée "déconvolution en aveugle" réalise conjointement l'estimation de PSF et la restauration de l'image, elle est nettement plus difficile à résoudre (Kundur and Hatzinakos [1996], Krahmer et al. [2006], Oliveira et al. [2007]).
L'hypothèse de l'invariance spatiale de flou dans l'image est l'inconvénient majeur des méthodes de déconvolution. En réalité, on ne peut pas considérer le flou comme constant en tout point dans l'image surtout dans de nombreuses situations comme par exemple l'image contenant des structures qui se déplacent selon des mouvements complexes. En plus, ces méthodes sont constituées d'un grand nombre d'étapes de traitement d'image intermédiaires complexes. Par conséquent, l'implémentation devient très coûteuse en ressources et en temps de calcul. Enfin, la plupart de ces méthodes sont basées sur un grand nombre d'hypothèses, de contraintes et d'améliorations mineures complexes, ce qui rend leur utilisation très difficile en pratique.

3.4 Méthodes de recalage des images

Parmi les problématiques principales les plus rencontrées dans le domaine du traitement d'image se trouvent "la mise en correspondance entre deux images", ceci afin essentiellement de pouvoir superposer d'une façon optimale les deux images. Cette mise en correspondance repose sur la recherche d'une transformation spatiale qui permet de passer d'une image de référence (image sur laquelle les autres images seront mises en correspondance) à une autre secondaire (image déformée géométriquement qui sera mise en correspondance sur l'image de référence) au sein d'un référentiel commun. Cette technique appelée aussi "recalage des images" est très utilisée en analyse d'images médicales (fusion d'images en multimodalité par exemple) et en traitement vidéo (suivi des objets par exemple), ainsi qu'en astronomie. Beaucoup de travaux de recherche introduisant la réalisation ou encore l'utilisation de cette technique ont été déjà publiés (Le Moigne et al. [2010], Lakshmi and Vaithiyanathan [2015], Patel and Shah [2014]).
La plupart des publications sont spécifiques au traitement d'images médicales (Maintz and Viergever [1996], Goshtasby [2005]), alors que d'autres couvrent aussi la vision par ordinateur (Brown [1992]) et(Zhang [1994]). Plusieurs éléments caractérisent une méthode de recalage :

- Les attributs : ce sont des caractéristiques extraites d'une image qui sont utilisées pour trouver la transformation entre les deux images, ils peuvent être appelés également primitives.
- Le type de transformation : c'est le modèle de déformation avec lequel l'image est géométriquement modifiée. Le choix du modèle de transformation dépend essentiellement de la nature de correspondance géométrique que l'on souhaite établir entre les deux images. Chaque type de transformation est caractérisé par un certain nombre de paramètres. On peut distinguer les transformations linéaires comme par exemple les transformations rigides (translation, rotation...), les transformations affines, les transformations projectives (homographie) et les transformations non-linéaires comme par exemple les thin-plate splines ou les combinaisons de B-Splines.
- Le critère de similarité : c'est le critère de mesure soigneusement choisi pour quantifier le degré de ressemblance entre les deux images en définissant une certaine distance à l'aide des attributs identifiés dans les deux images. On peut distinguer les critères de similarité géométriques (distances euclidiennes entre des coins identifiés par exemple, distance entre deux courbes, distance entre deux surfaces...) et les critères de similarité d'intensité denses utilisés pour trouver des liens entre les intensités de deux images.
- L'algorithme d'optimisation : c'est la méthode choisie qui sert à trouver la meilleure transformation au sens du critère de similarité prédéfini pour garantir une mise en correspondance optimale entre les deux images. On peut distinguer les méthodes directes, les méthodes exhaustives, les méthodes itératives, et les méthodes stochastiques.

3.4.1 Approches utilisant les attributs

Comme on l'a déjà dit ci-dessus, les attributs représentent les informations pertinentes extraites sur lesquelles s'appuie l'estimation de la transformation entre les deux images

à recaler. Le type des attributs doit être choisi avec soin parce qu'il dépend fortement de la nature des images. Globalement, on peut distinguer deux approches classifiées par attributs : les approches géométriques basées sur l'utilisation de quelques caractéristiques géométriques extraites de l'image (Lin et al. [2010], Gillan and Agathoklis [2010]) et les approches iconiques basées sur l'utilisation de l'intensité (niveaux de gris) d'un grand nombre de pixels de l'image (Rohde et al. [2003], Kim and Fessler [2004]). Par ailleurs, il existe aussi les approches hybrides qui combinent les deux approches citées ci-dessus (Sotiras et al. [2010]).

Méthodes géométriques

Les méthodes géométriques (Maurer et al. [1993]) reposent sur deux étapes complètement indépendantes. Dans un premier temps, elles consistent à identifier des caractéristiques géométriques distinctes dans les deux images : ces caractéristiques peuvent être intrinsèques à l'image (les points, les contours, les courbes, les surfaces...) ou extrinsèques à l'image (marqueurs sur les objets). Les caractéristiques intrinsèques peuvent être extraites soit automatiquement grâce à des algorithmes de détection en s'appuyant sur les propriétés locales des intensités des pixels, soit manuellement. On peut citer ici quelques types d'attributs très fréquemment utilisés d'une façon automatique pour faire du recalage : des points d'intérêt, des coins, des surfaces, des volumes ; cette étape est appelée segmentation. D'autre part, le choix d'attributs doit être fait en fonction de quelques propriétés : une extraction précise et préférablement rapide des caractéristiques, une répartition uniforme des caractéristiques sur toute l'image afin d'estimer au mieux la transformation, une robustesse acceptable aux différents changements liés à l'acquisition comme le bruit, la luminosité et le type de transformation (rotation...) etc. Une fois que les caractéristiques sont identifiées dans les deux images, la transformation spatiale peut être calculée à l'optimum à l'aide d'une stratégie d'optimisation généralement choisie au sens des moindres carrés en fonction du critère de similarité prédéfini.

Méthodes iconiques

Au contraire des méthodes géométriques, les méthodes iconiques (den Elsen et al. [1995]) ne nécessitent aucune étape de segmentation des images. Elles consistent à trouver une mesure de similarité en se basant uniquement sur l'utilisation dense des niveaux de gris des pixels de deux images (ou une transformation des niveaux de gris). Il existe à ce jour plusieurs mesures de similarité, parmi elles on peut citer la somme des carrés des différences SSD (sum of square differences), le coefficient de corrélation linéaire, la corrélation croisée ainsi que les critères statistiques comme l'information mutuelle (IM).

3.4.2 Conclusion

Le principal avantage des méthodes géométriques est la réduction du nombre de données utilisées comme attributs (due à l'étape de segmentation), ce qui réduit énormément le temps de calcul de l'algorithme de recalage. Cependant, la qualité de l'étape de segmentation (extraction et sélection des caractéristiques géométriques) est cruciale pour la robustesse de la transformation estimée, ainsi que par la suite pour la qualité du recalage obtenu. D'autre part, le fait que les méthodes iconiques utilisent toute l'information portée par l'image et non pas seulement quelques points ou structures d'intérêts

(avec l'appariement de ces points qui peut être non précis) améliore la robustesse et la précision de l'estimation de transformation. Par contre, ces méthodes sont sensibles au bruit et aux artefacts dans l'image et elles demandent beaucoup plus de ressources (coût algorithmique de calcul + mémoire) que les méthodes géométriques.
Le choix d'une de ces méthodes dépend fortement de l'application visée et de la plate-forme avec laquelle l'algorithme de recalage sera exécuté. Dans la littérature (Al-Mamun and Ibrahim [2007], Mehrdad et al. [2016], Zhi et al. [2016]), on trouve que la plupart des algorithmes qui nécessitent une performance temps-réel (stabilisation électronique temps-réel d'image, construction temps-réel d'une image panoramique, localisation temps-réel...) utilisent préférentiellement les méthodes géométriques.

3.5 Fusion des données IMU/Images

S'inspirant de la nature, l'homme a constaté qu'un robot doit avoir plusieurs capteurs pour obtenir des informations de son environnement afin d'accomplir des tâches plus complexes. C'est à cause de cela que les robots d'aujourd'hui sont équipés de différents capteurs comme les capteurs inertiels, les lidars, la vision, etc. Bien que la combinaison d'informations provenant de différents sens est un processus tout à fait naturel et ne nécessite pas un grand effort pour les êtres vivants, en reproduire la même capacité dans les machines fabriquées par l'homme est une tâche extrêmement difficile. Relever ce défi devient de plus en plus réalisable en raison de l'émergence de nouveaux capteurs et des techniques de traitement avancées. L'utilisation conjointe des sources d'informations multiples a donné naissance à une nouvelle discipline appelée "fusion de données de capteurs".
Les capteurs inertiels sont depuis longtemps parmi les capteurs essentiels utilisés pour la navigation (localisation et orientation) des systèmes. Avec la progression technologique en électronique et en informatique, la capacité du traitement des images a évolué énormément ce qui a permis aux capteurs de vision d'apporter de nouvelles complémentarités à des solutions basées uniquement sur l'utilisation des capteurs inertiels. Par ailleurs, certains systèmes utilisent des capteurs inertiels en coopération et en complémentarité avec des caméras pour se renseigner sur leurs vitesses et leurs positions instantanées, ce qui aboutit aussi à des nouvelles perspectives. Le principal avantage de la fusion de différents capteurs est de s'affranchir des limites de chaque capteur en partant des observations d'autres capteurs associés. Cette combinaison des informations permet d'offrir des informations supplémentaires nécessaires pour certaines applications et en plus d'améliorer la précision des différents paramètres estimés et donc offre par la suite une bonne robustesse/stabilité globale du système. C'est pour cela que les techniques de fusion de données inertielles et visuelles ont intéressé beaucoup les chercheurs, ce qui donne lieu chaque année à un grand nombre de travaux académiques, technologiques et industriels dans les domaines qui utilisent des systèmes multi-capteurs.
Les applications majeures qui profitent de la fusion de données inertielles et visuelles sont : la robotique (terrestre, aérienne, marine...), la réalité augmentée (lunettes, smartphone...), l'aérospatiale (satellites, navettes...), etc. De nombreuses techniques de fusion ont été proposées dans la littérature. Nous en présentons ici quelques-unes :

- (Joshi et al. [2010]) et (Shah and Schickler [2012]) suggèrent d'utiliser des capteurs inertiels à bas prix pour mesurer l'accélération et la vitesse angulaire de la caméra pendant l'exposition. Le but est de modéliser la fonction de flou (PSF) afin de

réduire ou supprimer le flou dans l'image à l'aide de la méthode de déconvolution. Leurs méthodes ne fonctionnent que pour de petits mouvements et sont limitées par la précision de ce type de capteur à bas prix et leur bruit. Les résultats obtenus par les travaux de (Shah and Schickler [2012]) montrent que les images aériennes avec 4 à 7 pixels de flou de bougé ont des améliorations mineures.

- Beaucoup de travaux utilisent la combinaison des capteurs inertiels avec la vision pour des applications de localisation. (Tsai et al. [2014]) ont implémenté un algorithme de suivi de trajectoire (pour des appareils mobiles) en améliorant la localisation en ayant recours à la fusion de données inertielles et de données visuelles (images). Certaines parties de leur approche sont semblables à celles de la nôtre. Ils utilisent les mesures des capteurs inertiels pour estimer les angles de rotation afin d'améliorer la précision de l'estimation de la trajectoire. En revanche, ils utilisent des cibles comme points de contrôle identifiés dans la partie de traitement d'image pour corriger la propagation des erreurs des capteurs inertiels. Par conséquent, leur méthode n'est pas optimale si aucune cible n'est disponible le long de la trajectoire.

- (You and Neumann [2001]), (Hol et al. [2007]), et (Caarls et al. [2003]) utilisent une combinaison de capteurs inertiels et d'une caméra pour le suivi des mouvements dans les applications de réalité augmentée. La plupart de ces travaux de fusion utilisent des capteurs inertiels pour faciliter les méthodes de vision par ordinateur, rendant le système plus précis et plus robuste.

3.6 Références

Al-Mamun, A. and Ibrahim, M. Y. (2007). Feature-based real-time object identification in industrial processes. In *2007 IEEE International Conference on Integration Technology*, pages 139–143. 24

Brown, L. G. (1992). A survey of image registration techniques. *ACM Computing Surveys*, 24 :325–376. 22

Caarls, J., Jonker, P., and Persa, S. (2003). *Sensor Fusion for Augmented Reality*, pages 160–176. Springer Berlin Heidelberg, Berlin, Heidelberg. 25

Chang, H.-C., Lai, S.-H., and Lu, K.-R. (2006). A robust real-time video stabilization algorithm. *Journal of Visual Communication and Image Representation*, 17(3) :659 – 673. Special Issue on Real-Time Imaging. 20

Chang, J.-Y., Hu, W.-F., Cheng, M.-H., and Chang, B.-S. (2002). Digital image translational and rotational motion stabilization using optical flow technique. *IEEE Transactions on Consumer Electronics*, 48(1) :108–115. 20

Choi, B. D., Jung, S. W., and Ko, S. J. (2008). Motion-blur-free camera system splitting exposure time. *IEEE Transactions on Consumer Electronics*, 54(3) :981–986. 21

den Elsen, P. A. V., Maintz, J. B. A., Pol, E. J. D., and Viergever, M. A. (1995). Automatic registration of ct and mr brain images using correlation of geometrical features. *IEEE Transactions on Medical Imaging*, 14(2) :384–396. 23

Duric, Z. and Rosenfeld, A. (1996). Image sequence stabilization in real time. *Real-Time Imaging*, 2(5) :271 – 284. 20

Ertürk, S. (2002). Real-time digital image stabilization using kalman filters. *Real-Time Imaging*, 8(4) :317 – 328. 20

Gillan, S. and Agathoklis, P. (2010). Image registration using feature points, zernike moments and an m-estimator. In *2010 53rd IEEE International Midwest Symposium on Circuits and Systems*, pages 434–437. 23

Goshtasby, A. A. (2005). *2DD and 3-D Image Registration : For Medical, Remote Sensing, and Industrial Applications*. Wiley-Interscience. 22

Han, L., Xue, F., Li, Z., and Chen, D. (2011). *A New Scheme for Preprocessing Image Sequence in Wearable Vision System*, pages 295–301. Springer Berlin Heidelberg, Berlin, Heidelberg. 20

Hol, J. D., Schön, T. B., Gustafsson, F., and Slycke, P. J. (2007). Sensor fusion for augmented reality. ***"9th International Conference on Information Fusion, Florence, 2006***. 25

Joshi, N., Kang, S. B., Zitnick, C. L., and Szeliski, R. (2010). Image deblurring using inertial measurement sensors. *ACM Trans. Graph.*, 29(4) :30 :1–30 :9. 24

Juanjuan, Z. and Baolong, G. (2008). Electronic image stabilization system based on global feature tracking. *Journal of Systems Engineering and Electronics*, 19(2) :228 – 233. 20

Katsaggelos, A. K. (2012). *Digital Image Restoration*. Springer Publishing Company, Incorporated. 21

Kim, J. and Fessler, J. A. (2004). Intensity-based image registration using robust correlation coefficients. *IEEE Transactions on Medical Imaging*, 23(11) :1430–1444. 23

Krahmer, F., Lin, Y., McAdoo, B., Ott, K., Wang, J., Widemann, D., and Wohlberg, B. (2006). Blind image deconvolution : Motion blur estimation. 21

Kundur, D. and Hatzinakos, D. (1996). Blind image deconvolution. *IEEE Signal Processing Magazine*, 13(3) :43–64. 21

Lakshmi, K. D. and Vaithiyanathan, V. (2015). Study of feature based image registration algorithms for navigation of unmanned aerial vehicles. *Indian Journal of Science and Technology*, 8(22). 22

Le Moigne, J., Netanyahu, N. S., and Eastman, R. D., editors (2010). *Image registration for remote sensing*. Cambridge University Press, Cambridge, New York. 22

Lim, S. H. and Silverstein, A. (2008). Estimation and removal of motion blur by capturing two images with different exposures. *HP Laboratories*. 21

Lin, H., Du, P., Zhao, W., Zhang, L., and Sun, H. (2010). Image registration based on corner detection and affine transformation. In *2010 3rd International Congress on Image and Signal Processing*, volume 5, pages 2184–2188. 23

Maintz, J. B. A. and Viergever, M. A. (1996). An overview of medical image registration methods. Technical report, In Symposium of the Belgian hospital physicists association (SBPH-BVZF. 22

Maurer, C. R., Jr., and Fitzpatrick, J. M. (1993). A review of medical image registration. In *Interactive imageguided neurosurgery*, pages 17–44. 23

Mehrdad, S., Satari, M., Safdary, M., and Moallem, P. (2016). Toward real time uavs' image mosaicking. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B1 :941–946. 24

Morimoto, C. and Chellappa, R. (1996). Fast electronic digital image stabilization. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 3, pages 284–288 vol.3. 20

Morimoto, C. and Chellappa, R. (1998). Evaluation of image stabilization algorithms. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 5, pages 2789–2792 vol.5. 20

Nayar, S. K. and Ben-Ezra, M. (2004). Motion-based motion deblurring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6) :689–698. 21

Oliveira, J. P., Figueiredo, M. A., and Bioucas-Dias, J. M. (2007). Blind estimation of motion blur parameters for image deconvolution. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 604–611. Springer. 21

Patel, P. and Shah, V. (2014). Image registration techniques : A comprehensive survey. *International Journal of Innovative Research and Development || ISSN 2278 – 0211*, 0(0). 22

Rohde, G. K., Aldroubi, A., and Dawant, B. M. (2003). The adaptive bases algorithm for intensity-based nonrigid image registration. *IEEE Transactions on Medical Imaging*, 22(11) :1470–1479. 23

Samarasinghe, P. D., Kennedy, R. A., and Li, H. (2009). On non-blind image restoration. In *2009 3rd International Conference on Signal Processing and Communication Systems*, pages 1–7. 21

Schuler, C. J., Hirsch, M., Harmeling, S., and Schölkopf, B. (2016). Learning to deblur. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7) :1439–1451. 21

Shah, C. A. and Schickler, W. (2012). Automated blur detection and removal in airborne imaging systems using imu data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIX-B1 :321–323. 24, 25

Shan, Q., Jia, J., and Agarwala, A. (2008). High-quality motion deblurring from a single image. *ACM Trans. Graph.*, 27(3) :73 :1–73 :10. 21

Sondhi, M. M. (1972). Image restoration : The removal of spatially invariant degradations. *Proceedings of the IEEE*, 60(7) :842–853. 21

Sotiras, A., Ou, Y., Glocker, B., Davatzikos, C., and Paragios, N. (2010). Simultaneous geometric - iconic registration. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2010, 13th International Conference, Beijing, China, September 20-24, 2010, Proceedings, Part II*, pages 676–683. 23

Tsai, F., Chang, H., and Su, A. Y. S. (2014). Combining mems-based imu data and vision-based trajectory estimation. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-4 :267–271. 25

Wang, Y., Hou, Z., Leman, K., and Chang, R. (2011). Real-time video stabilization for unmanned aerial vehicles. 20

Yitzhaky, Y., Mor, I., Lantzman, A., and Kopeika, N. S. (1998). Direct method for restoration of motion-blurred images. *J. Opt. Soc. Am. A*, 15(6) :1512–1519. 21

You, S. and Neumann, U. (2001). Fusion of vision and gyro tracking for robust augmented reality registration. In *Virtual Reality, 2001. Proceedings. IEEE*, pages 71–78. 25

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13 :119–152. 22

Zhi, X., Yan, J., Hang, Y., and Wang, S. (2016). Realization of cuda-based real-time registration and target localization for high-resolution video images. *Journal of Real-Time Image Processing*. 24

Chapitre 4

Plateforme matérielle

Sommaire

4.1 Présentation de la caméra photogrammétrique légère de l'IGN pour drones

Les appareils photographiques numériques ont rarement toutes les caractéristiques nécessaires pour une prise de vue photogrammétrique par drone : optique fixe, sauvegarde des images à pleine résolution en flux soutenu, capteurs d'orientation précise (gyromètres et accéléromètres) et de localisation précise (GPS)... Afin de remédier à cela, l'équipe LOEMI du laboratoire LaSTIG (Laboratoire des Sciences et Technologies de l'Information Géographique) de l'IGN s'est lancé depuis 2012 dans la conception d'une nouvelle caméra numérique ultra-légère pour drones, bien adaptée aux travaux photogrammétriques de très haute qualité, en s'appuyant sur des caractéristiques techniques pertinentes qui permettent de la distinguer des autres APN du commerce (Martin et al. [2014]) :

- Une faible masse, à peu près 160g sans optique, ce qui rend cette caméra adaptée aux différents modèles de drones.
- Un capteur plein format de 20 megapixels.
- Un obturateur global afin de capturer tous les pixels de la scène en même temps.
- Une cadence d'acquisition des images élevée de 30 images/seconde.

- Une capacité de sauvegarde de 5 images/seconde en pleine résolution (20 megapixels) et de 20 images/seconde en mode sous-échantillonné par 4 (5 megapixels).
- Une intelligence embarquée composée principalement d'un système sur une puce basée sur un FPGA et deux cœurs CPU faisant tourner un OS Linux. Ceci donne en plus de la gestion de l'acquisition la possibilité d'implémenter des algorithmes de traitement d'image.
- Des capteurs inertiels qui permettent de remonter à des informations sur la position et l'attitude de la caméra lors des prises de vue.
- Un récepteur GPS permettant une localisation précise de la caméra grâce à la possibilité de récupérer les données de phase.
- Une interface Wifi qui permet de contrôler la caméra à distance.
- Possibilité d'ajouter d'autres fonctionnalités grâce à une conception modulaire : batterie, connectique supplémentaire, réseau, usb...
- Possibilité de synchronisation avec d'autres dispositifs ou d'autres caméras.

FIGURE 4.1 – La caméra légère de l'IGN.

Toutes ces caractéristiques font de cette caméra un bon outil de développement et de recherche à plusieurs niveaux. Elle peut être utilisée comme plateforme d'acquisition d'images pour des travaux de photogrammétrie. Certains travaux de recherche de ce domaine sont déjà faits ou sont en cours de réalisation tels que (Tournadre et al. [2015]), (Daakir et al. [2015]) et (Daakir et al. [2016]).

4.1.1 La caméra : structure en modules

La caméra légère de l'IGN fonctionne grâce à une architecture en modules selon plusieurs configurations conditionnées par le type d'usage visé. La configuration minimale est composée d'un module appelé "imageur". D'autres modules peuvent être ajoutés pour des configurations plus complexes par exemple un module batterie de capacité simple ou double (respectivement 2100mAh et 4000mAh) pour les drones n'ayant pas la capacité d'alimenter la caméra. On ne détaillera ici que les différents composants qui constituent le module principal "imageur".

FIGURE 4.2 – Embarquement de la caméra légère de l'IGN sur plusieurs types de drones.

Le module "imageur"

Le module "imageur" est composé essentiellement d'un capteur d'image CMOS (Complementary Metal-Oxide Semiconductor), d'un SoC (System On Chip), d'une mémoire RAM, d'une carte SSD format M.2, d'une carte µSD et d'un module inertiel IMU (Inertial Measurement Unit).

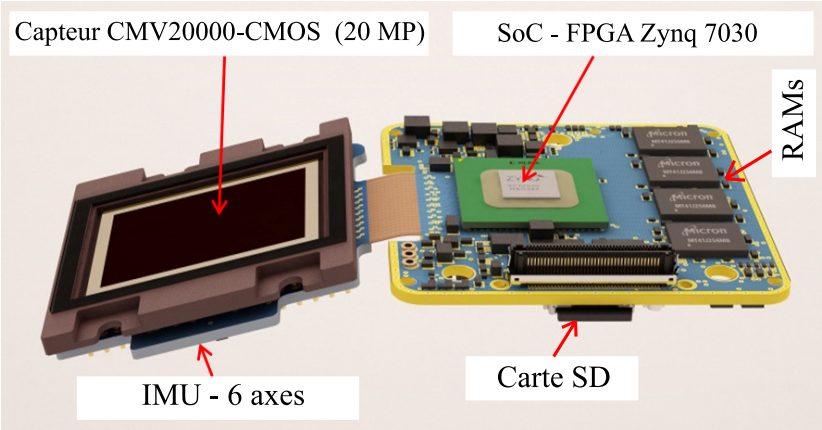


FIGURE 4.3 – Les composants principaux du module "imageur".

Le capteur d'image

Le capteur d'image est le composant le plus important de la caméra, il permet de transformer les rayonnements composés des photons en une image numérique.
Pendant la phase des choix techniques, les ingénieurs du LOEMI ont décidé que la caméra ne comprendrait pas un obturateur mécanique à cause des contraintes liées aux poids. En même temps, ils voulaient éviter le problème du "rolling shutter" qui cause une déformation non maîtrisée de l'image dans le cas d'un mouvement de la caméra (figure 4.4) ou du sujet photographié (figure 4.5). Donc, ils se sont orientés vers un capteur CMOS disposant d'un obturateur global qui permet d'assurer une exposition simultanée de tous les pixels.
Le capteur d'image intégré dans la caméra légère est de type CMV20000 (figure 4.6) de la société CMOSIS. Ce CMOS est plein format ($24.5 \times 33.7mm$) et possède une résolution d'image de 5120×3840 pixels. Il est caractérisé par une cadence d'acquisition

pleine résolution de 30 images/seconde. Cette vitesse d'acquisition élevée ouvre la porte à des applications intéressantes de traitement d'image, surtout avec la présence du FPGA Zynq-7030 qui pourra assurer une importante puissance de calcul. Dans notre travail, nous avons utilisé les images en niveaux de gris de 5MP (2560×1920 pixels) obtenues à partir des images d'origine type bayer en faisant la moyenne de chaque paire verte.

FIGURE 4.4 – L'effet rolling shutter dans le cas du mouvement de la caméra. Source : www.course-de-drone.fr.

FIGURE 4.5 – L'effet rolling shutter dans le cas du mouvement d'un objet. Source : www.leblogphoto.net.

Les capteurs inertiels

Les capteurs inertiels comprennent des accéléromètres et des gyromètres mesurant respectivement les accélérations selon trois axes X, Y, Z orthogonales et les vitesses angulaires autour de ces mêmes trois axes (figure 4.7). La centrale inertielle intégrée dans la caméra est du type ICM-20608-G de la société InvenSense. Elle est fixée sur la carte électronique de l'étage "imageur" (figure 4.8) de manière rigide.
Les gyromètres délivrent avec une faible latence jusqu'à une fréquence de 333Hz la vitesse angulaire autour des trois axes. Un sous-système logiciel a été développé antérieurement qui permet d'obtenir une estimation de la rotation dans un repère initial de chaque pose (R^n_{imu}) acquise par le capteur d'image.
Les capteurs inertiels de cette centrale à faible coût sont basés sur la technologie MEMS (Micro Electro Mechanical Systems). Le principal inconvénient de cette technologie est

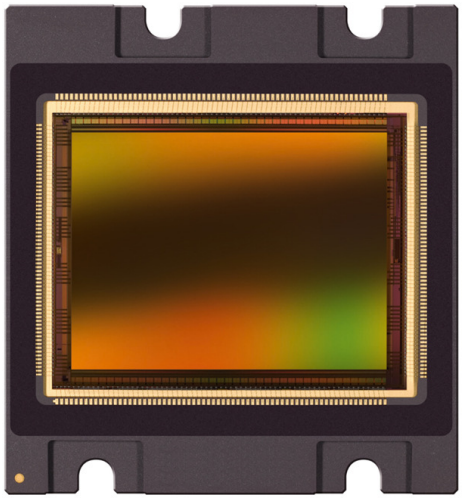


FIGURE 4.6 – Le capteur CMOS de type CMV20000 utilisé dans la caméra légère.

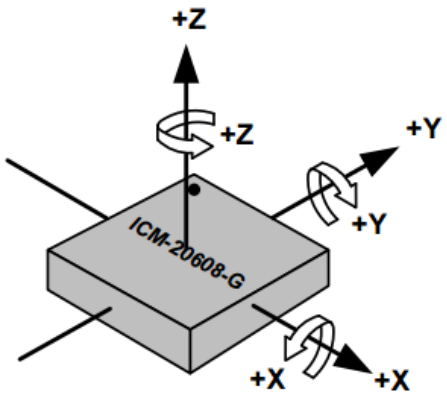


FIGURE 4.7 – L'orientation des axes et la polarité de rotation de l'IMU.

la performance réduite en termes de précision et de stabilité : à cause de l'intégration de la vitesse angulaire, l'instabilité du biais est le principal type d'erreur qui affecte les mesures. Pour mitiger ce problème, l'influence de la température sur le biais sera évaluée en laboratoire car la température dans la caméra peut être élevée (75 °C) à cause de la dissipation de la chaleur issue de l'électronique. Une autre estimation de biais est faite pendant quelques secondes lorsque le drone est immobile juste avant le décollage.

SoC basé sur un FPGA et un processeur ARM

Le cerveau de cette caméra est constitué d'un système sur puce (SoC) de type Zynq-7030 de la société Xilinx (figure 4.9). Il est composé d'un processeur double cœur ARM Cortex-A9 sur lequel tourne un système d'exploitation embarqué (linux) et d'une partie matérielle programmable sous la forme d'un FPGA. Le rôle principal du SoC est de gérer les liens avec le capteur CMV20000 afin d'acquérir les données image. Il dispose également d'un grand nombre de périphériques : contrôleurs mémoires RAM DDR3, Ethernet, USB2.0, lecteur de cartes SD, UART, SPI, I2C. La capacité de calcul que le Zynq-7030 peut fournir fait de cette caméra une plate-forme intéressante pour la

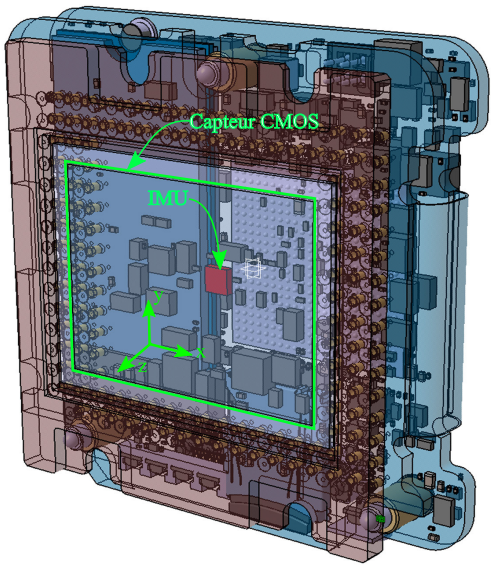


FIGURE 4.8 – L'IMU et le capteur CMOS à l'intérieur de la caméra IGN.

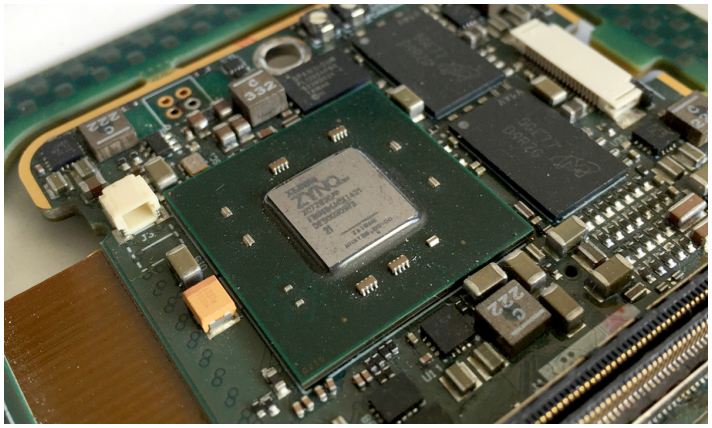


FIGURE 4.9 – Le SoC ZYNQ-7030 implanté dans la caméra.

recherche et pour le développement surtout au niveau de l'implémentation de nouvelles fonctionnalités temps-réel utiles pour d'autres applications plus complexes.

Optique

Dans notre travail, les prises de vue sont effectuées avec un objectif de type monture Leica M. à focale fixe de 35mm. On a choisi cette distance focale car elle est la plus adaptée à la photographie que l'on souhaite réaliser (rues, crues, batiments...). Une focale fixe permet de définir un modèle caméra unique qui peut être appliqué à l'ensemble des prises de vue. Une mise au point est réalisée manuellement avant la mission afin d'avoir une bonne netteté durant l'acquisition. Cette mise au point doit être effectuée à l'infini puisque l'altitude à laquelle les images aériennes sont acquises reste toujours supérieure à la distance hyperfocale. Enfin, la distance focale sera estimée d'une manière précise dans le processus de calibration ainsi que les distorsions dues à l'optique.

FIGURE 4.10 – La caméra de l'IGN sans et avec l'objectif.

