Proposition de mécanismes d'optimisation des données pour la perception temps-réel dans un système embarqué hétérogène

Quentin Picard

université PARIS-SACLAY

Proposition de mécanismes d'optimisation des données pour la perception temps-réel dans un système embarqué hétérogène

Data optimization mechanisms for real-time perception in a heterogeneous system

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique
Graduate School : Informatique et sciences du numérique
Référent : Université Evry Val d'Essonne

Thèse préparée dans les unités de recherche Institut LIST (Université Paris-Saclay, CEA) et IBISC (Université Paris-Saclay, Univ Evry) sous la direction de **Jean-Yves DIDIER**, Professeur, le co-encadrement de **Stéphane CHEVOBBE**, Ingénieur Chercheur et **Mehdi DAROUICH**, Ingénieur Chercheur.

Thèse soutenue à Paris-Saclay, le 20 décembre 2023, par

Quentin PICARD

Composition du jury

Samia BOUCHAFA Professeur, Université Paris-Saclay, Univ Evry, IBISC	Présidente
Bertrand GRANADO Professeur, Sorbonne Université, LIP6	Rapporteur
Ezio MALIS Directeur de recherche, Inria, Sophia Antipolis	Rapporteur
David FOFI Professeur, Université Bourgogne Franche-Comté, ImVIA VIBOT	Rapporteur
Jean-Philippe DIGUET Directeur de recherche, CNRS, IRL Crossing	Examinateur
Muriel PRESSIGOUT Maître de conférences, INSA Rennes, IETR	Examinatrice

THESE DE DOCTORAT
NNT : 2023UPASG039

La thèse a été réalisée au sein du Laboratoire d'Intelligence Artificielle Embarquée (LIAE) du CEA LIST en collaboration avec le laboratoire Informatique, BioInformatique et Systèmes Complexes (IBISC) de l'université Evry Paris-Saclay.

Remerciements

Dans un premier temps, je souhaite remercier la présidente du jury, Pr. Samia BOUCHAFA ainsi que Pr. Bertrand GRANADO, DR Ezio MALIS et Pr. David FOFI d'avoir rapporté ma thèse. Je remercie également les examinateurs, DR Jean-Philippe DIGUET et MCF Muriel PRESSIGOUT.

Je suis profondément reconnaissant envers mon directeur de thèse Pr. Jean-Yves DIDIER et mes encadrants Stéphane CHEVOBBE, Mehdi DAROUICH qui ont toujours été à l'écoute et m'ont guidé, recadré, supporté durant ma période de thèse. Leurs conseils, leur engagement et leur patience m'ont permis d'améliorer mon travail, ma réflexion pour terminer ma thèse dans les meilleures conditions.

Je remercie les chercheurs, ingénieurs, doctorants et stagiaires du laboratoire intelligence artificielle embarqué du CEA LIST pour toutes les discussions enrichissantes lors des trois années de thèse. Je souhaite remercier en particulier Thibault ALLENET, Vincent TEMPLIER, Vincent LORRAIN, Samy CHALI, Martyna POREBA, Michal SZCZEPANSKI, Dinh Khanh HO, Karim BENCHEHIDA et Olivier BICHLER.

Je remercie chaleureusement ma femme Inna KUCHER, qui me motive, m'inspire et pour son soutien inconditionnel.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Proposition de mécanismes d'optimisation des données pour la perception temps-réel dans un système embarqué hétérogène

Mots clés : SLAM, localisation, reconstruction 3D, temps-réel, systèmes embarqués, réduction de données

Résumé : Le développement des systèmes autonomes engendre des besoins en perception de l'environnement de plus en plus élevés dans les systèmes électroniques embarqués. Les voitures autonomes, les drones, ou encore les casques à réalité mixte présentent des facteurs de forme limités et un budget de puissance énergétique restreint pour les performances en temps-réel. Par exemple, les cas applicatifs cités ont un budget compris entre 300W-10W, 15W-10W et 10W-10mW respectivement. La thèse se place dans le domaine des systèmes autonomes et mobiles ayant un budget de consommation de 10mW à 15W avec l'utilisation de capteurs d'images et de la centrale inertielle (IMU).

La méthode *Simultaneous Localization And Mapping* (SLAM) fournit aux systèmes autonomes et mobiles une perception précise et robuste de l'environnement en temps-réel et sans connaissances préalables. La thèse a pour objectif de répondre à la problématique de l'exécution temps-réel du système SLAM dans son ensemble composé de fonctions de perception avancées allant de la localisation à la reconstruction 3D sur du matériel à ressources restreintes. Dans ce cadre, deux principales questions sont posées pour répondre aux défis de l'état de l'art. Comment réduire les besoins en ressources de ces fonctions de perception ? Quel est le partitionnement de la chaîne de traitement SLAM pour le système hétérogène intégrant plusieurs unités de calcul allant du circuit intégré au capteur d'image, au traitement proche capteur (FPGA) et dans la plateforme embarquée (ARM, GPU embarqué) ?.

La première question abordée dans la thèse concerne le besoin de réduire les ressources utilisées par la chaîne de traitement SLAM, de la sortie du capteur à la reconstruction 3D. Dans ce sens, les travaux détaillés dans le manuscrit apportent deux principales contributions. La première présente le traitement dans le circuit intégré au capteur en impactant les caractéristiques de l'image grâce à la réduction de la dynamique. La seconde impacte le pilotage du flux d'image envoyé dans la chaîne de traitement SLAM avec un traitement proche capteur. La première contribution vise à réduire l'empreinte mémoire des algorithmes SLAM en évaluant l'impact de la réduction de la dynamique du pixel sur la précision et la robustesse de la localisation et la reconstruction 3D temps-réel. Les expérimentations menées ont montré que l'on peut réduire la donnée d'entrée jusqu'à 75% correspondant à moins de 2 bits par pixel tout en obtenant une précision similaire à la référence 8 bits par pixel. Ces résultats ont été obtenus en évaluant la précision et la robustesse de quatre algorithmes SLAM différents sur deux bases de données.

La seconde contribution vise à réduire la quantité de données injectée dans le SLAM par le filtrage adaptatif qui est la stratégie de décimation pour contrôler le flux d'entrée des images. Le déclenchement des images provenant du capteur est initialement à un flux constant (20 images par seconde). Cela implique une consommation d'énergie, de mémoire, de bande passante et augmente la complexité calculatoire. Pouvons-nous réduire cette quantité de données à traiter ? Dans le cadre du SLAM, la précision et le nombre de calculs à effectuer dépendent fortement du mouvement du système. Grâce à l'IMU qui fournit les accélérations linéaires et angulaires, les données sont injectées en fonction du mouvement du système. Ces images clés sont obtenues grâce à la méthode de filtrage adaptatif (AF). Bien que les résultats dépendent de la difficulté des bases de données choisies, les expérimentations menées ont montré que l'AF permet de décimer jusqu'à 80% des images tout en assurant une erreur de localisation faible et similaire à la référence. L'étude de l'impact mémoire dans le contexte embarqué montre que le pic de consommation est réduit jusqu'à 92%.

Title : Data optimization mechanisms for real-time perception in a heterogeneous system
Keywords : SLAM, localization, 3D reconstruction, real-time, embedded systems, data reduction

Abstract : The development of autonomous systems has an increasing need for perception of the environment in embedded systems. Autonomous cars, drones, mixed reality devices have limited form factor and a restricted budget of power consumption for real-time performances. For instance, those use cases have a budget in the range of 300W-10W, 15W-10W and 10W-10mW respectively. This thesis is focused on autonomous and mobile systems with a budget of 10mW to 15W with the use of imager sensors and the inertial measurement unit (IMU).

Simultaneous Localization And Mapping (SLAM) provides accurate and robust perception of the environment in real-time without prior knowledge for autonomous and mobile systems. The thesis aims at the real-time execution of the whole SLAM system composed of advanced perception functions, from localization to 3D reconstruction, with restricted hardware resources. In this context, two main questions are raised to answer the challenges of the literature. How to reduce the resource requirements of advanced perception functions ? What is the SLAM pipeline partitioning for the heterogeneous system that integrates several computing units, from the embedded chip in the imager, to the near-sensor processing (FPGA) and in the embedded platform (ARM, embedded GPU) ?.

The first issue addressed in the thesis is about the need to reduce the hardware resources used by the SLAM pipeline, from the sensor output to the 3D reconstruction. In this regard, the work described in the manuscript provides two main contributions. The first one presents the processing in the embedded chip with an impact on the image characteristics by reducing the dynamic range. The second has an impact on the management of the image flow injected in the SLAM pipeline with a near-sensor processing. The first contribution aims at reducing the memory footprint of the SLAM algorithms with the evaluation of the pixel dynamic reduction on the accuracy and robustness of real-time localization and 3D reconstruction. The experiments show that we can reduce the input data up to 75% corresponding to 2 bits per pixel while maintaining a similar accuracy than the baseline 8 bits per pixel. Those results have been obtained with the evaluation of the accuracy and robustness of four SLAM algorithms on two databases.

The second contribution aims at reducing the amount of data injected in SLAM with a decimation strategy to control the input frame rate, called the adaptive filtering. Data are initially injected in constant rate (20 frames per second). This implies a consumption of energy, memory, bandwidth and increases the complexity of calculation. Can we reduce this amount of data ? In SLAM, the accuracy and the number of operations depend on the movement of the system. With the linear and angular accelerations from the IMU, data are injected based on the movement of the system. Those key images are injected with the adaptive filtering approach (AF). Although the results depend on the difficulty of the chosen database, the experiments describe that the AF allows the decimation of up to 80% of the images while maintaining low localization and reconstruction errors similar to the baseline. This study shows that in the embedded context, the peak memory consumption is reduced up to 92%.

Publications

Durant la thèse, deux articles ont été publiés.

1. Quentin Picard, Stephane Chevobbe, Mehdi Darouich and Jean-Yves Didier, "Image Quantization Towards Data Reduction : Robustness Analysis for SLAM Methods On Embedded Platforms," 2022 IEEE International Conference on Image Processing (ICIP), Bordeaux, France, 2022, pp. 4158-4162, doi : 10.1109/ICIP46576.2022.9897315.
   Abstract : Embedded simultaneous localization and mapping (SLAM) aims at providing real-time performances with restrictive hardware resources of advanced perception functions. Localization methods based on visible cameras include image processing functions that require frame memory management. This work reduces the dynamic range of input frame and evaluates the accuracy and robustness of real-time SLAM algorithms with quantified frames. We show that the input data can be reduced up to 62% and 75% while maintaining a similar trajectory error lower than 0.15m compared to full precision input images.
2. Quentin Picard, Stephane Chevobbe, Mehdi Darouich, Zoe Mandelli, Mathieu Carrier and Jean-Yves Didier, "Work-in-Progress : Smart data reduction in SLAM methods for embedded systems," 2022 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), Shanghai, China, 2022, pp. 23-24, doi : 10.1109/CASES55004.2022.00018.
   Abstract : Visual-inertial simultaneous localization and mapping methods (SLAM) process and store large amounts of data based on image sequences to estimate accurate and robust real-time trajectories. Real-time performances, memory management and low power consumption are critical for embedded SLAM with restrictive hardware resources. We aim at reducing the amount of injected input data in SLAM algorithms and, thereby, the memory footprint while providing improved real-time performances. Two decimation approaches are used, constant filtering and adaptive filtering. The first one decimates input images to reduce frame rate (from 20 to 10, 7, 5 and 2 fps). The latter one uses inertial measurements to reduce the frame rate when no significant motion is detected. Applied to SLAM methods, it produces more accurate trajectories than the constant filtering approach, while further reducing the amount of injected data up to 85%. It also impacts the resource utilization by reducing up to an average of 91% the peak of memory consumption.

Table des matières

Table des figures

Liste des tableaux

Glossaire

SLAM *Simultaneous Localization And Mapping* - Méthode de perception temps-réel pour fournir aux systèmes mobiles et autonomes une localisation et reconstruction 3D sans connaissance préalable.

COTS *Components off-the-shelf* - Plateforme matérielle standard disponible et prêt à l'emploi pour intégrer un système existant.

FE Le *front-end* (FE) fait partie du module de localisation du SLAM pour le traitement des images et l'estimation de déplacement entre les images consécutives.

BE Le *back-end* (BE) fait partie du module de localisation du SLAM et est responsable de gérer la cohérence topologique via des optimisations de poses et des points 3D de manière locale et/ou globale à partir des données du *front-end*.

VIO *Visual-Inertial Odometry* - Méthode d'odométrie se basant sur les capteurs d'images et les données inertielles. L'odométrie apporte une localisation temps-réel à partir d'une optimisation locale des poses 6DoF.

PI Les points d'intérêts (PI) sont représentés par la détection des coins de l'image et sont exploités dans la chaine de traitement SLAM.

RANSAC *RANdom SAmple Consensus* - RANSAC est une méthode itérative pour optimiser et corriger les valeurs aberrantes issues de l'association des données.

PnP La technique perspective n points (PnP) calcul le déplacement de la caméra par la stratégie 2D-3D qui minimise l'erreur de reprojection entre le point 2D et son homologue 3D.

KF Les *keyframes* (KF) sont les images dont l'information extraite par le *front-end* est utilisée dans l'estimation de pose du *back-end*.

BA Le *bundle adjustment* (BA) est une méthode d'optimisation où les seuls états pris en compte sont les poses, les points 3D, avec comme contraintes les erreurs de reprojection.

factor graph Le *factor graph* modélise la partie d'optimisation du SLAM sous la forme d'un graphe composé d'une multitude de facteurs, tels que les poses, les points, les données inertielles etc.

TSDF *Truncated Signed Distance Field* - Technique pour représenter l'envionnement 3D en grille de voxels à partir des données de profondeur des pixels.

DNN *Deep Neural Network* - Réseau composé de neurones pour de l'apprentissage profond à partir d'une grande quantité de données.

end-to-end La chaine de traitement *end-to-end*, allant de la sortie du capteur à l'estimation de pose repose exclusivement sur les algorithmes DNN pour la localisation à partir d'images consécutives.

CNN *Convolutional Neural Network* - Type d'algorithme d'apprentissage profond pour le traitement des images basé sur l'extraction des caractéristiques et sur la classification des données.

traitement hybride Le traitement hybride utilise des méthodes d'apprentissage profond pour des fonctions spécifiques du *front-end* ou du *back-end*, associées à des méthodes géométriques.

FPGA *Field-Programmable Gate Array* - Circuit intégré reprogrammable pour accélérer les calculs. Le FPGA SoC contient en une plateforme le FPGA et l'architecture de processeur.

SIMD *Single Instruction Multiple Data* - Méthode de calcul pour traiter plusieurs données avec une seule instruction.

ASIC *Application-Specific Integrated Circuit* - Circuit intégré conçu pour une tâche spécifique.

1 - Introduction

Les systèmes autonomes tels que les voitures, robots de service, drones UAV/MAV (*unmanned/micro air vehicle*) et les appareils mobiles RA/RV (réalité augmentée/virtuelle) requièrent une localisation et une reconstruction 3D de l'environnement ambiant précise et robuste pour les fonctions haut-niveaux basées sur l'évitement d'obstacles ou l'interaction avec son environnement physique. Chaque système présente plusieurs contraintes matérielles et logicielles avec un niveau de criticité différent pour un traitement temps-réel. Les voitures autonomes offrent plus d'espace pour inclure de puissants et coûteux matériels de calculs (Liu et al., 2020) que les drones ou les appareils RA/RV où le budget de consommation d'énergie et la robustesse de la localisation sont critiques (Chatzopoulos et al., 2017; Couturier and Akhloufi, 2021). Les applications liées aux drones, telles que la navigation dans des environnements non accessibles aux humains et les opérations de recherche et de sauvetage (Pinies et al., 2006) nécessitent une autonomie à long terme et une précision accrue. Les appareils RA/RV sont plus limités en termes d'espace libre et les applications ne sont plus limitées aux jeux vidéos et au divertissement, mais à des applications nécessitant une localisation précise en temps-réel dans les domaines médical, industriel ou éducatif (Garg et al., 2023).

Perception temps-réel

Pourquoi la perception est si importante ? La perception des systèmes mobiles et autonomes permet de comprendre ce qui est présent dans l'environnement ambiant nécessaire aux applications haut-niveaux comme l'évitement des obstacles, la planification d'une trajectoire ou encore la prise de décision de manière précise et en toute sécurité. Pour cela, les algorithmes de perception prennent en compte l'information issue de nombreux capteurs proprioceptifs et extéroceptifs disponibles permettant d'acquérir les données nécessaires à la compréhension de ce qui entoure le système, comme les caméras, les centrales inertielles, les lidars, etc. Les différents cas d'usage étudiés dans la thèse imposent le traitement des données issues des capteurs en temps réel. Mettre en place un système de perception robuste et précis met en évidence de nombreuses limitations dans un système réel d'un point de vue logiciel et matériel. Les ressources de calcul à notre disposition imposent des contraintes sur les algorithmes de perception d'un point vue complexité calculatoire pour avoir une exécution temps réel et à faible consommation d'énergie.

Les algorithmes conçus deviennent de plus en plus avancés pour obtenir une compréhension riche de l'environnement incluant des fonctions de base pour l'ego-localisation jusqu'à la reconstruction 3D. La méthode *Simultaneous Localisation And Mapping* (SLAM) est un domaine de recherche actif et largement utilisée par la communauté pour fournir une localisation et une reconstruction précise et robuste en temps-réel de l'environnement sans connaissance préalable. Le SLAM se reflète en trois principales questions (Davison, 2018) : où suis-je ? (partie localisation), comment est mon environnement ? (partie reconstruction) et quels sont les objets autour de moi ? (partie segmentation). Le principal défi du SLAM est d'obtenir une cohérence globale. L'utilisation de mesures relatives issues des capteurs visuels et inertiels entraîne une accumulation graduelle de l'incertitude et l'effet de dérive est visible au cours du temps. Les méthodes SLAM incluent un module d'optimisation

responsable pour la cohérence locale et globale. La détection d'une fermeture de boucle permet de corriger la dérive lorsque la scène reconstruite a déjà été visitée. Suivant la technique utilisée, le SLAM permet d'avoir plusieurs types de reconstruction, le nuage de points, les *surfels* (points avec l'information d'orientation), le maillage triangulaire et la reconstruction volumétrique basée sur les voxels. Le choix de la reconstruction 3D dépend du type d'application visé. Plus on souhaite interagir avec les éléments de la scène et d'avoir une représentation plus riche, plus la reconstruction doit être fine et exploitable géométriquement. C'est le cas avec la reconstruction volumétrique qui permet d'avoir une compréhension géométrique de l'espace utilisé par ce qui entoure le système.

Tandis que les méthodes SLAM visuelles et inertielles prennent en considération les capteurs d'images et les mesures de la centrale inertielle, plusieurs axes de recherche utilisent d'autres capteurs. Dans les références (Alliez et al., 2020a) et (Alliez et al., 2020b), un système multi-capteurs basé sur le LiDAR (Zhang and Singh, 2014) et une caméra monoculaire infrarouge (Kachurka et al., 2019) est utilisé pour la localisation en temps-réel. Dans la référence (Vidal et al., 2018), une chaine de traitement SLAM combine le capteur basé évènement, la caméra visible et la centrale inertielle. Les capteurs évènementiels permettent de surpasser les limitations des caméras visibles contre les mouvements rapides ou les changements d'intensités. A la place de capturer directement l'intensité lumineuse, ils acquièrent le changement d'intensité de la scène (Gallego et al., 2020). Aujourd'hui, ils sont principalement utilisés en complément d'autres capteurs pour prendre en compte la quantité d'information riche que fournissent les caméras visibles.