4.2 Références

Daakir, M., Pierrot-Deseilligny, M., Bosser, P., Pichard, F., and Thom, C. (2015). Uav onboard photogrammetry and gps positionning for earthworks. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3/W3 :293–298. 30

Daakir, M., Pierrot-Deseilligny, M., Bosser, P., Pichard, F., Thom, C., and Rabot, Y. (2016). Study of lever-arm effect using embedded photogrammetry and on-board gps receiver on uav for metrological mapping purpose and proposal of a free ground measurements calibration procedure. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3/W4 :65–70. 30

Martin, O., Meynard, C., Pierrot Deseilligny, M., Souchon, J.-P., and Thom, C. (2014). Réalisation d'une caméra photogrammétrique ultralégère et de haute résolution. *Colloque Drones et moyens légers aéroportés d'observation, Montpellier, France*. 29

Tournadre, V., Pierrot-Deseilligny, M., and Faure, P. H. (2015). Uav linear photogrammetry. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3/W3 :327–333. 30

Chapitre 5

Description de l'algorithme de stacking

Sommaire

5.1 Introduction

Un nouvel algorithme a été développé pour compenser le flou de bougé de la caméra légère de l'IGN en bénéficiant des capteurs inertiels (gyromètres) présents dans cette caméra. Ce chapitre est consacré principalement à la description de l'algorithme proposé de traitement d'images et de données IMU. Les hypothèses et les choix théoriques aboutissant à la méthode seront présentés au fur et à mesure. Des résultats seront aussi présentés et analysés tout au long de ce chapitre pour qualifier les différentes phases du processus.

5.2 Choix de la méthode d'élimination de bougé

Pour pouvoir faire une image nette avec un temps de pose long, on envisage d'acquérir plusieurs images avec un temps de pose assez court puis d'exploiter ces images afin de générer une image finale empilée au flou minimisé. Nous nous intéressons aussi à conserver une cadence d'acquisition entre deux séquences successives ; cela signifie qu'il est nécessaire que notre algorithme s'exécute en temps réel sur notre plateforme. Pour maintenir une cadence élevée entre deux séquences, on ne peut sauvegarder qu'une seule image dans le système de stockage car la bande passante de cet dernier est limitante. Donc au lieu d'acquérir plusieurs images et de les sauvegarder puis les traiter a posteriori au sol, on les acquiert dans la mémoire RAM de la caméra, puis on les traite en temps-réel pendant le vol pour ne sauvegarder qu'une seule image finale. L'ordre de grandeur du temps réel dans notre cas sera égal au temps d'écriture d'une image dans le système de stockage (Carte µSD).
Les images acquises pendant le vol présentent des déformations liées aux effets de mouvement du drone pendant l'acquisition. Dans le but de superposer toutes les images dans un référentiel commun, on a besoin de connaître les transformations géométriques qui existent entre ces images. Tout cela nécessite l'utilisation d'une méthode de recalage dont les paramètres seront choisis en fonction de notre application. En regardant la conclusion tirée de l'état de l'art des méthodes de recalage, on peut remarquer que l'approche géométrique consomme beaucoup moins de temps de calcul et de mémoire

que l'approche iconique parce qu'elle nécessite moins de données pour l'estimation de la transformation, c'est pourquoi nous l'avons choisie. La première image sera l'image de référence et toutes les autres images seront des images secondaires. Il reste maintenant à choisir les éléments de l'approche géométrique :

- Les attributs : il existe plusieurs types de caractéristiques géométriques qui peuvent être extraites des images. On peut distinguer : les points, les contours, les courbes et les surfaces. Nous avons choisi les points d'intérêt comme attributs parce que d'une part, ils sont les plus faciles à extraire et les moins coûteux en calcul et en utilisation de mémoire et sont présents fréquemment dans les images aériennes. Il existe un grand nombre de détecteurs, une prochaine partie de ce chapitre est consacrée à l'état de l'art des détecteurs existants et à notre choix à ce propos.

- Le type de transformation : dans la réalité, le monde est en 3D et le drone est en mouvement. Afin de faire des mesures sur les images en prenant en compte les déformations géométriques et les effets de perspective, notre problématique aurait besoin de disposer du modèle 3D (MNS) qui pourra être obtenu par la chaîne photogrammétrique basée sur la technique d'ajustement de faisceaux. Malheureusement, ce traitement n'est pas possible en temps réel sur notre plateforme. En absence de ce MNS, nous avons étudié deux approches possibles :

  - La première approche considère que le drone est quasi-stationnaire. Dans cette situation, le centre optique de la camera est considéré comme fixe, ce qui conduit à ne prendre en compte que l'orientation de la caméra et donc à ne corriger que la déformation due à la rotation. Ainsi, la transformation géométrique entre les images sera une projection 2D de la rotation pure des prises de vue dans l'espace 3D.
  - La deuxième approche considère que le drone peut subir en plus de la rotation, une translation. Dans cette situation, la scène est considérée comme planaire et la transformation géométrique sera une transformation projective de type homographie.

- Le critère de similarité : la distance géométrique euclidienne entre les points correspondants est classiquement la plus utilisée comme mesure de distance pour l'appariement entre les images. Dans notre cas, les points correspondants sont d'une part les points d'intérêt reprojetés dans les images et d'autre part leurs homologues.

- L'algorithme d'optimisation : on utilise la méthode de moindres carrés pour estimer la transformation géométrique entre les images à partir des points correspondants obtenus. Cette méthode est réitérée pour permettre d'éliminer sur des critères statistiques les points mal appariés aberrants.

5.3 Description détaillée de l'algorithme

L'algorithme est basé globalement sur le modèle de recalage choisi (figure 5.1).

- L'orientation initiale de la caméra est calculée en utilisant les mesures inertielles obtenues par le sous-système de l'IMU (R^1_{imu}).

- Acquisition en mode rafale de 10 images à la cadence maximale du capteur CMOS (30 images/s). Elles sont ensuite gardées dans la mémoire RAM. Le temps entre deux images acquises est 33ms.
- Détection d'un nombre raisonnable de points d'intérêt dans la première image (de référence).
- Séquentiellement, pour les autres images (secondaires) :
    1. Dans chaque image, la position prédite des points d'intérêt est calculée en utilisant la géométrie de la caméra et les données des capteurs inertiels. Cette étape permet largement d'accélérer l'étape d'identification des points homologues.
    2. Détermination de la position exacte des points homologues à l'aide d'une méthode de corrélation.
    3. (a) La rotation entre la première prise de vue (de référence) et la prise de vue actuelle (secondaire) (R^n_{img}) est estimée à l'aide de la méthode des moindres carrés en éliminant les points aberrants.
        (i) Cette information peut être utilisée dans le sous-système de l'IMU pour éliminer le biais afin d'améliorer la qualité globale des mesures inertielles.
    (b) Une homographie entre l'image de référence et l'image secondaire (H^n_{img}) est estimée à l'aide de la méthode des moindres carrés en éliminant les points aberrants.
    4. L'image (secondaire) est ré-échantillonnée dans la géométrie de la première image.
    5. Superposition de l'image (secondaire) avec l'image résultat.
- Sauvegarde de l'image empilée.

Chaque étape de l'algorithme sera détaillée par la suite en analysant les choix théoriques et techniques.

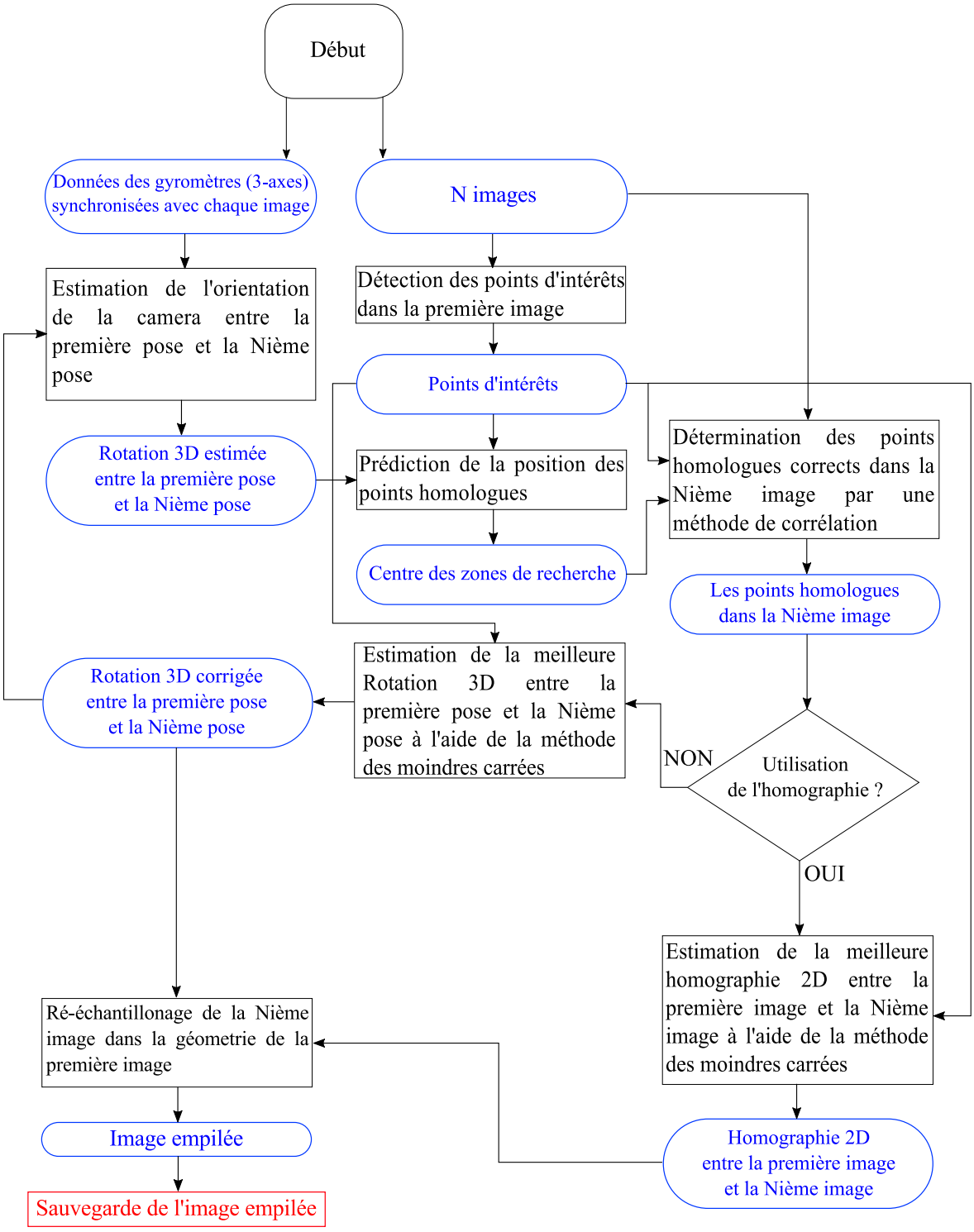


FIGURE 5.1 – Architecture de l'algorithme proposé.

5.4 Acquisition des données IMU et synchronisation avec l'acquisition des images

Par hypothèse, le drone est considéré quasi-stationnaire et donc il ne subit que des rotations. Pour cette raison, ce travail ne prend en compte que la rotation de la caméra. Le but principal de l'utilisation des données inertielles est d'avoir une estimation des

orientations relatives entre chaque prise de vue de la séquence. Cette information est utilisée pour rendre plus robuste et accélérer certaines phases de notre algorithme.
Afin de fusionner correctement les données visuelles (images) et les données inertielles (orientations), les images doivent être synchronisées de façon précise avec les mesures de la rotation fournies par le système inertiel.
L'orientation de la caméra est modélisée par la rotation qui fait passer d'un repère initial arbitraire au repère caméra à l'instant t ; les quaternions sont utilisés ici pour représenter les rotations.
Donc, à chaque déclenchement logiciel de la prise de vue, on envoie une requête à travers un socket au logiciel pilote de l'unité inertielle pour recevoir les mesures de la rotation correspondantes.
Une fois que la rotation absolue attribuée à chaque prise de vue est obtenue, la rotation relative entre deux poses est une composition de deux rotations des poses et peut être calculée de la manière suivante :

$$\hat{R}_{imu}^{1,2} = R_2 \circ R_1^{-1}$$

$\hat{R}_{imu}^{1,2}$ représente la rotation relative entre deux prises de vue acquises, R_1 représente l'orientation de la première prise de vue et R_2 représente l'orientation de la deuxième prise de vue.
La précision des mesures dépend directement de la précision de la centrale inertielle. Malheureusement, les capteurs inertiels bas coût utilisés dans la caméra ne sont jamais parfaits et les mesures fournies par les gyromètres sont affectées principalement par un biais et un bruit. En réalité, les rotations ($\hat{R}_{imu}^{1,n}$) sont obtenues en intégrant les mesures bruitées et biaisées, cela va donc entraîner une dérive au cours du temps. Après un certain temps, les données fournies ne seront plus exploitables à cause de l'accumulation de cette dérive. Lorsque c'est possible, une calibration initiale peut être faite pendant que les capteurs sont immobiles ; de plus, un étalonnage thermique est fait pour chaque caméra pour prendre en compte les effets de la température.

5.5 Détection des points d'intérêt

La détection de points d'intérêt dans les images est une étape fondamentale dans la grande majorité des problèmes de vision par ordinateur surtout dans les applications qui nécessitent une reconnaissance d'objets ou d'appariement d'images : stabilisation vidéo, modélisation 3D, images panoramiques, suivi, etc. Pour cette raison, il existe dans la littérature un grand nombre de travaux qui y sont consacrés. En général, les points d'intérêt se distinguent des autres points de l'image par certaines propriétés telle que l'intensité, la texture, l'orientation, la courbure, etc. Un point d'intérêt peut par exemple être caractérisé par une discontinuité des niveaux de gris dans deux directions. On peut citer quelques exemples : les coins, les contours ou les points de fortes variations d'intensité. En pratique, chaque point d'intérêt a une position 2D précise dans l'image et quelquefois lui est associée un descripteur ou un score.

5.5.1 État de l'art des différents détecteurs

D'après (Schmid et al. [2000]), les détecteurs de points d'intérêt sont classés suivant trois catégories principales :

- Les détecteurs basés sur les contours : Ces détecteurs existent depuis longtemps, ils s'appuient sur deux idées : soit trouver premièrement des contours puis chercher les points au niveau de la courbure maximale des contours (Shilat et al. [1997], Mokhtarian and Suomela [1998]). Soit détecter les points d'intersection entre les contours (Medioni and Yasumoto [1987], Horaud et al. [1990]).
- Les détecteurs basés sur des modèles paramétriques : ce type de détecteurs consistent à estimer les paramètres d'un modèle théorique qui s'ajuste le plus précis possible au niveau du point candidat sur l'image. (Baker et al. [1998]) proposent un détecteur de ce type.
- Les détecteurs basés sur l'intensité : la détection de points d'intérêt ici s'appuie directement sur les valeurs d'intensité des pixels dans l'image. Ils consistent à calculer pour chaque pixel une réponse dont la valeur dépend des intensités des pixels du voisinage. Ensuite le pixel considéré est dit point d'intérêt si la réponse calculée est maximale. Les détecteurs de ce type les plus connus sont ceux de Moravec, Harris, SIFT(Scale Invariant Feature transform), SURF(Speeded Up Robust Features), SUSAN et FAST(Features from Accelerated Segment Test).

En pratique, les détecteurs basés sur les contours peuvent aboutir à des contours mal détectés près des coins. De plus, les détecteurs basés sur les modèles paramétriques sont très limités parce qu'ils ne recherchent qu'un seul type spécial de points dans l'image et en même temps, ils dépendent de l'approximation du modèle théorique estimé. Enfin, les détecteurs basés sur l'intensité sont généralement les plus utilisés car ils sont relativement robustes et stables et ils ne dépendent pas de la détection de contours. Donc, nous nous sommes intéressés à la dernière catégorie de détecteurs.

5.5.2 Détecteurs basés sur l'intensité

On va détailler les différents détecteurs basés sur l'intensité qui existent dans la littérature afin de quantifier d'une part la qualité des points détectés et d'analyser d'autre part la complexité algorithmique de chacun d'eux.

Détecteur Moravec

Ce détecteur proposé par Moravec en 1977 (Moravec [1977]) est parmi les premiers détecteurs de points d'intérêt qui sont basés sur la fonction d'intensité de l'image. L'idée de Moravec est de considérer une fenêtre rectangulaire $w_{x,y}$ autour du pixel candidat (x, y) et de déterminer la moyenne des différences d'intensité pour de petits déplacements $(\Delta x, \Delta y)$ de la fenêtre $w_{x,y}$ dans différentes directions grâce à la somme des différences au carré (Harris and Stephens [1988]) :

$$E(\Delta x, \Delta y) = \sum_{u,v \in w_{x,y}} |I(u + \Delta x, v + \Delta y) - I(u, v)|^2 \tag{5.1}$$

où $I(u, v)$ est l'intensité du pixel candidat à la position (u, v) dans l'image. Les différents déplacements sont : $(\Delta x, \Delta y) \in \{(1, 0), (1, 1), (0, 1), (-1, 1)\}$. $E(\Delta x, \Delta y)$ représente donc la variation moyenne de l'intensité pour un petit déplacement $(\Delta x, \Delta y)$ de la fenêtre. On peut distinguer ici trois situations intéressantes :

- Si le pixel candidat se trouve dans une zone uniforme (intensité globalement constante), alors tous les écarts d'intensités seront assez faibles : $E(\Delta x, \Delta y) \approx 0$: cas(c) (figure 5.2).

- Si le pixel examiné est situé proche d'un contour, les déplacements non parallèles au contour engendreront des écarts d'intensités importants ($E(\Delta x, \Delta y) > 0$). En revanche, les déplacements parallèles au contour provoqueront des différences d'intensités faibles ($E(\Delta x, \Delta y) \approx 0$) : cas(b) (figure 5.2).
- Si le pixel candidat tombe dans un coin ou à proximité d'un coin, les écarts d'intensité sont tous importants : $E(\Delta x, \Delta y) > 0$: cas(a) (figure 5.2).
- Si le pixel à traiter est un pixel isolé, il sera considéré comme un coin.

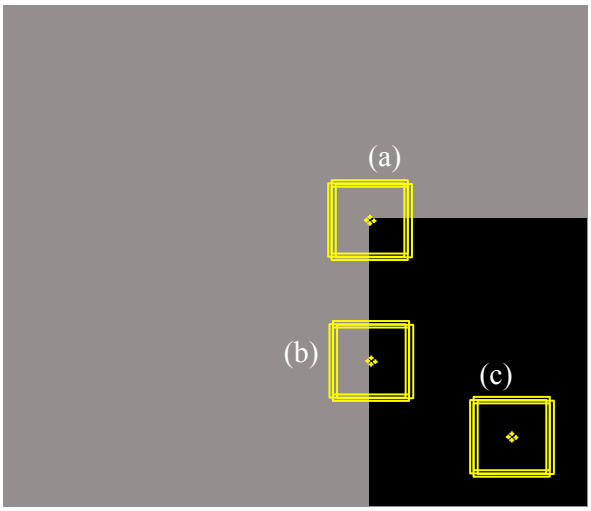


FIGURE 5.2 – Illustration du détecteur Moravec

Par conséquence, cette méthode revient formellement à rechercher les maxima locaux dans l'image du minimum de $E(\Delta x, \Delta y)$.
L'avantage du détecteur Moravec est qu'il permet de détecter des contours et des points isolés, mais il souffre de plusieurs limitations : premièrement, la réponse de ce détecteur est très sensible au bruit en raison de l'utilisation de la fenêtre binaire rectangulaire $w_{x,y}(u, v)$ considérée. En plus, seulement les déplacements discrets de 45 degrés sont pris en compte.

Détecteur Harris

Afin de passer outre les limitations identifiées du détecteur Moravec, Harris et Stephen (Harris and Stephens [1988]) ont proposé une nouvelle approche améliorée connue sous le nom de *détecteur de Harris*. Premièrement, la fenêtre rectangulaire discrète peut être remplacée par une fenêtre circulaire lisse par exemple une fonction gaussienne afin de réduire le bruit :

$$w_{x,y}(u, v) = exp(-\frac{(u^2 + v^2)}{2\sigma^2}) \tag{5.2}$$

La fonction du changement d'intensité $E(x, y)$ est améliorée en utilisant un développement de la formule de Taylor de second ordre :

$$E(x, y) = \sum_{u,v} w_{x,y}(u, v) \; [x\frac{dI}{dx} + y\frac{dI}{dy} + o(x^2, y^2)]^2 \tag{5.3}$$

où $o(x^2, y^2)$ peut être négligé pour les petits déplacements. Ainsi, $E(x, y)$ sera écrit sous la forme suivante :

$$E(x, y) = Ax^2 + 2Cxy + By^2 \tag{5.4}$$

avec :

$$A = \frac{d^2I}{dx^2} \otimes w,\ B = \frac{d^2I}{dy^2} \otimes w \text{ et } C = (\frac{dI}{dx}\frac{dI}{dy}) \otimes w \tag{5.5}$$

En forme matricielle, $E(x, y)$ peut être écrite de la façon suivante :

$$E(x, y) = (x, y).M.(x, y)^T \tag{5.6}$$

où M est une matrice symétrique de taille 2×2 :

$$\begin{bmatrix} A & C \\ C & B \end{bmatrix} \tag{5.7}$$

En effet, la matrice M est connue sous le nom de tenseur de structure du pixel. C'est en fait une caractérisation de l'information de tous les pixels dans la fenêtre. Les valeurs propres λ_1 et λ_2 de M sont proportionnelles aux courbures principales de la fonction E. A partir de ces valeurs, on peut distinguer trois situations :

- Si les deux courbures sont de faibles valeurs (λ_1 et λ_2 sont petites), E est approximativement constante, cela veut dire que le pixel candidat se trouve dans une zone uniforme.
- Si et seulement si une des deux courbures est de forte valeur ($\lambda_1 >> \lambda_2$ ou $\lambda_1 << \lambda_2$), alors le pixel candidat se situe à la proximité d'un contour.
- Si les deux courbures sont de fortes valeurs (λ_1 et λ_2 sont grandes) cela indique que la variation d'intensité est forte dans toutes les directions. Par conséquent, le point candidat est considéré comme un coin.

Finalement, au lieu de faire des classifications des régions à partir des valeurs propres de la matrice M, les auteurs ont proposé de calculer la réponse suivante pour examiner un tel point candidat :

$$R = Det(M) - k * Trace(M)^2 \tag{5.8}$$

où $Det(M) = AB - C^2$, $Trace(M) = A + B$ et k est un seuil à paramétrer. Les points qui ont une réponse négative ($R < 0$) se trouvent à proximité d'un contour, ceux qui ont une réponse positive ($R > 0$) sont au voisinage d'un coin et ceux qui ont une réponse faible ($R \approx 0$) résident dans une zone homogène dont l'intensité est constante. (Schmid et al. [2000]) ont montré que le détecteur de Harris présente des avantages non négligeables qui font de lui un détecteur très populaire. Il est connu par sa stabilité et sa robustesse au bruit. En revanche, sa complexité algorithmique importante nécessite beaucoup de ressources (mémoire, cpu..) et aboutit à des temps de traitement importants par rapport aux autres détecteurs.

Détecteur SIFT

Le détecteur SIFT (Scale-Invariant Feature Transform) a été proposé en 1999 par D. G. Lowe (Lowe [1999]) principalement pour effectuer des mises en correspondance d'images acquises de différents points de vue. L'idée essentielle de SIFT est de détecter des points d'intérêt qui sont invariants aux différents transformations (rotation, changements d'échelle, changements de luminosité, changements du point de vue, etc..). L'approche est composée en deux étapes : dans un premier temps, des points d'intérêt sont détectés. Ensuite, un descripteur qui caractérise localement le voisinage autour du point détecté est construit. On ne détaille que la partie détection de points d'intérêt

car on ne s'intéresse pas à utiliser un descripteur dans notre travail : l'extraction de points d'intérêt consiste à rechercher les extrema locaux dans l'espace-échelle (à plusieurs facteurs d'échelle) en se basant sur des gaussiennes. On note $L(x, y, \sigma)$ le résultat de la convolution de l'image $I(x, y)$ dans l'espace-échelle gaussienne :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{5.9}$$

où σ est le facteur d'échelle et $G(x, y, \sigma)$ est le noyau gaussien :

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{5.10}$$

La détection des extrema est faite généralement à l'aide de la fonction LoG (Laplacian of Gaussien) qui est coûteuse en calcul. *Lowe* propose de faire une approximation de la fonction LoG par une différence de gaussiennes DoG (Difference of Gaussien) :

$$DoG(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{5.11}$$

Les extrema locaux de DoG représentent alors les pixels qui ont une intensité maximale ou minimale par rapport à ses 8 voisins dans l'image ainsi qu'aux 9 voisins de l'échelle précédente et de l'échelle suivante. Ainsi on a besoin de faire 26 comparaisons pour chaque point.
Comme ces points d'intérêt obtenus sont assez nombreux, les points qui ont un faible contraste où se trouvent le long d'un contour sont éliminés en analysant la valeur du laplacien implémenté à l'aide de DoG. L'algorithme proposé comporte aussi des étapes supplémentaires destinées à apporter plus de précision dans la localisation de points d'intérêt.
SIFT montre une grande stabilité par rapport aux différents changements géométriques et radiométriques dans l'image. Avec son descripteur qui n'est pas détaillé ici, il permet de faire l'appariement de façon robuste. Malheureusement, malgré les différentes améliorations faites dans ce détecteur, la partie détection sans parler de la partie de description reste toujours coûteuse en calculs et en ressources.

Détecteur SURF

Le détecteur SURF (Speeded-Up Robust Features) proposé par H. Bay et L.V. Gool (Bay et al. [2006]) en 2006 s'appuie fortement sur le détecteur SIFT. Il est basé sur une approximation de la matrice hessienne qui leur permet de diminuer largement le temps de calcul en passant par les images intégrales et l'utilisation des filtres de type "box" :

$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix} \tag{5.12}$$

où L_{ij} est le résultat du produit de convolution de l'image $I(x, y)$ par la dérivée de deuxième ordre d'une gaussienne (LoG) $\frac{\partial^2 g(\sigma)}{\partial x^2}$ suivant les directions de i et j. SURF utilise les maxima locaux du déterminant de la matrice hessienne comme critère pour sélectionner les points d'intérêt. Par contre, le déterminant est approximé en utilisant des filtres de type box de diverses tailles directement sur l'image :

$$det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{5.13}$$

où D_{xx}, D_{yy} et D_{xy} sont les résultats du produit de convolution de l'image par l'ensemble de filtres "box" dans les directions x, y et xy. La valeur 0.9 est choisie empiriquement.

Parmi les caractéristiques importantes de SURF, il y a le fait qu'il utilise l'image intégrale pour accélérer le calcul du produit de convolution en calculant la somme des intensités des pixels de l'image. De cette façon, la somme des intensités de n'importe quelle région rectangulaire de l'image originale peut être calculée indépendamment de ses dimensions en utilisant seulement trois additions.
Cette approche tente de faire des approximations pour diminuer le temps de calcul face au détecteur SIFT en conservant toujours ses qualités de détection. Les tests faits par Bay ont montrés que SURF-9 (la taille de filtre box est 3×3) est 5 fois plus rapide que SIFT tout en détectant un nombre de points d'intérêt comparable avec celui du SIFT. Les tests ont montré aussi une bonne stabilité et une bonne robustesse aux changements d'échelle et aux changements de vue. Malgré ces améliorations, SURF utilise toujours beaucoup de ressources en mémoire et en calcul CPU.

Détecteur SUSAN

Le détecteur SUSAN (Smallest Univalue Segment Assimilating Nucleus) introduit par S.M. Smith et J.M. Brady (Smith and Brady [1997]) repose sur la mesure de la similarité des intensités au voisinage circulaire local autour du point candidat. Il consiste à construire un masque circulaire dont le centre appelé noyau est le pixel à analyser : chaque pixel x compris dans le masque est comparé avec le pixel noyau p. Ensuite, les pixels ayant un niveau de gris proche de celui du pixel centré p seront gardés afin de définir une zone du masque au voisinage local du pixel p appelée USAN. En pratique, la surface de la zone USAN détermine si le point détecté est un coin ou un contour (figure 5.3) :

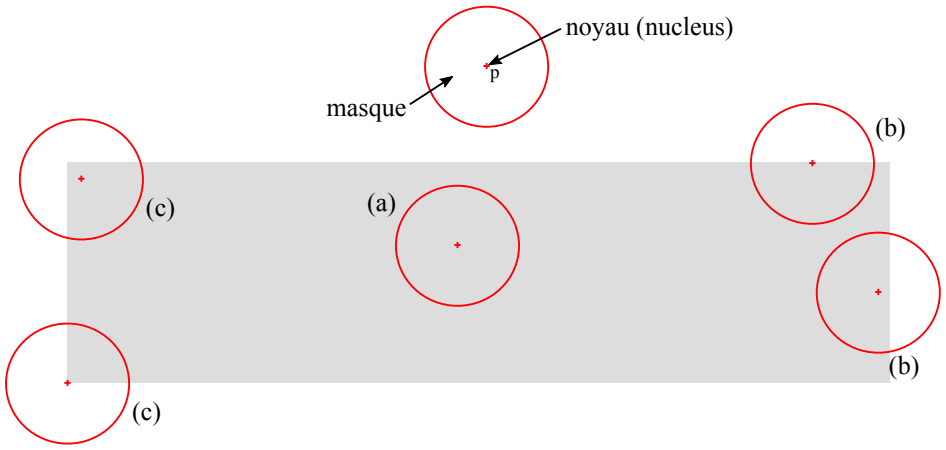


FIGURE 5.3 – Illustration du détecteur SUSAN

- Si $S_{USAN} = S_{masque}$ alors le pixel candidat se trouve dans une zone homogène (cas (a)).
- Si $S_{USAN} \simeq \frac{S_{masque}}{2}$ alors le pixel candidat est près d'un contour (cas (b)).
- Si $S_{USAN} < \frac{S_{masque}}{2}$ alors le pixel candidat se situe dans un coin (cas (c)).

Initialement, Les auteurs de SUSAN ont proposé une simple équation de différence des niveaux de gris afin de comparer un pixel localisé à l'intérieur du masque avec le pixel du noyau :

$$c(\vec{r}, \vec{r_0}) = \begin{cases} 1 \; si \; |I(\vec{r}) - I(\vec{r_0})| \leq s \\ 0 \; si \; |I(\vec{r}) - I(\vec{r_0})| > s \end{cases} \tag{5.14}$$

où $\vec{r_0}$ est la position du pixel candidat (noyau du masque) dans l'espace 2D de l'image, $\vec{r}$ est la position de tous les autres pixels à l'intérieur du masque. I représente l'intensité d'un pixel, s est le seuil de la différence d'intensité. La comparaison est effectuée pour chaque pixel situé dans le masque :

$$n(\vec{r}) = \sum_{\vec{r}} c(\vec{r}, \vec{r_0}) \tag{5.15}$$

A la sortie, la valeur de n correspond au nombre de pixels qui définit la surface de la zone USAN. Ensuite, n est comparé avec un seuil géométrique g fixé à $\frac{n_{max}}{2}$ où n_{max} est la valeur maximale que peut prendre n. La réponse $R(\vec{r})$ du détecteur :

$$R(\vec{r}) = \begin{cases} g - n(\vec{r_0}) \ si \ n(\vec{r_0}) < g \\ 0 \ sinon \end{cases} \tag{5.16}$$

Seuls les points qui représentent des maxima locaux sont considérés comme des coins possibles. Cependant, il est possible que la réponse $R(\vec{r})$ soit positive pour des points appelés faux positifs. Les auteurs proposent d'éliminer ces points en examinant la distance entre le centre de gravité de USAN et le pixel examiné, qui devra être éloigné du noyau pour considérer le point analysé comme coin.
L'avantage de ce détecteur est qu'il est robuste en présence du bruit (Smith and Brady [1997]) puisqu'il n'a pas recours à la dérivation. En plus, il est plus rapide que les autres détecteurs comme Harris, SIFT, etc. Mais, on ne l'a pas choisi puisque (Rosten and Drummond [2006], Rosten and Drummond [2005]) affirment que la partie détection de coins reste complexe pour être utiliser dans des applications temps-réel, alors qu'un autre détecteur existant appelé FAST utilise le même principe mais avec un coût moindre en calcul.

Détecteur FAST

Le détecteur FAST (Features from Accelerated Segment Test) est proposé par E. Rosten et T. Drummond en 2006 (Rosten and Drummond [2006], Rosten and Drummond [2005]) en s'appuyant sur l'idée du détecteur SUSAN. FAST est basé aussi sur l'analyse d'intensités des pixels au voisinage local du pixel candidat. Il consiste à comparer un voisinage de pixels situés sur un cercle de Bresenham de rayon 3 avec le pixel candidat centré p. Le critère de test fonctionne en considérant un cercle de 16 pixels (r = 3) autour du pixel considéré p. Initialement, un pixel candidat p est un point d'intérêt si et seulement si (3/4) pixels contigus (n = 12 pixels) parmi les 16 pixels sont plus clairs ($I_p + s$) ou plus sombres ($I_p - s$) que le point au centre p. On remarquera qu'il est possible d'éliminer directement un point candidat p en faisant un test sur les pixels d'extrémités (pixel 1, pixel 9, pixel 5, pixel 13). Les valeurs de n, r et s sont choisies empiriquement. Par la variation de s, on peut augmenter le nombre de points détectés. La détection de plusieurs points d'intérêt adjacents les uns aux autres est l'un des problèmes de la version initiale du détecteur. Cela a été résolu en appliquant une fonction de suppression non maximale après avoir détecté les points d'intérêt. Cette fonction consiste à calculer un score V pour chaque point d'intérêt détecté puis comparer les scores des points adjacents et prendre seulement le point qui a la valeur la plus grande. Le score V peut être défini par la somme de la différence absolue entre les pixels situés dans l'arc contigu x et le pixel central p :

$$V = max \begin{cases} \sum\limits_{x \in claire} |I_x - I_p| - s \\ \sum\limits_{x \in sombre} |I_p - I_x| - s \end{cases} \tag{5.17}$$

FAST se distingue principalement par sa rapidité élevée par rapport aux autres détecteurs et par sa répétabilité élevée au travers de prises de vue différentes de la même scène. (Rosten and Drummond [2006]) montrent qu'il est aussi robuste aux changements de luminosité. En revanche, il n'est pas très robuste en présence de niveaux de bruit élevés (Le and Gonzalez [2009], Pena [2012]) et il dépend aussi d'un seuil comme beaucoup d'autres détecteurs.

5.5.3 Choix du détecteur

Détecteur	Rapidité	Robustesse au bruit	Stabilité	Nb points
Harris	+	+++	+++	++
SIFT	+	+++	+++	+++
SURF	+	+++	++	++
SUSAN	++	+++	+++	+++
FAST	+++	+	+++	+++

TABLEAU 5.1 – *Résultats de notre analyse de la comparaison entre les différents détecteurs basés sur l'intensité.*

Le choix d'un détecteur de points d'intérêt dépend essentiellement de l'application visée. La performance d'un tel détecteur est qualifiée pour une application par l'évaluation pondérée de certains critères qui sont liés en général à l'appariement de points entre les images. On peut distinguer la stabilité, la robustesse au bruit et la complexité/rapidité parmi les critères les plus examinés (Schmid et al. [2000], Zou et al. [2008]).
Notre choix est concentré sur le critère de rapidité/complexité : la simplicité d'un détecteur de points d'intérêt, c.a.d sa rapidité et sa faible consommation de ressources, implique en général celle de son approche algorithmique et mathématique. La détection de points d'intérêt dans notre travail n'aura lieu que dans la première image, donc le critère de stabilité ne nous concerne pas. Comme l'appariement des points sera effectuée par une méthode de corrélation, on ne s'intéresse pas à construire un descripteur pour trouver les points homologues. En même temps, on veut que les points détectés aient des propriétés qui assurent sans ambiguïté un appariement fiable malgré les déformations géométriques et radiométriques qui peuvent se produire entre les images.
En analysant les avantages et les inconvénients de chaque détecteur (table 5.1), on trouve que FAST répond à nos besoins en termes de rapidité, de simplicité et de qualité. Ce détecteur par son concept simple permet une implémentation non coûteuse (en ressources et en temps) dans le logiciel ainsi que dans le matériel (FPGA). Il assure en même temps une performance de qualité suffisante pour notre application en raison de sa répétabilité élevée démontrée par (Rosten and Drummond [2006]). De l'autre côté, le détecteur FAST n'est pas robuste au bruit (table 5.1) et dépend comme beaucoup d'autres détecteurs d'un seuil qui doit être réglé en fonction du niveau du bruit. Cependant, dans notre caméra, le niveau de bruit est réduit, et est constant en RMS par rapport aux niveaux de gris grâce à la fonction racine carrée implémentée dans les LUT (Look Up Tables). Nous pouvons donc donner une valeur au seuil indépendamment du niveau de luminosité et du contenu de l'image.
On ne prend pas en compte la deuxième partie de l'algorithme FAST qui consiste à éliminer les points adjacents (à l'aide d'une fonction de maximum local) car nous utilisons une approche différente de filtrage qui sera détaillée dans le prochain paragraphe.

Nous avons décidé aussi de n'implémenter que la partie "claire" du détecteur (seulement si les 12 pixels contigus ont une valeur d'intensité supérieure à la valeur d'intensité du pixel candidat à un seuil près), car nos expériences ont prouvé que l'on obtenait ainsi un nombre raisonnable de points d'intérêt.

5.5.4 Uniformisation de la distribution des points d'intérêt dans l'image

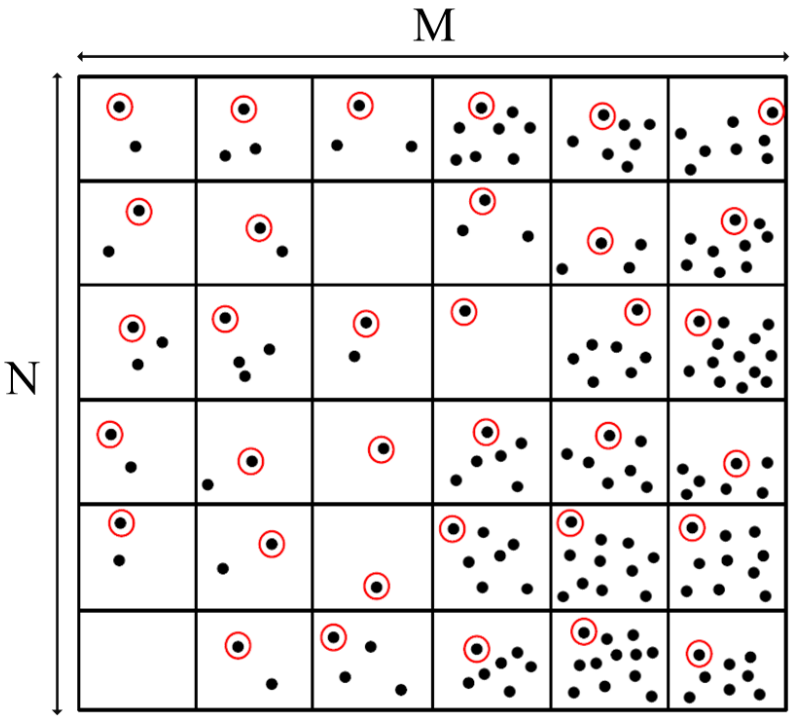
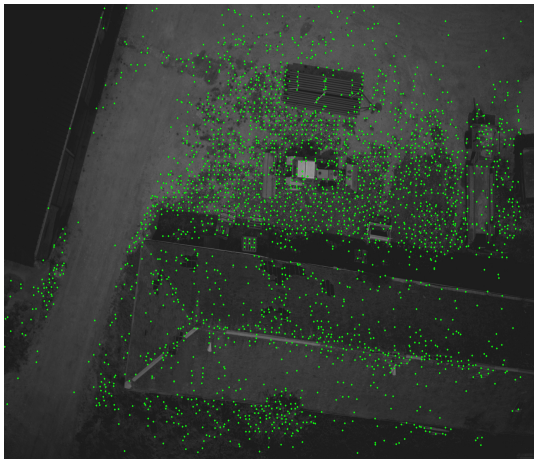
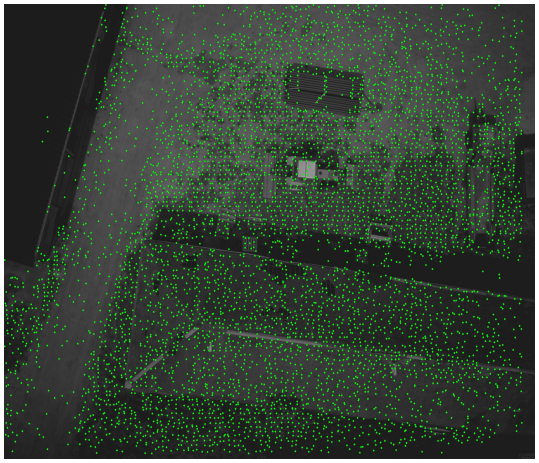


FIGURE 5.4 – Approche de sélection de points d'intérêt.

Quel que soit le modèle de la transformation géométrique entre les images, on a besoin de relativement peu de points pour résoudre le problème. En prendre plus permet d'éliminer les points aberrants et d'améliorer la précision de l'estimation au prix d'une augmentation du temps de calcul de l'appariement. Il faut par contre qu'ils soient répartis de manière correcte dans l'image (Behrens and Röllinger [2010]). L'application des détecteurs et notamment de FAST (Schauwecker et al. [2012]) sur les images qui ne présentent des textures ou des niveaux élevés de contraste que localement ne produit que des points d'intérêt groupés qui ne seront donc pas distribués correctement.
Afin que les zones de faible texturation aient aussi des points d'intérêt tout en limitant leur nombre dans les zones fortement texturées, nous paramétrons le détecteur de manière à augmenter sa sensibilité et appliquons une approche de sélection basée sur une grille. Cette approche consiste à diviser l'image en blocs et conserver seulement le premier point trouvé par le détecteur dans chaque bloc (figure 5.4). Cette méthode permet de maintenir un bon compromis entre une distribution uniforme de points d'intérêt et la simplicité de la mise en œuvre.



(a) Seuil = 6, Grille = 100×100 : 2854 points.



(b) Seuil = 3, Grille = 100×100 : 5935 points.

(c) Seuil = 3, Grille = 10×10 : 98 points.

FIGURE 5.5 – Distribution uniformisée de points d'intérêt détectés par FAST dans l'image : Cette image est acquise avec un faible temps d'exposition. La valeur d'intensité de seuil est réglée à 6 (a). La taille de la grille est réglée à 100 × 100 blocs (a, b). Nous pouvons voir sur (a) qu'aucun point d'intérêt n'est détecté dans les zones à texture très faible. Par contre, la probabilité d'obtenir des points d'intérêt dans les zones faiblement texturées est plus importante lorsqu'on diminue le seuil (s = 3) (b, c). Pour diminuer le nombre de points dans le cas de cette dernière configuration, on règle simplement la grille à 10 × 10 (c).

5.6 Recherche de points homologues dans les autres images

Le suivi de points d'intérêt, l'appariement ou encore la mise en correspondance de points d'intérêt est un point clé pour la plupart des applications de la vision par ordinateur et surtout la reconstruction 3D, la reconnaissance d'objets, la localisation, la stabilisation électronique vidéo, etc. Imaginons que N images ($N \geq 2$) de la même scène 3D sont acquises selon des points de vue différents, l'appariement consiste à détecter des points d'intérêt dans une image et à les identifier dans une autre image malgré les déformations géométriques et radiométriques. Les méthodes les plus connues sont regroupées en deux catégories (Brown et al. [2003]) : les méthodes locales et les méthodes globales. Les méthodes locales s'appuient généralement sur la mesure de la similarité des caractéristiques locales au voisinage de points d'intérêt. Alors que les méthodes globales prennent en compte la totalité de l'image pour minimiser une fonction de coût globale qui représente les erreurs de mise en correspondance.

- La première approche consiste à chercher localement les points homologues des points d'intérêt autour d'une position estimée avec des méthodes telles que la mesure de la corrélation ou de la somme des carrés des différences, etc. La taille de la zone de recherche dépend du contexte de l'application, elle peut être réduite largement selon le type de mouvement subi entre les prises de vue. Elle est connue par sa simplicité de mise en œuvre et sa rapidité, mais elle pose malheureusement certaines ambiguïtés dans le cas de zones homogènes ou dans le cas de zones à textures répétitives. Malgré tout, elle reste largement utilisée surtout dans les applications temps-réel tel que le suivi d'un objet dans une vidéo. Les méthodes locales basées sur la corrélation sont très utilisées ici à cause de leur avantage du faible coût en mémoire et en temps de calcul (Heo et al. [2011]).
- L'approche globale cherche à résoudre un problème d'optimisation sur la carte de disparité afin de minimiser l'erreur globale de mise en correspondance. Cette approche utilise, généralement, des méthodes complexes de minimisation d'énergie, comme la programmation dynamique (Forstmann et al. [2004]), la théorie des graphes (Zureiki et al. [2007])... Ces méthodes sont moins sensibles aux problèmes des zones peu texturées et donnent de bons résultats de mise en correspondance, mais leur inconvénient est le coût trop élevé en temps de calcul.

Le suivi de la position de points d'intérêt entre les images successives peut être facilement effectué avec la première approche citée ci-dessus si le mouvement établi entre les prises de vue successives est connu. Les capteurs inertiels intégrés dans la caméra peuvent intervenir ici pour donner une estimation du mouvement de la caméra lors de l'acquisition et donc fournir une bonne prédiction de la position de points d'intérêt tout au long de la séquence d'images (figure 5.6). Par hypothèse, les images sont acquises avec la vitesse du capteur CMOS (30 images/seconde) cela veut dire que les déplacements entre les images sont petits. Si les déplacements entre les images associés au mouvement subi sont connus parfaitement, la position de points homologues dans les autres images sera obtenue par une simple application de la projection du mouvement sur les points d'intérêt de la première image, mais ce n'est pratiquement jamais le cas. En réalité, les données issues de capteurs inertiels souffrent d'une imprécision due principalement aux problèmes de la dérive ce qui implique la recherche de la position de points homologues dans une voisinage du point estimé. On peut donc définir dans les autres images une zone de recherche centrée sur le point prévu initial. Il reste à

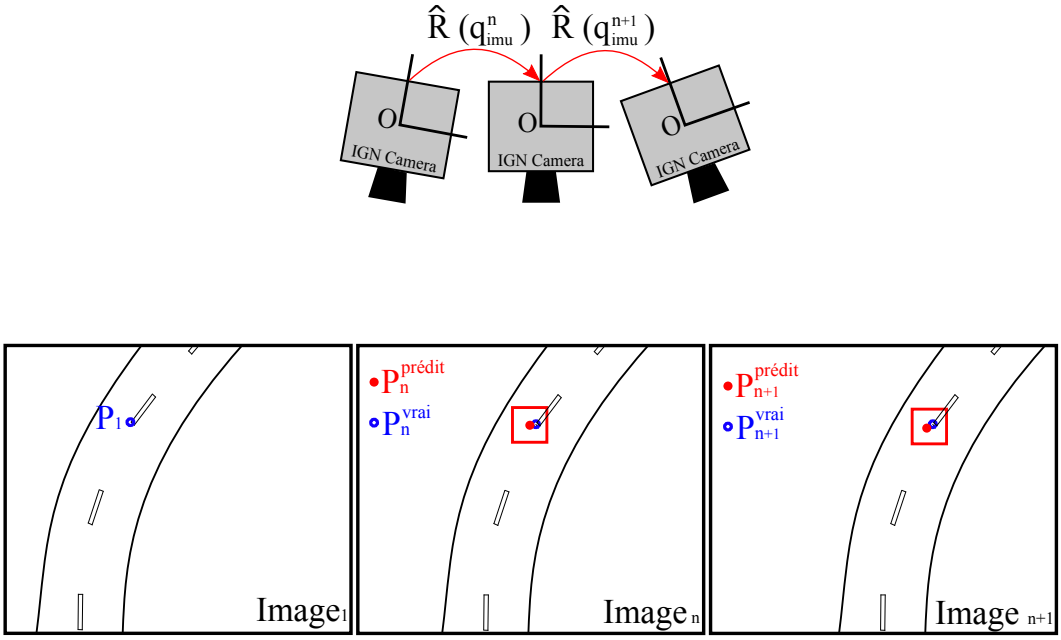


FIGURE 5.6 – Suivi des points sur deux images à l'aide d'une méthode locale. Les carrés rouges sont centrés sur les positions initiales ($P_n^{prédit}$) des points après un déplacement dû au mouvement.

définir une mesure de similarité pour évaluer la ressemblance entre le point d'intérêt et l'ensemble des points de la zone de recherche afin de retrouver le vrai point homologue qui ressemble le plus au point d'intérêt.
Cette étape de traitement représente le cœur de l'algorithme puisque le reste de l'algorithme s'appuie fortement sur la mise en correspondance et donc la qualité des résultats dépend de la fiabilité de cette étape.

5.6.1 Estimation de la position de points homologues à l'aide de l'IMU

Le capteur IMU fixé sur le même plan que le capteur CMOS permet de fournir à chaque prise de vue une estimation de l'orientation de la caméra. En utilisant les paramètres d'orientation interne produits par le processus d'étalonnage optique et la rotation relative inverse estimée de la caméra $\hat{R}_{imu}^{n-1}$ entre les poses, nous pouvons calculer géométriquement la position prédite de points homologues dans les autres images. Pour envisager des mesures géométriques dans le plan image d'une façon précise, il est nécessaire de restituer la localisation du point du plan corrigé de la distorsion géométrique due à l'optique. Soit $P^{1,i}$ un point d'intérêt situé dans le plan de la première image et $\hat{P}^{n,i}$ le point homologue prédit et situé dans le plan de la $N^{ième}$ image. A noter que PPA est le point principal d'auto-collimation et f est la distance focale. Donc après la correction de la distorsion :

$$P_c^{1,i} = \mathcal{D}^{-1}(P^{1,i}) \tag{5.18}$$

où $\mathcal{D}^{-1}(P^{1,i})$ est la fonction qui permet de corriger la distorsion au point P.
On construit dans le repère caméra canonique de la caméra le vecteur perspectif $\overrightarrow{V_c^{1,i}}(x_c^{1,i}, y_c^{1,i}, z_c^{1,i})$ à partir du point corrigé $P_c^{1,i}(u_c^{1,i}, v_c^{1,i})$ et des paramètres de l'orientation

interne de la caméra :

$$\begin{cases} x_c^{1,i} = u_c^{1,i} - u_{PPA} \\ y_c^{1,i} = v_{PPA} - v_c^{1,i} \\ z_c^{1,i} = -f \end{cases} \tag{5.19}$$

On applique ensuite à ce vecteur $\overrightarrow{V_c^{1,i}}$ l'inverse de la rotation $\hat{R}_{imu}^{n}$ pour obtenir le vecteur $\overrightarrow{\hat{V}_c^{n,i}}$:

$$\overrightarrow{\hat{V}_c^{n,i}} = \hat{R}_{imu}^{n^{-1}}(\overrightarrow{V_c^{1,i}}) \tag{5.20}$$

Le vecteur $\overrightarrow{\hat{V}_c^{n,i}}$ est ensuite re-projeté dans le plan de la $n^{ième}$ image pour pouvoir obtenir la position du point d'intersection $\hat{P}_c^{n,i}(\hat{u}_c^{n,i}, \hat{v}_c^{n,i})$ entre le plan de la $n^{ième}$ image et le faisceau perspectif dirigé par le vecteur $\overrightarrow{\hat{V}_c^{n,i}}$:

$$\begin{cases} \hat{u}_c^{n,i} = -\dfrac{x_c^{n,i}}{z_c^{n,i}} f + u_{PPA} \\ \hat{v}_c^{n,i} = \dfrac{y_c^{n,i}}{z_c^{n,i}} f + v_{PPA} \end{cases} \tag{5.21}$$

L'effet de la distorsion est appliqué à nouveau sur les points obtenus afin de pouvoir travailler avec les données brutes de l'image $\hat{P}^{n,i}$:

$$\hat{P}^{n,i} = \mathcal{D}(\hat{P}_c^{n,i}) \tag{5.22}$$

Une illustration simple des étapes intermédiaires du traitement est présentée (figure 5.7).

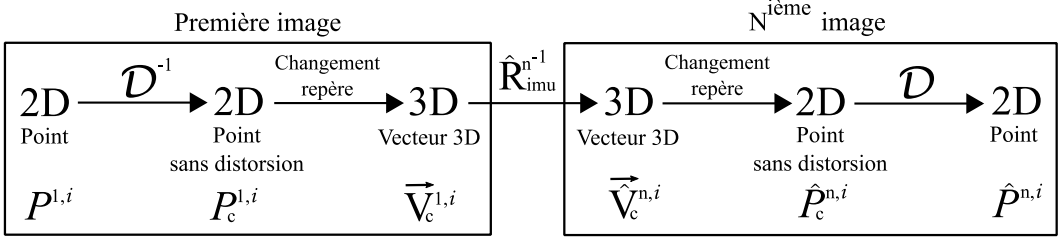


FIGURE 5.7 – Les différentes étapes pour retrouver la position prédite initiale du point homologue dans l'autre image en utilisant la rotation relative inverse $\hat{R}_{imu}^{n^{-1}}$ estimée entre les prises de vue.

Donc la position obtenue $\hat{P}^{n,i}$ n'est qu'une position approximative du point homologue. Nous considérons cette position comme le centre d'une zone de recherche dans laquelle on recherche par corrélation la position exacte. La figure 5.8 montre les trajectoires de points homologues prédits dans toute la séquence des images. On remarque que le déplacement de ces points n'est pas le même dans toute l'image. Donc, une méthode de déconvolution avec une seule fonction PSF constante dans toute l'image ne peut pas convenir ici.

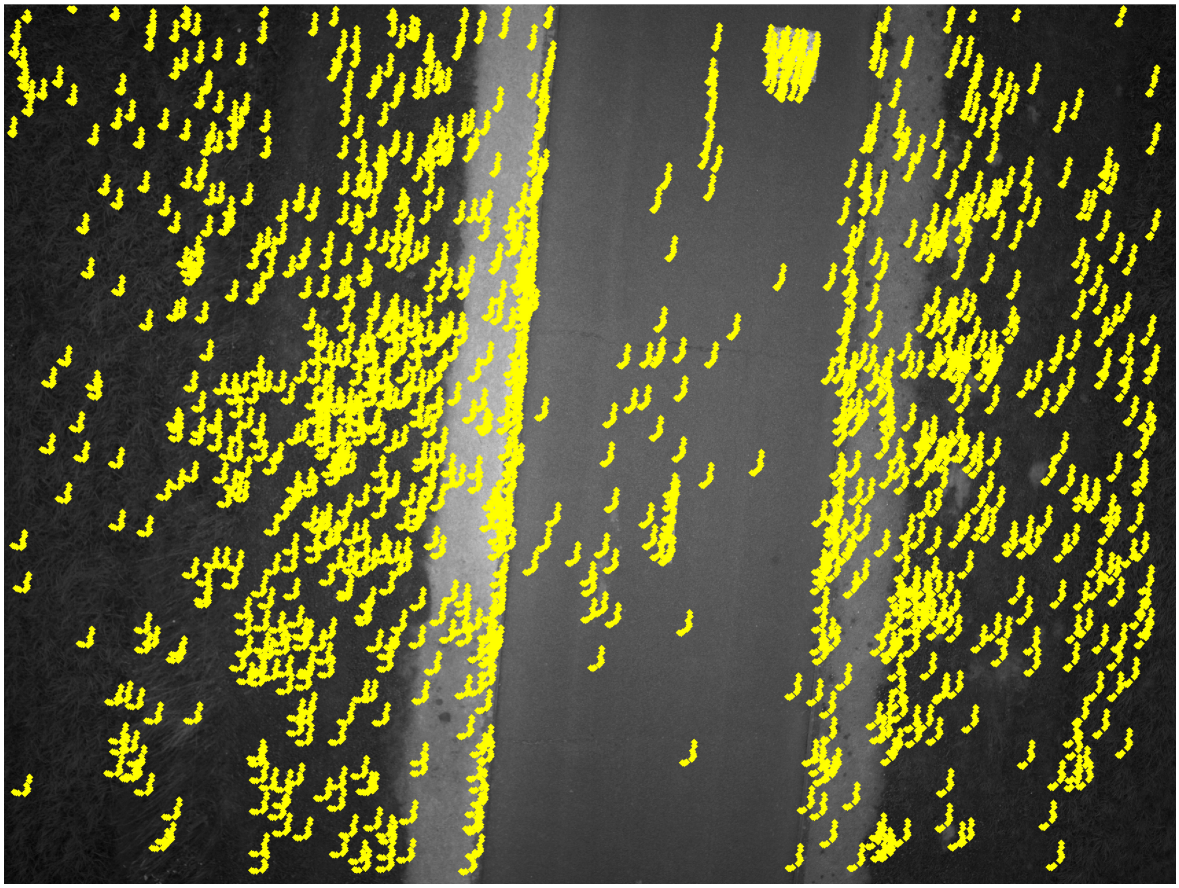


FIGURE 5.8 – Positions prédites par l'IMU des points homologues tout au long de la séquence des images.

5.6.2 Recherche de la position correcte de points homologues

Une fois que la position prédite $\hat{P}^{n,i}$ associée à chaque point homologue est trouvée dans les autres images de la séquence, la position exacte de ces points $P^{n,i}$ peut être obtenue en maximisant une mesure de corrélation des niveaux de gris dans la zone de recherche centrée sur la position prédite initiale $\hat{P}^{n,i}$. Cette mesure de corrélation a pour but de quantifier la similarité entre une vignette autour de chaque point d'intérêt de la première image (vignette de référence) et celles autour des points candidats qui se trouvent à l'intérieur de la zone de recherche considérée dans les autres images.
Les méthodes à base de corrélation présentent l'avantage d'une grande simplicité d'implémentation logicielle. En revanche, elles sont peu robustes dans certaines situations : les zones non texturées, les zones répétitives, les occultations et les déformations, où elles peuvent engendrer des points homologues mal appariés. La taille de la vignette de référence et celle de la zone de recherche sont des paramètres difficiles à affiner dans ces méthodes :

- Si la taille de la vignette est très petite, les statistiques ne seront pas robustes. Si la taille est trop grande, on risque d'avoir une imprécision dans la corrélation puisqu'un grand nombre de pixels appartenant aux deux vignettes ne seront probablement pas correspondants, surtout dans le cas de changements d'échelle et des rotations.
- Si la taille de la zone de recherche est très petite, il y aura un risque que cette

zone ne couvre pas le point correspondant. Si la taille est trop grande, on a un risque de subir un appariement incorrect dans le cas où la zone se trouve dans des régions homogènes ou à motifs répétitifs. Dans les deux cas, on obtient un mauvais appariement.

- De toute manière, agrandir ces zones est coûteux en calcul.

Dans notre travail, on a adapté la taille de la vignette à la taille de la fenêtre associée au détecteur de caractéristiques (7×7 pixels) puisqu'on cherche en réalité à trouver la vignette qui corresponde au mieux à la structure pixellaire autour du point d'intérêt. Concernant la taille de la zone de recherche, ceci dépend en pratique de la précision de l'IMU. On a testé plusieurs tailles pour la zone de recherche afin de choisir la meilleure qui sera utilisée sur la séquence des images, et on a laissé en même temps une petite marge pour garantir un recouvrement de points correspondants. Cependant, on peut diminuer la taille de la zone à partir de la deuxième image après l'élimination du biais de l'IMU.
Il existe dans la littérature (Chambon and Crouzil [2003]) un grand nombre de méthodes de corrélation qui sont généralement classifiées en quatre catégories :

- Les mesures de corrélation croisée (Aschwanden and Guggenbül [1992], Brown [1992], Garcia [2001]) qui sont basées sur l'utilisation d'un produit scalaire : la corrélation croisée simple (SCC), la corrélation croisée normalisée NCC (Normalized Cross Correlation) et la corrélation croisée centrée normalisée ZNCC (Zero Normalized Cross Correlation).

- Les mesures qui utilisent des outils de statistiques classiques (Aschwanden and Guggenbül [1992], Cox [1995]) en se basant sur la distribution des différences des niveaux de gris des deux fenêtres : les mesures de distance (la somme des valeurs absolues des différences (SAD), la somme des carrés des différences (SSD), etc.), les mesures de variance (la variance des différences (VD), la variance des valeurs absolues des différences élevées à la puissance P (VAD_p), etc.), etc.

- Les mesures qui s'appuient spécifiquement sur les dérivées des niveaux de gris : ces mesures utilisent les champs de vecteur de gradients au lieu de niveaux de gris. On peut citer les mesures de Nack (Nack [1975]), de Nishihara (Nishihara [1984]) et Seitz (Seitz [1989]).

- Les mesures robustes (Zhang [1995], Lan and Mohr [1995]) font intervenir des outils de statistiques robustes pour lutter contre le problème d'occultation. On peut distinguer les mesures partielles, les pseudo-normes et les mesures s'appuyant sur des variances robustes, des M-estimateurs et des R-estimateurs.

Chaque catégorie possède ses propres avantages et inconvénients. Des travaux d'améliorations ont été faits afin de pallier les lacunes de chaque méthode. Nous allons nous limiter à la présentation de la première catégorie qui est la plus couramment employée dans la littérature de la vision par ordinateur à cause de sa simplicité au niveau de l'implémentation logicielle et de sa fiabilité (Aschwanden and Guggenbül [1992], Bindu and Sheshadri [2014]).

La corrélation croisée (SCC)

La fonction de la corrélation croisée (Ferrari et al. [2003]), Di Stefano and Mattoccia [2003], Tsai et al. [2003]) est exprimée par :

$$SCC = \sum_{v=0}^{V_{length}} \sum_{u=0}^{V_{width}} V(u,v).Z(c+u,l+v) \tag{5.23}$$

où $V(u,v)$ est la vignette de référence et $Z(c,l)$ est la zone de recherche. Les valeurs maximales de $SCC(c,l)$ ne sont pas significatives pour la similarité des vignettes, ce qui rend cette mesure inexploitable directement. Donc, il est nécessaire de lui ajouter un facteur de normalisation.

La corrélation croisée normalisée

Après l'apport de l'étape de normalisation, la mesure de la corrélation croisée normalisée (Sun [1997], Zhang and Shan [2001]) est donnée par :

$$NCC = \frac{\sum_{v=0}^{V_{length}} \sum_{u=0}^{V_{width}} V(u,v).Z(c+u,l+v)}{\sqrt{\sum_{v=0}^{V_{length}} \sum_{u=0}^{V_{width}} V^2(u,v).\sum_{v=0}^{V_{length}} \sum_{u=0}^{V_{width}} Z^2(c+u,l+v)}} \tag{5.24}$$

Les valeurs de $NCC(c,l)$ varient maintenant dans l'intervalle $[0,1]$. La valeur maximale ($NCC(c,l) = 1$) correspondant à une ressemblance totale entre les deux vignettes considérées. Bien que cette mesure soit utilisée fréquemment, elle reste sensible aux conditions de prises de vue. Cependant, elle peut être corrigée par la moyenne et normalisée par la variance. Cela améliore la robustesse de $NCC(c,l)$ aux changements de luminosité et aboutit à une nouvelle mesure appelée $ZNCC(c,l)$.

La corrélation croisée centrée normalisée

Si on pose :

$$\overline{V} = \frac{1}{V_{width} \times V_{length}}. \sum_{v=0}^{V_{length}} \sum_{u=0}^{V_{width}} V(u,v) \tag{5.25}$$

$$\overline{Z}(c,l) = \frac{1}{V_{width} \times V_{length}}. \sum_{v=0}^{V_{length}} \sum_{u=0}^{V_{width}} Z(c+u,l+v) \tag{5.26}$$

Alors on a :

$$ZNCC = \frac{\sum_{v=0}^{V_{length}} \sum_{u=0}^{V_{width}} \Big(V(u,v) - \overline{V}\Big).\Big(Z(c+u,l+v) - \overline{Z}(c,l)\Big)}{\sqrt{\sum_{v=0}^{V_{length}} \sum_{u=0}^{V_{width}} \Big(V(u,v) - \overline{V}\Big)^2.\sum_{v=0}^{V_{length}} \sum_{u=0}^{V_{width}} \Big(Z(c+u,l+v) - \overline{Z}(c,l)\Big)^2}} \tag{5.27}$$

Cette mesure est donc centrée dans une plage de variation de $[-1,+1]$, elle est parmi les mesures les plus utilisées dans la littérature (Y. Raghavender Rao [2014], Rao and Prathapani [2014]) car elle résiste aux changements de luminosité et elle permet de sélectionner la vignette candidate ressemblant le plus à la vignette de référence en exploitant simplement les valeurs obtenues.

Le point homologue le plus ressemblant au point de référence correspond au point qui donne le score de corrélation le plus élevé :

$$P^{n,i} = \underset{c,l \in [zone\_recherche]}{\operatorname{argmax}} \Big(ZNCC^{n,i}(c,l) \Big) \tag{5.28}$$

On effectue ensuite un simple filtrage pour ne retenir que les paires ayant un score de corrélation supérieur à un seuil proche de la valeur maximale de corrélation (= 1) parce qu'on considère qu'il y a peu de déformations entre les images et qu'on détermine la valeur par l'analyse de jeux tests réalistes. Ce filtrage sert à éliminer dans une première étape les points mal appariés qui induisent en erreur l'estimation de la transformation géométrique entre les images à recaler.
Bien que la méthode de ZNCC est simple et fiable, un certain nombre d'inconvénients ont également été rapportés : premièrement, ZNCC est sensible au bruit dans les images. Un tel bruit peut entraîner une mauvaise corrélation conduisant à un mauvais appariement. Dans notre première phase de notre travail, ce problème n'a pas été abordé car les images que nous avons acquises ont un bon rapport signal bruit. Deuxièmement, la ZNCC est sensible aux différences d'échelle et de rotation entre les images (Zhao et al. [2006]). Cette limitation ne s'applique pas dans notre cas puisque le mouvement de rotation subi par la caméra est considéré comme petit par hypothèse. Troisièmement, la résolution de la ZNCC est en principe limitée à un pixel entier.