Système embarqué hétérogène

L'implémentation matérielle des méthodes SLAM pose de nombreuses problématiques pour le calcul temps-réel en raison des ressources restreintes disponibles sur les matériels embarqués en termes de puissance de calculs, d'utilisation mémoire et de consommation d'énergie. Les systèmes tels que les drones et les casques RA/RV nécessitent l'intégration des fonctions de perception avec un budget de consommation d'énergie de 10W à 15W et de 10mW à 10W respectivement. Comment réduire les besoins en ressources de ces fonctions de perception tout en assurant une précision de localisation et de reconstruction suffisante ? Quelle est la granularité de la reconstruction 3D qui pose la plus grande problématique pour l'implémentation dans un système embarqué ? Les travaux de la thèse permettent de répondre à ces questions et ont pour objectif est de minimiser la consommation des traitements effectués par le système SLAM dans son ensemble avec une représentation 3D riche et complexe tout en ayant une exécution temps réel à faible consommation d'énergie.

Les architectures hétérogènes sont de plus en plus utilisées dans les appareils électroniques permettant de distribuer et de partitionner les charges de calculs sur différentes plateformes. Le système embarqué hétérogène est défini comme l'utilisation de plusieurs unités de calculs allant du traitement très proche capteur à la plateforme embarqué contenant plus de ressources disponibles avec une implémentation plus ou moins flexible. Il est commun d'étudier des travaux portant sur l'accélération matérielle de certaines fonctions complexes du SLAM pour réduire le coût calculatoire de la plateforme principale. Quel est le partitionnement de la chaine de traitement SLAM qui permet d'avoir une implémentation efficace en termes de latence et de consommation des ressources à partir d'un système hétérogène intégrant la co-conception logicielle et matérielle ? Un système hétérogène est

proposé permettant de répondre aux besoins des systèmes de perception, soit l'exécution temps réel à faible consommation d'énergie avec une perception riche, exploitable pour les fonctions haut-niveaux, précise et robuste.

Contributions

Les travaux de la thèse visent à monter une chaine de traitement pour avoir une perception précise et riche, tout en prenant en compte les problématiques liées à l'implémentation matérielle à ressources restreintes. Deux contributions sont proposées dans la thèse et ont pour but d'optimiser les données qui transitent dans le système hétérogène. Ce dernier est défini comme un système composé de plusieurs unités de calcul. La première contribution vise à réduire l'empreinte mémoire des algorithmes SLAM en évaluant l'impact de la réduction de la dynamique du pixel sur la précision et la robustesse de la localisation et reconstruction 3D temps-réel (Picard et al., 2022a). Le but est de définir les méthodes de réduction de la dynamique qui permettent d'avoir ce compromis entre précision et consommation mémoire sur différents cas d'usage. La deuxième contribution vise à optimiser la quantité de données injectée dans le SLAM par le filtrage adaptatif qui est la stratégie de décimation pour contrôler le flux d'entrée des images (Picard et al., 2022b). Le déclenchement des images provenant du capteur est initialement à flux constant (20 images par seconde). Cela implique une consommation d'énergie, de mémoire, de bande passante et augmente la complexité calculatoire. Pouvons-nous réduire cette quantité de donnée à traiter ? Dans le cadre du SLAM, la précision et le nombre de calculs à effectuer dépend fortement du mouvement du système. Grâce aux mesures inertielles qui fournit les accélérations linéaires et angulaires, les données sont injectées en fonction du mouvement du système.

Organisation de la thèse

La thèse est répartie en six chapitres :

- Le chapitre 2 présente l'analyse de la littérature allant de la compréhension globale du SLAM d'un point de vue théorique vers les implémentations pratiques sur matériels embarqués. Une chaîne de traitement conventionnelle est détaillée allant de l'acquisition de l'image pour la localisation temps-réel jusqu'à la reconstruction 3D de l'environnement par nuage de points, maillage et voxels. A partir des travaux présents dans l'état de l'art, un système hétérogène composé de plusieurs unités de calculs est détaillé avec l'implémentation et le partitionnement des fonctions de perception dont le compromis entre les performances matérielles et logicielles est analysé et discuté.
- Le chapitre 3 présente les différents leviers pour réduire la complexité de la chaine de traitement SLAM et détaille la proposition du système hétérogène comprenant les unités de calculs proches capteur et dans la plateforme embarquée de type produit informatique standard (*Components off-the-shelf* - COTS). Les contributions de la thèse sont détaillées permettant de réduire la complexité du système SLAM dans son ensemble à partir de mécanismes d'optimisation des données pour la perception temps-réel.
- Le chapitre 4 détaille la méthodologie employée pour l'analyse et l'impact des mécanismes proposés. Les jeux de données, outils, métriques et algorithmes utilisés sont présentés et la

procédure de recherche réalisée pour les expérimentations est expliquée.

- Le chapitre 5 décrit les résultats qui se concentrent sur le compromis entre précision, robustesse et consommation mémoire des algorithmes SLAM allant de la localisation à la reconstruction 3D. Le chapitre présente une discussion sur le partitionnement des fonctions de perception et l'impact des mécanismes d'optimisation proposés sur le système.
- Pour finir, le chapitre 6 conclut les travaux de la thèse et donne la perspective des contributions et discute plus généralement du domaine dans le contexte embarqué.

2 - Analyse de la littérature sur le SLAM embarqué

Sommaire

La communauté SLAM a apporté de remarquables améliorations sur la précision et la robustesse des applications à grande échelle durant les dernières années (Cadena et al., 2016; Stachniss et al., 2016; Rosen et al., 2021), des approches probabilistes et association de données (Durrant-Whyte and Bailey, 2006; Bailey and Durrant-Whyte, 2006) jusqu'à l'utilisation des algorithmes d'apprentissage profond (Chen et al., 2020; Li et al., 2021). Les différentes parties du SLAM incluant les capteurs la localisation et la reconstruction 3D dans un contexte embarqué ont été intensivement étudiées (Salhi et al., 2019; Picard et al., 2021). Les études ont pour but de fournir une solution robuste à plusieurs applications, comme la conduite autonome (Bresson et al., 2017), les tâches de recherche et secours, les inspections d'infrastructure et la reconstruction 3D dans les environnements statiques et dynamiques avec conditions difficiles (Zollhöfer et al., 2018; Alkendi et al., 2021). La robustesse des méthodes temps-réel sous les conditions difficiles a été étudiée et quantifiée incluant la faible visibilité (Alkendi et al., 2021), les mouvements dynamiques, les changements d'illuminations, les changements de points de vue et les scénarios long-terme (Bujanca et al., 2021). Les expérimentations de la référence (Bujanca et al., 2021) montrent que les approches de l'état de l'art ont du mal avec ces conditions difficiles. Elles montrent également que la méthode SLAM basée sur l'extraction de points d'intérêts fournit le meilleur compromis en termes de robustesse. Cependant, l'extraction de points d'intérêts pour les méthodes basées vision accentue le manque de flexibilité à cause de la dépendance d'un certain type d'extraction et de la localisation difficile lors de la présence de bruit (Azzam et al., 2020). De la localisation temps-réel à la reconstruction 3D, les problématiques du SLAM ont été étudiées en utilisant également les approches basées apprentissage (Chen et al., 2020). L'évolution des réseaux de neurones profonds (DNN) et leur impact sur le SLAM ouvre plusieurs directions de recherche incluant le type de modèle, la scalabilité et le déploiement matériel.

Ce chapitre étudie la littérature du SLAM de la localisation à la reconstruction 3D en prenant en compte une chaine de traitement permettant d'avoir une perception précise et riche. Cela implique d'avoir une représentation de la scène détaillée et exploitable pour des fonctions haut-niveaux. L'analyse de la littérature est orientée dans le contexte embarqué de l'implémentation des fonctions avancées sur les plateformes matérielles à ressources restreintes. La thèse prend en compte les méthodes utilisant les capteurs d'images à faible coût et à faible consommation d'énergie et les mesures inertielles. Le chapitre est réparti en quatre sections : la section 2.1 détaille la chaine de traitement SLAM d'un point de vue théorique de la localisation à la reconstruction 3D en temps-réel. La section 2.2 met en évidence les travaux de la littérature sur les implémentations embarquées des fonctions de perception à plusieurs niveaux ayant du moins au plus de ressources disponibles, le traitement dans le capteur, proche capteur et dans le système embarqué de type COTS (*Components off-the-shelf*). Les solutions d'optimisation utilisées pour réduire la complexité calculatoire des algorithmes sont détaillées dans la section 2.3. Pour finir, les problématiques et questions de recherches sont discutées dans la section 2.4.

2.1. Chaine de traitement SLAM du capteur à la reconstruction 3D

Les méthodes SLAM ont évolué considérablement ces dernières années dans le but d'obtenir une cohérence spatiale de la scène plus robuste et plus précise. La figure 2.1 illustre la chaine de traitement conventionnelle du SLAM en utilisant les capteurs d'images et les mesures inertielles. On retrouve deux principaux modules, le module de localisation et le module de reconstruction 3D. Le premier est composé d'un *front-end* (FE) et d'un *back-end* (BE). Le FE traite les images et estime le déplacement entre les images consécutives. Le BE utilise la pré-intégration des mesures inertielles (Forster et al., 2017a) et gère la cohérence topologique via les optimisations des poses et des points 3D de manière locale et/ou globale (Engels et al., 2006). Le module de reconstruction 3D fournit un modèle de la scène et les propriétés géométriques de l'environnement ambiant. La reconstruction par le nuage de point permet d'avoir une représentation plus ou moins dense de la scène, éparse (Campos et al., 2021), semi-dense (Forster et al., 2017b) et dense (Engel et al., 2014). La plupart des méthodes se concentrent sur un système robuste et précis pour suivre le mouvement de la caméra localement. Le nuage de point est défini comme étant non structuré, non ordonné, sans échelle et représenté par seulement les coordonnées. Un nuage de point est alors une représentation suffisante pour fournir la précision de localisation requise. La reconstruction apporte une amélioration à la perception de l'environnement. Les travaux de la thèse visent à étudier plus en profondeur l'utilisation d'une représentation 3D dense, géométrique et exploitable pour les fonctions haut-niveaux nécessitant une interaction avec la scène que le nuage de point épars. Plus la représentation est dense, plus la complexité calculatoire est importante. C'est notamment le cas pour une reconstruction dense du nuage de point où tous les pixels de l'image sont pris en compte contrairement au nuage de point éparse où seulement les points d'intérêts de l'image d'entrée sont utilisés. La reconstruction par maillage permet une représentation 3D de l'environnement basée sur la triangulation 3D du FE et des poses 3D issues du BE. Cette représentation est utile pour les fonctions d'évitements d'obstacles. L'interaction physique avec l'environnement requiert une reconstruction volumétrique fournit par le calcul basé voxels et le rendu par un maillage des surfaces. Elle prend en entrée la carte de profondeur de l'estimation de

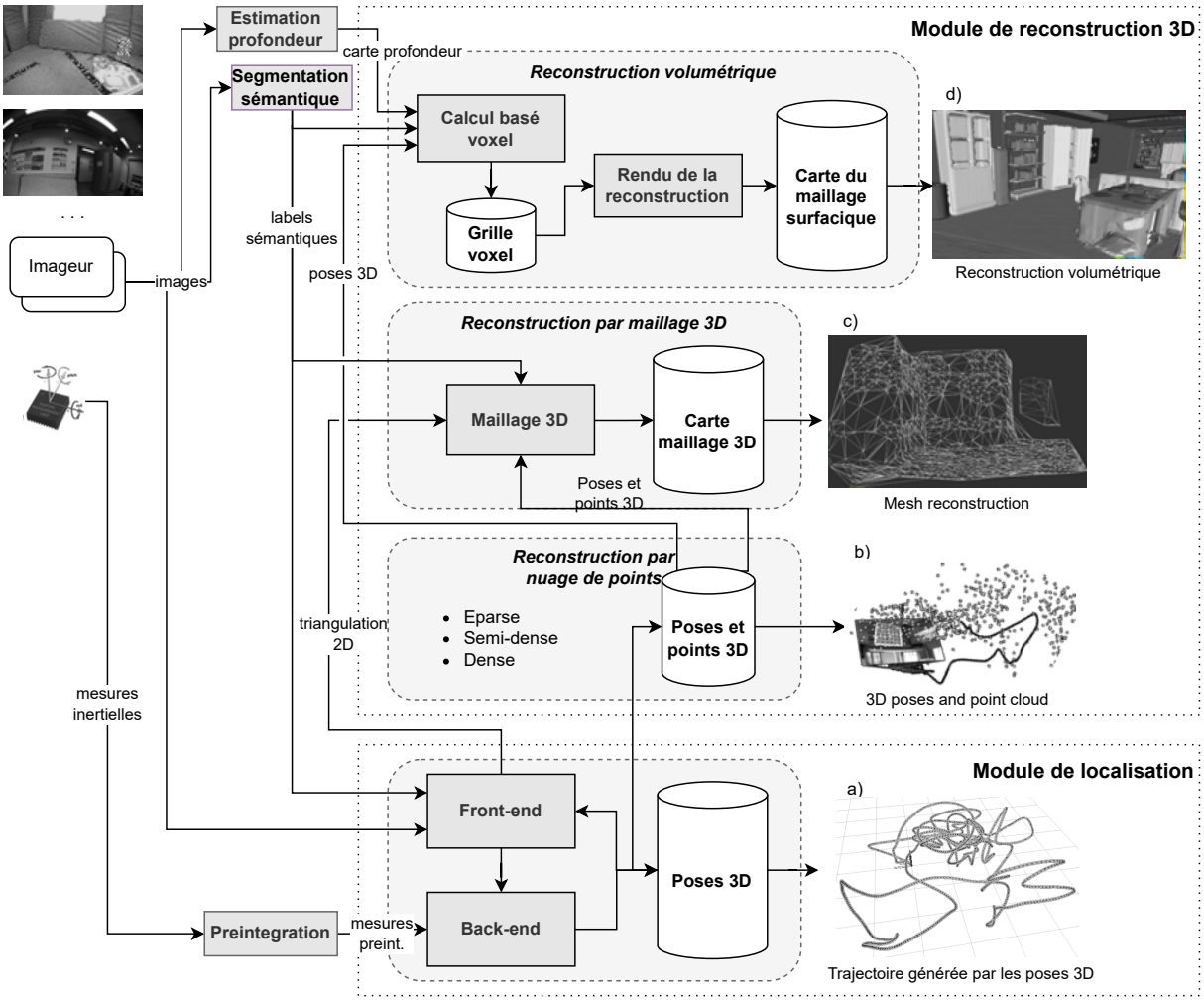


FIGURE 2.1 – Chaine de traitement conventionnelle du SLAM à partir de l'imageur et de la centrale inertielle. a) représente la trajectoire précise générée par les poses 3D. b) représentation de la scène par nuage de points éparse avec la trajectoire issue des poses 3D. c) correspond à la visualisation du maillage 3D (Greene and Roy, 2017). d) est la reconstruction volumétrique basée voxels (Rosinol et al., 2020).

profondeur de l'image et les poses 3D du BE. La segmentation sémantique donne l'information sur les objets autour du système en mouvement (Garg et al., 2020; Kostavelis and Gasteratos, 2015; Zhou et al., 2019). Par exemple, dans le cas d'images RGB, la sémantique correspond à la classification labellisée de chaque pixel, utilisée pour la reconstruction volumétrique, le maillage et le nuage de point.

2.1.1. Module de localisation

La reconstruction 3D temps-réel pour les robots mobiles et casques RA/RV dans un environnement inconnu requiert une localisation précise que le SLAM (Campos et al., 2021; Engel et al., 2014; Davison, 2003; Davison et al., 2007) et les méthodes d'odométrie visuelle(-inertielle) (VO, VIO) fournissent à partir de mesures relatives (Rosinol et al., 2020; Qin et al., 2018; Mourikis and Roumeliotis, 2007). Cette partie décrit le fonctionnement de la partie de localisation temps-réel de la chaine de traitement SLAM.

Front-end (FE)

Le *front-end* traite les images d'entrée. Les techniques indirectes et directes sont les deux principales approches et sont respectivement décrites ci-dessous :

- Les méthodes indirectes consistent à détecter les points d'intérêts (PI), à les mettre en correspondance ou les suivre entre deux images consécutives et à estimer le déplacement à partir des observations avec la vérification géométrique basée sur un ensemble de points issus de l'algorithme *random sample consensus* (RANSAC) (Nister, 2004; Horn, 1987; Laurent Kneip and Siegwart, 2011). La détection des points d'intérêts permet d'extraire majoritairement les angles sur une pyramide gaussienne à plusieurs niveaux n, générée à partir des images en niveaux de gris. Le processus de la pyramide gaussienne à plusieurs niveaux consiste à réduire la taille de l'image à travers n niveaux avec le même facteur (640×480 à 80×60 sur quatre niveaux avec un facteur 2) (Adelson et al., 1984). Dans la référence (Campos et al., 2021), un facteur d'échelle de 1.2 à travers 8 niveaux a été configuré par défaut. Cette technique extrait les points d'intérêts invariants à l'échelle. Plusieurs méthodes de détection ont été développées depuis Moravec (Moravec, 1977) qui offrent un compromis entre la robustesse et la complexité de calcul (Harris and Stephens, 1988; Jianbo Shi and Tomasi, 1994; Rosten and Drummond, 2006). La mise en correspondance des points d'intérêts inclut la détection et la description des PI pour mettre en correspondance les points sur les images consécutives (LoweDavid, 2004; Bay et al., 2006; Rublee et al., 2011) alors que le suivi de points consiste à estimer le déplacement des PI détectés sans description grâce à l'algorithme KLT (yves Bouguet, 2000). La prochaine étape consiste à estimer le déplacement entre deux images et à réaliser la vérification géométrique avec RANSAC pour la suppression de données aberrantes. Suivant l'approche utilisée, les PI sont spécifiés en deux ou trois dimensions pour les techniques 2D-2D, 3D-3D et 3D-2D. Le premier cas 2D-2D estime le déplacement à partir des points extraits de l'image en 2D. Cette méthode est généralement réalisée durant la phase d'initialisation en utilisant les contraintes épipolaires pour obtenir les relations géométriques entre deux images consécutives (Nister, 2004; Laurent Kneip and Siegwart, 2011; Civera et al., 2009). Dans le cas de l'estimation de déplacement 3D-3D, les PI à considérer sont basés sur la triangulation des points 3D en utilisant, par exemple, les caméras visibles en stéréovision. L'estimation 3D-3D est basé sur la minimisation des distances euclidienne entre les points correspondants 3D. Pour finir, l'approche d'estimation de déplacement 3D-2D est basé sur la perspective n points (PnP) qui minimise l'erreur de reprojection entre le point 2D et son homologue 3D décrit dans l'équation 2.1.

$$\mathbf{T}_k = \underset{T}{argmin} \sum_i \|u'_i - \pi(p_i)\|^2 \tag{2.1}$$

TABLE 2.1 – Description, avantages et inconvénients des approches directes et indirectes utilisées dans les méthodes de l'état de l'art

	Approche indirecte	Approche directe
Description	- Extraction de points d'intérêts (PI) - Suivi ou mise en correspondance des PI - Minimisation de l'erreur de reprojection	- Utilisation de tous les pixels - Minimisation de l'erreur photométrique
Avantages	- Rapide et robuste en raison de l'extraction des PI - Permet une reconstruction éparse à faible complexité	- Plus robuste et précis que l'indirecte - Permet une reconstruction dense de la scène
Inconvénients	- Dépendant de la qualité des PI - Requiert une estimation robuste pour pallier aux valeurs aberrantes	- Complexité élevée en raison de l'estimation du flot optique - Requiert une initialisation précise - Requiert une calibration photométrique précise

avec $\mathbf{T}_k$ la matrice de transformation entre le point de vue actuel et celui d'origine, u'_i le PI détecté, π la fonction de projection et p_i le point 3D correspondant. L'estimation de déplacement 3D vers 2D est plus précise que l'estimation 3D-3D à cause de l'incertitude générée par la triangulation des points 3D (Nister et al., 2004).