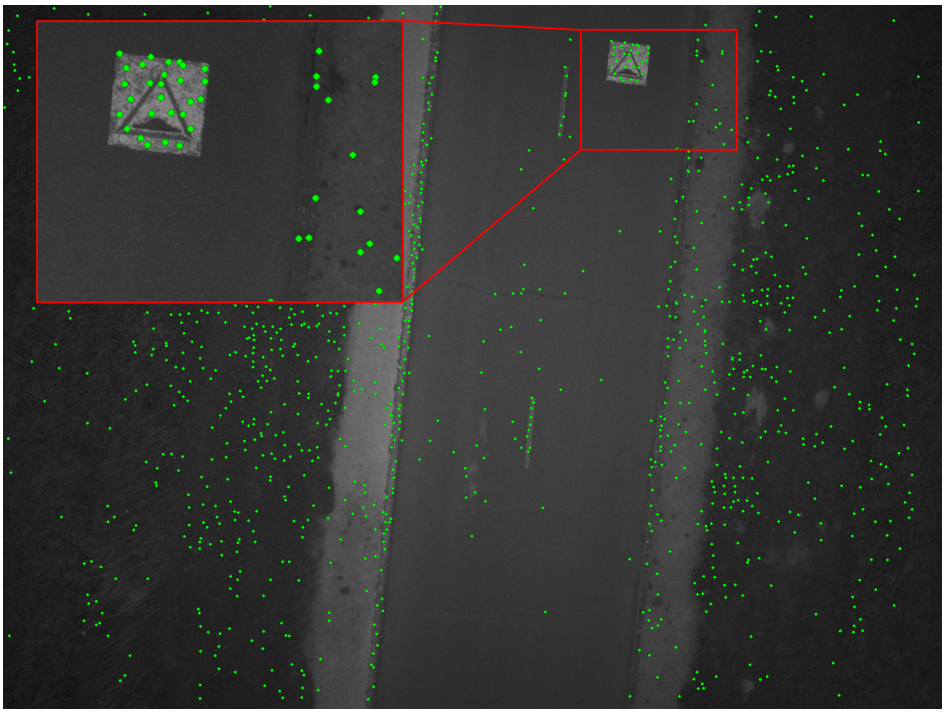
Corrélation subpixellaire

Pour obtenir une précision subpixellaire de la corrélation, il est possible d'interpoler la surface de la corrélation croisée (ZNCC) à une résolution spatiale plus élevée et ensuite localiser le pic de corrélation avec une précision subpixellaire (Huang et al. [1997]). L'interpolation peut être faite en ajustant un modèle bidimensionnel à la surface de la corrélation autour du pic obtenu. On peut calculer précisément la position maximum de ce modèle et l'utiliser comme position subpixellaire du maximum de la fonction de corrélation supposée. Les modèles paraboliques et gaussiens sont souvent utilisés (Nobach and Honkanen [2005]). On a utilisé ici un modèle parabolique simplifié qui traite les axes (x et y) de manière séparée qui est moins coûteux en temps de calcul et on verra plus tard qu'il assure en même temps une précision acceptable.
Supposons que la position entière du pixel correspondant au pic est (x_0, y_0). Ce pixel a deux voisins dans chacune des deux directions orthogonales : $(x_0 - 1, y_0)$ et $(x_0 + 1, y_0)$ dans la direction x et $(x_0, y_0 - 1)$ et $(x_0, y_0 + 1)$ dans la direction y. Pour trouver la position subpixellaire du pic dans chaque direction $(x_0 + \Delta x, y_0 + \Delta y)$, on définit une courbe parabolique qui relie les trois points de chaque direction et ensuite on calcule la position où la courbe atteint son maximum. Les positions non entières du pic dans les directions x et y seront ensuite ajoutées au pixel (nombre entier). On détaille seulement le calcul de Δx. La position Δy sera trouvée de la même façon. L'équation d'une parabole est donnée par :

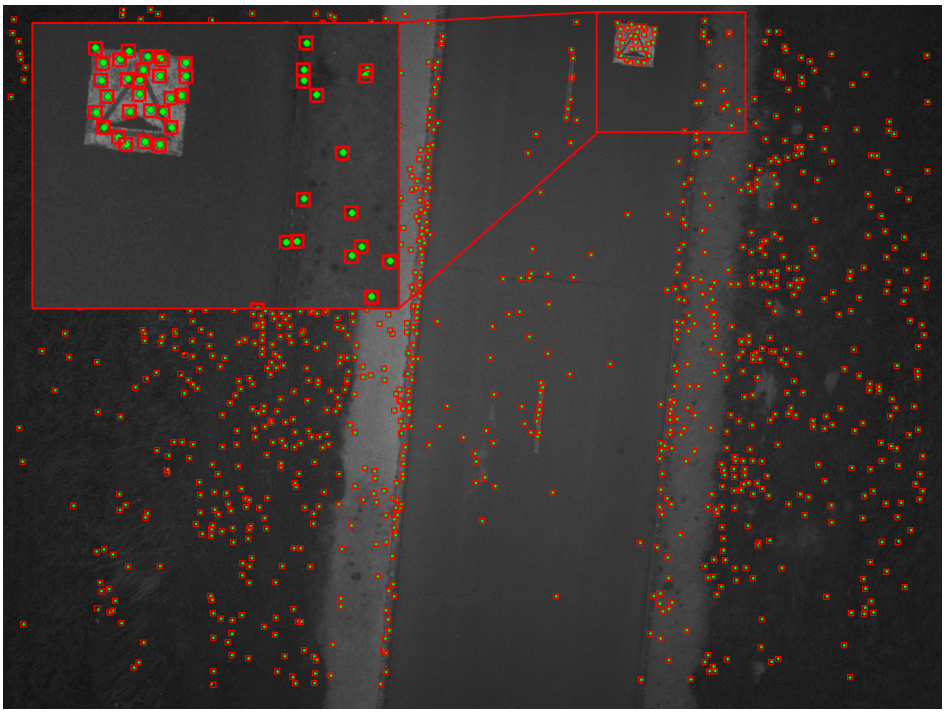
$$f(x) = ax^2 + bx + c \tag{5.29}$$

La position au maximum local (Δx) est déterminée en calculant la dérivée de la fonction f(x) qui s'annule à cette position :

$$\frac{df}{dx}(\Delta x) = 0 \Rightarrow 2a\Delta x + b = 0 \Rightarrow \Delta x = \frac{-b}{2a} \tag{5.30}$$



(a)



(b)

FIGURE 5.9 – (a) : points d'intérêt détectés à l'aide de l'algorithme FAST dans la première image, la valeur du seuil choisi ici est 7. Le nombre de blocs de la grille est configuré à 100×100 ; résolution d'image : 2560×1920 pixels. (b) : les zones de recherche obtenues à l'aide des gyromètres de la centrale inertielle entre la première image et la $10^{ème}$ image. Les points verts représentent les points homologues obtenus par la corrélation ZNCC. La taille de chaque zone de recherche est configurée à 11×11 pixels ; résolution d'image : 2560×1920 pixels.

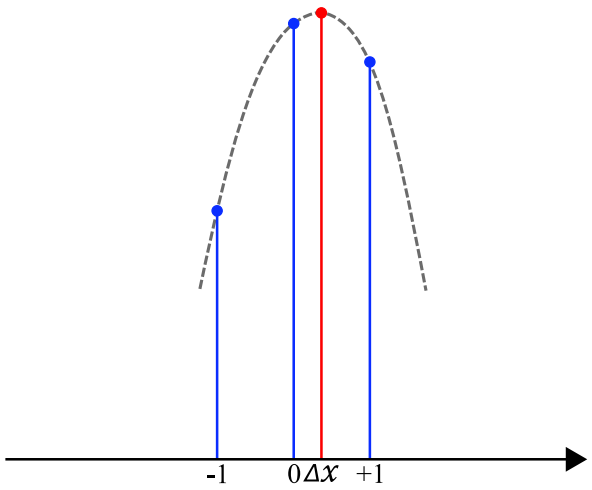


FIGURE 5.10 – Illustration du modèle parabolique défini selon la direction x

On détermine ensuite les paramètres a et b en substituant les valeurs de x_0, $x_0 + 1$, et $x_0 - 1$ dans l'équation de la parabole :

$$\Delta x = \frac{f(x_0 - 1) - f(x_0 + 1)}{2f(x_0 - 1) - 4f(x_0) + 2f(x_0 + 1)} \tag{5.31}$$

Enfin, Δx et Δy peuvent être écrites selon les deux directions x et y de la façon suivante :

$$\Delta x = \frac{ZNCC(x_0 - 1, y_0) - ZNCC(x_0 + 1, y_0)}{2ZNCC(x_0 - 1, y_0) - 4ZNCC(x_0, y_0) + 2ZNCC(x_0 + 1, y_0)} \tag{5.32}$$

Par symétrie, Δy est exprimé par :

$$\Delta y = \frac{ZNCC(x_0, y_0 - 1) - ZNCC(x_0, y_0 + 1)}{2ZNCC(x_0, y_0 - 1) - 4ZNCC(x_0, y_0) + 2ZNCC(x_0, y_0 + 1)} \tag{5.33}$$

Δx et Δy seront ajoutées à la position entière du pic (x_0, y_0).

5.7 Transformation géométrique entre les images

Maintenant que nous avons un ensemble de points homologues, l'étape suivante consiste à établir la meilleure estimation de la relation géométrique qui permet de mapper les coordonnées des pixels d'une image à l'autre en minimisant l'erreur résiduelle entre les points appariés. Le calcul de la meilleure transformation revient à minimiser un critère de distance en utilisant des méthodes de minimisation telles que les méthodes moindres carrés. Le critère qu'on a utilisé est la somme des erreurs de distance quadratiques entre les paires de points appariés.

5.7.1 Choix de la transformation

Deux transformations géométriques sont utilisées ici, la rotation entre les poses et l'homographie entre les images. Notre choix préférentiel sera la rotation puisqu'elle est plus rigide et permettra aussi de recalibrer les gyromètres. On prend l'erreur quadratique moyenne RMS (Root Mean Square) qui caractérise la précision de l'estimation comme facteur de qualité de contrôle. Dans le cas où cette erreur est grande, on utilisera l'homographie pour obtenir les meilleurs résultats dans l'image stackée finale.

5.7.2 Estimation de la rotation

Comme décrit dans le paragraphe 5.6.1, à partir des points homologues $P^{n,i}$, on construit les vecteurs perspectifs $\overrightarrow{V^{n,i}}$ dans leur repère caméra respectif. Puis à partir des couples de vecteurs $(\overrightarrow{V^{1,i}}, \overrightarrow{V^{n,i}})$, on peut estimer la rotation du repère camera. Théoriquement, on n'a besoin que de deux couples (du fait qu'elle ne possède que trois degrés de liberté) pour trouver la solution exacte. Mais en pratique, nous disposons d'un nombre de couples plus important lié à la taille de la grille. Donc, on estime la rotation à l'aide de la méthode de moindres carrés qui va minimiser les distances entre les vecteurs normalisés $\overrightarrow{K^{n,i}} = \frac{\overrightarrow{V^{n,i}}}{||\overrightarrow{V^{n,i}}||}$ et les images des vecteurs de la première image par la rotation recherchée $R^n_{img}(\overrightarrow{K^{1,i}})$:

$$\hat{R}^n_{img}(\overrightarrow{K^{1,i}}) = \underset{R}{\operatorname{argmin}} \sum_{i=1}^{N} ||R(\overrightarrow{K^{1,i}}) - \overrightarrow{K^{n,i}}||^2 \tag{5.34}$$

L'estimation de rotation donnée par l'IMU est utilisée comme solution initiale. Une seule itération est nécessaire pour être suffisamment proche pour que les itérations suivantes puissent servir à éliminer les observations aberrantes venant de fausses corrélations car il ne peut y avoir de points grossièrement faux à cause de la dimension limitée de la zone de recherche dont ils sont issus.

5.7.3 Estimation de l'homographie

Les couples d'images (première image et $N^{ième}$ image) se situent dans deux plans projectifs π_1 et π_n donc ils sont liés entre eux par une transformation linéaire projective appelée "homographie", tel que $P^{n,i}_c = H^n_{img} \times P^{1,i}_c$ avec $P^{1,i}_c \in \pi_1$ et $P^{n,i}_c \in \pi_n$ respectivement les points en correspondance corrigés de la distorsion de coordonnées homogènes $(u^{1,i}_c, v^{1,i}_c, 1)$ et $(u^{n,i}_c, v^{n,i}_c, 1)$ et H^n_{img} la matrice de l'homographie (3×3) contenant 9 éléments dont 8 indépendants (on pose généralement le paramètre $h^n_{33} = 1$ puisque nous considérons que les rotations ont des faibles amplitudes) :

$$H^n_{img} = \begin{bmatrix} h^n_{11} & h^n_{12} & h^n_{13} \\ h^n_{21} & h^n_{22} & h^n_{23} \\ h^n_{31} & h^n_{31} & 1 \end{bmatrix} \tag{5.35}$$

Les images par l'homographie des points $(u^{1,i}_e, v^{1,i}_e)$ sont calculées de la manière suivante :

$$u^{1,i}_e = \frac{u^{1,i}_e h^n_{11} + v^{1,i}_e h^n_{12} + h^n_{13}}{u^{1,i}_e h^n_{31} + v^{1,i}_e h^n_{32} + 1} \qquad v^{1,i}_e = \frac{u^{1,i}_e h^n_{21} + v^{1,i}_e h^n_{22} + h^n_{23}}{u^{1,i}_e h^n_{31} + v^{1,i}_e h^n_{32} + 1} \tag{5.36}$$

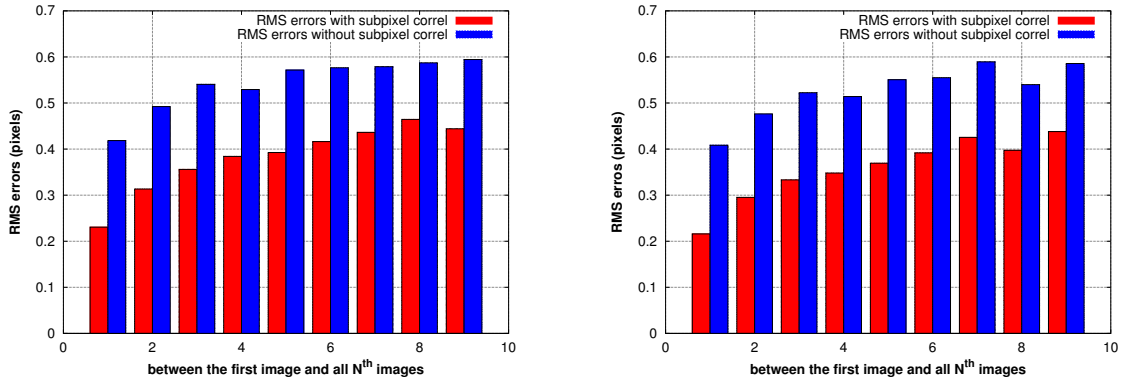
Notre problème revient donc à trouver les paramètres $h_{i,j}$ qui minimisent les distances aux carrés entre les $(u^{1,i}_e, v^{1,i}_e)$ et les $(u^{n,i}_c, v^{n,i}_c)$. Classiquement, ce problème non linéaire est linéarisé en prenant en compte que les rotations sont faibles et nous obtenons les équations d'observations suivantes :

$$\begin{aligned} u^{n,i}_c(u^{1,i}_c h^n_{31} + v^{1,i}_c h^n_{32} + 1) &\approx u^{1,i}_c h^n_{11} + v^{1,i}_c h^n_{12} + h^n_{13} \\ v^{n,i}_c(u^{1,i}_c h^n_{31} + v^{1,i}_c h^n_{32} + 1) &\approx u^{1,i}_c h^n_{21} + v^{1,i}_c h^n_{22} + h^n_{23} \end{aligned} \tag{5.37}$$

Le système pourra être résolu à l'aide de la méthode des moindres carrés en utilisant par exemple la décomposition de Cholesky de la matrice. De la même manière que la rotation, les points aberrants ayant des résidus supérieurs à 3 pixels seront éliminées à l'aide des itérations successives.

5.7.4 Précision de l'estimation subpixellaire

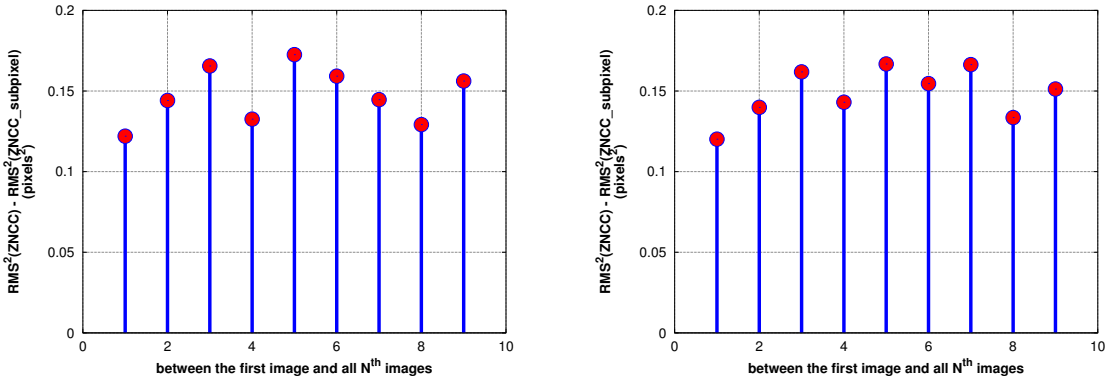
Nous avons maintenant un outil pour évaluer la précision de notre interpolation subpixellaire grâce aux résidus que nous obtenons lors de l'estimation de la transformation géométrique entre les images.
Nos résultats (figure 5.11) montrent une amélioration de la précision lorsque nous utilisons la corrélation subpixellaire dans les deux types de transformation. Les résul-



(a) Les erreurs RMS de la rotation estimée.

(b) Les erreurs RMS de l'homographie estimée.

FIGURE 5.11 – Les résidus de l'estimation de la rotation et de l'homographie.



(a) rotation.

(b) homographie.

FIGURE 5.12 – La différence des erreurs RMS^2 entre la corrélation pixellaire et la corrélation subpixellaire.

tats montrent qu'avec ce modèle simple, l'approche subpixellaire améliore la précision de l'appariement d'image basé sur le $ZNCC$ en éliminant environ de 0,15 $pixels^2$ à la variance des résidus par rapport à la précision initiale. L'arrondi d'une variable aléatoire uniformément répartie engendre une erreur dans l'estimation de sa variance de $\frac{1}{12}$ $pixels^2$. Puisqu'on a deux variables indépendantes x et y, la variance de l'erreur globale sera augmentée de $\frac{1}{6}$ $pixels^2$, soit 0.17.
La figure 5.12 montre que la différence des résidus RMS^2 entre la corrélation pixellaire et la corrélation subpixellaire vaut presque (2/12), ce qui signifie que la corrélation subpixellaire fonctionne comme espéré.

5.8 Ré-échantillonnage et empilement des images

La transformation géométrique estimée $\hat{T}_n^{img}$ relie les coordonnées des points $P^1(u^1, v^1)$ dans la première image aux coordonnées des points correspondants $P^n(u^n, v^n)$ dans la $N^{ième}$ image :

$$P^n = \mathcal{D}(\hat{T}_n^{img}(\mathcal{D}^{-1}(P^1))) \tag{5.38}$$

avec $\hat{T}_n^{img} = \hat{R}_n^{img}$ ou $\hat{H}_n^{img}$. On veut empiler les images dans la géométrie de la

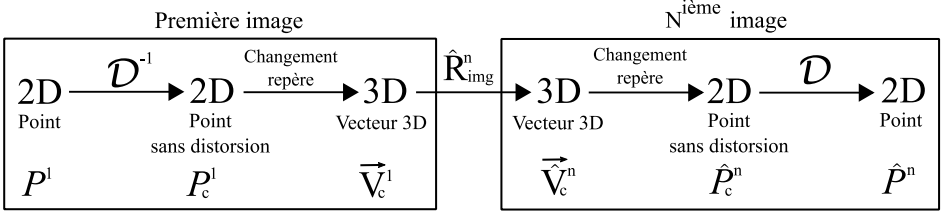


FIGURE 5.13 – Transformation par $\hat{R}_n^{img}$

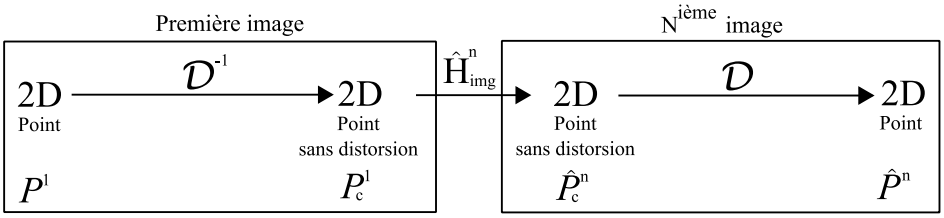


FIGURE 5.14 – Transformation par $\hat{H}_n^{img}$

première, donc pour tous les points P^1 de l'image finale, on doit calculer la moyenne des intensités des images aux points P^n. Bien que les coordonnées (u^1, v^1) soient entières, les coordonnées (u^n, v^n) sont des nombres réels. L'intensité au point (u^n, v^n) doit donc être estimée à partir des intensités des pixels voisins. Différentes méthodes sont utilisées pour réaliser cette estimation. On ne considérera ici que la méthode de ré-échantillonnage au plus proche voisin et la méthode de ré-échantillonnage bilinéaire.
Une fois calculée cette moyenne, dans le cas de faibles radiométries on peut bénéficier de l'amélioration de la dynamique en multipliant cette moyenne par un coefficient d'amplification avant de la passer sur 8 bits et de la stocker dans l'image résultat.

5.8.1 Ré-échantillonnage au plus proche voisin

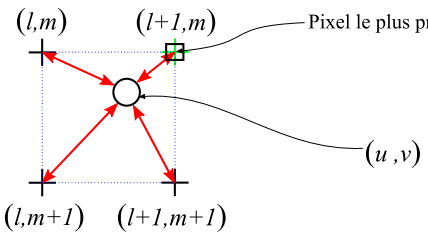


FIGURE 5.15 – Ré-échantillonnage au plus proche voisin.

Le ré-échantillonnage au plus proche voisin consiste donc à prendre le pixel de coordonnées correspondantes aux arrondis des coordonnées des pixels désirés.
On remarque évidemment que la mise en œuvre de cette méthode est très simple et que son calcul est très rapide et ne nécessite qu'un seul accès à la mémoire. La perte de précision devra être évaluée sur des cas concrets.

5.8.2 Ré-échantillonnage bilinéaire

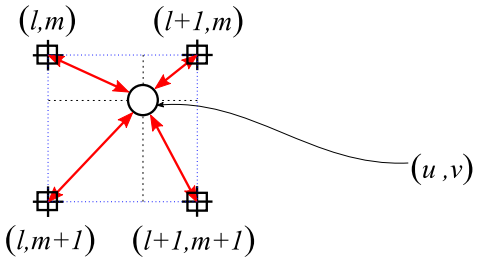


FIGURE 5.16 – Ré-échantillonnage bilinéaire.

Dans l'interpolation bilinéaire, l'intensité au point (u, v) est déterminée à partir de la somme pondérée des intensités des quatre pixels les plus proches. Étant donné les coordonnées (u, v) d'un point correspondant dans la $N^{ième}$ image où u et v sont des nombres réels et supposant que l et m sont respectivement les parties entières de u et v. L'intensité en (u, v) peut être estimée à partir des intensités en (l, m), $(l + 1, m)$, $(l, m + 1)$, et $(l + 1, m + 1)$ en effectuant deux interpolations linéaires successives selon les lignes puis les colonnes. On peut formuler $I(u, v)$ de la façon suivante :

$$I_{(u,v)} = \delta_{(l,m)} I_{(l,m)} + \delta_{(l+1,m)} I_{(l+1,m)} + \delta_{(l,m+1)} I_{(l,m+1)} + \delta_{(l+1,m+1)} I_{(l+1,m+1)} \tag{5.39}$$

où

$$\begin{aligned} \delta_{(l,m)} &= (l + 1 - u)(m + 1 - v) \\ \delta_{(l+1,m)} &= (u - l)(m + 1 - v) \\ \delta_{(l,m+1)} &= (l + 1 - u)(v - m) \\ \delta_{(l+1,m+1)} &= (u - l)(v - m) \end{aligned} \tag{5.40}$$

Cette méthode est très utilisée mais elle est assez complexe en termes de calcul et nécessite 4 accès à la mémoire.

5.8.3 Qualification de la méthode de ré-échantillonnage

En général, le choix d'une de ces deux méthodes dépend fortement des contraintes de qualité et de calcul imposées. Bien que la méthode de l'interpolation bilinéaire soit plus populaire, elle est moins rapide que la méthode au plus proche voisin. Dans notre cas, on s'intéresse à une méthode rapide qui fournisse des résultats satisfaisants.
Afin de comparer de façon précise la qualité des images empilées lorsqu'on utilise ces deux méthodes de ré-échantillonnage, nous avons calculé les profils radiométriques correspondants en visualisant la valeur radiométrique des pixels d'une partie d'une ligne de l'image. La figure 5.17 et le tableau 5.2 montrent que, sans que l'on puisse dire quelle est la méthode la meilleure, les résultats sont suffisamment proches pour que l'on préfère la plus simple.

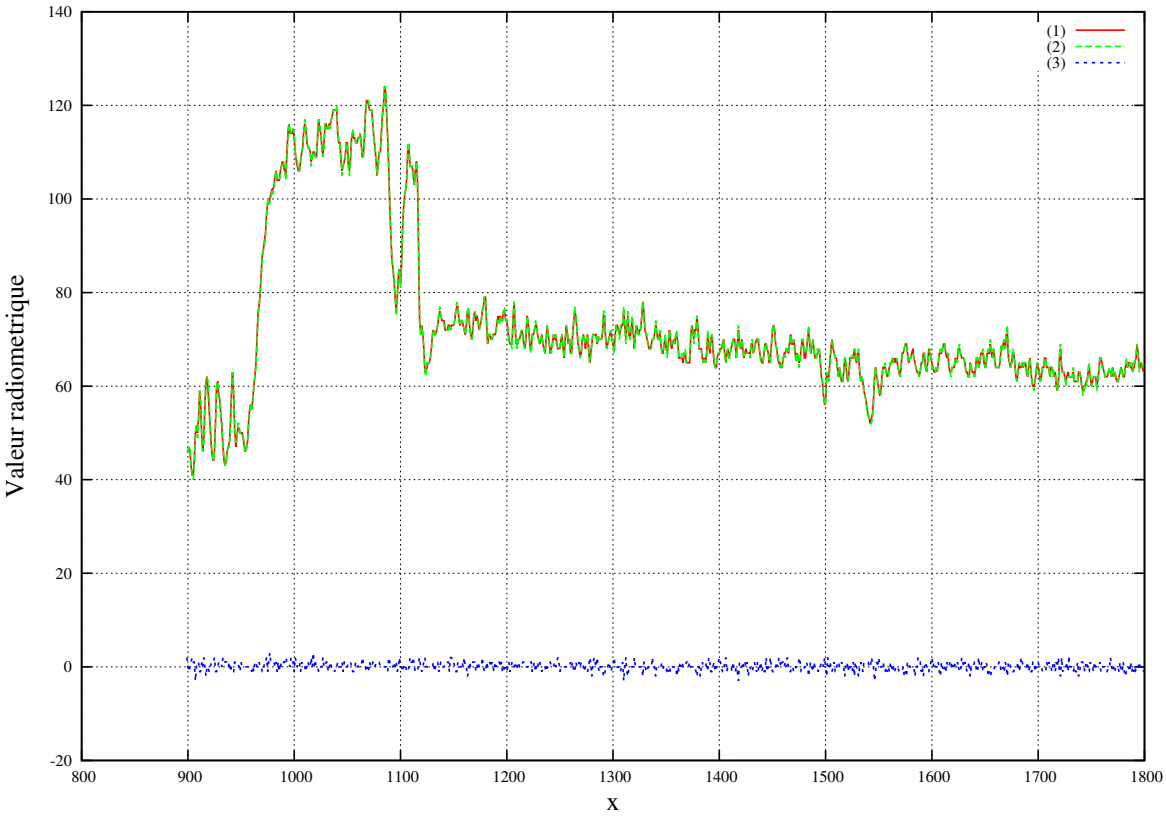


FIGURE 5.17 – Le profil radiométrique d'une partie d'un ligne à partir d'une des séquences d'images. (1 - rouge) : Image stackée obtenue avec l'interpolation bilinéaire. (2 - vert) : Image stackée obtenue avec le ré-échantillonnage au plus proche voisin. (3 - bleu) : Différence entre les deux.

Tests	Test 1	Test 2	Test 3	Test 4
Écart radiométrique maximal	2	3	2	5
Moyenne	-0.005	0.008	-0.007	-0.008
Variance	0.290	0.293	0.268	0.285

TABLEAU 5.2 – Statistiques de 4 jeux de données sur les résidus radiométriques entre l'image stackée obtenue par la méthode d'interpolation bilinéaire et l'image stackée obtenue en utilisant le ré-échantillonnage au plus proche voisin. Les statistiques sont réalisées sur l'image entière.

5.9 Accélération logicielle de la partie ré-échantillonnage

Les résultats du temps d'exécution présentés dans le tableau 6.1 montrent que cette phase du traitement reste très coûteuse en temps (environ 18 secondes en utilisant les 2 cœurs du processeur ARM). Si on revient sur son aspect algorithmique, on remarque que le processus de l'application de la transformation géométrique estimée $\hat{T}^n_{img}$ sur chaque pixel de chaque image de la séquence $P^n = \mathcal{D}(\hat{T}^n_{img}(\mathcal{D}^{-1}(P^1)))$ est la cause principale de ce temps (figure 5.13, figure 5.14), ce processus contenant à la fois les calculs liés à la distorsion et les calculs liés à l'application de la rotation ou de l'homographie. A noter qu'à cause du grand nombre de pixels à traiter, le temps utilisé pour les accès mémoire est aussi non-négligeable.
Tout d'abord, nous avons pensé à accélérer cette phase de traitement directement dans

le FPGA. Mais comme le nombre de multiplications effectuées dans l'ensemble de la transformation géométrique est assez élevé, cela nous obligerait à utiliser beaucoup de ressources dans le FPGA surtout des tranches DSP. Nous avons donc envisagé d'utiliser une nouvelle approche algorithmique qui va permettre d'accélérer déjà en logiciel cette phase de l'algorithme.

5.9.1 Approximation bilinéaire de la transformation entre les images

Notre nouvelle approche se concentre essentiellement sur la réduction d'utilisation du processus de la transformation géométrique (correction distorsion, application de la transformation géométrique, application de la distorsion) sur l'ensemble de l'image. Notre approche consiste à diviser la première image en une grille de $K \times L$ blocs et à n'appliquer la transformation géométrique que sur les pixels sommets de chaque bloc afin de trouver leurs correspondants dans toutes les autres images. Ensuite, les correspondants des autres pixels à l'intérieur de chaque bloc seront calculés à l'aide d'une interpolation bilinéaire. Tout cela permet d'éviter ainsi un grand nombre de multiplications.
Supposons que P_1, P_2, P_3, P_4 sont les points sommets du bloc traité de la première image, P_1', P_2', P_3', P_4' sont les points projetés dans la $N^{ième}$ image obtenus à l'aide de la transformation géométrique $\hat{T}^n_{img} = (\hat{R}^n_{img}$ ou $\hat{H}^n_{img})$ (figure 5.18). Les huit paramètres

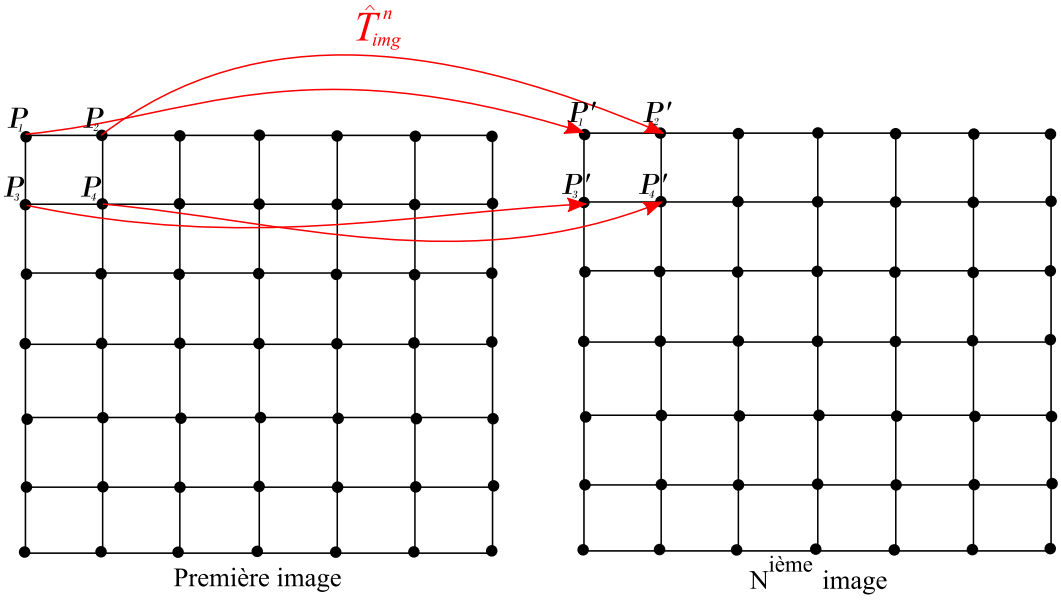


FIGURE 5.18 – L'approche d'interpolation bilinéaire.

de l'interpolation bilinéaire $(a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y \in \Re)$ sont obtenus en résolvant sur les quatre couples P, P' le système d'équations suivant :

$$\begin{cases} x' = a_x + b_x x + c_x y + d_x xy \\ y' = a_y + b_y x + c_y y + d_y xy \end{cases} \quad (6.5)$$

Une fois que les paramètres de l'approximation bilinéaire sont calculés, la position des pixels à l'intérieur du bloc peut être calculée en évitant d'utiliser un grand nombre de multiplications.