- Les méthodes indirectes extraient et utilisent les points d'intérêts pour l'estimation de pose, les approches directes utilisent tous les pixels disponibles de l'image (Engel et al., 2014; Newcombe et al., 2011a; Engel et al., 2018; Cremers, 2017). Cette technique ne contient pas d'étape de détection de PI. Elle estime le flot optique avec la minimisation de l'erreur photométrique de l'équation 2.2.

$$\mathbf{T}_{k,k-1} = \underset{T}{argmin} \sum_i \|I_k(u'_i) - I_{k-1}(u_i)\|^2 \tag{2.2}$$

avec $\mathbf{T}_{k,k-1}$ la matrice de transformation entre deux images, $I_k(u'_i)$ l'intensité I du pixel u'_i de l'image k. En présence d'une grande quantité de gradients d'intensités issus de l'image, les approches directes ont une complexité calculatoire importante. Les représentations denses avec l'alignement direct d'images utilisent du matériel GPU pour atteindre une performance temps-réel (Newcombe et al., 2011a; Pizzoli et al., 2014). Afin de réduire le temps de calcul, le filtrage de profondeur semi-dense a été développé (Engel et al., 2013) sur CPU (Engel et al., 2014; Engel et al., 2015). Dans les travaux de (Engel et al., 2018), un modèle direct et épars utilise seulement une sélection de point qui sont adéquatement distribués dans l'image avec une amplitude du gradient élevée.

La table 2.1 résume les avantages et les inconvénients des approches indirectes et directes. Les méthodes indirectes permettent plus de robustesse et d'efficacité en termes de calcul grâce à une détection éparse des points d'intérêts. Cependant, elles reposent exclusivement sur la détection de PI et requièrent une estimation robuste pour éviter les données aberrantes qui nuisent au processus de localisation. Les méthodes directes ont l'avantage d'estimer le déplacement par la minimisation de l'erreur photométrique. Cependant, elles requièrent une calibration photométrique précise. Par exemple, dans (Engel et al., 2018), le temps d'exposition, une fonction de réponse non-linéaire et la

vignette de la lentille du capteur qui permet une compréhension du modèle de la transformation de luminosité sont pris en compte.

Afin de combiner les avantages des approches directes et indirectes, les méthodes semi-directes exploitent les pixels avec un fort gradient tout en s'appuyant sur des points d'intérêts épars (Dong et al., 2021; Forster et al., 2017b). Le traitement temps-réel pour ces approches est atteint avec l'utilisation de techniques basées *keyframes* (KF) pour fournir un équilibre entre la précision et l'efficacité (Klein and Murray, 2007). Le module *front-end* estime si une image d'entrée est considérée comme étant une KF pour l'optimisation *back-end*. Les poses et les points 3D sont produits dans le temps par ces *keyframes* pour obtenir une cohérence locale et/ou globale. Les KF sont générées à un flux plus faible que les images d'entrées, par exemple 8 à 16 ips pour 30 à 60 ips pour le flux d'entrée respectivement.

Back-end (BE)

Les premiers travaux traitant des méthodes SLAM étaient composés d'un filtre de Kalman étendu (EKF) (Leonard et al., 1990; Smith et al., 1990; Leonard and Durrant-Whyte, 1991), d'un *back-end* séparé pour l'ajustement de faisceau ou plus communément appelé *bundle adjustment* (BA) (Klein and Murray, 2007) et sous la forme d'un graphe (Folkesson and Christensen, 2004; Frese and Schroder, 2006; Dellaert and Kaess, 2006). L'optimisation BE est une étape cruciale pour obtenir une estimation de pose précise en fonction des nouvelles mesures en entrée. Cette étape consiste à optimiser la carte composée de poses et de points 3D de manière globale ou locale. Les défis de l'optimisation adressés dans cette sous-section inclus la manière dont les mesures inertielles sont injectées dans le BE, les approches utilisées pour optimiser les poses et les points 3D de la carte, les fonctions qui fournissent une cohérence globale de la carte, qui est un sujet fondamental dans le SLAM (Cadena et al., 2016; Aulinas et al., 2008).

Dans un système visuel-inertiel, les mesures inertielles sont pré-intégrées pour améliorer la localisation en résolvant le problème de facteur d'échelle par l'accélération linéaire et angulaire entre chaque image. Les centrales inertielles fournissent les données à un flux important (100Hz à 1kHz) comparé au flux d'images (20-60 ips). Pour faire face à la différence d'acquisition, la méthode de pré-intégration est incluse dans la chaine de traitement de la localisation en temps-réel qui analyse et consolide toutes les données inertielles entre deux *keyframes* en une seule mesure (Forster et al., 2017a). Cette technique utilise une approche dite sans structure qui évite d'optimiser les points 3D pour améliorer le temps d'exécution durant la phase d'optimisation.

Afin de fusionner les données visuelles et inertielles et d'optimisation les poses générées par le module de localisation, deux principales approches sont utilisées : les approches basées filtrage et par graphe (*factor graph*) (Gui et al., 2015). Dans un système basé filtrage, seulement la dernière pose est estimée. Le filtre de Kalman étendu fusionne les données issues des différents capteurs, prédit l'état futur en relation avec l'estimation initiale et met à jour la prédiction. L'incertitude est représentée par l'utilisation de la matrice de covariance. La complexité de cette approche augmente au fur et à mesure du temps, car la carte générée et les points 3D estimés sont plus nombreux. La consommation mémoire augmente significativement lors de longues trajectoires en plus du coût calculatoire. Le traitement temps-réel est atteint en prenant en compte seulement un nombre limité de points d'intérêts (Davison et al., 2007). Afin d'adresser le coût calculatoire, les erreurs photométriques ont été utilisées dans la mise à jour de l'EKF et emploie une distance numérique minimale pour la

représentation des points d'intérêts. Des méthodes alternatives basées sur le *Multi-State Constraint Kalman Filter* (MSCKF) utilise une stratégie sans structure qui ne prend pas en compte les points 3D (Mourikis and Roumeliotis, 2007; Sun et al., 2018; Geneva et al., 2020).

L'autre technique d'optimisation est le *factor graph* (Dellaert and Kaess, 2017) qui représente tous les états, les points et les données reliés sous la forme d'un graphe non-linéaire (Dellaert and Others, 2019; Kaess et al., 2012; Kümmerle et al., 2011; Agarwal et al., 2022). La non-linéarité du graphe est résolue par une optimisation locale (*fixed-lag smoothing*) ou globale (*full smoothing*) des états et des points 3D tout en exploitant la parcimonie des algorithmes SLAM. Dans l'optimisation locale, seules les poses incluses dans une fenêtre de temps glissante sont optimisées alors que dans la configuration globale, tout l'historique des poses est pris en compte. L'optimisation basée sur le *factor graph* est décrit de la manière suivante :

1. Linéarisation des facteurs du graphe dans le système d'équation linéaire (mesures inertielles, visuelles, etc.)
$$H\Delta x = \varepsilon \tag{2.3}$$
avec H la matrice hessienne, ε est le vecteur qui décrit comment les mesures du FE affectent l'état de chaque *keyframes* et Δx le vecteur qui décrit les états mis à jour.
2. Utilisation de la factorisation Cholesky et *back-substitution* pour résoudre le système d'équation linéaire.
3. Marginalisation des états en dehors de la fenêtre glissante pour l'optimisation locale permettant de maintenir les performances temps-réel.
4. Mise à jour des états des *keyframes* restants à partir des solutions du système linéaire.

Dans (Usenko et al., 2020), un ensemble d'information non-linéaire visuel-inertiel sur l'estimation de déplacement entre les *keyframes* sont récupérés à partir des premières informations de l'odométrie et combinés en utilisant l'optimisation par *bundle adjustment* global. Basé sur la méthode d'optimisation non-linéaire (Qin et al., 2018), VINS-Fusion supporte l'utilisation de plusieurs capteurs intégrés dans la structure du graphe d'optimisation par la méthode du maximum de vraisemblance. Dans (Leutenegger et al., 2015), l'optimisation non-linéaire intègre les erreurs de reprojection et le terme d'erreur temporel à partir des mesures inertielles tout en supprimant les anciennes KFs de la fenêtre d'optimisation pour assurer un traitement temps-réel. Le module d'odométrie Kimera-VIO (Rosinol et al., 2020) est basé sur la pré-intégration des mesures inertielles (Forster et al., 2017a) et fournit des capacités SLAM avec le module *pose graph optimization* responsable des fermetures de boucles.

La cohérence globale du SLAM est complétée avec la technique de fermeture de boucle pour détecter qu'une scène a déjà été visitée et pour corriger l'accumulation de la dérive. Sans cette technique, le SLAM visuel devient l'odométrie visuelle pour une cohérence locale (Cadena et al., 2016). La fermeture de boucle est réalisée en trois principales étapes :

1. Détection des candidats entre les nouveaux points d'intérêts créés à partir des poses et la carte actuellement active.
2. Correction des poses détectées qui sont affectées par la fermeture de boucle
3. Optimisation de la carte afin de vérifier si l'accumulation de la dérive a bien été corrigée

La méthode dominante pour la fermeture de boucle est l'utilisation d'un vecteur de *bag-of-words* (BoW). Il implémente une base de donnée de reconnaissance d'un vocabulaire de mots visuels décrits par les points d'intérêts de l'image (Galvez-López and Tardos, 2012). La base de donnée est interrogée pour trouver les potentiels candidats. En termes de temps d'exécution sur PC de bureau, la fermeture de boucle du système ORB-SLAM3 (Campos et al., 2021) prend environ 10ms pour la détection, 124.77ms pour la correction, qui inclut la fermeture de boucle et la correction sur toute la carte et 1529.69ms pour l'optimisation de la carte entière. Afin de maintenir des performances temps-réel, la dernière étape est seulement réalisée si le nombre de *keyframes* à optimiser est en dessous d'un certain seuil. L'extraction et la comparaison des descripteurs des PI requièrent des ressources de calculs non négligeables. Des méthodes alternatives ont été développées, telles que l'extraction de forme de chaque objet comme information binaire (Wang et al., 2019a) et la vérification basée contours pour la fermeture de boucle (Schenk and Fraundorfer, 2019). L'extraction des contours permet de fournir une précision de 2.36cm, ce qui est deux fois moins précis comparé à ORB-SLAM2 (Mur-Artal and Tardós, 2017) avec 1.14cm sur la base de donnée RGB-D (Sturm et al., 2012).

2.1.2. Module de reconstruction 3D

Le module de reconstruction 3D fournit trois types de représentation de la carte, le nuage de point, le maillage et la reconstruction volumétrique basée sur les voxels pour fournir un maillage surfacique en temps-réel. Là où un nuage de point est épars et non ordonné, une reconstruction géométrique dense comme le maillage et volumétrique donnent une compréhension géométrique de la scène utilisée pour des fonctions de plus haut-niveaux. On s'intéresse dans cette section aux reconstructions par maillage et volumétrique.

Reconstruction par maillage 3D

La représentation par maillage est utilisée pour modéliser les surfaces, les formes et fournit une topologie de la scène basée sur les points (Rosinol et al., 2020) ou les *surfels* (Schöps et al., 2020). La triangulation de Delaunay est utilisée pour différentes applications et notamment en vision par ordinateur (Dinas and Bañón, 2014) pour fournir un maillage précis et de couvrir l'utilisation potentielle de surfaces planes (Rosinol et al., 2019). Plusieurs algorithmes dérivent de la triangulation de Delaunay en deux (Rosinol et al., 2020; Greene and Roy, 2017; Teixeira and Chli, 2016; Yokozuka et al., 2019) ou trois (Piazza et al., 2018) dimensions. En 2D, cette technique suit la propriété du cercle circonscrit qui permet de générer un triangle lorsque le cercle est le seul à passer sur les trois sommets. Aucun sommet n'est alors à l'intérieur du cercle et permet de maximiser l'angle minimum pour chaque triangle généré. Cette méthode permet d'obtenir un maillage de qualité et d'assurer une cohérence dans les triangulations.

Dans (Greene and Roy, 2017), un maillage Delaunay léger est généré avec une approche sans *keyframes* et une estimation de profondeur de l'image monoculaire pour les MAVs (Shewchuk, 2002). La reconstruction fournit un maillage par image sans prendre en compte la reconstruction issue des anciennes images pour améliorer le temps de calcul. Cette approche permet de traiter chaque image en embarqué sur MAV avec un Intel Skull Canyon NUC avec une performance de plus de 90Hz. Dans (Rosinol et al., 2020), le maillage multi-image donne une représentation unique reconstruite au fur et à mesure des images grâce à la fusion des maillages par image. Sur un Intel Xeon CPU E3-1505M v6 3GHz, le maillage multi-image est généré autour de 67Hz. Cette configuration montre une carte

3D fiable composée d'un large maillage reconstruite au cours de la séquence mis à jour avec chaque nouvelle représentation générée en temps-réel.

Reconstruction volumétrique

La reconstruction volumétrique est définie par le calcul basé voxels dans lequel un maillage surfacique est rendu à partir du volume qui permet un modèle 3D détaillé (Lorensen and Cline, 1987).

Calcul basé voxels : La méthode pour rapidement et précisément représenter les surfaces est la méthode *truncated signed distance field* (TSDF) (Curless and Levoy, 1996). Cette technique représente l'environnement 3D en grille de voxels. Dans un volume TSDF, qui intègre les données de profondeur, la valeur de chaque pixel correspond à la distance algébrique à la surface la plus proche. Les valeurs positives et négatives correspondent aux voxels en dehors et à l'intérieur du volume respectivement. La surface est définie par la limite d'iso-surface entre les valeurs négatives et positives, aussi appelé le *zero-crossing*. Au-delà d'une certaine distance, l'information devient non pertinente. La distance est alors tronquée pour utiliser les valeurs proches de la surface.

Chaque voxel stocke une distance signée tronquée D et les valeurs de points W mis à jour pour chaque point 3D p dans le volume à partir des images $1...k$ détaillé dans les équations 2.4 et 2.5.

$$D(p) = \frac{\sum w_k(p) d_k(p)}{\sum w_k(p)} \tag{2.4}$$

$$W(p) = \sum w_k(p) \tag{2.5}$$

avec d est la distance signée et w la fonction de pondération des mesures issues des capteurs. Le cumul $D_k(p)$ et $W_k(p)$ sont exprimés dans les équations 2.6 et 2.7 (Curless and Levoy, 1996).

$$D_{k+1}(p) = \frac{W_k(p) D_k(p) + w_{k+1}(p) d_{k+1}(p)}{W_k(p) + w_{k+1}(p)} \tag{2.6}$$

$$W_{k+1}(p) = W_k(p) + w_{k+1}(p) \tag{2.7}$$

Le choix de la pondération a un impact fort sur la précision de la représentation. Elle modélise l'incertitude des mesures de la surface (Oleynikova et al., 2017). Plus grande est l'incertitude, telle que des données bruitées, plus grande est la pondération. Par exemple, pour permettre une reconstruction 3D en temps-réel, KinectFusion (Newcombe et al., 2011b) a adopté une approche de pondération constante avec $w_k(p) = 1$ détaillée dans l'équation 2.8.

$$W_{k+1}(p) = min(W_k(p) + w_{k+1}(p), W_{max}) \tag{2.8}$$

Dans (Oleynikova et al., 2017), la comparaison de deux stratégies de pondération est proposée. Elle consiste à comparer la reconstruction 3D qualitativement et quantitativement avec une pondération constante et quadratique. L'équation 2.9 définie la distance tronquée d avec $\delta = 4v$ et $\epsilon = v$ où v est la taille du voxel.

$$w(p) = \begin{cases} \frac{1}{z^2}, & -\epsilon < d \\ \frac{1}{z^2}\frac{1}{\delta - \epsilon}(d + \delta), & -\delta < d < -\epsilon \\ 0, & d < -\delta \end{cases} \tag{2.9}$$

Qualitativement, la stratégie de pondération quadratique fournit une meilleure représentation de la structure avec moins d'erreurs que la pondération constante. Un plus haut niveau de robustesse avec moins de distorsion est observé avec une taille de voxel v allant de 0.02m à 0.20m.

Rendu de la reconstruction : Deux principales approches sont utilisées pour l'extraction du maillage : le *raycasting* et le *projection mapping* (Klingensmith et al., 2015). Le *raycasting* consiste à envoyer un rayon à partir du capteur jusqu'au voxel pour extraire le *zero-crossing* à partir des valeurs TSDF (Curless and Levoy, 1996; Parker et al., 1998). Le *projection mapping* calcule la distance entre le centre du voxel et la valeur de la profondeur de l'image. La quantité de donnée traitée par cette technique est limitée au nombre de voxels alors que la méthode de *raycasting* inclut le nombre et la longueur des rayons qui sont projetés. En terme de temps d'exécution, le *raycasting* est plus rapide avec un temps requis de 62ms contre 106ms pour le *projection mapping* sur la tablette Tango Yellowstone avec un processeur quatre cœurs, 4GB RAM et un GPU Tegra K1 (Klingensmith et al., 2015).

Dans les travaux pionniers de KinectFusion (Newcombe et al., 2011b), le capteur Kinect est utilisé pour fournir les images RGB et les mesures de profondeur avec une résolution de 640×480. Ils utilisent l'intégration TSDF et le rendu 3D avec la technique de *raycasting*. L'estimation de pose 3D est calculée avec l'approche *iterative closest point* (ICP) (Besl and McKay, 1992). La reconstruction volumétrique est limitée dans un volume de taille 512^3. L'implémentation sur NVIDIA GeForce GTX 560 permet de mettre à jour les valeurs TSDF à un temps d'exécution de 2ms (Izadi et al., 2011). Le système principal fonctionne en temps-réel à 40 ips avec la résolution du volume de 512^3. Entièrement implémenté avec CUDA afin d'utiliser les GPU NVIDIA, cette méthode consomme 512MB d'espace mémoire pour les voxels 32-bit dans l'espace de rendu restreint, ce qui est significatif par rapport à l'espace 3D du rendu final. La méthode de hachage (Nießner et al., 2013) est une solution pour obtenir une représentation large de la scène avec une consommation mémoire efficace (Kähler et al., 2015). L'approche algorithmique introduit la visualisation intermédiaire pour accélérer le calcul. Pendant que le *raycasting* est réalisé pour chaque image dans une configuration complète, le paramètre de projection utilise les plus récents résultats du *raycasting*. Après le seuil de n images dépassé, un *raycasting* complet est réalisé. Implémenté sur NVIDIA Tegra K1 et Apple iPad Air 2, la méthode atteint un temps de calcul par image de 21.04ms et 48.43ms respectivement.

Dans (Oleynikova et al., 2017), la même structure hiérarchique des données avec le hachage de voxels et l'approche de rendu est utilisée. L'approche de rendu fusionne le point visé par le rayon issu du *raycasting* avec tous les autres points 3D du même voxel pour exécuter le processus de rendu du voxel seulement une fois, sur la position moyenne des points. Cela permet d'avoir un temps d'exécution de 55ms et 5ms avec une taille de voxel de 0.05m et 0.20m respectivement, comparé à 250ms et 100ms avec l'approche standard.

2.1.3. Modules d'apprentissage dans le pipeline SLAM conventionnel

Les recherches avancées en réseaux de neurones profonds (DNN) permettent l'utilisation d'un large ensemble de données tout en fournissant des résultats précis et robustes pour plusieurs applications (Sze et al., 2017). Cette sous-section donne une vue d'ensemble des méthodes DNN pour la localisation temps-réel, une description exhaustive des représentations intermédiaires et de la complexité des algorithmes d'apprentissage profond dans le contexte des systèmes embarqués.