Le système d'équations (équation 6.5) peut être largement optimisé de telle sorte qu'il n'est pas nécessaire de faire toutes ces opérations (trois additions et quatre multiplications) pour chaque pixel dans le bloc.
Tout d'abord, $x'_{l,0}$ peut être calculé au début de chaque ligne en utilisant une addition et une multiplication, puis $x'_{l,c}$ sera calculé pour chaque colonne en utilisant seulement une addition (équation 6.6).

$$\begin{cases} x'_{l,c} = x'_{l,c-1} + k_{x_l} \\ x'_{l,0} = x'_{l-1,0} + c_x \\ x'_{0,0} = a_x \\ k_{x_l} = k_{x_{l-1}} + d_x \\ k_{x_0} = b_x \end{cases} \tag{6.6}$$

où $c \in$ [1..LARGEUR_BLOC[avec LARGEUR_BLOC $= \frac{LARGEUR\_IMAGE}{K}$
et $l \in$ [1..HAUTEUR_BLOC[avec HAUTEUR_BLOC $= \frac{HAUTEUR\_IMAGE}{L}$
$y'_{l,c}$ est calculé de la même manière que $x'_{l,c}$ mutatis mutandis.
La valeur radiométrique de chaque pixel dans les autres images sera alors obtenue à l'aide du ré-échantillonnage au plus proche voisin.
Le pseudo-code illustré dans le listing 5.1 représente la version logicielle optimisée de la partie du ré-échantillonnage d'un bloc de l'image.

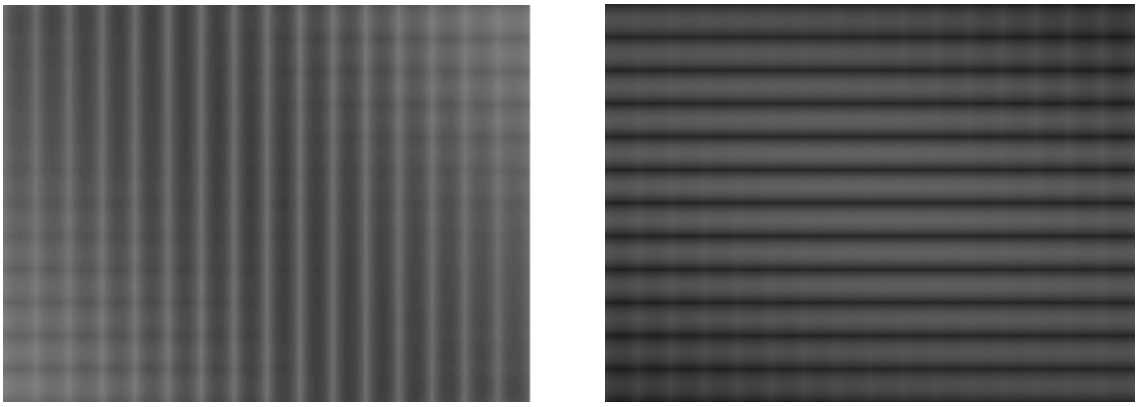
Listing 5.1 – Optimisation logicielle de la partie ré-échantillonnage sur un bloc.

```
for(j=0; j<TAILLE_BLOC_Y; j++){
   for(k=0; k<NB_IMAGES; k++){
      x_k' = ax_k;
      y_k' = ay_k;
   }
   for(i=0; i<TAILLE_BLOC_X; i++){
      cnt_pixel_ok = 0;
      sum_pixels = 0;
      for(k=0; k<NB_IMAGES; k++){
         (I_y, I_x) = NearestNeihbour(y_k, x_k);
         if(IsInImage(I_y, I_x)){
                sum_pixels += Image_k(I_y, I_x);
                cnt_pixel_ok++;
            }
         x_k' += bx_k;
         y_k' += by_k;
      }
      stacked_image(j,i) = sum_pixels/cnt_pixel_ok;
   }
   for(k=0; k<NB_IMAGES; k++){
      ax_k += cx_k;
      ay_k += cy_k;
      bx_k += dx_k;
      by_k += dy_k;
   }
}
```

5.9.2 Précision de la méthode accélérée

Afin de pouvoir synthétiser la qualité et la précision de la méthode accélérée, nous avons réalisé quelques tests de précision sur les images empilées obtenues. Nous présenterons

ici quelques résultats obtenus pour une taille de bloc configurée à 160×160.
La figure 5.19 représente la distance selon x et y entre le point projeté par la transformation géométrique et celui projeté par la fonction bilinéaire. On remarque que cette distance est continue, en particulier d'un bloc à un autre selon les deux axes x et y dans l'image. Cela assure qu'il n'y aura pas de sauts radiométriques dans les images interpolées au niveau des limites entre les blocs.



(a) $Diff_{max} = 0.0262$ - selon y (b) $Diff_{max} = 0.0248$ - selon x

FIGURE 5.19 – Différence en pixels entre les points projetés obtenus par les deux méthodes pour la dernière paire d'images. Taille de bloc : 160×160.

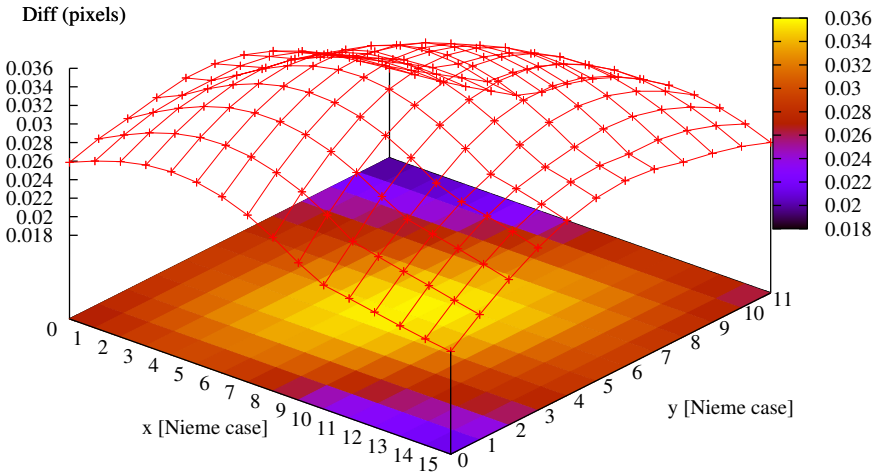
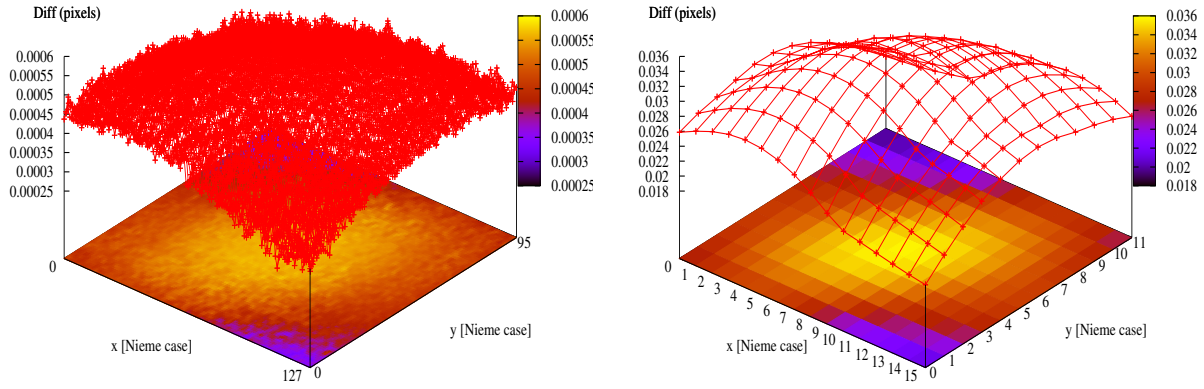


FIGURE 5.20 – Distance maximale (en pixels) entre le point projeté par la transformation géométrique et le point projeté par la fonction bilinéaire dans chaque bloc (160×160) de la grille de l'image.

D'un autre côté, la figure 5.20 illustre la distance géométrique euclidienne maximale (en pixels) entre le point projeté par la transformation géométrique et le point projeté par la fonction bilinéaire dans chaque bloc de la grille de l'image. On remarque que cette distance maximale est très petite et ne dépasse pas les 3 centièmes de pixels.

Les résultats montrent que la perte de précision qui peut avoir lieu en utilisant la méthode de ré-échantillonnage accélérée est négligeable dans la totalité de l'image, ce qui nous garantit par la suite la précision globale de la méthode de stacking. Le choix de la taille du bloc sera jugée dans le prochain paragraphe.

5.9.3 Choix de la taille des blocs de la grille



(a) $Dist_{max} = 0.0005\ px.\ \ Taille\_bloc = (20 \times 20)$ (b) $Dist_{max} = 0.035\ px.\ \ Taille\_bloc = (160 \times 160)$

FIGURE 5.21 – Différence en pixels entre les points projetés obtenus par les deux méthodes pour la dernière paire d'images.

Dans notre approche, on a besoin de qualifier la taille optimale à utiliser pour les blocs de la grille sur des critères de temps et de précision. Donc pour cette raison, on a réalisé plusieurs tests en utilisant différentes tailles de blocs (figure 5.21), (tableau 5.3).
Les résultats montrent que la distance maximale en pixels entre les deux points projetés

T_{bloc}	20×20	40×40	64×64	128×128	160×160	320×320	640×640
Dist	0.0005	0.002	0.005	0.022	0.035	0.14	0.55

TABLEAU 5.3 – *Distance maximale en pixels entre les points projetés obtenus par les deux méthodes pour la dernière paire d'images.*

augmente avec la taille de blocs, cela veut dire évidement que la qualité de l'image empilée dépend forcément de la taille de blocs. En plus, le temps du calcul de traitement en dépend aussi surtout à cause du calcul des sommets projetés de chaque bloc par la transformation géométrique. Afin de faire un compromis entre la précision et la rapidité, nous avons choisi finalement 160×160 comme taille de bloc.

5.10 Conclusion

La version logicielle accélérée de notre algorithme (3.56 secondes en utilisant les 2 cœurs du processeur ARM) peut être déjà utilisée dans certains types de missions qui

ne nécessitent pas un traitement très court de calcul. Cependant, il sera intéressant de bénéficier de la présence du FPGA pour accélérer certaines phases de l'algorithme.

5.11 Références

Aschwanden, P. and Guggenbül, W. (1992). Experimental results from a comparative study on correlation type registration algorithms. In *Robust computer vision : Quality of Vision Algorithms*, pages 268–282. Wichmann, Karlsruhe, Allemagne. 56

Baker, S., Nayar, S. K., and Murase, H. (1998). Parametric feature detection. *International Journal of Computer Vision*, 27(1) :27–50. 43

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). *SURF : Speeded Up Robust Features*, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg. 46

Behrens, A. and Röllinger, H. (2010). Analysis of feature point distributions for fast image mosaicking algorithms. *Acta Polytechnica Journal of Advanced Engineering*, 50(4) :12–18. 50

Bindu, N. and Sheshadri, H. (2014). An evaluation of correlation based stereo matching algorithms by considering various parameters. In *Proceedings of International Conference on Recent Trends in Signal Processing, Image Processing and VLSI, Bangalore*. 56

Brown, L. G. (1992). A survey of image registration techniques. *ACM Comput. Surv.*, 24(4) :325–376. 56

Brown, M. Z., Burschka, D., and Hager, G. D. (2003). Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8) :993–1008. 52

Chambon, S. and Crouzil, A. (2003). Dense matching using correlation : new measures that are robust near occlusions. In *Proceedings of British Machine Vision Conference (BMVC'2003), East Anglia*, pages 143–152. 56

Cox, G. S. (1995). Template matching and measures of match in image processing. 56

Di Stefano, L. and Mattoccia, S. (2003). Fast template matching using bounded partial correlation. *Machine Vision and Applications*, 13(4) :213–221. 57

Ferrari, V., Tuytelaars, T., and Gool, L. V. (2003). Wide-baseline multiple-view correspondences. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, pages I–718–I–725 vol.1. 57

Forstmann, S., Kanou, Y., Ohya, J., Thuering, S., and Schmitt, A. (2004). *Real-time stereo by using dynamic programming*, volume 2004-January. IEEE Computer Society, january edition. 52

Garcia, D. (2001). *Measurement of 3-D Shapes or 3-D Displacement Fields Using a Stereo-correlation Based Method*. Theses, Institut National Polytechnique de Toulouse - INPT. 56

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151. 43, 44

Heo, Y. S., Lee, K. M., and Lee, S. U. (2011). Robust stereo matching using adaptive normalized cross-correlation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4) :807–822. 52

Horaud, R., Veillon, F., and Skordas, T. (1990). *Finding geometric and relational structures in an image*, pages 374–384. Springer Berlin Heidelberg, Berlin, Heidelberg. 43

Huang, H., Dabiri, D., and Gharib, M. (1997). On errors of digital particle image velocimetry. *Measurement Science and Technology*, (12) :1427. 58

Lan, Z.-D. and Mohr, R. (1995). Robust Matching by Partial Correlation. Research Report RR-2643, INRIA. 56

Le, X. and Gonzalez, R. (2009). Pattern-based corner detection algorithm. In *Image and Signal Processing and Analysis, 2009. ISPA 2009. Proceedings of 6th International Symposium on*, pages 238–243. 49

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA. IEEE Computer Society. 45

Medioni, G. and Yasumoto, Y. (1987). Corner detection and curve representation using cubic b-splines. *Computer Vision, Graphics, and Image Processing*, 39(3) :267 – 278. 43

Mokhtarian, F. and Suomela, R. (1998). Robust image corner detection through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12) :1376–1381. 43

Moravec, H. (1977). Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*. 43

Nack, M. (1975). Temporal registration of multispectral digital satellite images using their edge images. In *AAS/AIAA/Astrodynamics Specialist Conference*, Nassau, Bahamas. Papier AAS75–104. 56

Nishihara, H. (1984). PRISM, a pratical real-time imaging stereo matcher. Rapport de recherche A. I. Memo 780, Massachusetts Institute of Technology, MIT, États-Unis. 56

Nobach, H. and Honkanen, M. (2005). Two-dimensional gaussian regression for subpixel displacement estimation in particle image velocimetry or particle position estimation in particle tracking velocimetry. *Experiments in Fluids*, pages 511–515. 58

Pena, M. (2012). *A Comparative Study of Three Image Matching Algorithms : Sift, Surf, and Fast*. BiblioBazaar. 49

Rao, Y. R. and Prathapani, N. (2014). Application of normalized cross correlation to image registration. 57

Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, pages 1508–1515, Washington, DC, USA. IEEE Computer Society. 48

Rosten, E. and Drummond, T. (2006). *Machine Learning for High-Speed Corner Detection*, pages 430–443. Springer Berlin Heidelberg, Berlin, Heidelberg. 48, 49

Schauwecker, K., Klette, R., and Zell, A. (2012). A new feature detector and stereo matching method for accurate high-performance sparse stereo matching. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5171–5176. 50

Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors. *Int. J. Comput. Vision*, 37(2) :151–172. 42, 45, 49

Seitz, P. (1989). Using local orientational information as image primitive for robust object recognition. pages 1630–1639. 56

Shilat, F., Werman, M., and Gdalyahn, Y. (1997). Ridge's corner detection and correspondence. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 976–981. 43

Smith, S. M. and Brady, J. M. (1997). Susan—a new approach to low level image processing. *International Journal of Computer Vision*, 23(1) :45–78. 47, 48

Sun, C. (1997). A fast stereo matching method. *proceedings of Digital Image Computing : Techniques and Applications*. 57

Tsai, D.-M., Lin, C.-T., and Chen, J.-F. (2003). The evaluation of normalized cross correlations for defect detection. *Pattern Recogn. Lett.*, pages 2525–2535. 57

Y. Raghavender Rao, Nikhil Prathapani, E. (2014). Application of normalized cross correlation to image registration. 57

Zhang, Z. (1995). Parameter Estimation Techniques : A Tutorial with Application to Conic Fitting. Research Report RR-2676, INRIA. 56

Zhang, Z. and Shan, Y. (2001). A progressive scheme for stereo matching. In *Revised Papers from Second European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, SMILE '00, pages 68–85, London, UK, UK. Springer-Verlag. 57

Zhao, F., Huang, Q., and Gao, W. (2006). Image matching by normalized cross-correlation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 2, pages II–II. 58

Zou, L., Chen, J., Zhang, J., and Dou, L. (2008). The comparison of two typical corner detection algorithms. In *2008 Second International Symposium on Intelligent Information Technology Application*, volume 2, pages 211–215. 49

Zureiki, A., Devy, M., and Chatila, R. (2007). Stereo matching using reduced-graph cuts. In *2007 IEEE International Conference on Image Processing*, volume 1, pages I – 237–I – 240. 52

Chapitre 6

Implémentation en co-design de l'algorithme dans la caméra

Sommaire

6.1 Introduction

La présence dans la caméra d'un SoC qui est constitué de deux 2 CPUs et d'un FPGA nous donne une possibilité d'accélérer encore plus notre traitement en déléguant une partie des calculs au FPGA suivant les méthodes de co-design.
Nous choisissons dans ce chapitre les parties de l'algorithme à accélérer et nous présenterons les implémentations que nous avons retenues et les gains apportés.

6.1.1 Le co-design

Le co-design (ou partitionnement logiciel/matériel) est une méthodologie de conception qui consiste à concevoir des modules logiciels et matériels qui travaillent en harmonie et en parallèle afin d'obtenir les performances finales souhaitées :

- La partie logicielle est constituée d'un programme exécutable sur un processeur. Dans notre cas, le processeur est de type ARM Cortex A9 à deux cœurs avec une fréquence de 600Mhz.
- La partie matérielle est constituée de la configuration d'un ensemble d'éléments logiques programmables. Elle est composée d'un certain nombre de modules dédiés chacun à une fonction appelée communément IP (Intellectual Property).
- La partie d'interfaçage entre le matériel et le logiciel est constituée essentiellement d'un ensemble de pilotes.

Le partitionnement entre le matériel et le logiciel devra être dicté par les caractéristiques de deux types de dispositifs, mais le développement sur le FPGA étant moins commode que sur le processeur, c'est souvent des raisons plus pragmatiques qui dicteront notre démarche.

6.1.2 Identification des parties coûteuses en temps de calcul

Pour identifier les parties qui consomment le plus de ressources CPU dans notre algorithme, nous avons fait afficher par le programme le temps pris par chaque phase. Nous avons également utilisé un outil de profilage (callgrind).

	Détection points d'intérêt (85 points)	**Appariement (85 points) & Estimation de la transformation**	**Ré-échantillonnage**			
			(1)	**(2)**	**(3)**	**(4)**
PC (Soft)	0.21	0.017	4.72	4.00	0.94	0.66
PC (Soft/2 Threads)	0.10	0.008	2.46	2.06	0.42	0.38
Camera (Soft)	1.64	0.17	36.94	25.17	7.24	5.28
Camera (Soft/2 Threads)	0.83	0.08	18.48	12.57	3.81	2.65

TABLEAU 6.1 – Temps d'exécution en secondes des différentes phases du traitement. Tous les tests ont été exécutés sur deux plates-formes : premièrement, une station de travail HPZ210 avec un processeur Intel Xeon 3.30 GHz et 8,0 Go de RAM. Deuxièmement, la caméra IGN photogrammétrique avec un processeur ARM Cortex-A9 à double cœur à 666 MHz. (1) : transformation géométrique à l'aide de la rotation, (2) : transformation géométrique à l'aide de l'homographie, (3) : transformation géométrique accélérée à l'aide de la fonction bilineaire et l'interpolation bilinéaire, (4) : transformation géométrique accélérée à l'aide de la fonction bilineaire et le ré-échantillonnage au plus proche voisin.

Le tableau 6.1 montre que la partie ré-échantillonnage est exigeante dans la caméra quelque soit le type de transformation même si on utilise la version logicielle accélérée,

ce qui limite son utilisation en temps-réel dans le cas de certaines applications. Le temps de calcul de la partie détection de points d'intérêt n'est pas lui non plus négligeable. Bénéficiant de la présence d'un processeur à deux cœurs dans la caméra, nous avons utilisé la bibliothèque OpenMP pour accélérer les différentes parties de l'algorithme en logiciel, le temps de traitement a été réduit de moitié pour un nombre de threads égal à deux.

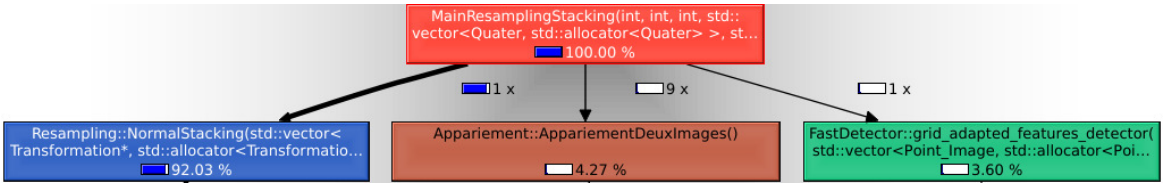


FIGURE 6.1 – Profilage de l'algorithme de stacking : la partie détection de points d'interets prend 3.60%, la partie d'appariement prend 4.27% et la partie ré-échantillonnage prend 92.03%.

Le profilage (figure 6.1) confirme que la partie ré-échantillonnage s'avère très coûteuse. En effet, un peu plus de 90% du temps de calcul est consommé dans cette phase. Cette phase est donc clairement candidate à une implantation dans le matériel. Toutefois, c'est par la détection et la répartition des points d'intérêt que nous avons commencé notre travail parce que cela nous a paru plus simple surtout pour les accès à la mémoire, bien qu'elle ne consomme pas beaucoup de ressources.
La partie d'appariement ne présente pas d'intérêt à être implémentée dans le FPGA car son temps d'exécution est négligeable relativement au reste.

6.2 Les FPGAs

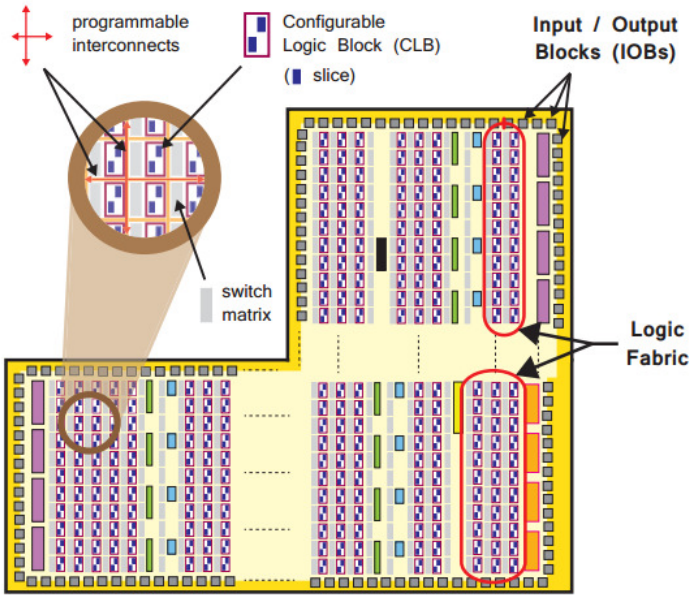


FIGURE 6.2 – Les différents éléments constituant un FPGA. Source : the zynq book.

Les FPGAs sont des circuits intégrés reprogrammables constitués essentiellement

d'une matrice de cellules logiques configurables CLBs (Configurable Logic Block) reliées entre elles par un réseau d'interconnexions configurable. Chaque bloc CLB est composé à son tour de deux parties : des bascules (flip-flops) et des tables de correspondance LUTs (Look-Up Tables). Ceci permet d'implémenter des fonctions séquentielles et/ou combinatoires. Il existe aussi dans le réseau logique des blocs entrées-sorties pour que le circuit implanté puisse s'interfacer avec le monde extérieur. En plus, certains types de FPGAs comprennent deux types de composants spéciaux : des blocs mémoires RAM et des tranches DSP adaptées aux calculs arithmétiques.
Le FPGA est une puce qui est parfaitement adaptée aux traitements en pipeline sur des flux de données. Elle permet aussi d'instancier plusieurs fois les IPs utilisées de manière à paralléliser les traitements.

6.2.1 Le SoC FPGA ZYNQ-7030 de la société Xilinx®

Le ZYNQ-7030 de la famille ZYNQ-7000 est basé sur une architecture de "système sur puce". Il contient deux unités principales : une unité de traitement logicielle (PS) contenant un processeur double cœur de type ARM Cortex-A9 avec ses périphériques et une unité de logique programmable (PL) constituée principalement d'un FPGA de la génération Kintex-7® .

Vivado IDE

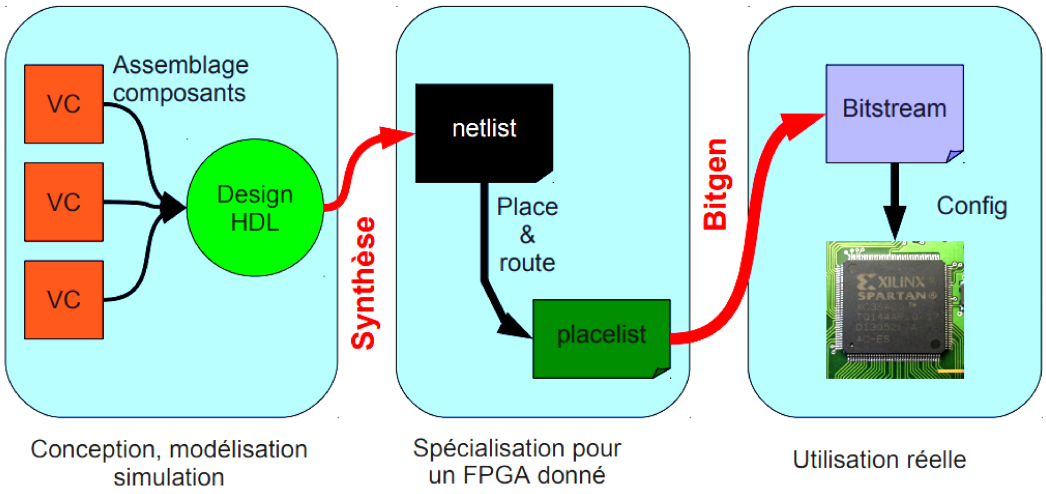


FIGURE 6.3 – Chaîne de développement. Source : documents de Xilinx.

Xilinx fournit aux clients un environnement de développement appelé Vivado IDE afin de réaliser les nombreuses étapes nécessaires pour la conception d'un circuit numérique intégré (figure 6.3). Ce logiciel supporte les deux langages de description matérielle VHDL (VHSIC Hardware Description Language) et Verilog. Dans notre travail, nous avons utilisé le VHDL qui reste le plus populaire en Europe. Vivado nous permet dans un premier temps de modéliser/coder notre architecture électronique sous la forme d'une description structurelle, comportementale ou flot de données. Ensuite, le synthétiseur associé convertit le code en une liste (netlist) constituée de portes et de leurs interconnexions qui peut être modélisée en RTL (Register Transfert Level). Finalement, Vivado prend en charge l'implémentation (placement et routage) du circuit et

génère le fichier bitstream correspondant pour la configuration du FPGA.
Pour déboguer et valider la fonctionnalité du design, il est possible d'effectuer différents types de simulations avec Vivado : fonctionnelle, post-synthèse et post-implémentation.

VivadoHLS® Design Suite HLx Editions

En parallèle avec les langages de description matérielle (VHDL, Verilog), il est possible de modéliser le système sous la forme d'une description algorithmique séquentielle écrite en un langage haut niveau type C, C++ ou SystemC. Le code sera converti en modèle RTL à l'aide d'un outil qui s'appelle HLS (High Level Synthesis). Ces outils permettent de simplifier le développement d'un système complexe en offrant une synthèse haut niveau. Le code produit reste généralement moins optimisé que celui qui pourrait être produit par un programmeur VHDL expérimenté. Cependant, un grand nombre d'optimisations peuvent être faites au niveau du code séquentiel afin de réduire au maximum la latence et les ressources utilisées.
Xilinx fournit le logiciel VivadoHLS qui supporte les langages haut niveau comme C/C++ et systemC. Il est possible de générer le module sous la forme d'une IP qui pourra être utilisée dans un système global. Dans notre travail, nous avons utilisé le langage C++ pour implémenter la partie ré-échantillonnage.

6.3 Accélération matérielle de la partie détection et répartition des points d'intérêts

Dans cette section, nous allons tout d'abord parler des implémentations matérielles du détecteur FAST que nous avons trouvées dans la littérature, puis nous allons détailler notre propre implémentation du détecteur et de l'approche de répartition de points d'intérêt dans l'image.

6.3.1 État de l'art des implémentations matérielles du détecteur FAST

Il existe dans la littérature deux implémentations matérielles de FAST qui sont très proches l'une de l'autre. (Kraft et al. [2008]) ne testent pas seulement si les pixels candidats sont des points d'intérêt, mais aussi calculent un score spécifique (5.17) pour chaque point détecté afin de l'utiliser dans un autre module appelé NMS (Non-Maximal Suppression) qui sert à réduire le nombre de points d'intérêt adjacents. L'ajout de ce module augmente considérablement le nombre de ressources utilisées dans cette implémentation. D'un autre coté, (Dinh et al. [2014]) proposent une autre approche pour la réduction du nombre de points d'intérêt en découpant l'image en une grille de blocs et en ne prenant par bloc que le point d'intérêt ayant le meilleur score.
L'architecture matérielle proposée par (Kraft et al. [2008]) utilise 6 blocs mémoires RAM comme étant des tampons FIFO dont la longueur est égale à la largeur de l'image, afin d'avoir un accès simultané aux pixels examinés (les 16 pixels situés dans le cercle de Bresenham et le pixel central). En plus, 49 (7×7) registres de 8 bits sont utilisés pour stocker les valeurs de pixels de la fenêtre candidate. L'architecture est constituée en global d'une chaîne de modules : le module de threshold, le module de contiguïté, le module de score et le module qui sélectionne le maximum local (Non Maximum Suppression). Ce module consomme le plus de ressources car il nécessite plusieurs

étages de pipeline pour son exécution. Enfin, le module NMS utilise 4 blocs mémoires RAM comme tampons FIFO pour avoir un accès simultané à une fenêtre de score de taille 5×5 au voisinage du pixel candidat.
Cette architecture utilise au total 12 blocs RAMs, 1547 flipflops et 2368 LUTs dans le cas des images de résolution 512×512. La profondeur de pipeline de calcul de cette architecture est égale à 11. Elle est capable de traiter un pixel par cycle.
D'une manière similaire, (Dinh et al. [2014]) mettent en place une architecture de détection de points proche de celle qui est utilisée dans (Kraft et al. [2008]). Cependant, afin de réduire l'apparition de points d'intérêt adjacents, ils n'utilisent pas la fonction de sélection de maximum local (NMS). Mais, ils divisent l'image en $m \times n$ blocs et seul le point d'intérêt qui a le score le plus grand dans chaque bloc est pris en compte. Cette méthode utilise moins de ressources mémoire que le module NMS dans (Kraft et al. [2008]). Le nombre de scores stockés ne dépend que du nombre de colonnes de blocs. Cette architecture utilise au total 5 blocs RAMs, 1247 flipflops et 1575 LUTs dans le cas des images de résolution 640×480. La profondeur de pipeline de calcul de cette architecture est égale à 6.
Dinh et al. [2014] utilisent 21% moins de ressources que (Kraft et al. [2008]). En plus, le nombre de points d'intérêt détectés est réduit (88%) par rapport à ceux qui sont obtenus avec l'architecture de (Kraft et al. [2008]). D'un autre coté, la latence de pipeline ainsi que la latence totale dans (Dinh et al. [2014]) sont moins importantes que celles dans (Kraft et al. [2008]).

6.3.2 Architecture matérielle

Avant de détailler notre architecture, il est nécessaire de lister les caractéristiques techniques que nous avons choisies pour cette conception :

- Les dimensions d'image et le seuil sont configurés par le logiciel.
- L'intensité des pixels d'entrée est de 8 bits (niveaux de gris entre 0 et 255).
- Le débit d'entrée est constant à 1 pixel/cycle. L'image est lue à partir de la mémoire.
- Les coordonnées (x, y) des points d'intérêt (2×16 bits) sont écrites dans la mémoire.

Notre architecture est constituée essentiellement de deux modules : un module "détecteur FAST" qui va permettre de détecter les points d'intérêt au fur et à mesure dans l'image et un module "sélecteur" destiné à réduire et à améliorer l'uniformité de la distribution des points détectés.
Le module "détecteur FAST" signale au module "sélecteur" les points d'intérêt via un drapeau de validité. Ensuite, le module "sélecteur" ne conserve que le premier point d'intérêt détecté dans chaque bloc.
Le module "détecteur FAST" est divisé en deux sous-modules : le module "fenêtre (7×7)" et le module "test point d'intérêt".
Dans le module fenêtre, pour accéder aux 16 pixels du cercle de Bresenham autour du pixel candidat, on utilise classiquement 6 FIFOs de profondeur égale à la largeur de l'image ainsi que 49 registres (7×7) contenant ce cercle.
Afin de réduire le nombre de points détectés, notre module "détecteur FAST" ne calcule que la partie "claire" (équation 5.17). Dans le premier cycle d'horloge, la valeur de

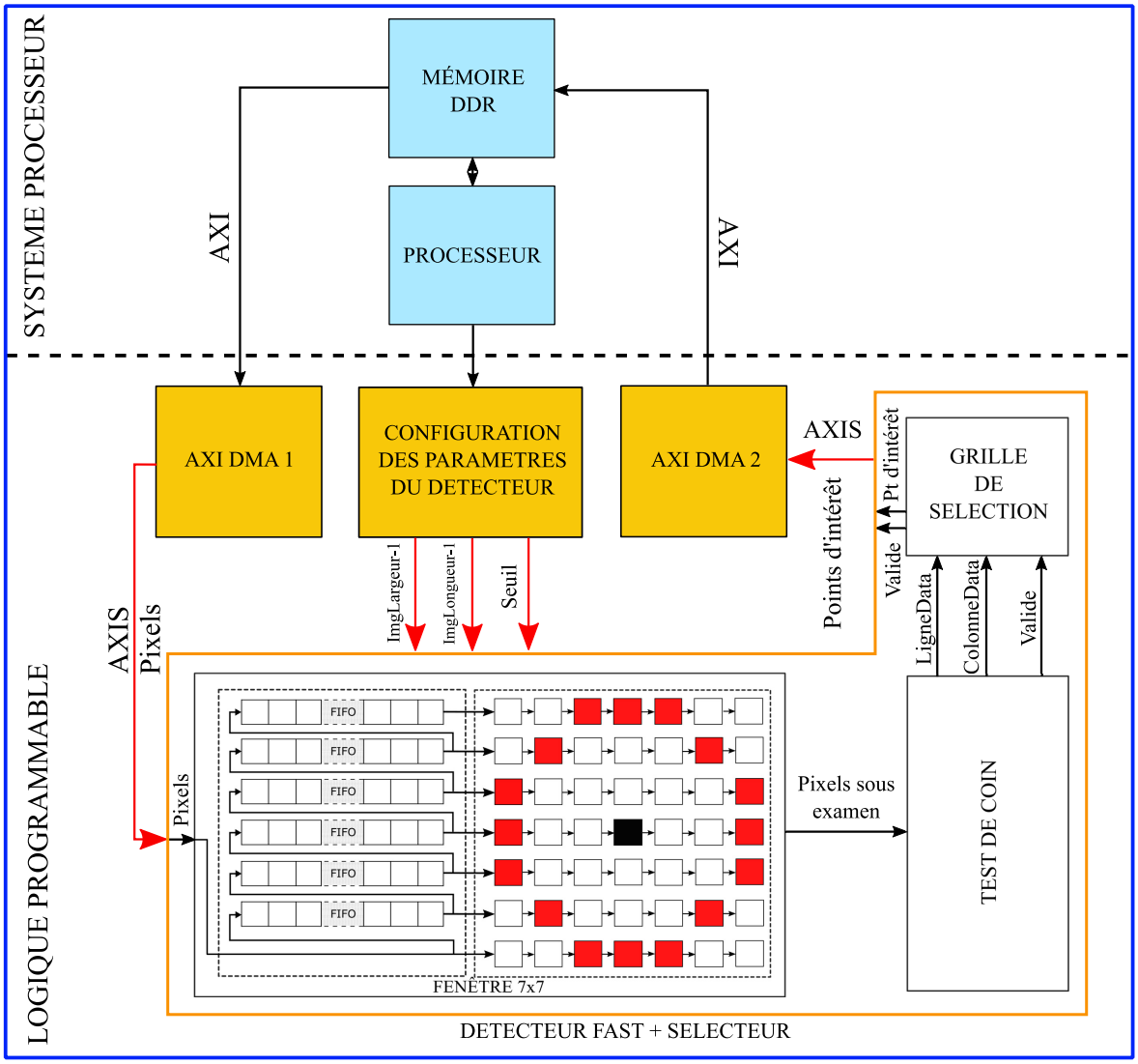


FIGURE 6.4 – L'architecture globale contenant le module "détecteur FAST" et le module "sélecteur".

chaque pixel situé sur le cercle de Bresenham est comparée au pixel candidat plus le seuil. La sortie de ces 16 comparaisons forme une suite de 16 bits. Dans le deuxième cycle d'horloge, la séquence de 16 bits est comparée à 16 rotations de 12 valeurs de '1' contigues et 4 valeurs contigues de '0'. La sortie du module "détecteur FAST" est égale à "1" s'il y a une correspondance. Le profondeur de pipeline de calcul de ce module est de deux cycles.

Le module "sélecteur" utilise un bit par colonne de la grille pour mémoriser que la case est servie. Le profondeur de pipeline de calcul de ce module est d'un cycle.

Notre architecture a été conçue en utilisant le langage de description du matériel VHDL puis testée, synthétisée, et implémentée sur le SoC Xilinx/Zynq de la caméra de l'IGN étant comme une IP. On a utilisé la mémoire RAM (DDR) pour stocker à la fois l'image et les coordonnées des points d'intérêt. Le bus AXI a été utilisé pour connecter le système du processeur avec la mémoire : le bus AXI permet d'avoir un accès direct à la mémoire DDR. L'AXIS transmet le flux de données entre les modules en mode maître/esclave. L'image dans la RAM est lue par un premier moteur DMA et transmise au circuit de notre architecture. De l'autre coté, les coordonnées de points d'intérêt

résultants sont écrites dans la mémoire par un deuxième moteur DMA. Le seuil et les dimensions de l'image sont configurés par le logiciel dans des registres accessibles par le FPGA pour que le design ait un accès permanent à ces paramètres. Les images utilisées ont une résolution de 2560 × 1920 et le nombre de points contigus (n) est fixé dans le design à 12. Le temps total obtenu est égal à 0.024 secondes qui est en effet le temps nécessaire pour transférer (2560 × 1920) pixels de la RAM. Le profondeur de pipeline de calcul de notre architecture est de trois cycles.

6.3.3 Ressources utilisées

Le tableau 6.2 illustre le nombre de ressources FPGA utilisées dans notre architecture.

Resources	**Utilisation**	**Disponible**	**Utilisation %**
LUTs	232	78600	0,30
Mémoire (blocs de 36 Kb)	6	265	2,26
Registres	472	157200	0,30

TABLEAU 6.2 – Les ressources utilisées dans notre architecture implémentée dans le Xilinx/Zynq-7030 FPGA.

Resources	**Notre architecture**	**Architecture Dinh et al. [2014]** Xilinx XC3S200 Spartan-3	**Architecture Kraft et al. [2008]** Xilinx XC3S200 Spartan-3
LUTs	232	1575	2368
Mémoire RAM	6 × 36 Kb	5 × 18 Kb	12 × 18 Kb
Registres	472	1503	1547

TABLEAU 6.3 – Comparaison des ressources utilisées dans notre architecture par rapport aux autres implémentations existantes.

Par rapport aux deux autres implémentations matérielles existantes dans la littérature (tableau 6.3), notre design utilise beaucoup moins de ressources matérielles malgré le fait que les dimensions de nos images soient plus grandes que celles des autres implémentations (640 × 480).
L'architecture présentée dans (Kraft et al. [2008]) exige environ 10 fois plus de LUTs, car elle ne teste pas seulement si le pixel candidat est un point d'intérêt mais calcule aussi un score approprié et utilise un module supplémentaire (NMS). Par ailleurs, malgré que (Dinh et al. [2014]) calculent aussi un score pour chaque point détecté, ils demandent 21% moins de ressources que (Kraft et al. [2008]) à cause de la simplification du niveau d'élimination de points adjacents.
Dans notre architecture, nous avons préféré ne pas utiliser le score pour l'élimination de points adjacents. Cela a réduit largement le nombre de ressources utilisées, si on compare le nombre de nos ressources avec celui de l'architecture de (Dinh et al. [2014]) (6.7 fois moins de LUTs, 3 fois moins de registres et presque 3 fois moins de Flipflops) malgré le fait que la résolution de l'image soit beaucoup plus grande.
Afin de valider notre design, on a vérifié que les points FAST obtenus par le matériel sont les mêmes que ceux obtenus par le logiciel.

6.4 Accélération matérielle de la partie ré-échantillonnage

Bien que la méthode accélérée du ré-échantillonnage ait montré des résultats satisfaisants en termes de qualité et de précision, son temps d'exécution reste trop lent pour notre application. Pour cette raison, on a cherché à accélérer cette méthode dans le FPGA. Cette fois, on a choisi d'utiliser une approche appelée synthèse de haut niveau (HLS) parce que l'algorithme s'exprime facilement en C/C++, et que l'environnement de développement permet de simuler facilement le comportement du résultat. On utilisera l'outil Vivado HLS fournit par Xilinx.

6.4.1 Implémentation de la partie ré-échantillonnage dans le SoC/FPGA

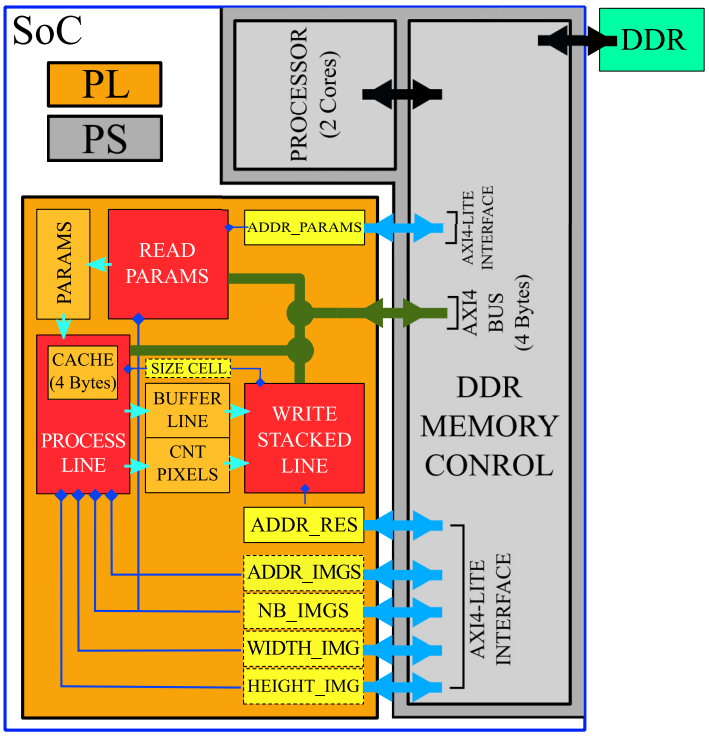


FIGURE 6.5 – Implementation de l'IP de la partie ré-échantillonnage dans le FPGA. PS : Processing system, PL : Programmable logic.

Pour bénéficier au maximum de la présence du SoC dans la caméra, nous avons utilisé ici la méthodologie du "codesign" qui permet d'assurer un développement coopératif de la partie logicielle et matérielle.
Nous avons donc décidé que la partie logique programmable dans le FPGA réalisera la superposition des pixels de la ligne courante du bloc courant et que le processeur calculera les paramètres nécessaires à cette opération.
Le calcul des adresses des pixels à moyenner nécessite le passage des paramètres de l'équation linéaire correspondant à la ligne courante :

$$\begin{cases} x^{',n} = A_x^n + B_x^n x^n \\ y^{',n} = A_y^n + B_y^n x^n \end{cases} \quad (6.7)$$

Ces paramètres sont donc à calculer par la partie logicielle selon les équations suivantes :

$$\begin{cases} A_x^n = a_x^n + c_x^n y^n \\ A_y^n = a_y^n + c_y^n y^n \\ B_x^n = b_x^n + d_x^n y^n \\ B_y^n = b_y^n + d_y^n y^n \end{cases} \quad (6.8)$$

Les paramètres (a_x^n, b_x^n, c_x^n, d_x^n, a_y^n, b_y^n, c_y^n, d_y^n) sont calculés aussi par la partie logicielle pour chaque bloc.

La figure 6.5 illustre notre implémentation matérielle de la partie ré-échantillonnage dans le SoC/FPGA. Les images ont été acquises dans la mémoire, les paramètres de la fonction bilinéaire sont calculés par le logiciel et se trouvent aussi dans la mémoire. Enfin, les pixels de l'image empilée seront aussi enregistrés au fur et à mesure dans la mémoire. La taille du bloc est fixée à 160×160 dans le code.

La dimension de chaque port est optimisée à un bit près afin de réduire au maximum les ressources utilisées dans le FPGA (tableau A.1).

Nous avons essayé de maintenir dans nos choix un compromis entre l'obtention d'une latence minimale et l'optimisation des ressources utilisées, les motivations des choix techniques que nous avons pris dans le design sont les suivantes :

- La partie logicielle calcule les paramètres de la fonction bilinéaire (a_x^n, b_x^n, c_x^n, d_x^n, a_y^n, b_y^n, c_y^n, d_y^n) en virgule flottante. Le traitement des nombres en virgule flottante est souvent complexe pour un FPGA, nous avons décidé d'utiliser une représentation virgule fixe de ces paramètres à l'aide d'un facteur d'échelle qui vaut 1024. Nous avons choisi ce facteur d'échelle en fonction de la taille des lignes des blocs (160) afin de préserver une précision de l'ordre du dixième de pixels.

- Nous avons utilisé un bloc RAM local dans le design pour récupérer depuis la RAM principale les $4 \times n$ paramètres (A_x^n, B_x^n, A_y^n, B_y^n) des fonctions bilinéaires de chaque image associés à la ligne en question. Ceci permet au design d'avoir un accès rapide à ces paramètres.

- Au début, nous avons pensé utiliser plusieurs bus AXI pour accélérer le traitement, mais comme ils sont branchés sur la même mémoire RAM, ceci complique la partie logique et n'apporte rien sur l'accélération du traitement. Nous avons donc utilisé un seul bus AXI (32 bits) dans tout le design pour accéder à la mémoire : pour lire les pixels des différentes images, pour lire les quatre paramètres (A_x^n, B_x^n, A_y^n, B_y^n) de la fonction bilinéaire de la ligne actuelle de toutes les images, et pour écrire les pixels de l'image finale. Nous avons utilisé également un bus AXI LITE pour contrôler l'IP depuis le logiciel par l'intermédiaire de registres comme les registres de contrôle et de statut et les paramètres liés au traitement comme le nombre et la taille de l'image, les adresses physiques des images dans la mémoire, l'adresse des $4 \times n$ paramètres de la fonction bilinéaire et l'adresse à laquelle la ligne traitée sera écrite. Ce bus prend en charge un seul transfert de données par transaction.

- Étant donné que la rotation entre les images est généralement petite, il y a une grande probabilité que les projections de points adjacents soient elles mêmes

adjacentes en mémoire. Nous avons donc décidé d'utiliser une mémoire cache de 4 pixels afin de réduire les accès mémoire dans les cas favorables. Pour faciliter la gestion de ce micro cache, nous avons été amenés à mettre la boucle sur les images à l'extérieur et la boucle sur les colonnes à l'intérieur.

- La sommation des pixels se fait dans un tampon ligne de taille TAILLE_BLOC_X dans la boucle de calcul. Les valeurs de ce tampon sont normalisées et envoyées en mémoire à la fin du traitement à raison d'un cycle d'horloge par valeur.

- Afin de maximiser le débit et en considérant que les données sont organisées en flux, nous avons décidé d'utiliser au maximum une architecture de type pipeline en insérant des directives *#pragma HLS_PIPELINE* dans les différentes boucles du processus. Nous avons utilisé aussi des directives de partition *#pragma HLS ARRAY_PARTITION* pour fractionner le tableau des paramètres des fonctions bilinéaires en plusieurs tableaux de taille plus petite pour imposer une organisation des données permettant de maximiser le débit en limitant les effets de goulot d'étranglement des accès mémoire. Enfin, les valeurs de même indice des tableaux contenant respectivement les pixels résultants et les compteurs de pixels étant toujours utilisées simultanément, nous pouvons minimiser les ressources utilisées en fusionnant ces deux tableaux dans un seul bloc RAM avec la directive de fusion *#pragma HLS ARRAY_MAP*.

6.4.2 Ressources utilisées

Ressources	Utilisation	Disponible	%
LUTs	1741	78600	2.21
Mémoire (blocs de 36 KB)	2.5	265	0.94
Registres	2364	157200	1.50
DSPs	1	400	0.25

TABLEAU 6.4 – *Les ressources utilisées dans l'implémentation matérielle de la partie ré-échantillonnage dans le FPGA Xilinx/Zynq-7030.*

Le tableau 6.4 montre que notre design utilise peu de ressources par rapport à ce qui est disponible dans le Zynq 7030.

6.4.3 Utilisation de plusieurs copies de l'IP

Comme remarqué ci-dessus, notre implémentation requiert peu de ressources logiques du FPGA. Ceci permet d'intégrer plusieurs copies de l'IP (figure 6.6) afin de traiter en parallèle plusieurs lignes du bloc en question, surtout que les tâches effectuées sur les lignes du bloc sont indépendantes. Chaque copie de l'IP sera responsable du traitement d'une seule ligne à la fois, elle aura donc son propre accès à la mémoire, et son propre accès à ses registres de contrôle.

6.4.4 Temps d'exécution

Les temps de calcul présentés ici sont le résultat de tests effectués sur la caméra légère de l'IGN. Dans un premier temps, nous n'avons pas activé la fonctionnalité du cache de

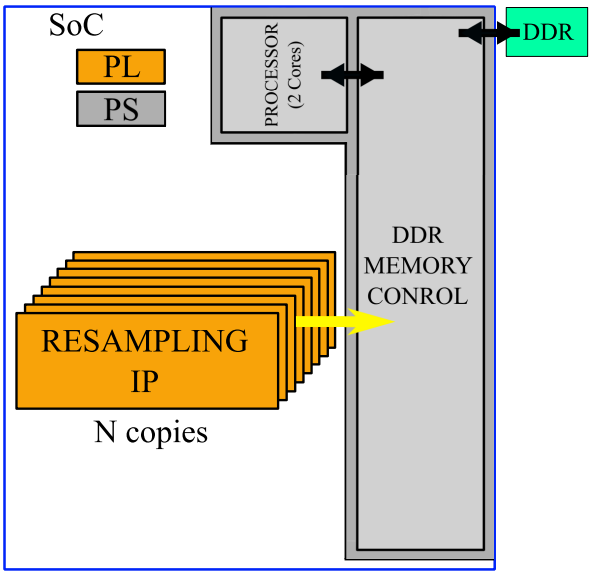


FIGURE 6.6 – Utilisation de plusieurs copies de l'IP.

lecture de pixels dans le design afin d'analyser son impact. La figure 6.7 montre l'évolution du temps de calcul de la partie ré-échantillonnage en fonction des différentes tailles de bloc testées et en fonction du nombre de copies de l'IP utilisées. Les 8 courbes présentées correspondent aux 8 différentes tailles de bloc étudiés. Tout d'abord, on peut

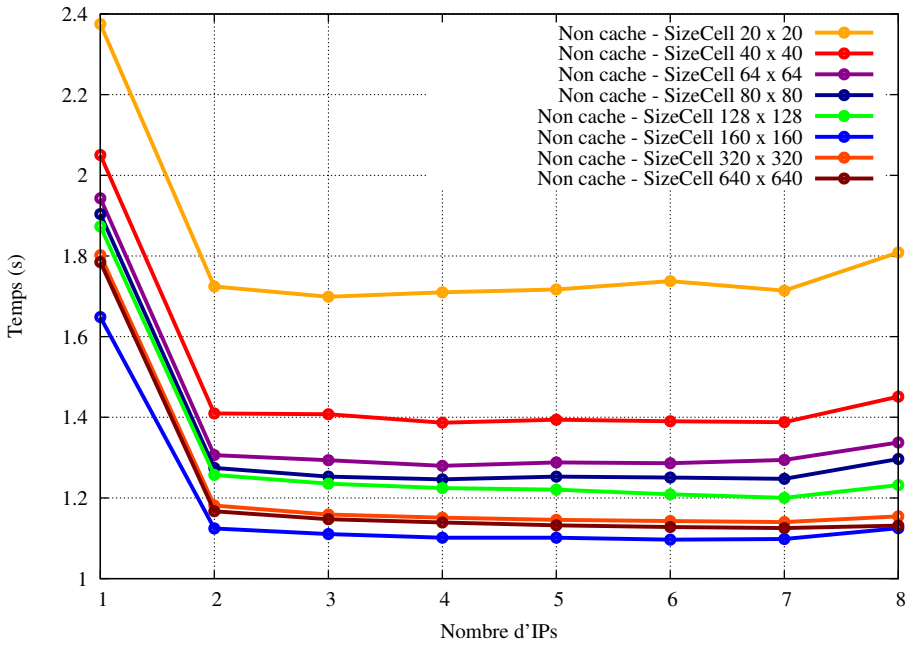


FIGURE 6.7 – Le temps de calcul en fonction de la taille de bloc et du nombre d'IP sans utilisation du cache.

remarquer que le temps de calcul diminue lorsqu'on augmente la taille de bloc jusqu'à une certaine limite "160 × 160". Cette diminution est due au fait que la collaboration entre le FPGA et le processeur n'est pas parfaite ce qui est plus sensible dans le cas des petits blocs. D'un autre coté, on observe que le temps de calcul diminue quand on passe d'une à deux IPs puis stagne jusqu'à 7 et réaugmente à 8 pour toutes les tailles

de bloc.
La stagnation que l'on observe au delà de deux IPs est probablement due à la saturation du bus mémoire. Cette saturation intervient en fait même pour deux IPs ce qui limite l'efficacité du procédé : on voit en effet que le temps de calcul pour deux IPs n'est pas la moitié de celui obtenu avec une IP.
Dans un deuxième temps, on a ajouté la fonctionnalité du cache dans le design HLS et on a refait les mêmes tests. Les résultats illustrés dans la figure 6.8 et la figure 6.9

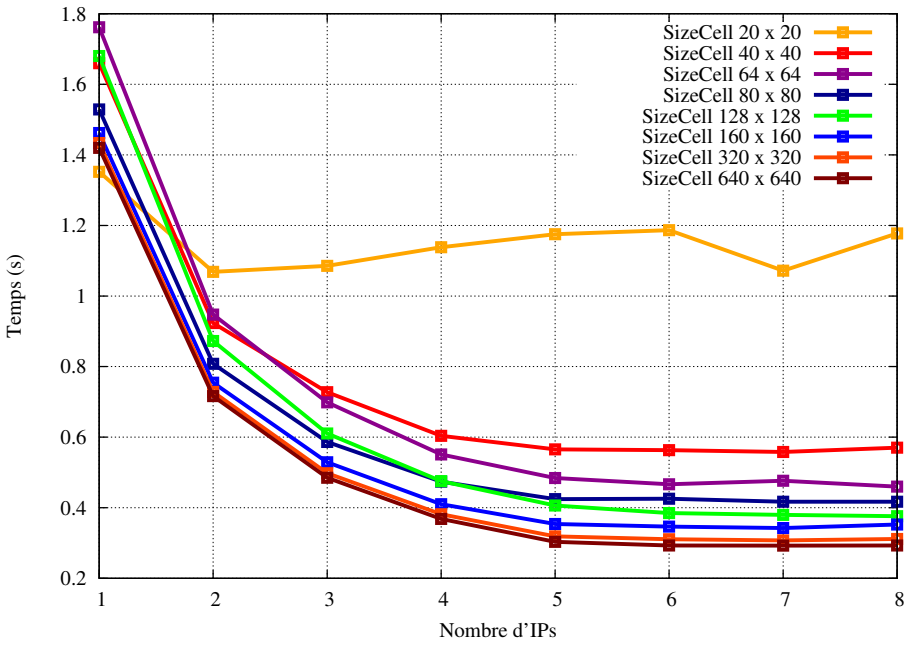


FIGURE 6.8 – Le temps de calcul en fonction de la taille du bloc et du nombre de blocs avec utilisation du cache.

montrent que l'utilisation du cache a diminué efficacement le temps de traitement et que la saturation n'apparaît qu'à partir de 5 IPs.
A partir de ces résultats (figure 6.9), notre choix s'est porté sur 4 copies de l'IP seulement et une taille de bloc de "160×160" afin d'aboutir à un compromis entre le temps de calcul et l'utilisation des ressources matérielles du FPGA.

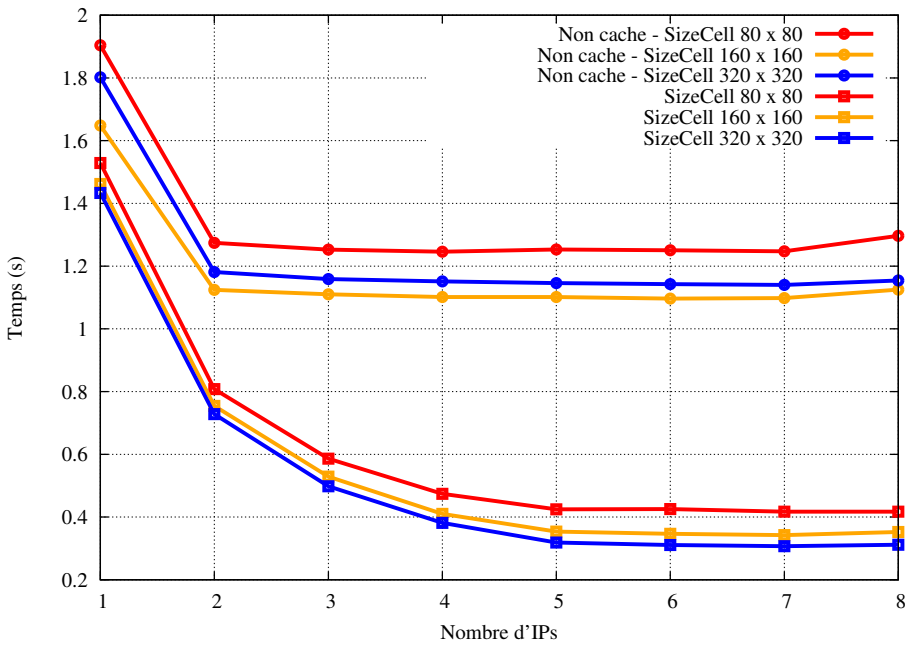


FIGURE 6.9 – Le temps de calcul en fonction de la taille du bloc, du nombre de blocs et de l'utilisation du cache ou pas.

6.5 Conclusion

L'implémentation de la phase "détection et répartition des points d'intérêt" et de la phase "ré-échantillonnage" dans le FPGA a accéléré largement l'exécution de l'algorithme dans la caméra, ce qui convient mieux à certaines applications.
Il est possible d'améliorer l'implémentation de la phase "détection et répartition des points d'intérêt" en traitant le flux de pixels directement à partir du capteur afin de gagner encore plus du temps.
De plus, il serait intéressant dans l'implémentation de la phase "ré-échantillonnage" de partager le calcul de la partie logicielle entre les deux cœurs du processeur pour accélérer un peu plus l'exécution globale de cette phase.

6.6 Références

Audi, A., Pierrot Deseilligny, M., Meynard, C., and Thom, C. (2015). Implementation of real-time functionalities in the hardware of an intelligent ultra-light camera specialized for aerial photography. *GeoUAV - ISPRS Geospatial Week*.

Dinh, T. H., Vu, D. Q., Ngo, V.-D., Ngoc, N. P., and Truong, V. T. (2014). High throughput fpga architecture for corner detection in traffic images. In *2014 IEEE Fifth International Conference on Communications and Electronics (ICCE)*, pages 297–302. 77, 78, 80

Kraft, M., Schmidt, A., and Kasiński, A. (2008). High-speed image feature detection using fpga implementation of fast algorithm. In *Proceedings of the Third International Conference on Computer Vision Theory and Applications - Volume 1 : VISAPP, (VISIGRAPP 2008)*, pages 174–179. INSTICC, ScitePress. 77, 78, 80

Chapitre 7

Résultats

Sommaire

7.1 Introduction

Pour tester la validité et les limites de notre procédé, nous avons réalisé certains tests en conditions réelles dont les résultats sont présentés dans ce chapitre.
Ces tests ont été effectués pour la plupart sur le site d'Ifsttar à Nantes avec leur drone hélicoptère "copter 1B" (figure 7.1) et pour les autres sur le site de Viabon avec le drone quadricopter "3DR" de Vinci. Le déclenchement d'acquisition est fait à distance par un signal radio. Chaque ensemble de données consiste en une séquence de dix images acquises avec la fréquence maximale du CMOS (30 images/s) et les données inertielles associées. La valeur du temps de pose est mise à 1 ms pour se rapprocher du cas de l'imagerie bande spectrale étroite. Ces tests ayant servi à mettre au point les algorithmes, les images n'ont pas été traitées en temps-réel mais ont été sauvegardées pour être post-traitées.

FIGURE 7.1 – le drone type Copter 1B développé par Wikhydro et équipé d'une caméra légère de l'IGN pendant une des expériences réalisées.

7.2 Limites attendues du procédé

Les principales limitations du procédé qu'on peut anticiper sont :

- Non prise en compte des mouvements de translation pendant la prise de vue :
  - Ces mouvements ont un impact sur la précision de la prédiction des positions des points homologues, ce qui a comme conséquences soit qu'on ne trouvera pas les points homologues, soit si on agrandit la zone de recherche, qu'on risquera de faire de faux appariements.
  - Lorsque la caméra se déplace, la transformation entre les images acquises ne suit plus un modèle géométrique simple et nécessite la connaissance du modèle numérique de surface.
- La rotation selon l'axe z pourrait perturber la détermination des points homologues par corrélation.
- L'acquisition de bandes spectrales fines ou dans des conditions d'éclairement faible pourrait aboutir à des images ayant un mauvais rapport signal à bruit, ce qui pourrait empêcher la détection des points d'intérêt et fausser l'appariement.

Malheureusement, l'IGN ne disposant pas de drone, il a été difficile de réaliser des tests en conditions réelles et toutes ces limites n'ont pas pu être explorées.

7.3 Résultats sur plusieurs séquences

Pour toutes les tests réalisés, la taille de la grille des points d'intérêt est fixée à 10×10, la taille des zones de recherche est fixée à 11×11, la taille des blocs pour la méthode

de ré-échantillonnage accélérée est fixée à 160×160, les images ont une résolution de 2560×1920. Les images sont éclaircies pour mieux visualiser les résultats. Le nombre maximum de points d'intérêt qu'on peut avoir ici est 100 puisque la taille de grille est 10×10.

7.3.1 Cas de mouvement de rotation pure de la caméra

Dans cette partie, on présente deux séquences dans lesquelles le drone a subi un mouvement de rotation pure.

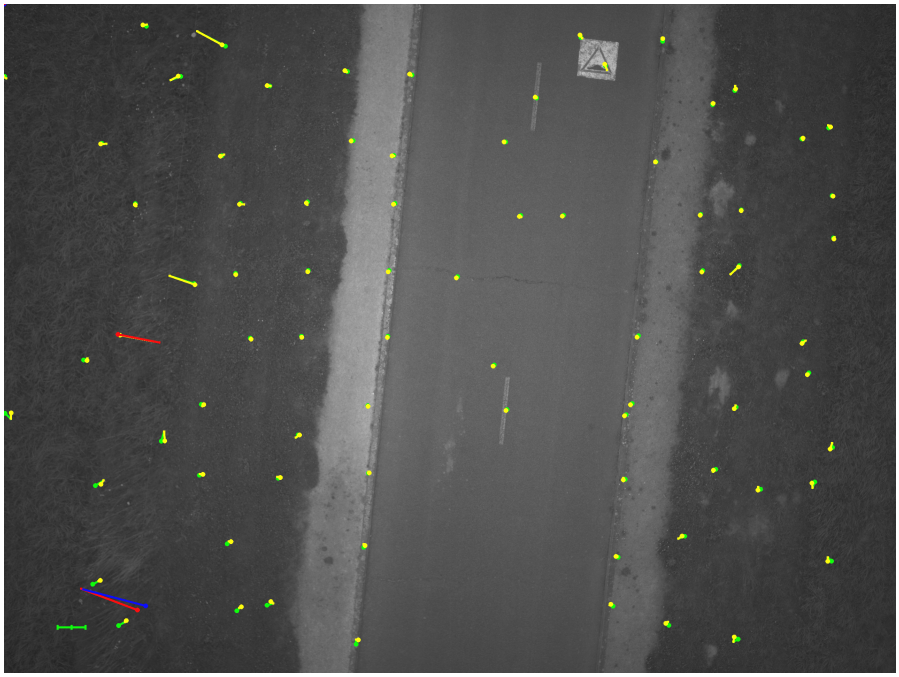


FIGURE 7.2 – Les résidus de l'estimation de la rotation et de l'homographie entre la $1^{ière}$ et la $10^{ième}$ image de la séquence 1. Les flèches vertes représentent les résidus de l'estimation de la rotation et les flèches rouges représentent ceux qui sont considérés comme aberrants. Les flèches jaunes représentent les résidus de l'estimation de l'homographie et les flèches bleues représentent ceux qui sont considérés comme aberrants.