Apprentissage profond pour l'estimation de pose en temps-réel

Plusieurs recherches discutent de l'utilisation des réseaux de neurones pour l'estimation de pose en temps-réel avec une implémentation *end-to-end* ou hybride. L'implémentation *end-to-end* repose exclusivement sur les algorithmes DNN pour estimer la pose à partir d'images consécutives. Dans (Wang et al., 2017), le réseau de neurone convolutif (CNN) (Dosovitskiy et al., 2015) est utilisé pour extraire les caractéristiques d'images RGB, et le réseau de neurone récurrent (RNN) est utilisé pour modéliser les informations séquentielles. L'approche mémoire longue à court terme (LSTM) est utilisée pour la représentation séquentielle avec un espace mémoire qui conserve les précédents états. Dans le cas de la configuration visuelle-inertielle, un LSTM à flux multiple est utilisé pour traiter les mesures inertielles (Clark et al., 2017). Les méthodes *end-to-end* basées sur de l'apprentissage supervisé (Wang et al., 2017) ou non supervisé (Li et al., 2021) évoluent, mais les performances des approches actuelles sont limitées par les données d'apprentissage pour fournir une estimation précise et robuste dans toutes les situations de chaque cas d'usage (voiture, drone, robot mobile, casque).

Une chaine de traitement hybride utilise l'apprentissage profond pour les fonctions spécifiques du *back-end* et du *front-end*, incluant l'optimisation locale (Tang and Tan, 2019; Czarnowski et al., 2020) ou globale (Arshad and Kim, 2021; Zhang et al., 2017a) et l'extraction de points d'intérêts du FE (Li et al., 2020; Sons et al., 2019; Xu et al., 2020; Zhou et al., 2021). Dans (Tang et al., 2019), les résultats basés sur un détecteur et descripteur CNN démontrent une meilleure distribution des points d'intérêts avec un nombre de détections plus faible comparé à ORB (Rublee et al., 2011). L'extraction s'exécute à 40 ips et le reste de la chaine de traitement SLAM basé sur ORB-SLAM2 (Mur-Artal and Tardós, 2017) à 20 ips sur Jetson TX2 avec la version réduite du réseau. La précision de la méthode hybride se détériore pour les séquences qui requièrent des détails fins, comme *fr1_floor* and *fr1_360* de la base de données TUM RGB-D (Sturm et al., 2012) à cause de l'échelle des caractéristiques de l'image issues du réseau de neurone.

Représentations intermédiaires basées sur les réseaux de neurones

Les représentations intermédiaires sont utiles pour les fonctions haut-niveaux basées sur les modèles allant du traitement du pixel vers l'action. Dans le contexte de la conduite autonome dans un environnement urbain où un agent doit atteindre une localisation précise, le pourcentage de réussite est amélioré d'environ 15% avec l'addition de données de profondeur de l'image et de 20% avec l'ajout de la segmentation sémantique comparé au traitement avec seulement les images RGB (Zhou et al., 2019).

Segmentation sémantique : Afin de reconnaître les objets ambiants, la segmentation d'image comme la sémantique permet de labelliser un ensemble d'objets au niveau du pixel (Garg et al., 2020; Minaee et al., 2021). Dans (Xia et al., 2020), le concept de sémantique pour la reconnaissance d'objets et la segmentation sémantique dans le SLAM est détaillé et affirme qu'elle fournit des solutions

efficaces pour l'association de données (Lianos et al., 2018) et la cohérence long terme afin d'obtenir une localisation fiable (Gawel et al., 2018). Les techniques d'apprentissages profonds améliorent les capacités du SLAM en utilisant la segmentation sémantique pour la détection de fermeture de boucle et pour une consistance à moyen terme (Lianos et al., 2018; Zhang et al., 2020) ou encore pour gérer les environnements dynamiques. Les méthodes SLAM considèrent les caractéristiques dynamiques comme étant des données aberrantes, la sémantique est utilisée pour labelliser les éléments dynamiques et statiques pour améliorer la précision et la robustesse de la localisation ce qui impact donc la qualité de la reconstruction (Wen et al., 2021; Yu et al., 2018a). Des recherches actives se penchent sur la segmentation de la carte 3D, ce qui mène à la reconnaissance d'objets en temps-réel basée sur la représentation par nuages de points (Tateno et al., 2015), *surfels* (McCormac et al., 2017; Wald et al., 2018), maillage (Rosinol et al., 2020; Rosu et al., 2019) et voxels (Rosinol et al., 2020; Grinvald et al., 2019; Narita et al., 2019). La reconstruction sémantique est atteint par deux principales approches, la labellisation des images et celle de la carte 3D (Landgraf et al., 2020). L'extraction de la sémantique des images traite les images d'entrée et réalise des opérations redondantes alors que la segmentation de la carte reconstruite implique l'utilisation d'un CNN 3D. Ce dernier évite les calculs redondants, mais dépend de la qualité de la reconstruction 3D. En utilisant la métrique d'intersection sur l'union (IoU), la segmentation atteint une précision de 0.89 et 0.92 après 1000 images pour la segmentation sémantique par images et de la carte 3D respectivement (Landgraf et al., 2020).

Plusieurs problématiques restent ouvertes sur la précision requise de la segmentation pour les méthodes SLAM et son impact sur la précision de la reconstruction. La complexité de génération de la sémantique sur le coût calculatoire et sur l'utilisation des ressources matérielles pour la segmentation 2D et 3D est également une direction de recherche future dans le contexte embarqué.

Estimation de la profondeur de l'image : Les réseaux de neurones ont été utilisés pour l'estimation de la profondeur des images monoculaires (Czarnowski et al., 2020; Tateno et al., 2017; Bloesch et al., 2018). L'estimation de la profondeur basée sur un CNN permet de surmonter les limitations des caméras monoculaires en gérant les déplacements rotationnels pures (Tateno et al., 2017) et en résolvant le problème de facteur d'échelle par la mise à jour de la matrice de transformation avec le paramètre d'échelle α (Yin et al., 2017) :

$$\mathbf{T}_{k,k-1} = \begin{bmatrix} \mathbf{R}_{k,k-1} & \alpha \mathbf{t}_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{2.10}$$

avec $\mathbf{T}_{k,k-1}$ la matrice de transformation entre deux images (rotation $\mathbf{R}_{k,k-1}$, translation $\mathbf{t}_{k,k-1}$) et α le paramètre d'échelle défini comme l'estimation de vraisemblance maximum en prenant en compte l'estimation de profondeur de chaque pixel.

Bien que cette représentation a plusieurs avantages dans le cas du SLAM monoculaire, sa précision reste limitée pour de longues séquences comparé aux méthodes traditionnelles avec une trajectoire qui dérive fortement (Li et al., 2018a). Dans (Martins et al., 2018), la carte de profondeur est estimée par la fusion de la chaine de traitement basée stéréovision avec l'estimation de profondeur monoculaire. Qualitativement, la carte de profondeur fusionnée affine les résultats bruités de la profondeur issue de la stéréovision et donne une composition globale de l'image grâce à l'estimation de profondeur monoculaire. L'erreur absolue médiane entre l'estimation de profondeur et la vérité terrain (GT) pour la stéréovision et monoculaire est moins de 5m et 20m respectivement avec une GT de 80m. Avec

une GT de 10m, la précision est de 1m et moins de 4m respectivement. La fusion des deux techniques permet surmonter les limitations de la stéréovision lorsqu'il y a peu de textures et d'affiner la qualité de l'estimation de profondeur monoculaire grâce à la stéréo.

Dans le contexte du calcul embarqué avec des ressources restreintes, le compromis entre la précision de la profondeur requise pour la reconstruction 3D et le coût calculatoire en termes d'utilisation mémoire et de temps de traitement reste une direction de recherche ouverte.

Complexité de l'apprentissage profond pour systèmes embarqués

La complexité des modèles d'apprentissage profond pour la localisation et la reconstruction temps-réel présente trois caractéristiques : 1/ le modèle utilisé, 2/ l'utilisation d'une grande quantité de données pour l'adaptabilité du réseau et 3/ la consommation de ressources et l'implémentation matérielle pour le traitement temps-réel. Les modèles *end-to-end* sont actuellement coûteux en calcul, car ils incluent la représentation séquentielle des états. Les modèles hybrides ont l'avantage de combiner les forces des algorithmes d'apprentissage profond avec la maturité des méthodes géométriques actuelles pour les fonctions spécifiques afin de surmonter les limitations des capteurs et de calculer les représentations intermédiaires pour une compréhension haut-niveau de l'environnement ambiant.

La précision des méthodes DNN est dépendante des données d'apprentissage. Cependant, afin d'obtenir un système fiable, l'étape d'apprentissage doit apprendre à partir de données issues de plusieurs situations, notamment pour la localisation en temps-réel. Les méthodes d'odométries visuelles basées sur les DNN obtiennent une faible précision si le modèle n'est pas bien ajusté (Wang et al., 2017). La majorité des méthodes de localisation sont entrainées et évaluées sur la base de données KITTI (Geiger et al., 2012) qui fournit des séquences enregistrées dans une voiture en milieu urbain sans changement de rotation significatif. Cela peut engendrer des erreurs dans d'autres situations, comme les drones dont les mouvements rotationnels sont importants.

En plus du temps d'exécution et des ressources matérielles requises pour la phase d'apprentissage des DNN, les plateformes matérielles spécifiques permettent un traitement temps-réel pour l'inférence des réseaux, notamment avec la parallélisation des calculs pour une meilleure utilisation des GPU. Les chaines de traitement hybrides sont exécutées à la fois sur CPU et GPU pour le calcul des contraintes géométriques et le réseau de neurone respectivement (Czarnowski et al., 2020). Pour un déploiement applicatif, la taille, la consommation d'énergie et les contraintes de ressources doivent être considérées. Dans (Yu et al., 2020), l'extraction des points d'intérêts avec un CNN a été accélérée sur FPGA (Yu et al., 2018b) et le modèle d'odométrie visuel s'exécute sur CPU. Cette implémentation atteint un temps d'exécution de 5ms sur FPGA et 340ms sur CPU avec une caméra monoculaire dont le flux d'entrée est de 20 ips. Cette implémentation montre que l'extraction de points est une étape calculatoire coûteuse dans la chaine de traitement SLAM et que l'accélération de cette fonction permet de réduire la complexité de la plateforme embarquée tout en prenant avantage du système hétérogène par le partitionnement efficace de la fonction d'extraction.

2.1.4. Outils d'évaluation et méthodes existantes

Cette sous-section décrit les outils d'évaluation utilisés pour les méthodes de l'état de l'art de manière qualitatif et quantitatif. La comparaison des performances des méthodes existantes est détaillée avec les techniques utilisées, de la localisation et reconstruction 3D temps-réel aux DNN.

Outils d'évaluation

De nombreux outils d'évaluation présentent des bases de données avec des séquences enregistrées à différents endroits pour l'évaluation des méthodes basées vision sous des conditions difficiles telles que le changement de luminosité, les longues trajectoires, les variations de vitesse de déplacement du système et les environnements dynamiques.

La table 2.2 détaille les bases de données existantes pour plusieurs cas applicatifs avec le choix des capteurs, les caractéristiques des séquences et les capteurs qui ont été utilisés pour capturer la vérité terrain. Suivant le cas d'usage, les bases de données publiques fournissent des séquences dans des environnements extérieurs pour les voitures autonomes avec des zones densément peuplées (Wen et al., 2020; Hsu et al., 2021), une diversité de l'environnement urbain (Jeong et al., 2019) et de longues séquences à grande échelle avec différentes vitesses de déplacement du véhicule allant jusqu'à 80km/h (Geiger et al., 2012). Les séquences avec des environnements dynamiques et long-terme avec des changements d'illumination sont également à notre disposition pour la cas d'usage des robots de service (Shi et al., 2020; Carlevaris-Bianco et al., 2016) où l'enregistrement des séquences inclut de l'activité humaine (Pronobis and Caputo, 2009). Les principaux défis liés au déplacement du système dans un environnement intérieur et extérieur sont liés aux bases de données pour drones MAV avec des trajectoires agressives (Delmerico et al., 2019), des déplacements rapides avec du flou lié au mouvement (Burri et al., 2016) et dans les rues urbaines à faible altitude (5-15m au-dessus du sol) (Majdik et al., 2017).

TABLE 2.2 – Bases de données existantes suivant les cas d'usages applicatifs

Cas d'usages	Bases de données	Capteurs	Caractéristiques				Année
			Seq.(#)	(m)/(s)	Int./Ext.	Conditions	
Voitures	UrbanLoco (Wen et al., 2020)	x6 FLIR Blackfly S USB3 (2048*1536, 10 ips) x1 fisheye lens Grasshopper3 5.0 MP x1 Velodyne HDL 32E (10Hz, range 80m) GPS, IMU (Xsens Mti 10 100Hz)	13	40000m	ext.	Zone densément peuplée	2020 2021 (Hsu et al., 2021)
	KAIST Urban (Jeong et al., 2019)	x2 FL3-U3-20E4C-C (1280*560, 10 ips) x4 LiDAR sensors (x2 VLP-16, x2 LMS-511) GPS, IMU (MTi-300 200Hz)	19	190000m	ext.	Environnement urbains diversifiés	2019
	KITTI (Geiger et al., 2012)	x2 PointGrey Flea2 (1382*512, 10 ips) x1 Velodyne HDL-64E (10Hz, range 100m) GPS, IMU (OXTS RT 3003)	22	39200m	ext.	Séquences à grande échelle Différentes vitesses jusqu'à 80km/h	2012
Robots mobiles	OpenLORIS (Shi et al., 2020)	x1 RealSense D435i x1 RealSense T265	22	2244s	int.	Environnements dynamiques Changements d'éclairage et de points de vue	2020
	NCLT (Carlevaris-Bianco et al., 2016)	x1 Pointgrey Ladybug3 (1600*1200, 5ips) x3 LiDAR (x1 Velodyne HDL-32E, x2 Hokuyo lidar) GX3 IMU (100Hz), GPS, RTK	27	147400m 125640s	int. ext.	Environnements dynamiques et à long-terme Changements de conditions météos et d'éclairages Changements saisonniers	2016
	COLD (Pronobis and Caputo, 2009)	x2 Videre Design MDCS2 (640*480 up to 60 ips) SICK laser scanners	76	920m 3576s	int.	Variations d'illumination et activité humaine	2009
MAVs	UZHFPV (Delmerico et al., 2019)	x1 miniDAVIS346 (346*260, 50 ips, events) x1 Qualcomm Flight Board (640*480, IMU	27	6166m 1307s	int. ext.	Trajectoires aggressives	2020
	ZUMAV (Majdik et al., 2017)	x1 GoPro Hero 4 (1920*1080, 30ips) GPS, IMU	1	2000m	ext.	Rues urbaines Basse altitude (5-15m au-dessus du sol)	2017
	EuRoC (Burri et al., 2016)	x2 Aptina MT9V034 (720*480p, 20ips) IMU (ADIS16448, 200 Hz)	11	894m 1349s	int.	Différentes vitesses de déplacement Différentes luminosités d'image	2016
Portatif Casque RA/RV	LaMAR (Sarlin et al., 2022)	HoloLens2 iPad/iPhone NavVis VLX backpack NavVis M6 trolley	3	40000m 100h	int. ext.	Déplacement de personnes Changements climatiques, cycle jour/nuit Changements à long terme	2022
	Aria Pilot (Karakas et al., 2022)	x1 RGB camera x2 mono cameras + IR x2 IMU (1KHz) GPS 1Hz	159	-	int. ext.	Activités quotidiennes Activités de bureau	2022
	Bonn Dynamic (Palazzolo et al., 2019)	ASUS Xtion Pro LIVE RGBD sensor	26	-	int.	Séquences dynamiques	2019
	TUM-VI (Schubert et al., 2018)	x2 uEye UI-3241LE-M-GL (1024*1024, 20 ips) IMU (BMI160 200Hz)	28	19596m	int. ext.	Séquences courtes et longues	2018
	ETHI RGBD (Park et al., 2017)	1-st generation of the Kinect sensor	3	141s	int.	Changements d'illumination globale et locale	2017
Synthétique	uHumans (Rosinol et al., 2021)	Stereo cameras Depth camera IMU	12	-	int. ext.	Simulation d'environnements surpeuplés	2021
	ETHI (Park et al., 2017)	RGB-D data based on (Handa et al., 2014)	2	70s	int.	Cinq variations d'éclairage par séquence	2017
	ICL-NUIM (Handa et al., 2014)	RGB-D data	8	58.46m 312s	int.	Deux scènes synthétiques	2014

Les données enregistrées avec les capteurs portés à la main sont particulièrement utiles pour le cas d'usage applicatif des casques à réalité mixte. De récentes bases de données ont été publiées mettant en scène de longues trajectoires sur plusieurs niveaux que ce soit à l'intérieur ou à l'extérieur avec de l'activité humaine (Karakas et al., 2022), des séquences dynamiques (Palazzolo et al., 2019), des changements d'illuminations de manière locales ou gloables (Park et al., 2017) et des changements longs-terme dus aux sites de constructions ou au déplacement de mobilier (Sarlin et al., 2022). Les bases de données TUM fournissent plusieurs trajectoires avec le capteur porté à la main incluant différents capteurs d'images pour des séquences courtes et longues (Schubert et al., 2018; Engel et al., 2016; Sturm et al., 2012). L'enregistrement de données dans des environnements réels a plusieurs limitations qui peuvent être surmontées avec les séquences synthétiques (Wang et al., 2020). La précision de la surface de reconstruction avec les informations RGB-D sous des conditions réalistes (Handa et al., 2014), comme les changements de luminosités est également évaluée sur la base des données synthétiques (Park et al., 2017; Rosinol et al., 2021).

L'évaluation de l'estimation de la trajectoire et la précision de la reconstruction de la scène requièrent la vérité terrain qui est généralement fourni par le RTK-GPS ou système de capture des mouvements.

Comparaison des performances de méthodes existantes

La table 2.3 détaille l'état de l'art des méthodes existantes SLAM de la reconstruction 3D par nuage de points à la reconstruction volumétrique. Elle met en évidence les différentes stratégies employées pour le *front-end* et le *back-end* des méthodes de localisation temps-réel et le type de représentation généré de la cartographie nuage de point et *surfels* à la reconstruction volumétrique. Les méthodes de reconstruction 3D ne fournissent pas de stratégie de localisation (noté : -), ce qui signifie qu'elles prennent les poses 3D générées par les méthodes SLAM. La stratégie de localisation montre que la plupart des méthodes emploient le suivi de points d'intérêts dans le FE qui se réfère à l'algorithme KLT (yves Bouguet, 2000). Cette technique permet d'améliorer le temps de calcul pour une optimisation locale comparé à la détection et description de points d'intérêts (Det+Des). Dans (Rosinol et al., 2020), le suivi de points prend en moyenne 4.5ms avec 300 PI par image alors que la détection et description prend 15ms pour extraire 1200 points ORB par image (Campos et al., 2021). Afin d'obtenir un algorithme avec une capacité SLAM prenant en compte la fermeture de boucle, la description des points est requise. Dans ce contexte, ORB se focalise sur le temps d'exécution comparé à l'algorithme SIFT qui est trop exigeant en termes de calcul pour des applications temps-réel.

La plupart des méthodes sont basées sur le *back-end* par graphe qui permet d'exploiter la parcimonie du SLAM et de fournir une précision de l'estimation de la trajectoire plus accrue que les techniques basées filtrage (Delmerico and Scaramuzza, 2018). La table 2.4 montre l'utilisation potentielle des DNN et l'implémentation matérielle des méthodes. Les mesures d'images par seconde (ips) correspondent à la performance de la chaine traitement dans sa globalité. Elle met en évidence la complexité des approches pour les plateformes embarquées. Par exemple, la méthode de filtrage basée MSCKF (Zhu et al., 2017) permet d'avoir une meilleure performance en termes d'images par seconde que VINS-Fusion (Qin et al., 2019) qui est basée sur un graphe BE sur la plateforme UpBoard avec 20 ips et 7 ips respectivement. Cependant, la méthode basée graphe SVO (Forster et al., 2017b) tourne à 40 ips sur la même plateforme. Les paramètres du *front-end*, les stratégies *multi-threading*, la granularité de la reconstruction 3D sont les facteurs à prendre en compte pour des performances

temps-réel.

TABLE 2.3 – Méthodes existantes d'odométrie visuelle(-inertielle), de SLAM et de reconstruction 3D avec l'utilisation de différents capteurs, la stratégie de localisation et le type de reconstruction.

Méthodes	Année	Capteurs				Stratégie de localisation		Type de reconstruction		
		Cam. visible		IMU	Autres					
		mono	stereo			Front-end	Back-end	Carto.	Maillage	Volum.
DeepSLAM (Li et al., 2021)	2021	✓	✓	-	-	End-to-end DL model		Dense	-	-
ORB-SLAM3 (Campos et al., 2021)	2020	✓	✓	✓	-	Det.+Des.	Graph	Sparse	-	-
Basalt (Usenko et al., 2020)	2020	-	✓	✓	-	Tracking	Graph	Sparse	-	-
OpenVINS (Geneva et al., 2020)	2020	✓	✓	✓	-	Tracking	Filtering	Sparse	-	-
DXSLAM (Li et al., 2020)	2020	✓	-	-	-	Det.+Des.	Graph	Sparse	-	-
(Alliez et al., 2020a)	2020	✓	-	✓	IR+LiDAR	Det.+Des.	Graph	Sparse/Dense	-	Offline (Verdie et al., 2015)
SurfelMeshing (Schöps et al., 2020)	2020	-	-	-	RGB-D	-	-	Surfels	✓	-
Kimera (Rosinol et al., 2020)	2020	✓	✓	✓	-	Tracking	Graph	Sparse	✓	✓
DeepFactor (Czarnowski et al., 2020)	2020	✓	-	-	-	Semi-direct	Graph	-	-	✓
ST-VIO (Zhang et al., 2020)	2020	✓	-	✓	-	Direct	Graph	-	-	-
(Yu et al., 2020)	2020	✓	-	-	-	End-to-end DL model		-	-	-
GCN-SLAM (Tang et al., 2019)	2019	-	-	-	RGB-D	Det.+Des.	Graph	Sparse	-	-
RESLAM (Schenk and Fraundorfer, 2019)	2019	-	-	-	RGB-D	Edge	Graph	Sparse	-	-
(Sons et al., 2019)	2019	✓	✓	-	-	Det.+Des.	Graph	Sparse	-	-
VINS-Fusion (Qin et al., 2019)	2019	✓	✓	✓	LiDAR	Tracking	Graph	Sparse	-	-
PanopticFusion (Narita et al., 2019)	2019	-	-	-	RGB-D	-	-	-	-	✓
MSCKF-based (Sun et al., 2018)	2018	-	✓	✓	-	Tracking	Filtering	Sparse	-	-
(Piazza et al., 2018)	2018	-	-	-	-	-	-	-	✓	-
DS-SLAM (Yu et al., 2018a)	2018	-	-	-	RGB-D	Det.+Des.	Graph	-	-	✓
CNN-SLAM (Tateno et al., 2017)	2017	✓	-	-	-	Direct	Graph	Dense	-	-
FLaME (Greene and Roy, 2017)	2017	✓	-	-	-	-	-	-	✓	-
DeepVO (Wang et al., 2017)	2017	✓	-	-	-	End-to-end DL model		-	-	-
VINet (Clark et al., 2017)	2017	✓	-	✓	-	End-to-end DL model		-	-	-
Voxblox (Oleynikova et al., 2017)	2017	-	-	-	RGB-D	-	-	-	-	✓
SVO (Forster et al., 2017b)	2016	✓	✓	✓	-	Semi-direct	Graph	Sparse	-	-
DSO (Engel et al., 2018)	2016	✓		-	-	Direct	Graph	Sparse	-	-
(Teixeira and Chli, 2016)	2016	✓	-	-	-	-	-	-	✓	-
BundleFusion (Dai et al., 2017)	2016	-	-	-	RGB-D	Det.+Des.	GPU-solver	-	-	✓
Chisel (Klingensmith et al., 2015)	2015	✓	-	✓	RGB-D	Tracking	Filtering	-	-	✓
ElasticFusion (Whelan et al., 2015)	2015	-	-	-	RGB-D	Direct	Graph (Sumner et al., 2007)	Surfels	-	-
InfiniTAM (Kähler et al., 2015)	2015	-	-	✓	RGB-D	Direct	ICP / (Madgwick et al., 2011)	-	-	✓
ROVIO (Bloesch et al., 2015)	2015	✓	✓	✓	-	Direct	Filtering	Sparse	-	-
OKVIS (Leutenegger et al., 2015)	2015	✓	✓	✓	-	Det.+Des.	Graph	Sparse	-	-
LSD-SLAM (Engel et al., 2014)	2014	✓	-	-	-	Direct	Graph	Dense	-	-
SLAM++ (Salas-Moreno et al., 2013)	2013	-	-	-	RGB-D	Direct	Graph	-	-	✓
KinectFusion (Newcombe et al., 2011b)	2011	-	-	-	RGB-D	Direct	ICP	-	-	✓
PTAM (Klein and Murray, 2007)	2007	✓	-	-	-	Detection	LBA / GBA	Sparse	-	-
MonoSLAM (Davison et al., 2007)	2007	✓	-	-	-	Detection	Filtering	Sparse	-	-

TABLE 2.4 – Méthodes existantes d'odométrie visuelle(-inertielle), de SLAM et de reconstruction 3D avec l'utilisation potentielle des réseaux de neurones profonds (DNN) et l'implémentation matérielle (HW)

Méthodes	Année	Capteurs				Utilisation des DNN	Implémentation HW
		Cam. visible		IMU	Autres		
		mono	stereo				
DeepSLAM (Li et al., 2021)	2021	✓	✓	-	-	Depth (48ms), poses (25ms), loop (120ms)	i7-6820HK 2.7GHz, GeForce GTX 980 M (20 ips)
ORB-SLAM3 (Campos et al., 2021)	2020	✓	✓	✓	-	(Mur-Artal and Tardós, 2017) on Jetson TX2 (28 ips) (Aldegheri et al., 2019) (Mur-Artal and Tardós, 2017) on Pi 3B+ (6 ips), Jetson Nano (10 ips) (Silveira et al., 2020)	
Basalt (Usenko et al., 2020)	2020	-	✓	✓	-	E5-1620 CPU (19 ips global opti., 128 ips local opti.)	
OpenVINS (Geneva et al., 2020)	2020	✓	✓	✓	-	Depth estimation (Merrill et al., 2021)	Jetson TX2 (36 ips), Jetson Nano (28 ips) (Merrill et al., 2021)
DXSLAM (Li et al., 2020)	2020	✓	-	-	-	Feature extraction (46.2ms w/ opti.)	Core i7-10710U (15W)
(Alliez et al., 2020a)	2020	✓	-	✓	IR+LiDAR	-	NI IC-3173 (10 ips visual-SLAM, 20 ips lidar-SLAM)
SurfelMeshing (Schöps et al., 2020)	2020	-	-	-	RGB-D	-	Core i7 6700K, Geforce GTX 1080 (178 ips for mesh)
Kimera (Rosinol et al., 2020)	2020	✓	✓	✓	-	Semantic segmentation	Jetson TX2 (Rosinol et al., 2021)
DeepFactor (Czarnowski et al., 2020)	2020	✓	-	-	-	Depth estimation	GTX 1080 GPU (network, camera tracking) CPU (geometric error factors)
ST-VIO (Zhang et al., 2020)	2020	✓	-	✓	-	Semantic segmentation (Wang et al., 2019b)	Jetson TX2
(Yu et al., 2020)	2020	✓	-	-	-	Feature extraction on FPGA (5ms) VO on ARM CPU (340ms)	Xilinx ZCU102 MPSoC
GCN-SLAM (Tang et al., 2019)	2019	-	-	-	RGB-D	Feature extraction (25ms)	Jetson TX2 (20 ips)
RESLAM (Schenk and Fraundorfer, 2019)	2019	-	-	-	RGB-D	-	i7-4790 desktop computer with 32 GB RAM
(Sons et al., 2019)	2019	✓	✓	-	-	Feature extraction (around 16ms)	GeForce GTX Titan X
VINS-Fusion (Qin et al., 2019)	2019	✓	✓	✓	LiDAR	-	UP Board (7 ips), ODROID XU4 (7 ips) (Delmerico and Scaramuzza, 2018)
PanopticFusion (Narita et al., 2019)	2019	-	-	-	RGB-D	Semantic segmentation (315ms) (He et al., 2017)	Image segmentation on GPU
MSCKF-based (Sun et al., 2018)	2018	-	✓	✓	-	-	UP Board (20 ips), ODROID XU4 (20 ips) (Delmerico and Scaramuzza, 2018)
(Piazza et al., 2018)	2018	-	-	-	-	-	Core i7-4770S, 3.10 GHz with (Mur-Artal et al., 2015)
DS-SLAM (Yu et al., 2018a)	2018	-	-	-	RGB-D	Semantic segmentation (37.6ms) (Badrinarayanan et al., 2017)	Intel i7 CPU, P4000 GPU
CNN-SLAM (Tateno et al., 2017)	2017	✓	-	-	-	Semantic segmentation (Wang et al., 2015) Depth estimation (Laina et al., 2016)	Intel Xeon CPU, 2.4GHz Quadro K5200 GPU for CNN networks
FLaME (Greene and Roy, 2017)	2017	✓	-	-	-	-	Intel Skull Canyon NUC (90 ips) with (Steiner et al., 2017)
DeepVO (Wang et al., 2017)	2017	✓	-	-	-	Feature extraction LSTM pose estimation	Training on Tesla K40 GPU
VINet (Clark et al., 2017)	2017	✓	-	✓	-	Features extraction (160ms) IMU LSTM (5ms) and Core LSTM	Training on Tesla K80 GPU
Voxblox (Oleynikova et al., 2017)	2017	-	-	-	RGB-D	Semantic segmentation (Grinvald et al., 2019)	i7 2.1 GHz CPU (Oleynikova et al., 2020)
SVO (Forster et al., 2017b)	2016	✓	✓	✓	-	-	UP Board (40 ips), ODROID XU4 (50 ips) (Delmerico and Scaramuzza, 2018)
DSO (Engel et al., 2018)	2016	✓	-	-	-	-	i7-4910MQ CPU (7 ips)
(Teixeira and Chli, 2016)	2016	✓	-	-	-	-	i7 4700MQ (around 143 ips per KF with (Leutenegger et al., 2015))
BundleFusion (Dai et al., 2017)	2016	-	-	-	RGB-D	-	Titan X GPU (around 36 ips)
Chisel (Klingensmith et al., 2015)	2015	✓	-	✓	RGB-D	-	Tango tablet 4GB RAM, quadcore CPU, Tegra K1 GPU Tango mobile phone, 2GB RAM, quadcore CPU
ElasticFusion (Whelan et al., 2015)	2015	-	-	-	RGB-D	Semantic segmentation (McCormac et al., 2017)	Core i7-4930K, GeForce GTX 780 Ti (32 ips)
InfiniTAM (Kähler et al., 2015)	2015	-	-	✓	RGB-D	-	Tegra K1 (47 ips), iPad Air 2 (21 ips) DE5 PCIe board (44 ips), DE1 FPGA SoC (2 ips) (Gautier et al., 2019)
ROVIO (Bloesch et al., 2015)	2015	✓	✓	✓	-	-	UP Board, ODROID XU4 (22 ips) (Delmerico and Scaramuzza, 2018)
OKVIS (Leutenegger et al., 2015)	2015	✓	✓	✓	-	-	UP Board (11 ips), ODROID XU4 (3 ips) (Delmerico and Scaramuzza, 2018)
LSD-SLAM (Engel et al., 2014)	2014	✓	-	-	-	-	FPGA Zynq-7020 SoC (4.55 ips) (Boikos and Bouganis, 2016)
SLAM++ (Salas-Moreno et al., 2013)	2013	-	-	-	RGB-D	-	GPGPU implementation
KinectFusion (Newcombe et al., 2011b)	2011	-	-	-	RGB-D	-	Zynq UltraScale+ MPSoC ZCU102 (27.5 ips) (Gkeka et al., 2021)
PTAM (Klein and Murray, 2007)	2007	✓	-	-	-	-	ODROID XU4 (up to 12 ips) (Pire et al., 2017)
MonoSLAM (Davison et al., 2007)	2007	✓	-	-	-	-	Intel Pentium M 1.60 GHz (53 ips)

Plusieurs approches sont basées sur l'apprentissage profond avec une configuration *end-to-end* ou hybride. Pour le moment, les méthodes *end-to-end* n'atteignent pas les performances des méthodes géométriques en termes de précision de la trajectoire estimée. Par exemple, la configuration géométrique monoculaire (Geiger et al., 2011) est surpassée par l'approche d'apprentissage supervisée (Wang et al., 2017), mais la configuration géométrique stéréovision reste la plus précise avec une dérive moyenne de la trajectoire t_{rel} de 17.48%, 5.96% et 1.89% respectivement. Dans (Li et al., 2021), l'approche *end-to-end* donne un t_{rel} de 5.58% comparé à 3.21% et 1.89% pour les approches géométriques (Mur-Artal et al., 2015; Geiger et al., 2011) sur la base de donnée KITTI (Geiger et al., 2012). Les chaines de traitement hybride donnent une solution viable pour inclure des fonctions DNN spécifiques. Dans (Merrill et al., 2021), l'estimation de profondeur basée DNN fourni un temps de traitement de 17.07ms et 7.09ms sur le GPU embarqué Jetson Nano et TX2 respectivement et 66.41ms et 105.07ms sur le CPU respectivement avec l'optimisation Apache TVM (Wofk et al., 2019). Sur les mêmes plateformes, la méthode classique *block matching* (BM) de la librairie OpenCV atteint une performance de 19.24ms, 12.38ms sur GPU et 27.76ms, 19.49ms sur CPU respectivement. Cela démontre que la précision issue du réseau de neurone est plus rapide sur GPU, mais le BM fournit des performances temps-réel sur les deux implémentations avec un temps de traitement plus rapide sur CPU. La précision de la profondeur n'a pas été quantifiée. La robustesse de l'estimation de profondeur issue du réseau de neurone montre une carte plus nette que l'approche géométrique avec des résultats bruités (Martins et al., 2018).

Dans le cadre d'un système hétérogène où différentes plateformes matérielles sont disponibles, l'utilisation de la chaine de traitement hybride avec des calculs partitionnés sur GPU et CPU permet d'alléger le nombre d'opérations au lieu d'avoir un traitement entièrement exécuté sur CPU.

2.2. Système hétérogène embarqué pour la localisation et reconstruction 3D

Les fonctions de localisation et de reconstruction 3D requièrent beaucoup de ressources matérielles. La principale implémentation matérielle mise en évidence dans la table 2.3 inclut les CPU/GPU (i7/GTX) qui sont flexibles et consomment beaucoup d'énergie, les CPU/GPU embarqués (Pi 3B+, ORDROID, Up Board / Jetson TX2, Nano), et les co-conceptions HW/SW spécifiques FPGA SoC. Les systèmes applicatifs tels que les drones, les robots miniatures et les appareils à réalité augmentée/virtuelle ont un facteur de forme restreint et un budget de consommation d'énergie limité. Ces contraintes affectent le choix d'implémentation pour répondre aux performances de précision et de temps-réel requises. Par exemple, le budget de consommation pour les voitures autonomes est de 10W à 300W (Liu et al., 2020), de 10 à 15W pour les drones MAV (Delmerico and Scaramuzza, 2018; Sky, 2019) et de 10mW à 10W pour les robots miniatures et appareils AR/VR (Chatzopoulos et al., 2017; Suleiman et al., 2019; Palossi et al., 2019; Terry, 2019).

2.2.1. Implémentation et partitionnement des fonctions

Un système hétérogène est composé de plusieurs ressources de calcul allant du traitement dans le capteur, proche capteur et sur *Components off-the-shelf* (COTS). Les COTS, accélérateurs matériels à faible consommation d'énergie ou les systèmes compacts comme les circuits intégrés au capteur d'image font partie des plateformes embarquées disponibles permettant le partitionnement des fonctions avancées du traitement de l'image à la reconstruction 3D illustré dans la figure 2.2.

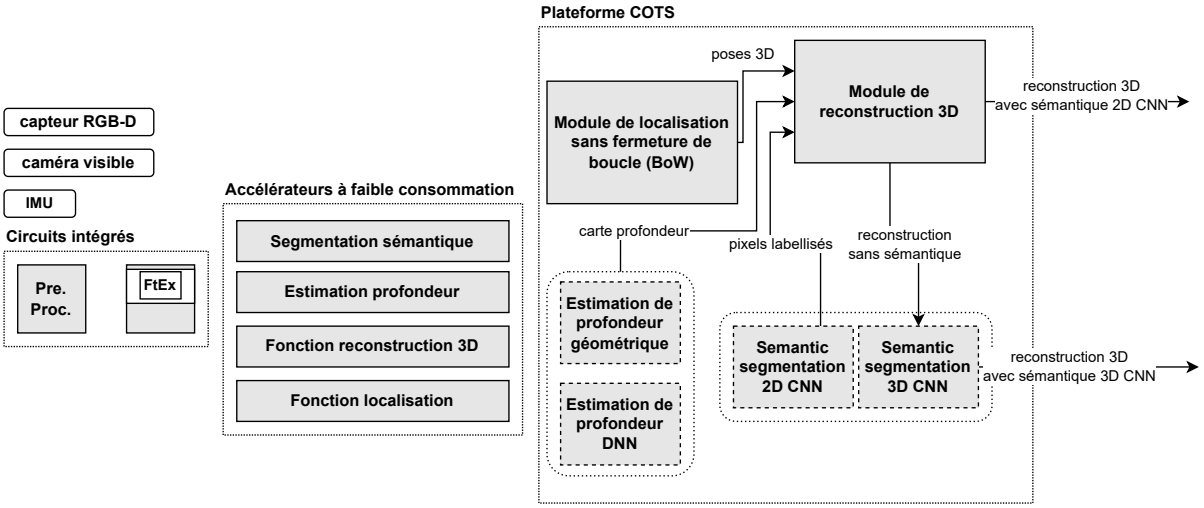


FIGURE 2.2 – Système hétérogène avec l'implémentation et le partitionnement des fonctions pour la localisation et reconstruction 3D du circuit intégré au capteur d'image à la plateforme COTS.

Components off-the-shelf (COTS)

Afin d'atteindre un traitement temps-réel avec des ressources matérielles limitées, un compromis doit être atteint entre la précision, la robustesse, le temps d'exécution, la consommation de mémoire et d'énergie (Delmerico and Scaramuzza, 2018). Plusieurs méthodes VIO ont été implémentées sur deux plateformes matérielles pour les drones MAV (Forster et al., 2017b; Bloesch et al., 2015; Qin et al., 2018; Leutenegger et al., 2015; Zhu et al., 2017). Elles incluent le Up Board avec un quad-core Intel Atom x5-Z8350 1.44GHz CPU, 4Go RAM, une consommation d'énergie d'environ 12W et le ODROID XU4 contenant un ARM hybride, un quad-core ARM A7 1.5GHz et un ARM avec la configuration big.LITTLE quad-core A15 à 2.0GHz. ODROID a 2Go de RAM et une consommation d'énergie de 10W.