Les figures 7.2, 7.3 et 7.4 montrent que pour les deux types de transformations, les erreurs RMS pour toutes les images sont inférieures à 0.5 pixel dans les deux séquences. La figure 7.4 montre que l'estimation de l'homographie a moins de résidus que la rotation dans les deux séquences, ce qui est compatible avec le fait que l'homographie a plus de degrés de liberté. Toutefois, cette différence est suffisamment faible pour que l'on considère que le modèle homographique n'est pas mieux adapté que le modèle basé sur la rotation, ce qui signifie que le mouvement de translation des sommets de prises de vue est effectivement négligeable.
On remarque aussi que les résidus des deux transformations augmentent linéairement au cours des images. La figure 7.5 montre que cela est causé par le mouvement des herbes à côté de la route : les résidus dans la zone routière sont presque constants alors que ceux de la zone herbeuse sont plus élevés et en augmentation. Ceci indique également que le bruit de notre processus n'est que de l'ordre de 0.2 pixel (séquence 1) et 0.3 pixels (séquence 2) en RMS. Les figures 7.6 et 7.7 montrent qu'aucun flou n'affecte visuellement l'image finale empilée. Ceci suggère fortement que notre méthode

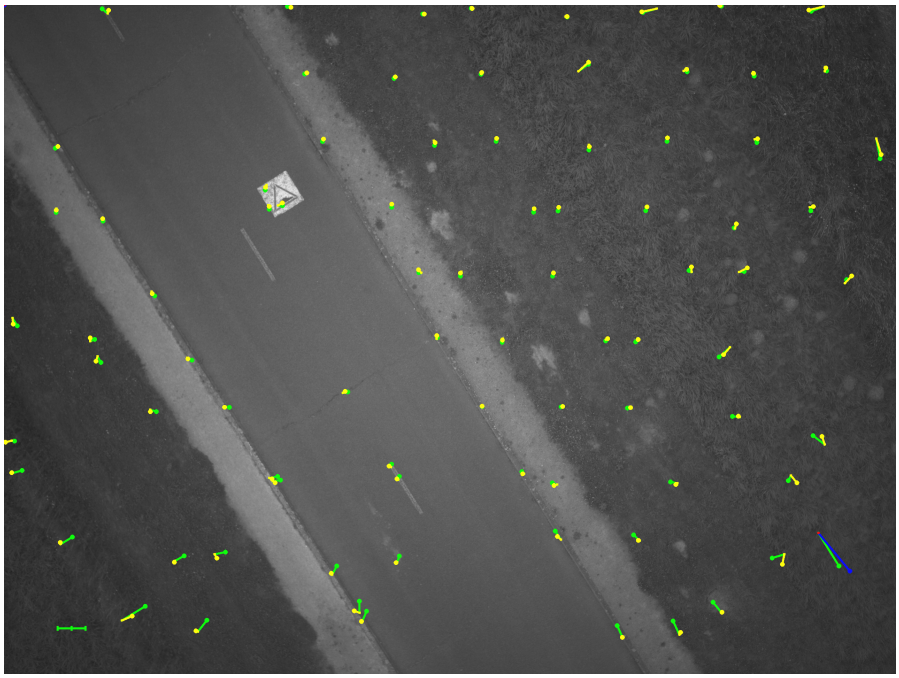
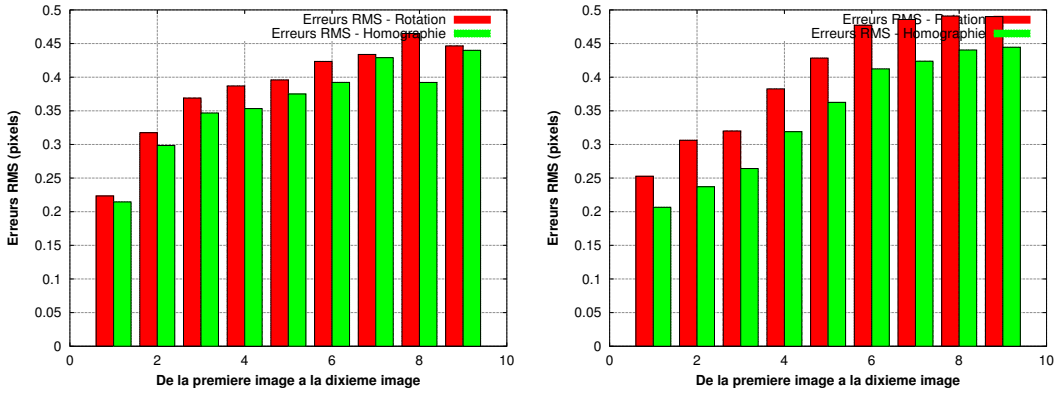


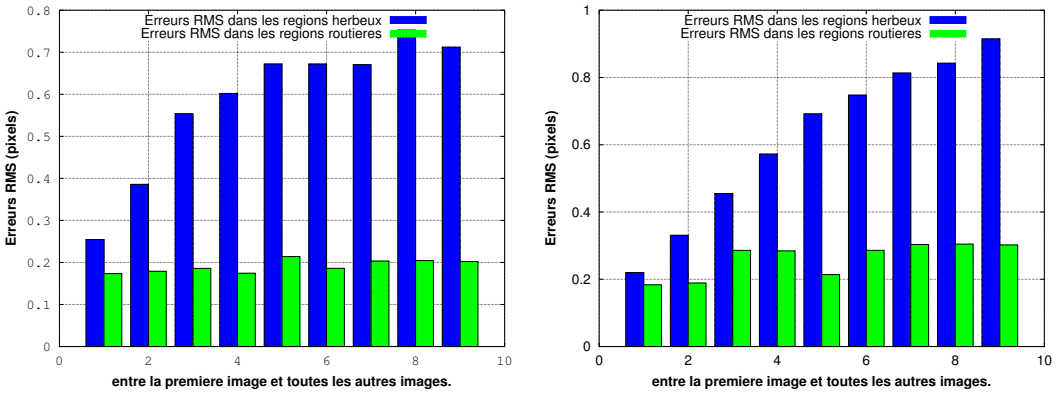
FIGURE 7.3 – Les résidus de l'estimation de la rotation et de l'homographie entre la $1^{ière}$ et la $10^{ième}$ image de la séquence 2. Les flèches vertes représentent les résidus de l'estimation de la rotation et les flèches rouges représentent ceux qui sont considérés comme aberrants. Les flèches jaunes représentent les résidus de l'estimation de l'homographie et les flèches bleues représentent ceux qui sont considérés comme aberrants.



(a) Séquence 1. (b) Séquence 2.

FIGURE 7.4 – Les résidus de l'estimation de la rotation et de l'homographie entre la $1^{ière}$ et la $10^{ième}$ image.

peut être effectivement utilisée pour éliminer le flou de bougé causé par les mouvements erratiques de la caméra. En fait, le flou est presque indétectable dans les images résultants, même si le mouvement de la caméra est complexe et même s'il est de l'ordre de plusieurs dizaines de pixels contrairement aux méthodes de dé-convolution. Nous avons calculé le profil de la première image et de l'image empilée afin de les comparer pour quantifier au mieux la qualité de l'image obtenue. Les figures 7.8 et 7.9 montrent que les deux profils sont très proches l'un de l'autre. Le profil radiométrique de l'image empilée est plus lisse. L'écart qui se trouve entre les deux profils de la séquence 2 vient du fait que la première image est plus sombre que les autres images.



(a) Séquence 1. (b) Séquence 2.

FIGURE 7.5 – Les résidus de l'estimation de la rotation et de l'homographie entre la $1^{ière}$ image et toutes les autres images dans les zones herbeuses et routières.

FIGURE 7.6 – Image empilée finale de la séquence 1. La zone orange représente un zoom de l'image finale empilée. La zone jaune représente la même zone dans première image de la séquence. La zone rouge représente la même zone dans la moyenne de 10 images non recalées.

La figure 7.10 présente les estimations de la rotation de la caméra par l'IMU et par la photogrammétrie. On remarque que les deux courbes sont très proches. Les biais se manifestent comme les pentes des courbes en pointillé, on voit qu'ils sont quasi-nuls dans tous les cas sauf autour de l'axe x dans la séquence 2.

7.3.2 Cas de mouvement de la caméra (rotation non pure)

On présente ici des résultats de deux séquences qui ont été acquises pendant que le drone subissait un mouvement.

FIGURE 7.7 – Image empilée finale de la séquence 2. La zone orange représente un zoom de l'image finale empilée. La zone jaune représente la même zone dans première image de la séquence. La zone rouge représente la même zone dans la moyenne de 10 images non recalées.

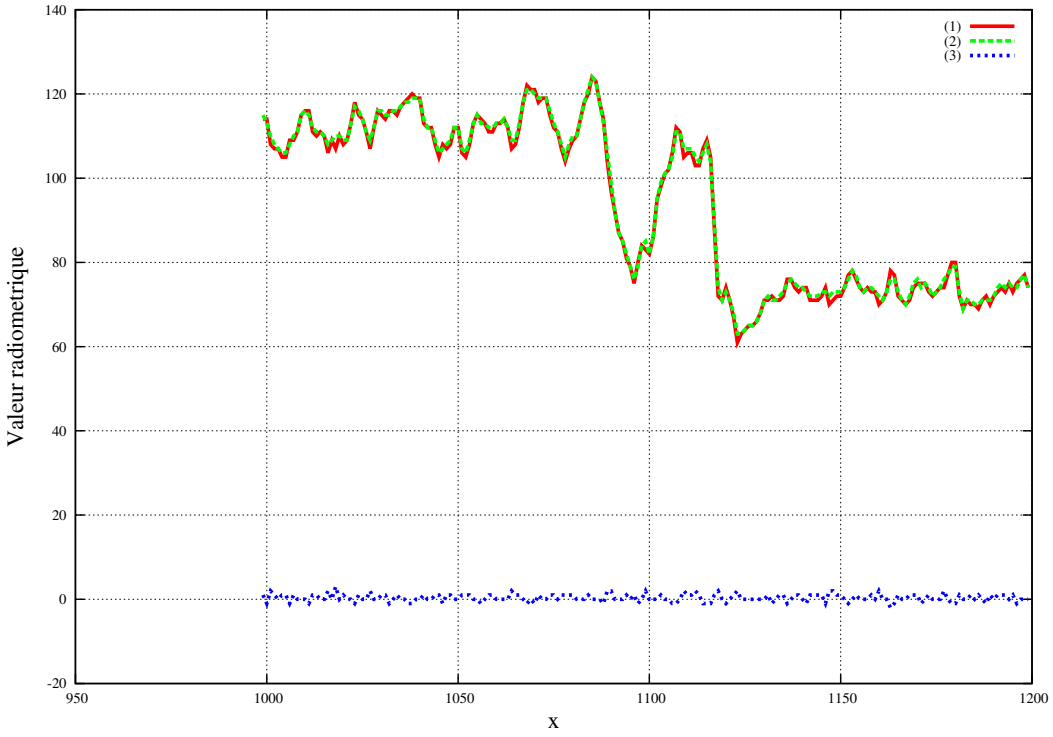


FIGURE 7.8 – Séquence 1 : profil radiométrique d'une partie d'une ligne de l'image empilée et de la première image. (rouge) : première image, (verte) : image finale empilée, (bleu) : la différence entre les deux profils.

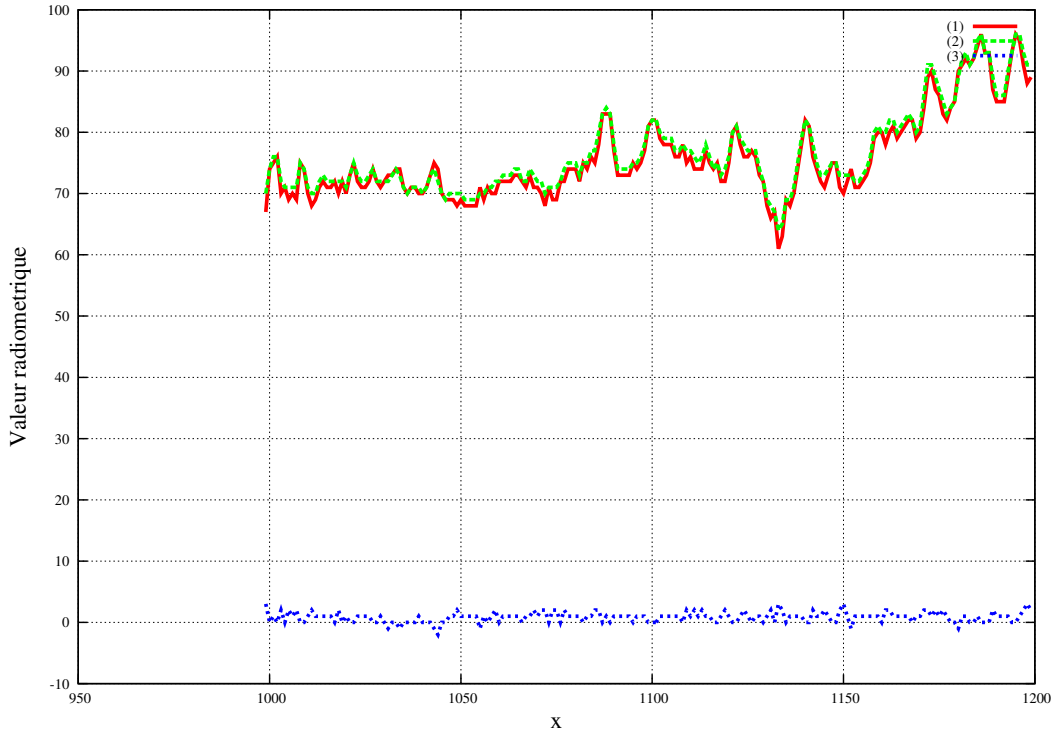
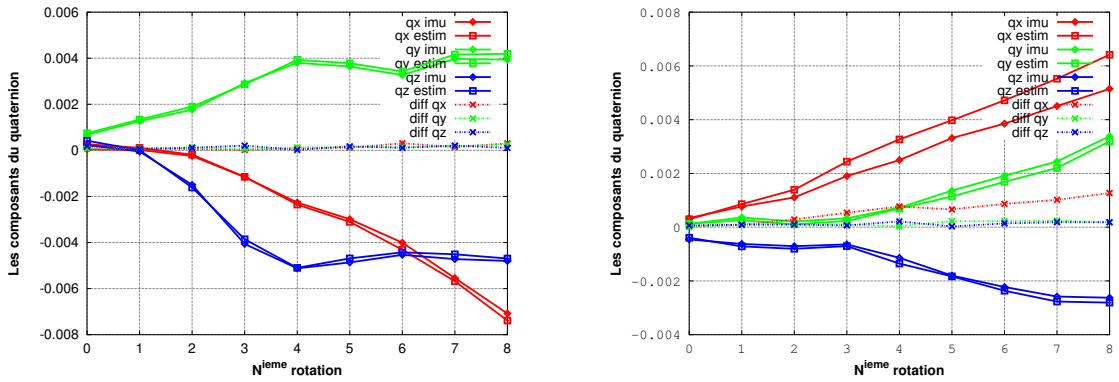


FIGURE 7.9 – Séquence 2 : profil radiométrique d'une partie d'une ligne de l'image empilée et de la première image. (rouge) : première image, (verte) : image finale empilée, (bleu) : la différence entre les deux profils.



(a) Première séquence.

(b) Deuxième séquence.

FIGURE 7.10 – Les deux rotations estimées obtenues dans les première et deuxième séquences représentées par les composantes de quaternions qui les modélisent. Symbole carré : méthode photogrammétrique. Symbole en losange : IMU. Les courbes en pointillé représentent la différence entre les deux.

Cas d'un mouvement en Z

Cette séquence représente une situation particulière du mouvement de drone où ce dernier a subi une chute libre pendant l'acquisition. Dans ce cas, la rotation estimée avec notre méthode photogrammétrique n'est pas optimale car le centre optique est déplacé pendant l'acquisition des images. Ceci est confirmé par le fait que les résidus obtenus sont importants et convergent vers le centre de l'image (figure 7.11). Ce mouvement

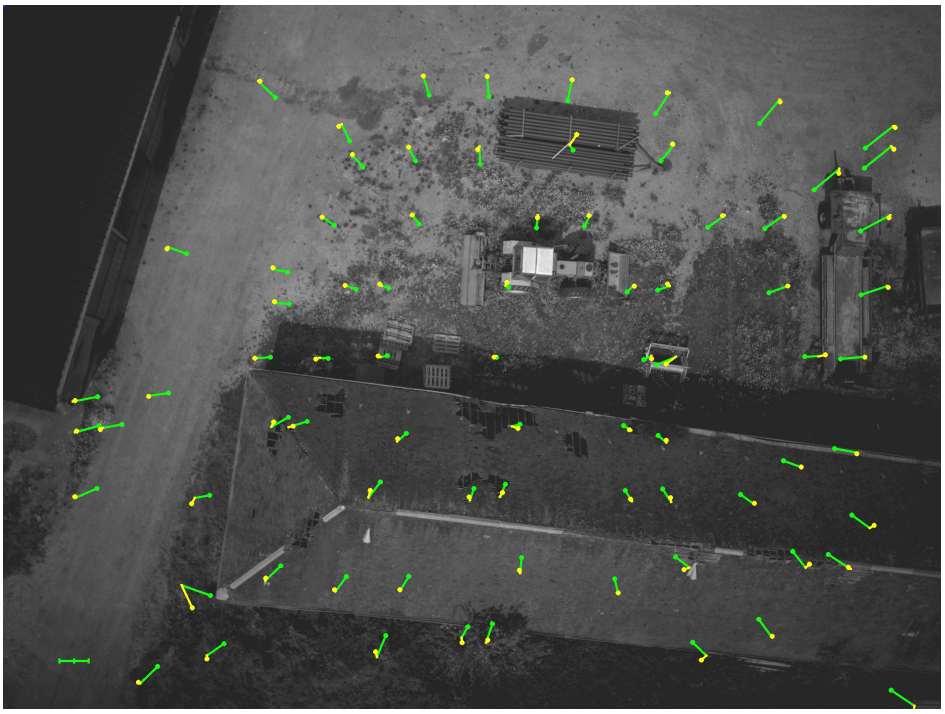


FIGURE 7.11 – Les résidus de l'estimation des deux transformations entre la $1^{ière}$ image et la $10^{ième}$ image. Les flèches vertes représentent les résidus de l'estimation de la rotation. Les flèches jaunes représentent les résidus de l'estimation de l'homographie.

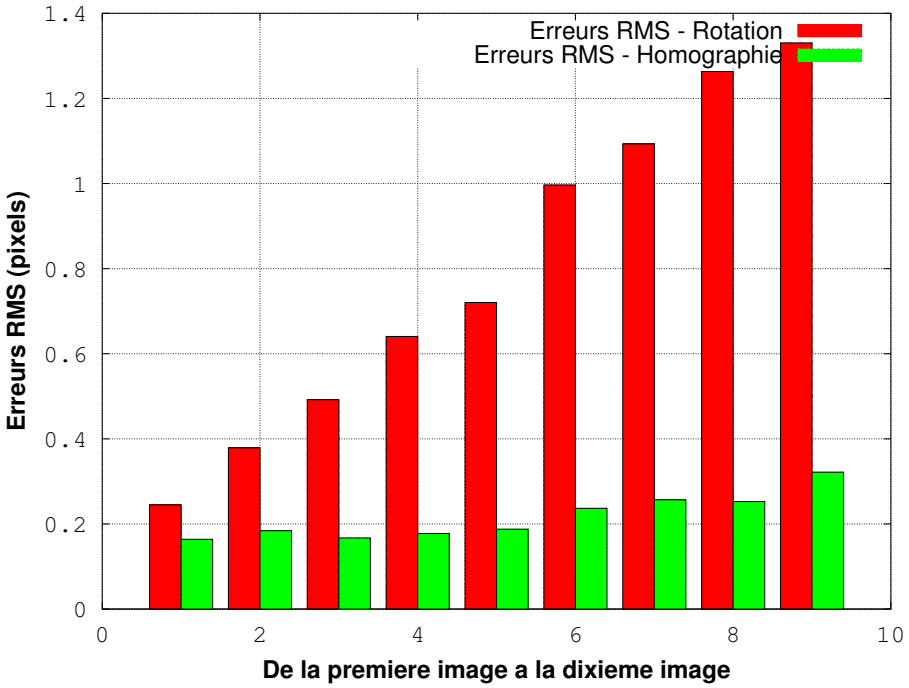


FIGURE 7.12 – Les résidus de l'estimation de la rotation et de l'homographie entre la $1^{ière}$ image et toutes les autres images en cas de mouvement de drone.

résiduel, qui n'est pas compensé par la rotation, peut être absorbé par l'homographie qui comprend un facteur d'échelle (figure 7.11). La figure 7.12 montre que les résidus de l'homographie estimée sont constants au cours des images et sont aussi beaucoup plus

FIGURE 7.13 – Image empilée finale de la séquence 3. La zone orange représente un zoom de l'image finale empilée. La zone jaune représente la même zone dans première image de la séquence. La zone rouge représente la même zone dans la moyenne de 10 images non recalées.

faibles que ceux de la rotation estimée, mais ils sont plus importants dans les zones où la scène est plus haute que le plan moyen comme le toit par exemple (figure 7.11). Grâce aux valeurs RMS de ces résidus, nous avons un facteur de qualité de contrôle qui peut aider notre traitement à déterminer quel type de transformation doit être utilisé pour obtenir les meilleurs résultats.
La figure 7.13 montre que l'image finale empilée ne manifeste pas un flou de mouvement puisqu'on a utilisé l'homographie comme transformation géométrique entre les images. Cette netteté est confirmée par le profil radiométrique illustré dans la figure 7.14.
La figure 7.15 montre des différences importantes entre les rotations estimées et les rotation mesurées par l'IMU qu'on ne peut malheureusement pas interpréter comme des biais car certains mouvements parasites du drone peuvent polluer l'estimation de la rotation.

Cas d'une translation de la caméra

Dans un autre jeu de données, la forme du flux des vecteurs de résidus (figure 7.16) montre que le drone a subi un mouvement de translation selon x pendant les prises de vue.
Ici, la rotation estimée par notre méthode photogrammétrique n'est pas représentative de la vraie rotation car le centre optique a bougé pendant l'acquisition des images, tout en laissant les résidus plus importants que d'habitude (figure 7.16). Le mouvement de translation peut être absorbé par l'homographie, et la figure 7.16 montre que les résidus de l'homographie estimée sont plus uniformes dans toute l'image et beaucoup plus faibles que ceux de la rotation estimée comme illustré dans la figure 7.17.
Enfin, les figures 7.18 et 7.19 montrent une bonne qualité radiométrique de l'image

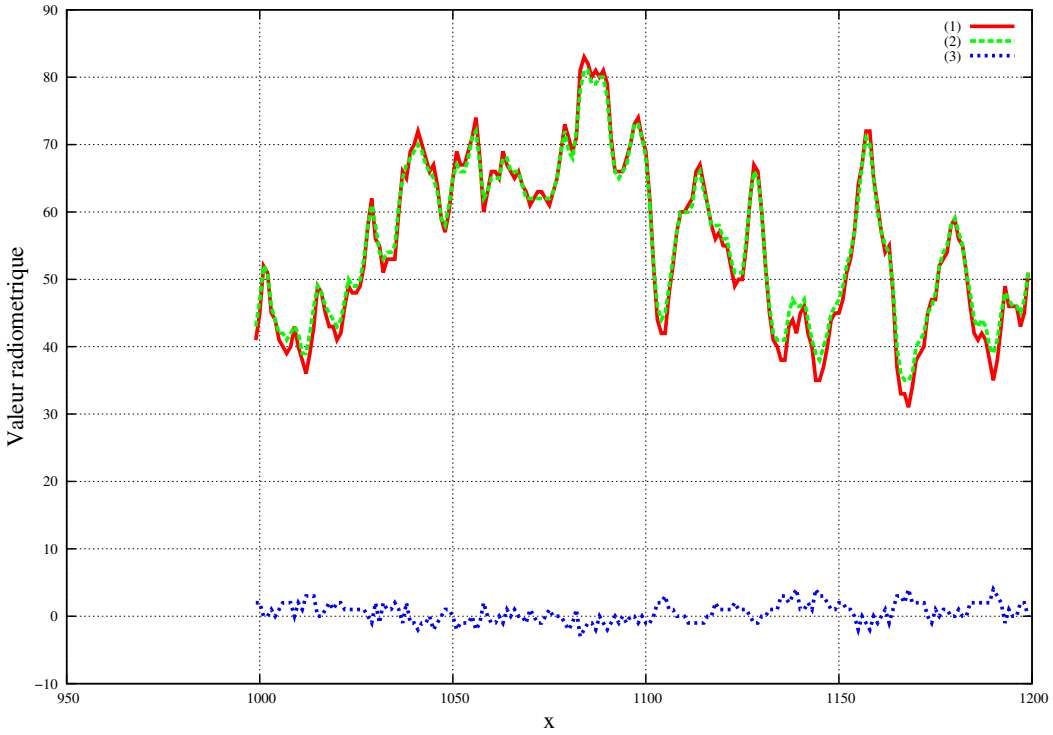


FIGURE 7.14 – Séquence 3 : profil radiométrique d'une partie d'une ligne de l'image empilée et de la première image. (rouge) : première image, (verte) : image finale empilée, (bleu) : la différence entre les deux profils.

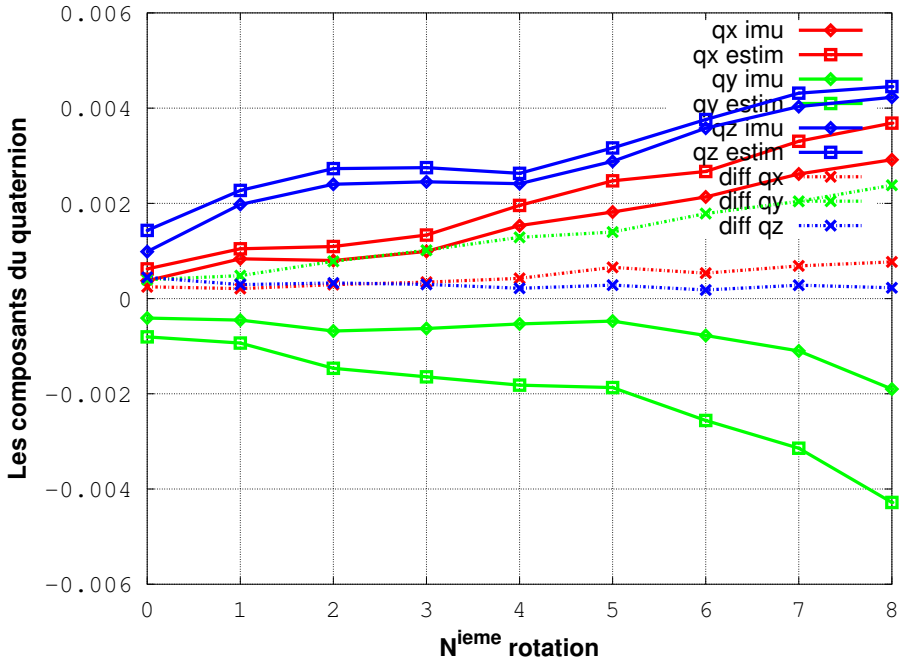


FIGURE 7.15 – Les deux rotations estimées obtenues dans la troisième séquence.

empilée finale en utilisant l'homographie comme transformation géométrique entre les images.
La figure 7.20 montre de la même manière que dans le cas précédent on ne peut pas

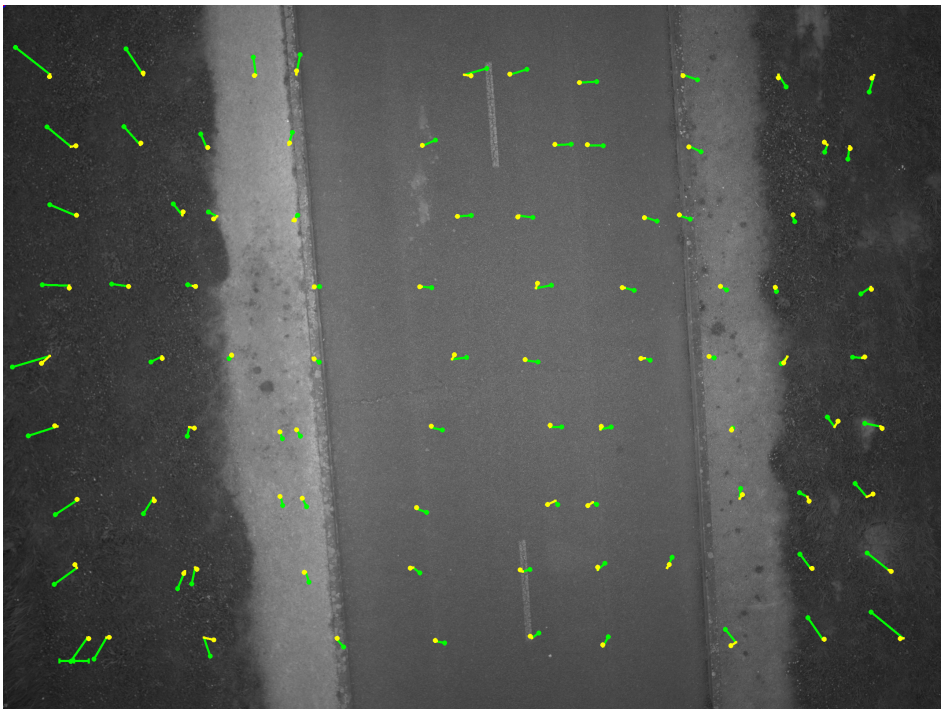


FIGURE 7.16 – Les résidus de l'estimation des deux transformations entre la $1^{ière}$ image et la $10^{ième}$ image. Les flèches vertes représentent les résidus de l'estimation de la rotation. Les flèches jaunes représentent les résidus de l'estimation de l'homographie.

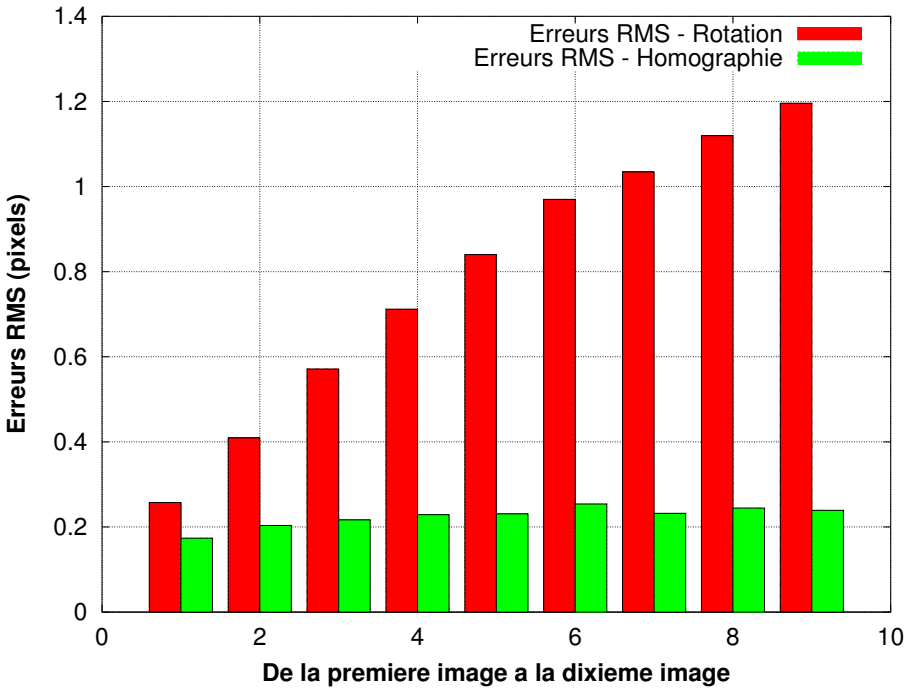


FIGURE 7.17 – Les résidus de l'estimation de la rotation et de l'homographie entre la $1^{ière}$ image et toutes les autres images en cas de mouvement de drone.

considérer que les rotations estimées sont représentatives des vraies rotations de la caméra.

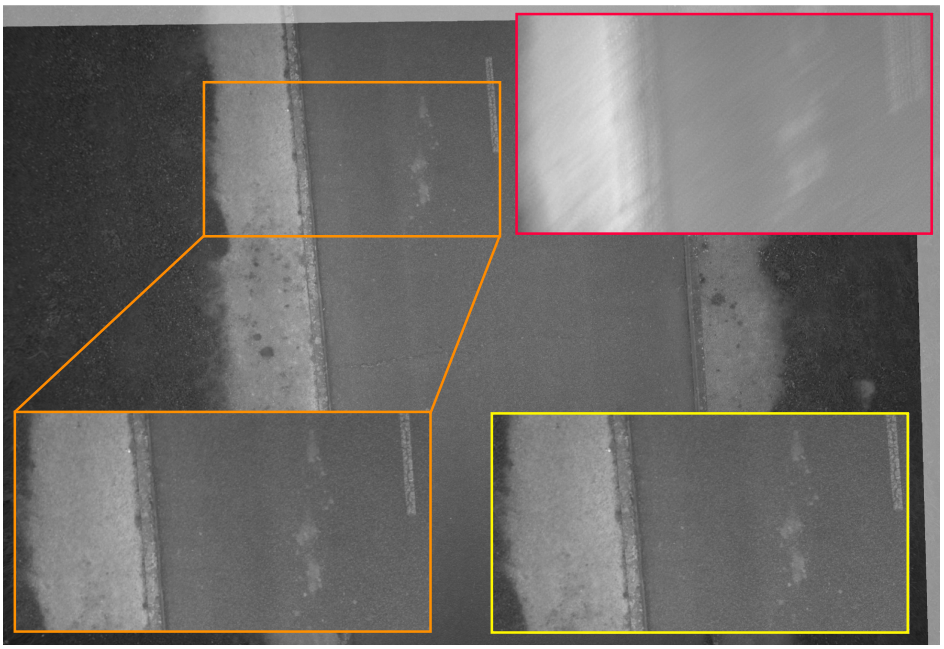


FIGURE 7.18 – Image empilée finale de la séquence 4. La zone orange représente un zoom de l'image finale empilée. La zone jaune représente la même zone dans première image de la séquence. La zone rouge représente la même zone dans la moyenne de 10 images non recalées.