La table 2.5 montre les performances temps-réel en images par seconde pour chaque méthode VIO sur les plateformes COTS. Par exemple, la méthode basée graphe VINS-Mono (Qin et al., 2018) fournit une performance de 7 ips tandis que MSCKF (Zhu et al., 2017) est à 20 ips sur ODROID comparé à 20 ips et 40 ips sur la référence PC respectivement. Le temps-réel est atteint en réduisant le nombre de point d'intérêts à détecter par image, la taille de la fenêtre glissante d'optimisation et en incluant les instructions *single instruction multiple data* (SIMD) comme les optimisations Intel SSE et ARM NEON pour Up Board et ODROID respectivement. Trouver le bon compromis, c'est également prendre en compte la précision de chaque méthode. La précision de VINS-Mono n'est pas impactée lorsqu'on passe d'un PC à une plateforme embarquée avec une précision globale de 0.16m, 0.16m et 0.15m pour le PC, Up Board et ODROID respectivement. Ce n'est pas le cas pour MSCKF où la précision globale est de 0.41m, 0.53m et 0.56m par rapport à la vérité terrain respectivement des plateformes de calculs. La méthode ROVIO (Bloesch et al., 2015) est la seule non implémentée sur Up Board en raison de la fréquence d'horloge du CPU.

TABLE 2.5 – Performances des implémentations matérielles sur plateformes embarquées. $(*)$ représente le taux *back-end* pour mettre à jour les états et la carte éparse 3D (Suleiman et al., 2019). $(**)$ correspond au temps mesuré entre l'image d'entrée et l'état mis à jour. $(\dagger)$ indique que la méthode a échoué sur une ou plusieurs séquences qui n'ont pas été prises en compte dans le calcul de performance moyenne (Delmerico and Scaramuzza, 2018).

Méthodes	Année	Implémentation HW	↑Taux	↓Puissance	↓Précision	Jeu de donnée
CNN-SLAM proc. (Li et al., 2019)	2019	ASIC	80 ips	243.6mW 61.8mW	97.90% in tr. ; 99.34% in rot.	KITTI
Navion (Suleiman et al., 2019)	2018		71 ips 19 ips*	24mW	0.23m	EuRoC
VINS-Mono (Qin et al., 2018)	2018	ODROID	7 ips**	10W	0.16m	EuRoC
MSCKF-based (Zhu et al., 2017)	2017		20 ips**		0.56m	
SVO+MSF (Forster et al., 2017b), (Lynen et al., 2013)	2016		50 ips**		0.69m†	
SVO+GTSAM (Forster et al., 2017b), (Kaess et al., 2012)	2016		66 ips**		0.11m†	
ROVIO (Bloesch et al., 2015)	2015		22 ips**		0.35m	
OKVIS (Leutenegger et al., 2015)	2013		3 ips**		0.26m†	
VINS-Mono (Qin et al., 2018)	2018	UP Board	7 ips**	15W	0.15	EuRoC
MSCKF-based (Zhu et al., 2017)	2017		20 ips**		0.53m	
SVO+MSF (Forster et al., 2017b), (Lynen et al., 2013)	2016		40 ips**		0.69m†	
SVO+GTSAM (Forster et al., 2017b), (Kaess et al., 2012)	2016		50 ips**		0.12m†	
ROVIO (Bloesch et al., 2015)	2015		-		-	
OKVIS (Leutenegger et al., 2015)	2013		11 ips**		0.27m†	

Partitionner les fonctions d'une méthode SLAM avec la fonction de fermeture de boucle demande de grandes ressources de calculs. C'est le cas avec ORB-SLAM2 (Mur-Artal and Tardós, 2017) qui a été optimisée en utilisant les instructions NEON dans les processeurs ARM des Raspberry Pi 3B+ et Jetson Nano (Silveira et al., 2020). La méthode atteint une performance moyenne de 6.11 ips sur Pi 3B+ et 9.64 ips sur Nano avec les images d'entrée d'une résolution 752×480. Ces travaux se sont concentrés sur le temps de traitement et non sur la précision de la méthode pour l'implémentation embarquée. La littérature fournit peu d'information sur les optimisations à effectuer pour les systèmes embarqués. Minimiser les opérations à réaliser dans l'exécution des fonctions de perception avancées est un axe de recherche à étudier.

La méthode SVO (Forster et al., 2017b) fournit des poses 3D avec une implémentation sur ODROID U3, la reconstruction 3D basée nuage de point dense est réalisée sur un PC W530 Lenovo grâce au GPU NVIDIA quadro K2000M (Pizzoli et al., 2014; Faessler et al., 2016). Les poses 3D et les images d'entrée sont envoyées à une fréquence de 5Hz entre la plateforme embarquée et le PC grâce à une communication Wi-Fi. Les structures de données efficaces en termes de mémoire et de temps d'exécution font face aux limitations de la reconstruction 3D dans un volume de taille fixe et la grande quantité de mémoire requise les reconstructions volumétriques (Newcombe et al., 2011b). Un volume TSDF en mouvement (Whelan et al., 2012), l'approche basée *octree* (Hornung et al., 2013; Steinbrücker et al., 2014) et le schéma de hachage (Nießner et al., 2013) permettent la reconstruction 3D d'environnements à grande échelle avec des structures de données compactes. Le schéma de hachage a été utilisé dans plusieurs recherches (Oleynikova et al., 2017; Klingensmith et al., 2015; Muglikar et al., 2020) pour réduire la consommation mémoire et le budget calculatoire car il permet d'avoir une complexité $\mathcal{O}(1)$ comparée à $\mathcal{O}(\log n)$ pour les structures *octree* (Hornung et al., 2013).

Afin de réduire la charge de calcul de la plateforme embarquée COTS, le partitionnement de la chaine de traitement avec les implémentations matérielles personnalisées est utile pour faire face à la complexité calculatoire des algorithmes de perception avancés comme l'extraction de points d'intérêts.

Accélérateurs à faible consommation d'énergie

Des parties spécifiques de la localisation avec une reconstruction 3D nuage de points semi-dense (Engel et al., 2014) ont été accélérées sur FPGA avec la compilation par le synthétiseur haut-niveau HLS (Boikos and Bouganis, 2016). L'outil HLS est utilisé pour effectuer des optimisations de conception matérielle bas niveau basées sur les algorithmes écrits en langage C pour améliorer les performances du système. Cela permet d'atteindre une performance de 4.55 ips comparée à 2.27 ips avec une implémentation logicielle et une consommation d'énergie d'environ 2.5W. Dans (Nikolic et al., 2014), le système visuel-inertiel se compose de deux capteurs d'image avec une résolution de 752×480 (Aptina MT9V034) synchronisé avec la centrale inertielle (ADIS16488) par le ARM-FPGA Xilinx Zynq 7020. La carte FPGA SoC a été utilisée pour accélérer la détection des points d'intérêts Harris (Harris and Stephens, 1988) et FAST (Rosten and Drummond, 2006) afin d'allouer plus de ressources CPU à d'autres tâches (Nikolic et al., 2014). La table 2.6 montre les performances des méthodes d'extractions de PI sur les solutions logicielles CPU pour PC (Qin et al., 2018; Nikolic et al., 2014), Jetson TX2 (Rosinol et al., 2021) et conception matérielle FPGA (Lepecq and Darouich, 2020; Liu et al., 2019; Nikolic et al., 2014) et les *application-specific integrated circuit* (ASIC) (Suleiman et al., 2019; Li et al., 2019). Les solutions logicielles sont plus simple à programmer avec plus de flexibilité, mais moins efficace que les matériels dédiés FPGA et ASIC. Le nombre de megapixels traité par seconde (MP/s) est relié à la résolution de l'image (Res.) de 752×480 (WVGA) et 640×480 (VGA) et la performance de traitement (ips) augmente significativement avec l'implémentation sur FPGA car le matériel est entièrement conçu pour la fonction d'extraction de points d'intérêts. L'ASIC VIO et ASIC CNN-VO ne sont pas concentrés seulement sur cette fonction. Ils sont conçus pour intégrer la chaine de traitement VO/VIO complète ce qui explique la différence de performance comparé aux FPGA.

Les matériels spécialisés comme les ASICs donnent plus de liberté pour concevoir des fonctions spécifiques de localisation et de reconstruction 3D. Ils offrent la possibilité d'obtenir les performances temps-réel avec une basse consommation d'énergie. Ils sont également plus coûteux en termes de

TABLE 2.6 – Nombre de pixels traités par seconde pour l'extraction de caractéristiques basé sur les implémentations logicielles (SW) et matérielles (HW).

Implementation HW	Caractéristiques	Res.	IPS	MP/s
Intel i7-4790 (Qin et al., 2018)	Shi-Tomasi	WVGA	66	23.82
Intel Core2Duo (Nikolic et al., 2014)	Harris	WVGA	40	14.44
Intel i7 (Liu et al., 2019)	ORB	VGA	30	9.45
Jetson TX2 (Rosinol et al., 2021)	KLT Shi-Tomasi	WVGA	100	36.10
ARM Cortex-A15 (Suleiman et al., 2019)	KLT Shi-Tomasi	WVGA	19	6.86
ARM Cortex-A9 (Liu et al., 2019)	ORB	VGA	3	1.05
FPGA Xilinx Zynq (Nikolic et al., 2014)	Harris	WVGA	333	120.20
FPGA Xilinx Zynq (Lepecq and Darouich, 2020)	Harris+SURF	VGA	320	98.30
FPGA Xilinx Zynq (Liu et al., 2019)	ORB	VGA	110	33.79
ASIC CNN-VO (Li et al., 2019)	CNN features	VGA	80	24.58
ASIC VIO (Suleiman et al., 2019)	KLT Shi-Tomasi	WVGA	71	25.63

temps de développement et de fabrication (Zhang et al., 2017b).

L'appareil AR/VR HoloLens2 est un des systèmes de perception les plus avancés permettant d'avoir une compréhension précise de la topologie de l'environnement avec la génération d'un maillage 3D en s'appuyant sur de multiples capteurs avec des ressources restreintes (Microsoft, 2016, 2019). Il intègre un ASIC pour traiter les fonctions de localisation 3D et de reconstruction 3D appelé *holographic processus unit* (HPU). Le HPU traite les données d'entrée issues des capteurs inertielles, de profondeur avec le capteur *time-of-flight* (ToF) et quatre caméras en niveau de gris avec un flux de 30 ips représentant une image de résolution 640×480. L'ASIC consomme moins de 10W et peut traiter plus de 1 TOPS (*tera operations per second*) et contient 125MB de SRAM. Il contient 2 milliards de transistors dans un espace de 79mm^2, 7 SIMD en virgule fixe pour le traitement des images 2D, 6 processeurs en virgule flottante pour le traitement 3D et un cœur dédié au traitement DNN programmable par Microsoft (Terry, 2019).

Navion est un accélérateur VIO à faible consommation d'énergie et consiste en trois parties principales : le *vision front-end* (VFE), le *IMU front-end* (IFE) et le *back-end* (BE) avec l'optimisation locale des poses 3D. Le suivi des points d'intérêts est la seule fonction réalisée par image. Le reste de la chaine de traitement est basée sur le traitement des *keyframes*. La détection de PI et le suivi atteignent une performance moyenne de 71 ips ce qui est comparable aux autres plateformes matérielles de la table 2.6 avec les images d'entrée de résolution 752×480. La performance est similaire à l'implémentation logicielle sur Intel i7. L'ASIC fournit plusieurs optimisations pour réduire la consommation de mémoire. Cela inclut la compression d'image dans le VFE, la réduction de la taille mémoire dans le BE et la manière dont les PI suivis sont stockés dans le BE. Les optimisations permettent de réduire la taille mémoire initiale de 3.5Mo à 854Ko. A notre connaissance, l'accélérateur à faible consommation d'énergie n'a pas été publiquement évalué sur un système applicatif réel, mais sur la base de donnée

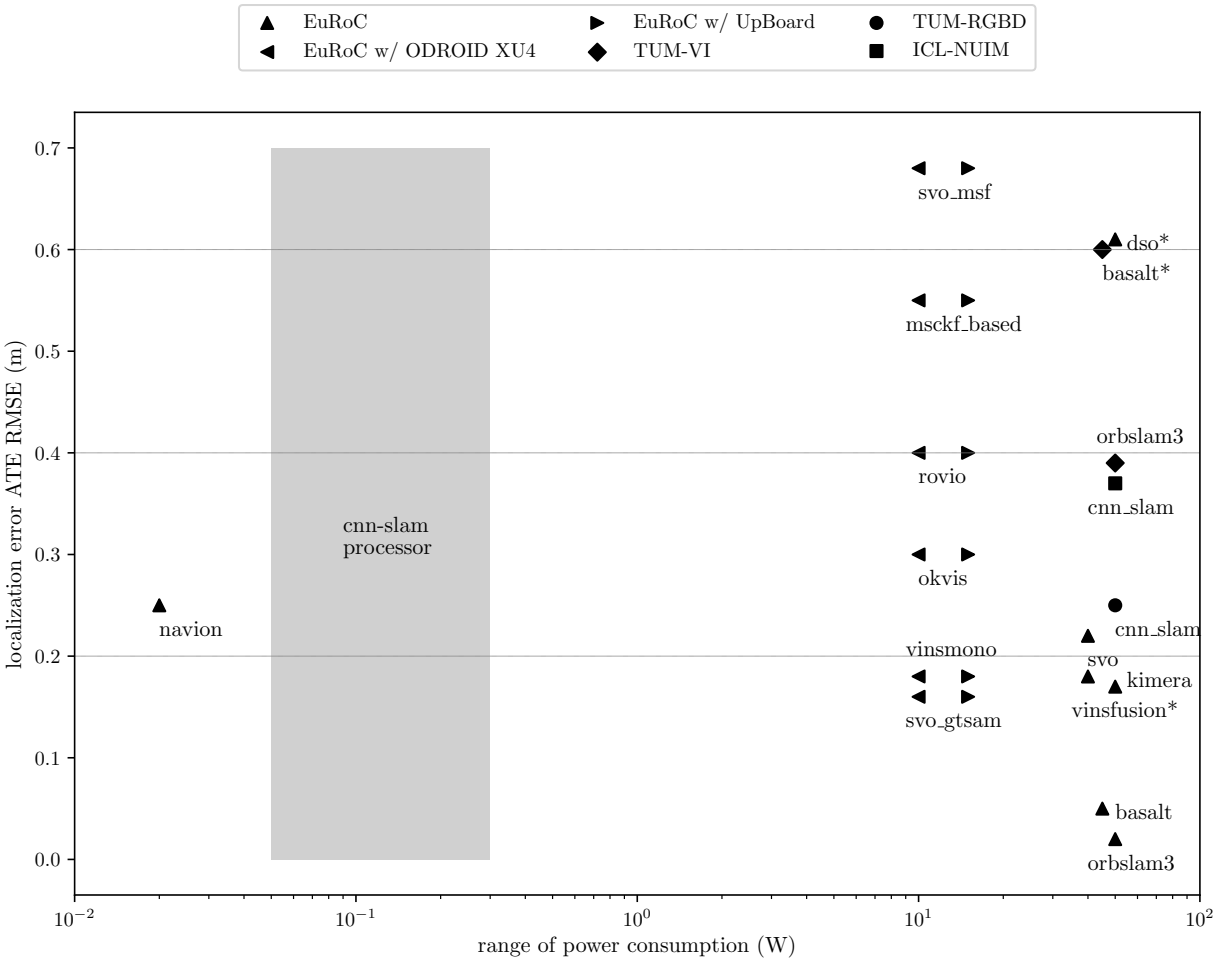


FIGURE 2.3 – Comparaison des méthodes de l'état de l'art en termes de précision (ATE RMSE) et suivant la plage de consommation d'énergie (W) entre les différentes plateformes matérielles, de l'ASIC au CPU. L'axe vertical correspond à la précision moyenne en mètres pour toutes les séquences des bases de données. Les méthodes dont l'erreur a été obtenue par (Campos et al., 2021) sont dénotées (*).

Euroc incluant des séquences enregistrées avec un drone MAV (Burri et al., 2016). Navion fournit une performance de 19 ips à la fréquence des *keyframes* et une consommation autour de 24mW avec 0.43 à 2.5 TOPS/W.

L'accélérateur Navion est entièrement basé sur les méthodes géométriques. Dans (Li et al., 2019), l'architecture VO est organisée en trois parties : une architecture CNN pour extraire les points d'intérêts à partir des images VGA, une partie perspective n points (PnP) pour calculer le déplacement de la caméra par la stratégie 2D-3D et la partie optimisation par *bundle adjustment* (BA) pour optimiser les poses générées à partir des 20 dernières *keyframes*. L'ASIC basé CNN atteint une efficacité énergétique de 3.6 à 5.34 TOPS/W, une latence de 12.5ms et consomme 243.6mW avec un flux d'entrée de 80 ips VGA et est réduit à 61.8mW avec un flux VGA de 30 ips. Les accélérateurs détaillés ne comprennent pas la fonction de fermeture de boucle pour le SLAM. Ce partitionnement dans la chaine de traitement

a été adressé par NeuroSLAM qui intègre l'architecture SLAM avec un réseau de neurones à impulsions (SNN) (Yoon and Raychowdhury, 2021). Alors que les accélérateurs VIO sont conçus avec des signaux numériques, NeuroSLAM ajoute des signaux analogiques pour imiter le SNN. Il atteint une efficacité énergétique de 7.25 à 8.79 TOPS/W avec une consommation d'énergie de 17.27 à 23.82mW. Motivé par les applications à très faible consommation d'énergie, l'utilisation des SNN pour le SLAM ouvre un nouvel axe de recherche qui n'est pas adressé dans cette thèse.

La figure 2.3 illustre les implémentations ASIC dans la plage de consommation d'énergie du mW. Plusieurs bases de données sont utilisés pour évaluer la précision de ces implémentations sous différentes conditions. La précision des méthodes diffère considérablement d'une base de donnée à une autre. Par exemple, Navion (Suleiman et al., 2019) a une erreur moyenne de 0.23m dans les séquences d'Euroc, alors que la méthode Basalt (Usenko et al., 2020) fourni une erreur moyenne de 0.051m avec les mêmes séquences et 0.6m sur les séquences de TUMVI par rapport à la vérité terrain. La figure expose également l'écart entre le nombre de méthodes temps-réel développées avec beaucoup de ressources matérielles disponible (à droite de la figure), les matériels COTS embarqués (autour de 10W) et les accélérateurs à faible consommation d'énergie (à gauche de la figure).

Circuits intégrés au capteur d'image

Les circuits intégrés au capteur d'image offrent une implémentation spécifique pour l'intégration d'algorithmes de traitement de l'image complexes à très basse latence (Millet et al., 2019; Dudek and Hicks, 2005). En plus du capteur d'image, les unités de traitement sont intégrées pour le calcul dans le capteur, ce qui améliore les performances globales du système. Ces circuits intégrés ont la capacité d'effectuer les fonctions d'extraction de PI à très haute cadence dans un système compact et à très basse consommation d'énergie. Dans (Murai et al., 2020), le système de vision est conçu pour extraire les points d'intérêts FAST et les décrire avec un descripteur de contours binaire de 44 bit (Chen et al., 2017). Ce système permet d'avoir une performance d'exécution de 300 ips. Le reste de la chaine de traitement VO tourne sur un Intel i7-6700HQ CPU avec en entrée les contours binaires et les coins tolérant au flou cinétique comparé aux caméras visibles.

2.2.2. Compromis et performances des méthodes algorithmiques

Localisation temps-réel

HoloLens (Microsoft, 2016, 2019) permet la reconstruction spatiale avec la génération d'un maillage, le traitement des mailles pour avoir des plans dans l'environnement et la compréhension de la scène avec les labels sémantiques. Le HPU intègre la chaine de traitement de la localisation pour fournir une estimation de pose de la caméra précise (Ebstyne et al.). Basé sur l'utilisation des mesures inertielles sur les capteurs d'images, l'algorithme mis en place permet la fusion multi-capteurs et la localisation est basée filtrage avec le EKF. Sur une trajectoire en boucle fermée de 287m, le système de localisation HoloLens dérive de 2.39m du point de départ au même point d'arrivée (Hübner et al., 2020; Khoshelham et al., 2019). La qualité globale du modèle 3D fourni par l'appareil RA/RV a quelques trous dans la reconstruction. Sa précision a été mesurée à partir de la vérité terrain Lidar. Avec un environnement mettant en scène plusieurs bureaux reliés par un couloir, le système donne une précision moyenne par la distance Euclidienne de 0.023m.

La stratégie d'implémentation de la localisation sur le HPU donne une vision globale sur le type d'algorithme qui peut être implémenté qui fournit des estimations de poses en temps-réel. Le système

autonome quadrotor met en évidence les compromis à effectuer pour limiter l'utilisation des ressources grâce aux paramètres de la localisation (Faessler et al., 2016). La méthode d'odométrie visuelle SVO (Forster et al., 2017b) implémentée sur ODROID U3 utilise deux *threads* pour estimer le déplacement de la caméra et pour l'insertion des *keyframes* dans la carte. Le paramètre *fast* a été utilisé et limite le nombre de points d'intérêts détectés à 120 par image et garde en mémoire un maximum de 10 *keyframes* générés dans la carte. Afin d'obtenir un système robuste, les données inertielles et les poses sont fusionnées par la méthode MSF qui utilise le filtre de Kalman étendu (Lynen et al., 2013). Les expérimentations ont été menées sur une trajectoire intérieure de 20m de long et de 1.7m de hauteur et en extérieur sur une longueur de trajectoire de 100m et 20m. Le système SVO+MSF atteint une dérive maximale de 0.5% de la distance parcourue et une erreur de trajectoire globale de 0.05m.

Reconstruction 3D temps-réel

TABLE 2.7 – Implémentation des méthodes de reconstruction 3D sur plateformes embarquées.

Méthodes	Reconstruction 3D	Implémentation HW	Taux	Cas d'usage
FLaME (Greene and Roy, 2017)	Maillage	Intel Skull Canyon NUC i7 CPU flight computer	90 ips	MAV
Voxblox (Oleynikova et al., 2017)	Volumétrique	Asctec Firefly Intel i7 2.1 GHz CPU	>4 ips	
InfiniTAM (Gautier et al., 2019)	Volumétrique	Cyclone V Terasic DE1 FPGA SoC	2 ips	Synthétique
KinectFusion (Gkeka et al., 2021)		SoC FPGA Zynq UltraScale+ MPSoC ZCU102	27.5 ips	
InfiniTAM (Gautier et al., 2019)		Stratix V Terasic DE5 PCIe board	44 ips	

La table 2.7 détaille les quelques méthodes de reconstruction 3D du maillage à la reconstruction volumétrique basée voxels utilisées sur plateformes embarquées pour drone MAV ou en prenant compte de la conception FPGA SoC. La méthode de reconstruction par maillage FLaME a été implémentée sur Intel Skull Canyon NUC pour drone. Le MAV est équipé de la caméra Point Grey Flea 3 avec un flux d'image de 60 ips et une résolution de 320×256 en plus de la centrale inertielle. Les expérimentations ont été conduites dans des environnements intérieurs et extérieurs avec une vitesse du drone allant de 2.5m/s à 3.5m/s en intérieur et extérieur respectivement. La méthode permet de reconstruire un maillage détaillé de la scène à 90 ips. Le même cas d'usage applicatif a été utilisé pour la reconstruction volumétrique Voxblox (Oleynikova et al., 2017, 2020). Les estimations de poses sont envoyées à la méthode de reconstruction 3D grâce à l'approche ROVIO (Bloesch et al., 2015). La méthode Voxblox a été évaluée sur la plateforme MAV équipée d'un Intel i7 2.1 GHz et de caméras en stéréovision synchronisées avec la centrale inertielle. Avec une représentation de la taille de voxel de 0.2m, le temps de calcul du système complet prend moins de 250ms. Afin d'obtenir une représentation de la scène précise avec une granularité plus fine, l'utilisation d'optimisations matérielles est particulièrement utile pour maintenir une performance temps-réel.

La table 2.7 montre plusieurs reconstructions basées voxels implémentées sur FPGA SoC. La méthode pionnière KinectFusion (Newcombe et al., 2011b) a été implémentée sur un SoC FPGA Xilinx UltraScale+ MPSoC ZCU102 (Gkeka et al., 2021). La partie du rendu détaillée dans la section 2.1.2 est la seule fonction de la chaine de traitement à être calculée sur ARM à cause d'accès mémoire dans le FPGA. La méthode de reconstruction atteint une performance de 27.5 ips avec une résolution d'image de 320×240 à partir de la base de donnée ICL-NUIM. Les optimisations sont concentrées sur les paramètres permettant d'améliorer le temps d'exécution, l'erreur de précision augmente alors de

0.02m à 0.08m d'une séquence à une autre. Dans (Gautier et al., 2019), c'est la méthode InfiniTAM (Kähler et al., 2015) qui a été implémentée sur le Terasic DE1 FPGA SoC qui est une plateforme à faible coût et sur le Terasic DE5 PCIe qui est une plateforme à coût élevé. Les performances temps-réel diffèrent considérablement suivant les ressources matérielles disponibles avec une performance de 2 ips et de 44 ips respectivement en sortie de la chaine de traitement avec en entrée les images de profondeur de résolution 320×240.

Bien que les FPGA SoC permettent l'accélération de fonctions avancées sur matériel avec le partitionnement de traitements sur le CPU. La reconstruction volumétrique Voxblox (Oleynikova et al., 2020) permet d'avoir une implémentation basée CPU qui offre la possibilité d'intégrer de nouveaux traitements de perception avancées dans le cadre du système hétérogène afin d'obtenir une perception de la scène plus précise et plus robuste.

2.3. Optimisation des données pour la réduction de la complexité calculatoire

Le partitionnement des fonctions sur différentes unités de calculs est une solution apportée pour accélérer les traitements les plus coûteux dans des plateformes matérielles adaptées. Lorsqu'elle est possible, l'accélération matérielle sur FPGA apporte des avantages considérables, tels qu'une plus grande parallélisation des opérations, le traitement en flux de données sans mémorisation intermédiaire et l'exécution d'opérations spécifiques pour la fonction développée. Cette technique d'implémentation réduit l'énergie consommée et apporte l'efficacité requise pour une exécution en temps-réel par rapport à une implémentation logicielle CPU. A titre d'exemple, l'exécution de la méthode d'extraction de caractéristiques ORB sur le processeur Intel i7, ARM Cortex-A9, FPGA Xilinx Zynq XCZ7045 SoC permet de traiter respectivement 9.45 MP/s, 1.05 MP/s et 33.79 MP/s (Liu et al., 2019). L'accélération FPGA augmente significativement le nombre de pixels traité par seconde par rapport à l'exécution de l'algorithme sur processeur Intel et ARM. Dans le but de réduire la complexité calculatoire et la consommation d'énergie des algorithmes de perception, l'optimisation des données est une des solutions également mise en place pour atteindre un traitement temps-réel sur plateforme matérielle embarquée. Dans ce cas, l'optimisation consiste à réduire la quantité de données nécessaires pour atteindre un compromis acceptable entre la complexité de calcul et la précision.

Quatre principales techniques d'optimisation sont identifiées.
La première est l'utilisation du format RAW des images. Les travaux menés par (Buckler et al., 2017) montrent que l'énergie consommée des capteurs pouvait être réduite de 75% en se passant des traitements d'images effectués par le processeur d'images (ISP) tels que le débruitage et la compression JPEG. L'apport principal de ne pas utiliser l'ISP d'un point de vue énergétique concerne le capteur en lui-même. Quel impact a l'utilisation des images sans pré-traitements dans la chaine de perception ? Dans les travaux de (Christie et al., 2020), le format RAW est utilisé avec l'algorithme ORBSLAM2.

La deuxième technique identifiée est la quantification de l'image d'entrée. Deux méthodes sont utilisées : 1/ la quantification basée sur les gradients où plus les pixels ont un fort gradient, plus l'encodage des bits est élevé. Cette technique préserve la qualité des zones d'intérêts de l'image, mais est fortement limitée lorsque la scène est dense en information. 2/ La quantification logarithmique effectuée via l'ADC (*analog-to-digital converter*) du capteur permet d'approximer le mappage tonal de l'image sans l'utilisation de l'ISP tout en obtenant un compromis entre précision et consommation

TABLE 2.8 – Erreur de localisation (ATE) de la méthode ORBSLAM2 en fonction de la résolution de l'image d'entrée (Christie et al., 2020) avec la base de donnée fr2-xyz (Sturm et al., 2012).

Résolution (pixels)	640×480	533×400	427×320	320×240
Réduction de résolution (%)	0	-30	-55	-75
ATE (cm)	0.29	3.48	6.46	10.90
Erreur à la référence (cm)	0	+3.19	+6.17	+10.61

d'énergie du capteur. La méthode basée sur l'extraction de point d'intérêts ORBSLAM2 est propice à la quantification de l'image jusqu'à 5 bits avec une erreur de localisation similaire à la référence (+0.69cm). Deux principales limitations sont identifiées : 1/ la méthode de quantification ne permet pas de descendre en dessous de 5 bits sans impacter significativement l'erreur de localisation (+2.53cm à 4bits et +12.52cm à 3bits). 2/ Le nombre de points à extraire par image doit être augmenté au fur et à mesure de la quantification à partir de 4bits afin de préserver une initialisation rapide de l'algorithme et un suivi précis des points caractéristiques. Cette augmentation a un impact sur le temps d'exécution des étages les plus coûteux de la chaine de traitement, à savoir l'extraction et la mise en correspondance des points d'intérêts. La quantification a également été utilisée pour le cas d'usage du drone (Suleiman et al., 2019). La technique de quantification par bloc de pixels est utilisée permettant d'obtenir une représentation de l'encodage des bits inférieur à 2 bits. Elle permet de réduire de 1.5× l'espace mémoire utilisée par la chaine de traitement SLAM tout en assurant une erreur de localisation faible et similaire à la référence. La principale limitation de ces travaux est que la quantification n'est pas utilisée à tous les étages de l'algorithme de perception. La fonction d'extraction de caractéristiques de l'image qui représente une des fonctions les plus coûteuses en termes de calcul prend en considération l'image en pleine précision pour éviter d'augmenter l'erreur de localisation.

La troisième technique est de réduire la résolution de l'image. Les travaux (Silveira et al., 2020) montrent que réduire la résolution de l'image VGA de 25% engendre une perte de qualité de la localisation qui impacte la reconstruction 3D d'un point de vue visuel et qui n'est pas quantifiée. Les expérimentations réalisées montrent que l'erreur de localisation (ATE) augmente de 3.19cm avec une réduction de 30% de la résolution de l'image (Christie et al., 2020) dans la table 2.8.

La quatrième et dernière technique identifiée porte sur la manière dont la donnée est injectée dans la chaine de traitement, soit le flux d'image d'entrée qui est initialement constant. Les caméras permettant d'avoir un flux d'image pouvant aller à très haute cadence (jusqu'à 100 ips). Pour le SLAM, l'état de l'art met en évidence que la majorité des méthodes sont basées sur la gestion des *keyframes* (KF) qui sont générées à un rythme plus faible que celui des images d'entrées (Younes et al., 2017). La décision de génération des *keyframes* prend en compte l'association des données entre l'image actuelle et les points de la carte de la dernière KF. Si le déplacement du système est trop important et que la quantité de donnée associées est faible, une nouvelle KF est générée. Les images qui ne sont pas prises en compte dans l'estimation de pose représentent un surplus de calcul sur les premiers étages du SLAM. Cela implique une consommation de mémoire, de bande passante et une complexité calculatoire élevée. Pour palier à ce problème, un système SLAM a été développé basé sur une sélection adaptative des *keyframes* à partir des mesures inertielles pré-intégrées (Piao

and Kim, 2019). Un modèle d'estimation de paramètres est mis en place et exécuter à chaque nouvelle image pour désigner les pondérations appliquées aux facteurs de vitesse et de position et de rotation entre l'image actuelle et l'image de référence. Plus les paramètres estimés sont élevés, plus l'image actuelle a de l'importance dans l'estimation de pose. Le système SLAM développé permet d'avoir une erreur de localisation moyenne sur la base de données Euroc de 0.06m comparé à 0.12m pour VINS-Mono (Qin et al., 2018). Deux principales limitations de la méthode proposée sont identifiées : 1/ les paramètres de l'algorithme sont modifiés en fonction de la difficulté de la base de données pour adapter l'utilisation des ressources de calcul. Cela ne permet pas d'avoir un système robuste et précis dans chaque cas d'usage. 2/ La méthode n'atteint pas un traitement temps-réel sans dégrader les performances du système. Le traitement temps-réel est obtenu en limitant l'utilisation du modèle d'estimation ce qui engendre une perte de précision dans la désignation des *keyframes*. L'erreur de localisation augmente en moyenne sur toutes les séquences à 0.12m avec un pic d'augmentation d'erreur de 0.21m comparé à 0.06m en moyenne avec un traitement qui n'est pas temps-réel.

2.4. Discussions et enjeux

L'implémentation de la chaine de traitement SLAM sur du matériel embarqué pose de nombreuses problématiques et enjeux qui sont discutés dans les sections suivantes. Quatre principaux points sont identifiés à partir de l'étude de l'état de l'art : 1/ l'utilisation des réseaux de neurones pour SLAM, 2/ la granularité de la reconstruction 3D, 3/ le calcul temps-réel sur plateformes embarquées et 4/ les enjeux pour l'optimisation des données.

Utilisation des réseaux de neurones pour le SLAM

Les méthodes SLAM emploient différentes stratégies pour les fonctions de localisation et de reconstruction 3D, incluant l'utilisation potentielle des réseaux de neurones profonds (DNN). L'utilisation des DNN est particulièrement utile dans la configuration d'une chaine de traitement hybride pour combiner les forces de l'apprentissage profond et la maturité des méthodes géométriques pour des fonctions spécifiques comme la détection, la description, la mise en correspondance de points d'intérêts et les représentations intermédiaires de l'image. La précision requise pour les représentations intermédiaires incluant l'estimation de la profondeur et la segmentation sémantique est un des axes de recherche à explorer. Ces méthodes sont coûteuses en termes de calculs et de mémoire. L'intégration des réseaux de neurones sur systèmes embarqués est un axe de recherche à part entière et inclut plusieurs limitations. Le modèle utilisé et la quantité de données à l'apprentissage sont deux des caractéristiques essentielles pour assurer un système précis, robuste et modulable.

Notre étude s'intéresse au système de perception SLAM dans son ensemble et à son exécution en temps réel sur systèmes à ressources restreintes de la localisation à la reconstruction 3D.

Granularité de la reconstruction 3D

La reconstruction 3D est une des fonctions les plus complexes à implémenter lorsque les ressources matérielles sont restreintes. Seulement quelques méthodes de reconstruction permettant la reconstruction 3D volumétrique de l'environnement sont implémentées sur du matériel embarqué. C'est une fonction coûteuse qui ne permet pas d'être embarquée avec la fonction de localisation du SLAM. Comment réduire la complexité calculatoire de la reconstruction sans perdre en précision pour avoir

un environnement 3D exploitable par des fonctions haut-niveaux ? L'une des questions clés qui se pose est le type de représentation. Un maillage 3D est moins complexe qu'une reconstruction volumétrique, mais ne permet pas d'avoir la granularité suffisante pour des applications qui nécessitent une interaction avec les objets de la scène. La granularité de la reconstruction 3D pose également plusieurs questions. Par exemple : quelle est la granularité requise ou la limitation de la reconstruction dans l'espace suivant le type d'application ? Quel est le compromis pour une localisation et une reconstruction 3D temps-réel avec du matériel à ressources limitées ?

Calcul temps-réel sur plateformes embarquées

Le calcul temps-réel considéré dans cette thèse est défini par la minimisation du temps de calcul entre l'information issue du capteur jusqu'à la sortie de l'algorithme SLAM, soit la position de l'image dans l'espace 3D ainsi que la reconstruction de l'environnement. Ce temps de calcul correspond à la latence de la chaine de perception qui est minimisée par différentes techniques dont l'accélération des traitements.

Avoir à notre disposition plusieurs unités de calculs permet de partitionner les fonctions de la chaine de traitement mise en place et de minimiser la latence pour avoir un traitement temps-réel. Les implémentations matérielles détaillées dans l'état de l'art montrent la quantité importante des méthodes algorithmiques développées avec beaucoup de ressources disponibles par rapport à celles développées pour les systèmes embarqués ayant avec des ressources limitées, tels que les MAV, les robots miniatures et les appareils mobiles AR/VR. D'un point de vue implémentation embarqué, au cours des deux dernières décennies, les travaux de la littérature se sont principalement concentrés sur l'accélération basée FPGA avec une co-conception logicielle et matérielle (Eyvazpour et al., 2022). Comment partitionner efficacement les fonctions de perception pour obtenir le traitement temps-réel souhaité ? Une des principales problématiques concerne alors la quantité de données qui transite et le besoin de réduire la complexité calculatoire des fonctions de perception avancées.

Enjeux pour l'optimisation des données

Dans cette thèse, nous nous intéressons à la réduction de données en entrée de la chaine de traitement SLAM dans l'optique de réduire la charge calculatoire et mémoire et trouver le compromis optimal avec la précision. Nous pouvons influer sur la quantité de données injectée en entrée de la chaine de traitement de deux manières, la réduction de l'encodage des pixels et le contrôle du flux d'entrée. La majorité des flux étant de 8b/pixel, c'est un premier levier sur lequel agir pour réduire le temps de calcul du traitement de l'image tout en influant sur les mémoires tampons qui peuvent être présentes dans le système hétérogène (Suleiman et al., 2019). Dans (Suleiman et al., 2019), la quantification est utilisée dans un seul cas d'usage, le drone, et n'est pas appliquée à tous les étages de la chaine de traitement. Comment prendre en compte la quantification de l'image dans tous les étages de la chaine de traitement sans impacter la précision de localisation de l'algorithme ? La quantification logarithmique ou basée sur les gradients de l'image réduit la consommation d'énergie tout en conservant une erreur de localisation faible jusqu'à 5 bits/pixel (Christie et al., 2020). Peut-on quantifier plus fortement la donnée tout en assurant une erreur de localisation faible et similaire à la référence sur des bases de données plus complexes présentant de longues séquences et des changements de luminosité intenses ? Comment se comportent les algorithmes de localisation moins sophistiqués que ORBSLAM2 avec la quantification dont l'intégration embarquée pose de nombreux défis en termes

de calcul temps-réel et de consommation mémoire ? Peut-on généraliser cette approche pour impacter la chaine de traitement allant jusqu'à la reconstruction 3D avec une méthode générique ?

Le deuxième levier à considérer pour la réduction de données concerne le contrôle du flux d'entrée. La fréquence d'images qui peut aller jusqu'à 100 images par seconde, génère une quantité importante de calculs pour la chaine de perception. Les méthodes basées sur la gestion des *keyframes* effectuent de nombreux calculs sur chaque image sans qu'elle soit prise en compte dans l'estimation de pose finale, car les informations consécutives sont redondantes. Les travaux de l'état de l'art montrent que la prise en compte du mouvement du système est un des axes à étudier pour éviter d'injecter un flux constant d'images (Piao and Kim, 2019). Pouvons-nous injecter efficacement les données en entrée de la chaine de traitement en assurant une exécution temps-réel du système et une faible erreur de localisation ? Comment adapter l'injection du flux en fonction des cas d'usages dont les mouvements détectés diffèrent en termes d'accélérations linéaires et angulaires ? Quel est le gain calculatoire par rapport à un flux d'entrée constant ? Telles sont quelques-unes des questions posées sur les leviers identifiés pour la réduction des données auxquelles nous voulons apporter des réponses.