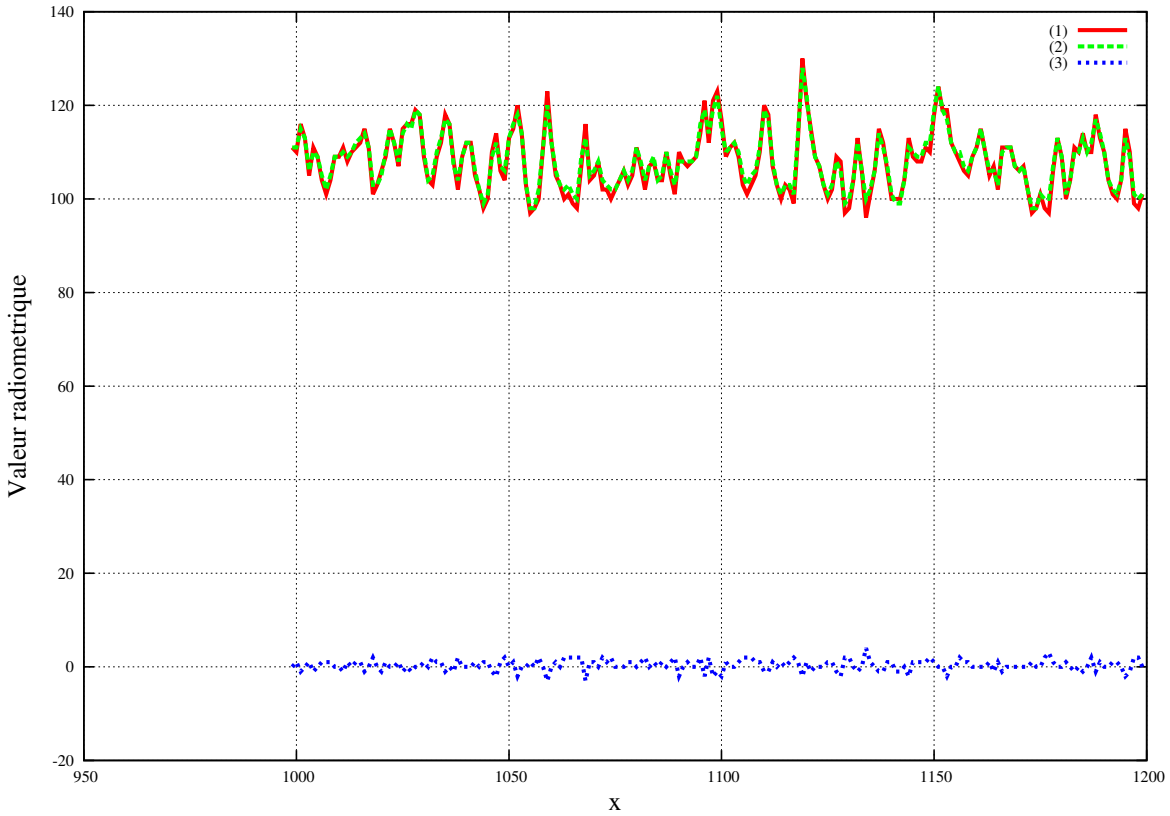


FIGURE 7.19 – Séquence 4 : profil radiométrique d'une partie d'une ligne de l'image empilée et de la première image. L'homographie est utilisée comme transformation géométrique entre les images. (rouge) : première image, (verte) : image finale empilée, (bleu) : la différence entre les deux profils.

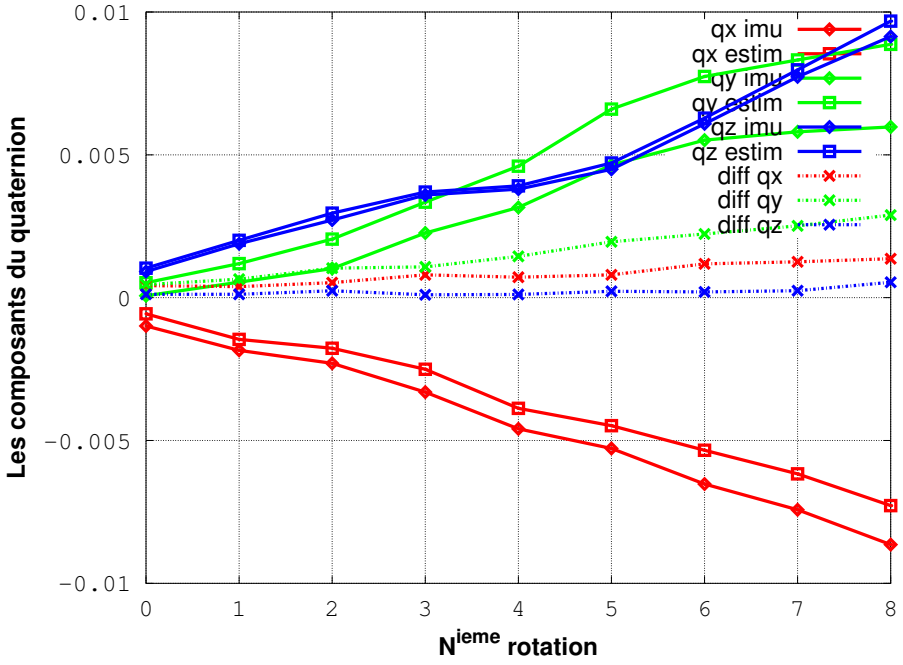


FIGURE 7.20 – Les deux rotations estimées obtenues dans la quatrième séquence.

7.4 Temps total d'exécution

Détection points d'intérêt (85 points)	**Appariement (85 points) & Estimation de la transformation**	**Ré-échantillonnage**		**Temps max total**
		Rotation	**Homographie**	
Hard	Soft	Co-design	Co-design	
0.02	0.08	0.42	0.40	**0.52**

TABLEAU 7.1 – Temps d'exécution en secondes des différentes phases du traitement. Les tests ont été exécutés dans la caméra IGN photogrammétrique.

Le tableau 7.1 montre que le temps d'exécution de notre implémentation est compatible avec nos applications car ce temps n'excède pas 0.5 seconde.

7.5 Conclusion

Les résultats montrent que notre méthode est efficace dans tous les cas testés. Si le drone ne subit qu'une rotation pure, on peut utiliser la rotation comme transformation géométrique, sinon on utilise l'homographie, et on voit que même dans ce cas les images résultantes stackées ne présentent aucun flou. Les résidus RMS jouent ici un rôle de facteur de qualité et peuvent guider le choix de la transformation géométrique pour la suite du traitement. Le fait qu'on utilise nettement plus de points de recalage que nécessaire bien répartis dans la scène garantit que les zones éventuellement mal superposées seront détectées.
On n'a pas pu tester le cas des images à très faible luminosité même si des temps de pose assez faibles ont été utilisés. De même le cas de fortes rotations autour de l'axe z ne s'est pas présenté lors de ces vols.
Enfin, le temps d'exécution de notre algorithme est tout a fait utilisable dans notre application.

Chapitre 8

Conclusion

Sommaire

8.1 Les contributions de cette thèse

Les contributions principales de cette thèse sont :

- On dispose maintenant de la fonctionnalité recherchée implémentée dans la caméra, ce qui ouvre la porte aux nouvelles applications qui étaient visées.
- L'image résultante est de bonne qualité radiométrique et géométrique, ce qui prouve qu'on peut maintenant compenser effectivement le flou de bougé sur des images à haute résolution même si le mouvement de la caméra est complexe.
- On a prouvé aussi qu'il est possible d'utiliser l'architecture de la caméra de l'IGN pour accélérer des algorithmes de traitement d'image dans un système embarqué (SoC + FPGA).
- On a confirmé que l'utilisation des capteurs inertiels de bas-coût peut accélérer grandement la phase de suivi des points d'intérêt.
- Les deux IPs qui ont été réalisées peuvent probablement être utilisées dans d'autres applications de vision par ordinateur.

8.2 Perspectives

En partant de l'algorithme développé au cours de cette thèse, des développements supplémentaires permettraient d'étendre les capacités de la caméra légère de l'IGN dans le domaine de la vision nocturne et super-spectrale.
Dans l'avenir, il conviendra d'affiner tous les paramètres de l'algorithme pour optimiser les résultats et éventuellement le temps d'exécution du processus. Il faudra aussi continuer les tests afin de mieux déterminer les limites du procédé.
Il serait intéressant d'exploiter la présence du récepteur GNSS et des accéléromètres

de manière à améliorer la phase de prédiction des positions de points homologues, ce qui permettrait d'utiliser ce procédé sur des drones à ailes fixes. En allant plus loin, on pourrait dans ce cas par le calcul simultané d'un MNS améliorer la qualité de résultats. De plus, il permettrait de mesurer les vraies orientations de la caméra et donc de corriger à chaque prise de vue les biais des gyromètres, ce qui pourrait bénéficier à d'autres capteurs embarqués solidaires de la caméra.
Par extension, ce travail sera appliqué sur des images plein format de résolution 20MP (5120×2840) dans le cadre du projet de réalisation d'un prototype de système d'imagerie aéroporté avec de nombreux canaux spectraux composés de plusieurs caméras IGN synchronisées pour des applications de télédétection.

8.3 Conclusion générale

Cette thèse s'est avérée très enrichissante et m'a apporté beaucoup de choses sur plusieurs niveaux.
Déjà, elle m'a permis d'avoir une première expérience de recherche scientifique dans le domaine algorithmique de traitement d'images et leur implémentation dans le logiciel et le matériel (SoC + FPGA).
Grâce à la forte expérience des ingénieurs du LOEMI, j'ai acquis aussi des compétences techniques en programmation, traitement d'image, FPGA et synthèse haut niveau.
De plus, les congrès auxquels j'ai participé m'ont permis d'échanger avec d'autres chercheurs qui ont travaillé sur des sujets proches des nôtres.
Pendant ma thèse, j'ai eu l'occasion de donner des cours d'informatique aux étudiants en master 1 et 2 à l'ENSG, Cela m'a apporté une nouvelle expérience d'enseignement, des compétences pédagogiques et de communication.

Annexe A

Annexes

Sommaire

A.1 Modélisation de la caméra

Afin de pouvoir effectuer des traitements géométriques à partir d'une image, on a besoin d'un modèle physique qui décrit la projection du monde 3D sur une image 2D. Dans la littérature, il existe un grand nombre de modèles qui décrivent les caractéristiques d'une caméra. Le modèle le plus simple et le plus utilisé en vision par ordinateur et en photogrammétrie est appelé sténopé ou pinhole (Sturm [2014]) illustré dans la figure A.2. Dans ce modèle, tous les rayons lumineux passent par le même point appelé centre optique (ou centre de projection). Il est caractérisé par les paramètres intrinsèques et les paramètres extrinsèques. Parmi les paramètres intrinsèques, on peut citer :

- La distance focale f : la distance entre le centre optique et le plan image (ou encore la distance entre le centre optique et le point principal)
- Le point principal (appelée aussi point d'auto-collimation PPA) : c'est la projection orthogonale du centre optique sur le plan image (ou encore l'intersection de l'axe optique avec le plan image).

A noter que l'axe optique est la perpendiculaire menée du centre optique sur le plan image. Ces paramètres sont appelés internes car ils dépendent uniquement de la caméra et sont estimés au cours du processus de calibration. D'un autre coté, les paramètres extrinsèques représentés par une matrice de rotation de taille (3×3) et un vecteur de translation de taille (3×1) décrivent respectivement l'orientation et la position du centre optique de la caméra dans le repère du monde 3D.
Dans notre démarche, on considère que la caméra est quasi-stationnaire. Une conséquence immédiate de cette hypothèse est qu'on devrait s'intéresser seulement aux paramètres intrinsèques de la caméra : on n'a pas besoin d'estimer la position de la caméra, alors que son orientation relative peut être estimée à l'aide des capteurs inertiels (IMU).

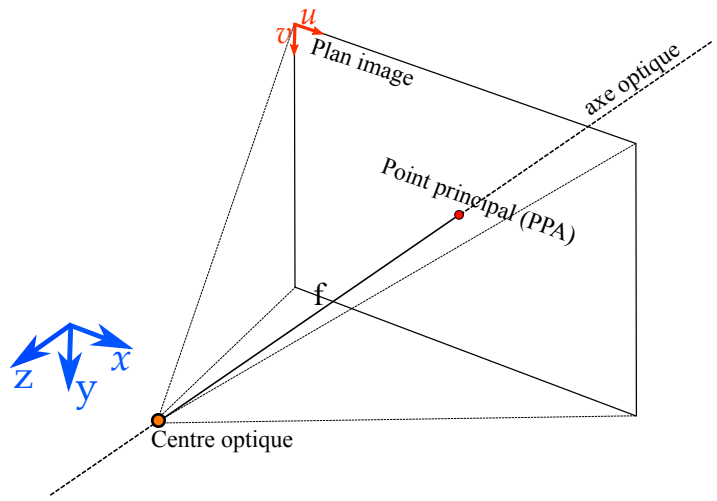


FIGURE A.1 – Le modèle sténopé - les paramètres intrinsèques.

Tous les calculs et les traitements géométriques seront effectués uniquement dans le repère caméra et dans le repère image. En effet, on ne cherchera pas dans notre approche à faire des passages du repère monde au repère caméra. Par contre, on aura besoin de faire des passages du repère caméra 3D au repère image 2D.

A.1.1 Modélisation de la distorsion optique

L'objectif représente le système optique de la caméra. Il est composé d'une ou plusieurs lentilles qui permettent de projeter la scène réelle sur la surface sensible du capteur en faisant converger les rayons lumineux incidents afin de former une image réelle. Cependant, des aberrations liées au fait que les lentilles utilisées ne vérifient pas d'une manière exacte les conditions de gauss (rayons lumineux peu inclinés par rapport à l'axe optique et peu écartés de cet axe), introduisent une distorsion géométrique qui est en général une distorsion radiale (Drap and Lefèvre [2016]), (Wang et al. [2016]), (Hartley and Kang [2007]). Les effets de distorsion radiale les plus rencontrés sont : la déformation en barillet (les lignes droites sont incurvées vers l'extérieur) et la déformation en coussinet (les lignes droites du sujet sont incurvées vers l'intérieur).

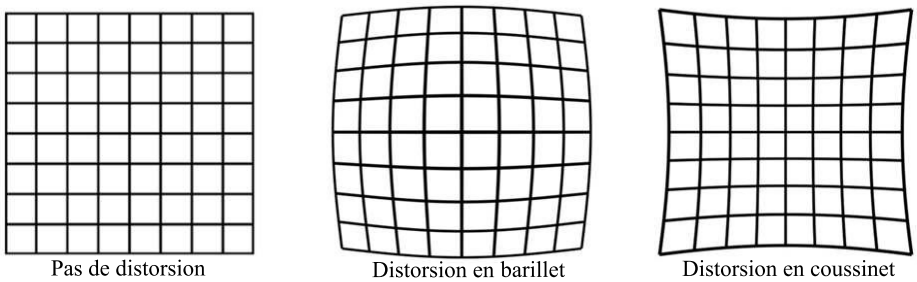


FIGURE A.2 – Les types de distorsion.

Toute fonction radiale est caractérisée par son centre de symétrie et par une fonction de distorsion $\phi(r)$ qui ne dépend que du rayon r. Généralement, $\phi(r)$ peut être modélisée par un polynôme :

$$\phi(r) = 1 + c_1 r^2 + c_2 r^4 + ... \tag{A.1}$$

$$r = \sqrt{(u_c - v_{PPS})^2 + (v_c - v_{PPS})^2} \tag{A.2}$$

$$\mathcal{D}(P_c) = PPS + (P_c - PPS)\phi(r) \tag{A.3}$$

$$P_d = \mathcal{D}(P_c) \tag{A.4}$$

où PPS est le centre de distorsion, P_d est le point distordu, P_c est le point corrigé, $\mathcal{D}$ représente l'application de la distorsion et c_i sont les coefficients de la distorsion radiale.
La distorsion radiale est modélisée généralement sous la forme d'un polynôme pair de degré 6 ($\phi(r) = 1 + c_1 r^2 + c_2 r^4 + c_3 r^6$), ce qui est réputé pour être suffisant. Cependant, il est possible d'utiliser un polynôme de degré 4 ou de degré 8 (il en existe jusqu'à 12). L'application de la distorsion peut être représentée sous la forme suivante :

$$\mathcal{D}\begin{pmatrix} u_c \\ v_c \end{pmatrix} = \begin{pmatrix} u_{PPS} \\ v_{PPS} \end{pmatrix} + (1 + c_1 r^2 + c_2 r^4 + c_3 r^6)\begin{pmatrix} u_c - u_{PPS} \\ v_c - v_{PPS} \end{pmatrix} \tag{A.5}$$

D'un autre coté, il est aussi possible de corriger la distorsion de façon directe en utilisant les paramètres de la distorsion inverse exportés par MicMac. Pour avoir une meilleure précision, le polynôme possède généralement un degré de plus :

$$\phi^{-1}(r) = 1 + c_1^{inv} r^2 + c_2^{inv} r^4 + c_3^{inv} r^6 + c_4^{inv} r^8 \tag{A.6}$$

$$r = \sqrt{(u_d - v_{PPS})^2 + (v_d - v_{PPS})^2} \tag{A.7}$$

$$\mathcal{D}^{-1}(P_d) = PPS + (P_d - PPS)\phi^{-1}(r) \tag{A.8}$$

$$P_c = \mathcal{D}(P_d) \tag{A.9}$$

La correction de la distorsion peut être représentée sous la forme suivante :

$$\mathcal{D}^{-1}\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \begin{pmatrix} u_{PPS} \\ v_{PPS} \end{pmatrix} + (1 + c_1^{inv} r^2 + c_2^{inv} r^4 + c_3^{inv} r^6 + c_4^{inv} r^8)\begin{pmatrix} u_d - u_{PPS} \\ v_d - v_{PPS} \end{pmatrix} \tag{A.10}$$

Les paramètres de distorsion : le centre de distorsion (u_{PPS}, v_{PPS}) et les coefficients de distorsion radiale (c_1, c_2, c_1^{inv}, c_2^{inv}...) sont estimés lors de la calibration intrinsèque de la caméra.
Afin d'illustrer cette distorsion, on a réalisé une carte de distorsion en calculant la distance euclidienne entre le point distordu P_d et le point corrigé P_c (figure A.3). Cette distance représente l'amplitude de la distorsion existante. Pour plus de lisibilité, on a fait le calcul avec un pas de 50 pixels de l'image. On remarque bien sur cette figure la forme de la distorsion radiale qui est d'autant plus importante aux bords qu'au centre où la distorsion est nulle.

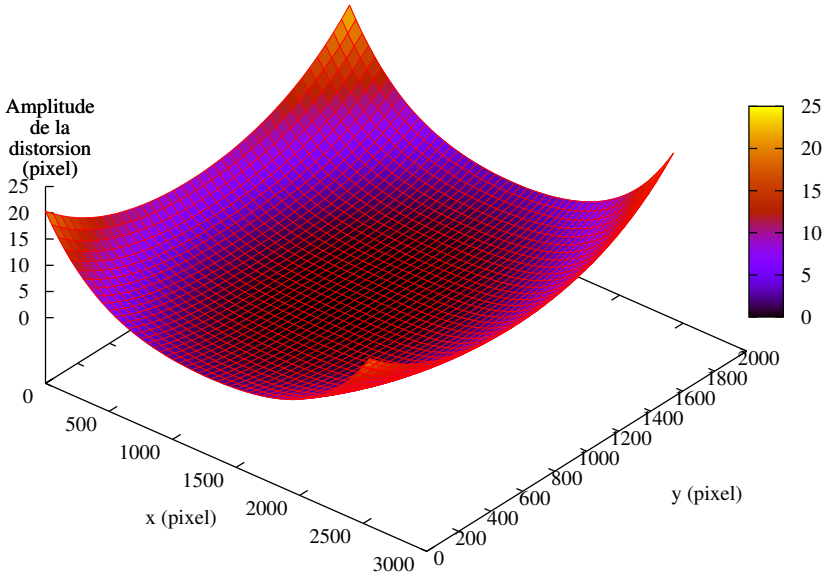


FIGURE A.3 – La carte de distorsion obtenue à partir des paramètres estimés lors de la calibration intrinsèque de la caméra (objectif $35mm$) : l'amplitude de la distorsion est de l'ordre de 20 pixels aux coins de l'image.

A.2 Calibration géométrique de la caméra

La calibration géométrique consiste en général à déterminer les paramètres intrinsèques et extrinsèques du modèle géométrique de la caméra (position, orientation, focale, point principal..) afin d'établir une relation mathématique entre les coordonnées de points 3D de la scène et leurs homologues dans le plan image. Cette étape est obligatoire dans la plupart des études métrologiques qui nécessitent une précision assez fine au niveau des mesures géométriques comme par exemple en photogrammétrie ou en vision par ordinateur.
Pour la raison citée précédemment à propos de la stationnarité du drone, on ne prend en compte que les paramètres intrinsèques de la caméra qui sont principalement : le point principal d'auto-collimation (point d'intersection entre l'axe optique et le plan image), la distance focale et les paramètres de distorsion optique introduits par l'objectif utilisé (centre de distorsion, coefficients de distorsion ..).
Il existe plusieurs méthodes de calibration dans la littérature. La méthode la plus célèbre utilise une mire de calibration généralement constituée de points connus de la scène (DOUSKOS et al. [2007]). Cependant, il existe des méthodes automatiques permettant de calibrer la caméra à partir des points de correspondance entre les images (Furukawa and Ponce [2009]), (Schneider and Förstner [2013]). La première étape de ces méthodes consiste à estimer les positions relatives des caméras à partir des points de correspondance, puis à appliquer l'ajustement des faisceaux (bundle adjustement en anglais) qui permet de minimiser l'erreur de reprojection des points $3D$ sur les points correspondants en question. Ces méthodes assurent en général une bonne estimation de paramètres intrinsèques.
Pour estimer les paramètres intrinsèques de notre caméra, on réalise une calibration de cette dernière à l'aide du logiciel OpenSource de photogrammétrie de l'IGN appelé

MicMac (Pierrot Deseilligny and Clery [2011]). Ce logiciel utilise globalement la méthode automatique de calibration citée ci-dessus. On suit la stratégie suivante avant chaque mission :

- On prend une série d'images (environ 20 à 30) d'une scène contenant du relief (un coin par exemple) et beaucoup de textures. Ces images devraient être centrées sur le même détail en déplaçant le point de vue et en tournant la caméra de 90° de temps en temps.
- On utilise les différentes commandes nécessaires de MicMac pour générer le fichier de calibration contenant la distance focale en pixels, les coordonnées du point principal dans le repère image, les coordonnées du centre de distorsion dans le repère image et les coefficients du modèle de distorsion.

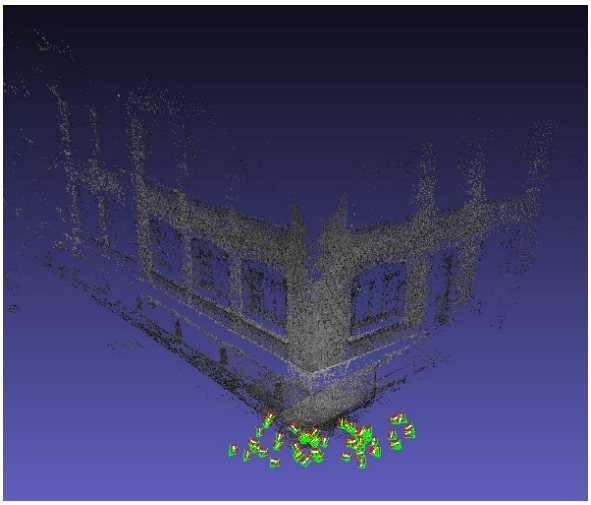


FIGURE A.4 – Les points nuages 3D obtenus à partir de la calibration automatique à l'aide de MicMac. On voit aussi la position et l'orientation de la caméra au moment des prises de vue. A partir de cette calibration, on a un fichier qui contient les paramètres de la distorsion (application de la distorsion + correction de la distorsion) et les paramètres intrinsèques de la caméra.

A.3 Code HLS de la partie ré-échantillonnage

Le code C/C++ du design HLS est le suivant :

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <stdlib.h>
#include <ap_int.h>
#include <assert.h>

#define SIZE_CELL 160
#define NB_IMAGES_MAX 16
#define NB_PARAMS_X 4
```

```
#define SCALE 1024

union UnionPix
{
    uint32_t pixel4;
    uint8_t tab[4];
};

void stacking_per_line(ap_uint<16> width_im,
                       ap_uint<16> height_im,
                       ap_uint<8> nb_images,
                       ap_uint<32> addr_res,
                       ap_uint<32> addr_param,
                       ap_uint<32> addr_imgs[NB_IMAGES_MAX],
                       uint32_t* ram_32){

   #pragma HLS INTERFACE m_axi port=ram_32
   #pragma HLS INTERFACE s_axilite port=addr_imgs
   #pragma HLS INTERFACE s_axilite port=addr_param
   #pragma HLS INTERFACE s_axilite port=addr_res
   #pragma HLS INTERFACE s_axilite port=nb_images
   #pragma HLS INTERFACE s_axilite port=height_im
   #pragma HLS INTERFACE s_axilite port=width_im
   #pragma HLS INTERFACE s_axilite port=return

   UnionPix upix;
   ap_int<24> params[NB_PARAMS_X*NB_IMAGES_MAX];
   #pragma HLS ARRAY_PARTITION variable=params cyclic factor=4 dim=1
   ap_int<24> x, y;
   ap_uint<5> im;
   ap_uint<12> pixel[SIZE_CELL];
   #pragma HLS ARRAY_MAP variable=pixel instance=pix_ctr vertical
   ap_uint<5> cnt_pixels[SIZE_CELL];
   #pragma HLS ARRAY_MAP variable=ctr_pixels instance=pix_ctr vertical
   ap_uint<30> prev_addr28;
   ap_uint<30> addr_res28;

   prev_addr28 = 0;

   for(unsigned i=0; i<NB_PARAMS_X*nb_images; i++){
      #pragma HLS LOOP_TRIPCOUNT min=40 max=40 avg=40
      #pragma HLS_PIPELINE
      params[i] = ram_32[addr_param+i];
   }
   for(im=0; im<nb_images; im++){
      #pragma HLS LOOP_TRIPCOUNT min=10 max=10 avg=10
      for(unsigned i=0; i<SIZE_CELL; i++){
         #pragma HLS PIPELINE
         if(i == 0){
```

```
            x = params[im*NB_PARAMS_X + 0] + SCALE/2;
            y = params[im*NB_PARAMS_X + 2] + SCALE/2;
          }

          ap_uint<32> p = addr_imgs[im] + x/SCALE + (y/SCALE) * width_im;
          ap_uint<30> addr28 = p / 4;

          if(x/SCALE >= 0 &&
             y/SCALE >= 0 &&
             x/SCALE < width_im &&
             y/SCALE < height_im){

            if(addr28 != prev_addr28){
              upix.pixel4 = ram_32[addr28];
              prev_addr28 = addr28;
            }
            pixel[i] += upix.tab[p%4];
            cnt_pixels[i] += 1;
          }
          x += params[im*NB_PARAMS_X + 1];
          y += params[im*NB_PARAMS_X + 3];
       }
    }
    for(unsigned i=0; i<SIZE_CELL; i++){
       #pragma HLS PIPELINE
       upix.tab[i%4] = pixel[i] / cnt_pixels[i];
       if(i%4 == 3){
         ram_32[addr_res+i/4] = upix.pixel4;
         pixel[i] = 0;
         cnt_pixels[i] = 0;
       }
    }
}
```

Remarque : nous n'avons pas implémenté dans le code la fonction augmentation de luminosité parce qu'on voulait voir comparer l'image finale empilée avec la première image durant nos tests.
L'interface du design HLS inclut à la fois les ports d'entrée/sortie et les protocoles utilisés. Les détails de tous les ports (en termes de types, de dimensions et de directions) sont spécifiés dans le fichier source C/C++ :
Le design HLS est composé essentiellement de trois parties :

- La première partie consiste à recopier le tableau des paramètres depuis la RAM CPU dans un bloc RAM FPGA.

- La deuxième partie consiste à traiter la ligne concernée en calculant les paramètres de la partie "x" ($x^{'n} = A_x^n + B_x^n x^n$, $y^{'n} = A_y^n + B_y^n x^n$) de la fonction bilinéaire à partir des paramètres de la partie "y" (A_x^n, B_x^n, A_y^n, B_y^n). Afin de faciliter l'implémentation matérielle, nous avons créé une image virtuelle avec laquelle on débute le calcul de projection des pixels dans toutes les images (la transformation identité est la transformation géométrique entre l'image virtuelle et la

Nom du port	Type et Dims	Type de protocole
$width\_im$	$ap\_uint < 16 >$	$s\_axilite\,(Input)$
$height\_im$	$ap\_uint < 16 >$	$s\_axilite\,(Input)$
nb_images	$ap\_uint < 8 >$	$s\_axilite\,(Input)$
$addr\_res$	$ap\_uint < 32 >$	$s\_axilite\,(Input)$
$addr\_param$	$ap\_uint < 32 >$	$s\_axilite\,(Input)$
$addr\_imgs[nb\_images]$	$ap\_uint < 32 >$	$s\_axilite\,(Input)$
ram_32	$uint32\_t*$	$m\_axi\,(Inout)$

TABLEAU A.1 – *L'interface du design implémenté.*

première image). Nous avons utilisé aussi un tampon de taille "$TAILLE\_BLOC$" ($ap\_uint < 5 > ctr\_pixels[TAILLE\_BLOC]$) dans lequel chaque case contient un compteur (par pixel) qui s'incrémente si le pixel projeté concerné existe dans chaque image de la séquence. Comme décidé avant, on a utilisé l'interpolation au plus proche voisin.

- La troisième partie consiste à écrire dans la mémoire la ligne de pixels traités en mode "burst" sous la forme d'un paquet de 4 pixels par transaction. Ensuite, les deux tampons contenant respectivement les pixels traités et les compteurs sont réinitialisés à zéro.

Le design HLS est traduit en une IP (figure A.5).

FIGURE A.5 – L'IP de stacking et son interface telle qu'elle dans VIVADO.

A.4 Références

DOUSKOS, V., KALISPERAKIS, I., and KARRAS, G. (2007). Automatic calibration of digital cameras using planar chess-board patterns. IV

Drap, P. and Lefèvre, J. (2016). An exact formula for calculating inverse radial lens distortions. *Sensors*, 16(6). II

Furukawa, Y. and Ponce, J. (2009). Accurate camera calibration from multi-view stereo and bundle adjustment. *International Journal of Computer Vision*, 84(3) :257–268. IV

Hartley, R. and Kang, S. B. (2007). Parameter-free radial distortion correction with center of distortion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8) :1309–1321. II

Pierrot Deseilligny, M. and Clery, I. (2011). Apero, an open source bundle adjusment software for automatic calibration and orientation of set of images. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-5/W16 :269–276. V

Schneider, J. and Förstner, W. (2013). Bundle adjustment and system calibration with points at infinity for omnidirectional camera systems. *PFG Photogrammetrie, Fernerkundung, Geoinformation*, 2013(4) :309–321. IV

Sturm, P. (2014). *Pinhole Camera Model*, pages 610–613. Springer US, Boston, MA. I

Wang, Q., Wang, Z., and Smith, T. (2016). Radial distortion correction in a vision system. *Appl. Opt.*, 55(31) :8876–8883. II

Liste des acronymes

AXI Advanced eXtensible Interface

AXIS AXI Stream

CLB Configurable Logic Block

CMOS Complementary Metal-Oxide Semiconductor

DoG Difference of Gaussien

FAST Features from Accelerated Segment Test

FPGAs Field Programmable Gate Arrays

HLS High Level Synthesis

IGN Institut National de l'Information Géographique et Forestière

IMU Inertial Measurement Unit

IP Intellectual Property

LaSTIG Laboratoire des Sciences et Technologies de l'Information Géographique

LOEMI laboratoire d'Optique, Electronique, Métrologie et Instrumentation

LoG Laplacian of Gaussien

LUT Look Up Tables

MEMS Micro Electro Mechanical Systems

NCC Normalized Cross Correlation

NMS Non-Maximal Suppression

PL Programmable Logic

PS Processing System

PSF Point Spread Function

RMS Root Mean Square

SCC Simple Cross Correlation

SIFT Scale Invariant Feature Transform

SoC System On Chip

SSD Sum of Square Differences

SURF Speeded Up Robust Features

SUSAN Smallest Univalue Segment Assimilating Nucleus

UAVs Unmanned Aerial vehicles

ULM Ultra-Léger Motorisé

VHDL VHSIC Hardware Description Language

VHSIC Very High Speed Integrated Circuits

ZNCC Zero Normalized Cross Correlation

Table des figures

Liste des tableaux