3 - Système hétérogène à faible complexité : du capteur à la reconstruction 3D

Sommaire

Une des principales questions de recherche abordée dans la thèse concerne le besoin de réduire les ressources utilisées par la chaine de traitement SLAM, de la sortie des capteurs à la reconstruction 3D. La littérature détaillée dans le chapitre 2 montre que peu de méthodes temps-réel sont développées avec les contraintes embarquées. Comment réduire les besoins en ressources des algorithmes de type SLAM ? Quel est le partitionnement de la chaîne de traitement approprié pour le système hétérogène intégrant le traitement dans le capteur, proche capteur (FPGA) et dans la plateforme embarquée COTS (processeur ARM, GPU embarqué) ?

Ce chapitre discute des leviers pour réduire la complexité des algorithmes dans la section 3.1 et détaille le système hétérogène proposé dans la section 3.2.1 à partir des différentes plateformes matérielles. Le partitionnement des fonctions de perception temps-réel allant jusqu'à la reconstruction 3D est réalisé. Les contributions proposées dans la thèse répondent à la question de la réduction des besoins en ressources par l'optimisation de la quantité de données injectées par le capteur à la reconstruction 3D de l'environnement. A partir du système hétérogène composé de plusieurs unités de calcul, la section 3.2.2 montre comment la complexité du système est optimisée grâce aux contributions qui se concentrent sur la manière dont les données sont injectées. La section 3.3 décrit les mécanismes d'optimisation proposés pour la perception temps-réel dans un système embarqué hétérogène.

3.1. Réduction de la complexité calculatoire des algorithmes

L'état de l'art détaillé dans le chapitre 2 montre la quantité de méthodes SLAM développées sans imposer de contraintes matérielles alors que c'est crucial pour des systèmes réels tels que les drones, robots de services et casques à réalité mixte. Plusieurs leviers sont mis en évidences pour réduire la complexité des algorithmes.

Le premier est l'accélération matérielle qui permet d'implémenter des fonctions coûteuses en termes de calcul dans un matériel dédié. Cela permet d'accélérer le calcul en utilisant, par exemple, un FPGA et d'alléger le traitement dans la plateforme principale COTS où l'ensemble de la chaine de traitement est exécutée. La première étape est d'identifier ces fonctions qui ralentissent le traitement

de l'algorithme. L'extraction de caractéristiques de l'image est la fonction qui prend la majeure partie du temps de calcul avec près de 50% du temps total de la partie *tracking* d'ORBSLAM3 (Campos et al., 2021). Le *front-end* de KimeraVIO (Rosinol et al., 2020) qui prend en considération la détection, le suivi et la mise en correspondance des points caractéristiques prend environ 46% du temps total de l'exécution de l'algorithme. L'accélération de cette partie sur FPGA permet d'obtenir un traitement temps-réel à 497.86 MP/s avec une résolution de l'image à 3840×2160 et une performance de 60 ips (Wasala et al., 2022). D'autres travaux mettent en évidence l'intérêt de l'accélération matérielle permettant d'obtenir une performance temps-réel des algorithmes de types SLAM (Nikolic et al., 2014; Lepecq and Darouich, 2020). Les implémentations peuvent se caractériser de deux manières, le traitement en flux et en différé. Les traitements différés impliquent l'utilisation de mémoires tampons permettant de stocker les images d'entrée. L'utilisation des algorithmes pour des systèmes restreints, comme les drones ou les casques à réalité mixte requière une faible latence, un haut débit et de précision à faible consommation d'énergie. Cela implique une utilisation minime de ces mémoires tampons. Pour cette raison, le traitement en flux est la meilleure option pour l'implémentation matérielle des fonctions de perception. Cette méthode permet d'avoir un traitement pixel à l'instant où la donnée est reçue. Au lieu d'attendre que l'image complète soit envoyée dans la chaine, le traitement en flux prend en compte le calcul pixel par pixel. L'utilisation de l'espace mémoire est minimisée, car l'image entière de résolution $M \times N$ n'est pas stockée. Le choix de cette technique d'implémentation convient aux algorithmes de type SLAM dont le premier étage est le traitement d'image pour extraire les points caractéristiques. L'extraction des points d'intérêts ne requiert pas d'avoir toute l'image en mémoire, mais seulement des lignes de pixels. Au mieux de notre connaissance, aucune méthode qui présente une accélération matérielle des premiers étages du SLAM porte les travaux jusqu'à la reconstruction 3D. C'est pour cette raison que nous proposons d'utiliser ce levier pour réduire la complexité du SLAM allant jusqu'à une représentation 3D de l'environnement dense et riche en information géométrique.

Une des principales caractéristiques du SLAM est l'utilisation des *keyframes* (KF). Elles sont générées à un flux plus faible que les images d'entrées. La quantité de données à traiter entre l'image d'entrée et la génération de cette KF dans la carte est alors importante. Comment réduire la quantité de données dans la chaine de traitement SLAM ? La différence entre le flux d'image d'entrée à 20 ips et la performance du SLAM à la fréquence des *keyframes* qui est de l'ordre de 10 ips est importante et questionne la nécessité ou non de traiter chaque image du flux d'entrée. Dans le cadre du SLAM, la précision et le nombre de calculs à effectuer dépendent fortement du mouvement du système. L'intuition est alors d'injecter seulement les images qui permettent de générer les *keyframes* dans le but de réduire considérablement le nombre d'opérations réalisé au total lors de l'exécution de l'algorithme. Par exemple, en considérant que le flux d'entrée à 20 ips représente 100% des images à traiter, la gestion de *keyframes* ne représente que 50% des images qui sont prises en compte dans l'estimation de pose et la génération de la carte 3D du SLAM. L'injection des données en fonction du mouvement du système est un des leviers pris en compte dans les travaux de la thèse pour réduire la complexité des méthodes algorithmiques.

Un autre levier pris en compte est la réduction de la donnée injectée dans la chaine de traitement. L'état de l'art a montré que la quantification des pixels permet d'avoir une réduction d'environ 74% de l'espace mémoire utilisée dans l'architecture Navion (Suleiman et al., 2019). Cette quantification n'est pas appliquée dans tous les étages de l'algorithme et n'est utilisée que dans un seul cas d'usage,

le drone. Un des points critiques concernant la robustesse des performances du SLAM en termes de localisation est d'assurer la qualité des points extraits de l'image. Un des axes sur lesquels la thèse se concentre est la généralisation de la compression pour injecter dans la chaine de traitement l'image quantifiée.

Les contributions proposées visent à réduire la complexité des méthodes de localisation et de reconstruction 3D dans le but de diminuer le nombre d'opérations réalisées dans la chaine de traitement tout en assurant une précision élevée, l'exécution en temps réel et une faible consommation d'énergie. La littérature montre plusieurs contributions sur différents systèmes embarqués avec une implémentation plus ou moins flexible suivant les plateformes matérielles utilisées. Le fait de prendre en compte le système hétérogène dans son ensemble est plus efficace pour minimiser la latence du système afin d'assurer une exécution temps-réel grâce aux implémentations des fonctions dans le circuit intégré au capteur à la plateforme embarquée COTS (processeur ARM ou GPU embarqué) en passant par les différents accélérateurs FPGAs. Comment partitionner les fonctions de perception pour réduire la latence et optimiser l'espace mémoire ?

3.2. Partitionnement des fonctions de perception

3.2.1. Description du système hétérogène

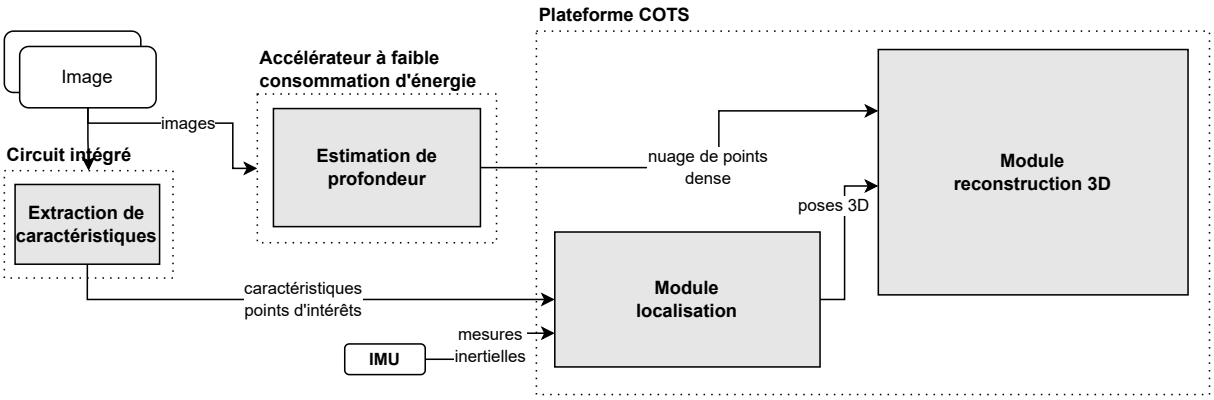


FIGURE 3.1 – Proposition du système hétérogène pour la chaine de traitement SLAM allant du circuit intégré au capteur d'image à la plateforme COTS.

Le partitionnement des fonctions sur les différentes unités de calculs consiste à répartir et accélérer les fonctions de perception de la chaine de traitement SLAM allant jusqu'à la reconstruction 3D basée voxel. Ce partitionnement prend en compte la co-conception matérielle et logicielle dans le but de réduire au maximum la latence. La figure 3.1 illustre la proposition de la thèse sur le développement du système hétérogène pour la chaine de traitement SLAM. Au mieux de notre connaissance, les implémentations issues des travaux de la littérature ne proposent pas de partitionnement des fonctions de perception SLAM permettant d'obtenir une reconstruction 3D à partir des données de profondeur.

A partir des images issues des capteurs, l'extraction des caractéristiques est la partie du module de localisation qui est accélérée dans le circuit intégré au capteur (Murai et al., 2020). Cette

implémentation permet de réduire la latence et ainsi d'accélérer le traitement de l'image avec des performances de 300 ips pour la fonction d'extraction des caractéristiques de l'image. L'intégration de la fonction n'est pas triviale et manque de flexibilité. Les travaux de la thèse ne se concentrent pas sur l'implémentation matérielle de cette fonction sur le circuit intégré, mais est prise en compte pour réduire considérablement la latence du système. Les points d'intérêts calculés sont passés en entrée du module de localisation pour le calcul de l'estimation de pose. Ce calcul est effectué dans la plateforme COTS avec un processeur ARM ainsi que le reste de la chaine de traitement. Dans cette configuration, le module de localisation ne comprend pas le paramètre de fermeture de boucle qui est coûteux en mémoire et dont la complexité calculatoire augmente en fonction du temps (Li et al., 2018b). Les poses 3D générées par le module de localisation sont envoyées dans le module de reconstruction 3D basé voxel (Oleynikova et al., 2017) qui prend également en entrée le nuage de points dense généré par l'estimation de profondeur.

L'accélérateur à faible consommation d'énergie met en place le traitement de l'estimation de profondeur des images en stéreovision sur FPGA. Cette implémentation permet d'augmenter les performances de l'algorithme en termes de vitesse d'exécution d'environ 50% par rapport à une implémentation CPU (Firmansyah and Yamaguchi, 2021). D'autres travaux de l'état de l'art implémentent l'algorithme *semiglobal matching* (SGM) sur FPGA (Lee and Kim, 2022) qui représente un défi en raison du nombre de calcul important à réaliser avec 2 tera opérations par seconde (TOP/s), une bande passant de 39 terabits par seconde (Tb/s) et une consommation mémoire d'environ 386MB pour une image ayant 2 millions de pixels à 30 images par seconde (Li et al., 2018c). L'implémentation FPGA permet d'accélérer le calcul jusqu'à atteindre un maximum de 103 images par seconde sur la plateforme Zynq ultrascale+ MPSoC (Lee and Kim, 2022).

3.2.2. Injection des données dans le système hétérogène

,

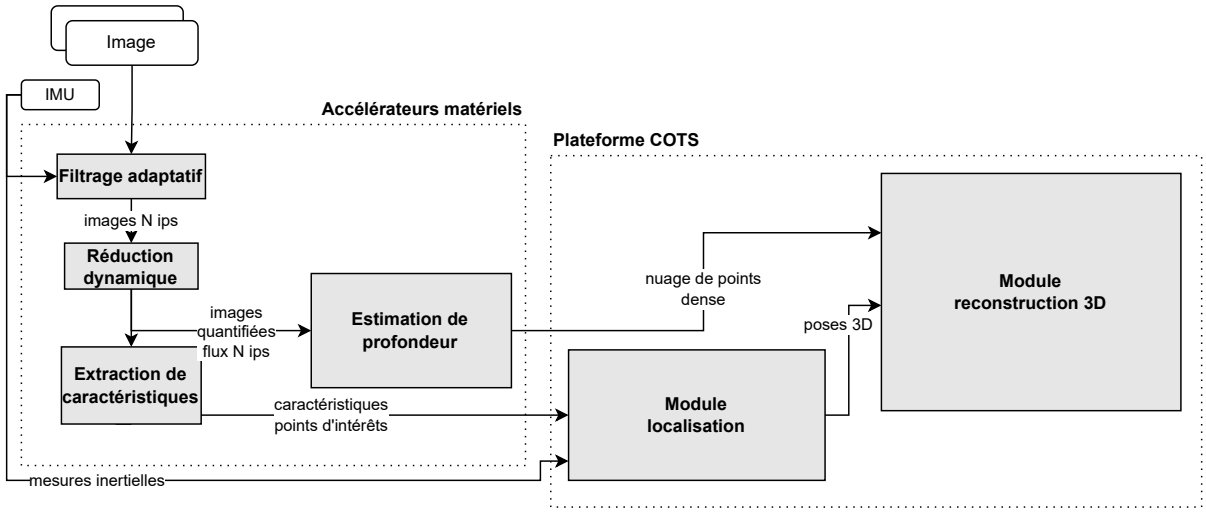


FIGURE 3.2 – Proposition des mécanismes d'optimisation pour réduire la complexité du système hétérogène proposé.

La figure 3.2 montre à quel niveau les propositions interviennent dans la chaine de traitement. Elles ont pour objectif de réduire la complexité du système hétérogène par le biais de deux principales contributions. Le filtrage adaptatif et la réduction de la dynamique qui sont les fonctions implémentées en flux au plus proche du capteur.

Le premier levier est le contrôle du flux d'entrée pour réduire la quantité de données dans la chaine de traitement avec le filtrage adaptatif. Ce mécanisme répond à la problématique posée dans la section 2.4 sur l'injection des données. Une des limitations adressées concerne l'ajout d'un algorithme d'optimisation dont l'impact sur l'exécution temps-réel de la chaine de perception est négligeable et ne détériore pas les performances de précision des méthodes SLAM. Le flux d'entrée d'images de 20 ips est réduit de manière adaptative en fonction du mouvement du système capté par les mesures inertielles. Théoriquement, cette technique d'injection des données permet de générer une *keyframe* pour chaque image d'entrée en raison du changement d'incidence d'observation suffisant pour prendre en compte les données dans l'estimation de la pose 3D. Plus il y a de mouvements, plus le nombre d'images injectées est élevé. Le flux n'est plus constant, mais variable et on évite ainsi d'effectuer des calculs redondants. Cette approche apporte deux principaux avantages : 1/ plus la vitesse de déplacement du système est faible, plus la bande passante est réduite. 2/ En injectant les images suivant les mesures d'accélérations, le nombre d'opérations à effectuer est réduit.

Le dernier levier pris en compte est la réduction de la dynamique de l'image. A partir des images déclenchées par le filtrage adaptatif implémenté au plus proche du capteur, l'encodage des pixels est réduit grâce à la réduction de la dynamique. La quantification mise en place impacte tous les étages de la chaine de traitement et vise à réduire l'encodage des bits inférieur à 5 bits/pixel tout en assurant une qualité de performance élevée. L'implémentation matérielle de ce mécanisme permet de quantifier l'image jusqu'à obtenir une représentation à 1 bit par pixel (1b/px) grâce à deux méthodes : la

median cut (MC) (Heckbert, 1982) qui partitionne de manière équilibrée la quantification en gardant la dynamique de l'image et la quantification par bloc, intitulée *block-wise* (BW) (Suleiman et al., 2019). L'implémentation de la fonction de quantification sur du matériel spécifique permet d'avoir le contrôle sur chaque bit au lieu d'être contraint à une architecture conventionnelle 8 bits. La réduction de la dynamique impact les fonctions de traitement d'images et permet de réduire la complexité matérielle des fonctions d'extraction de caractéristiques et d'estimation de profondeur. A quel point peut-on quantifier l'image tout en minimisant la perte de performance ? Les travaux visent à trouver le compromis entre l'optimisation des données avec la quantification de l'image et la performance de l'algorithme de perception.

3.3. Mécanismes d'optimisation des données

Cette section apporte les éléments théoriques qui permettent de proposer les mécanismes d'optimisation des données et ce qu'ils impliquent du point de vue de la chaine de traitement SLAM. Pour chacune des propositions, le gain potentiel du système hétérogène est détaillé. Les sections 3.3.1 et 3.3.2 décrivent respectivement la réduction de la dynamique de l'image et le contrôle du flux par le filtrage adaptatif.

3.3.1. Réduction de la dynamique de l'image

Les méthodes SLAM basées sur les caméras visibles incluent les fonctions de traitement d'images dès le premier étage de la chaine de traitement. Ces fonctions se concentrent sur l'extraction des caractéristiques de la scène. De nombreuses techniques permettent d'exploiter les pixels dans le but d'estimer la pose 3D de la caméra. Parmi ces techniques, la littérature indique que l'information pertinente est caractérisée par l'exploitation de tous les pixels de l'image, l'extraction des contours ou des points d'intérêts, représentés par les coins. Les travaux de la thèse se concentrent sur le calcul des points d'intérêts de l'image exploités par l'approche indirecte. Le chapitre 2 a montré que cette technique permet d'avoir une extraction robuste et rapide pour un calcul en temps réel grâce à l'exploitation éparse et distribuée des caractéristiques de l'image. La qualité de localisation dépend alors fortement de l'association des données qui se base sur les points d'intérêts extraits.

Ce premier étage du traitement SLAM est coûteux en termes de calcul et de mémoire. Plusieurs méthodes sont utilisées permettant d'avoir une détection précise et une description des caractéristiques de l'image en se basant sur la différence de gradients et sur la représentation pyramidale de l'image permettent d'être invariant à l'échelle. Le mécanisme proposé vise à réduire l'empreinte mémoire et la complexité calculatoire des algorithmes SLAM en réduisant la dynamique de l'image (DR) par deux méthodes différentes, le *median cut* (MC) qui permet de garder la dynamique de l'image et la deuxième est la quantification par bloc, intitulée *block-wise* (BW).

DR avec Median cut

Algorithm 1 Algorithme *Median Cut*

1. Déplacer tous les pixels en un seul tableau
2. Trier chaque valeur d'intensité
3. Trouver la médiane du tableau
4. Séparer le tableau par l'index de la médiane
5. Calculer la moyenne de chaque sous-tableau
6. Répéter à partir de 3. pour obtenir la palette de 2^n valeurs d'intensité

La méthode *median cut* (MC) génère une plage de 2^n valeurs d'intensités, avec n le nombre de bits tout en gardant la dynamique de l'image. L'algorithme 1 décrit les différentes étapes pour réaliser la quantification qui se base sur le calcul de la médiane à partir des pixels de l'image triés dans un tableau. Le tableau est séparé en deux par l'index de la médiane permettant de calculer la moyenne de chaque sous-tableau ce qui correspond aux deux valeurs d'intensités. Le calcul de la médiane et la séparation de chaque sous-tableau est répété pour obtenir la palette de 2^n valeurs d'intensités.

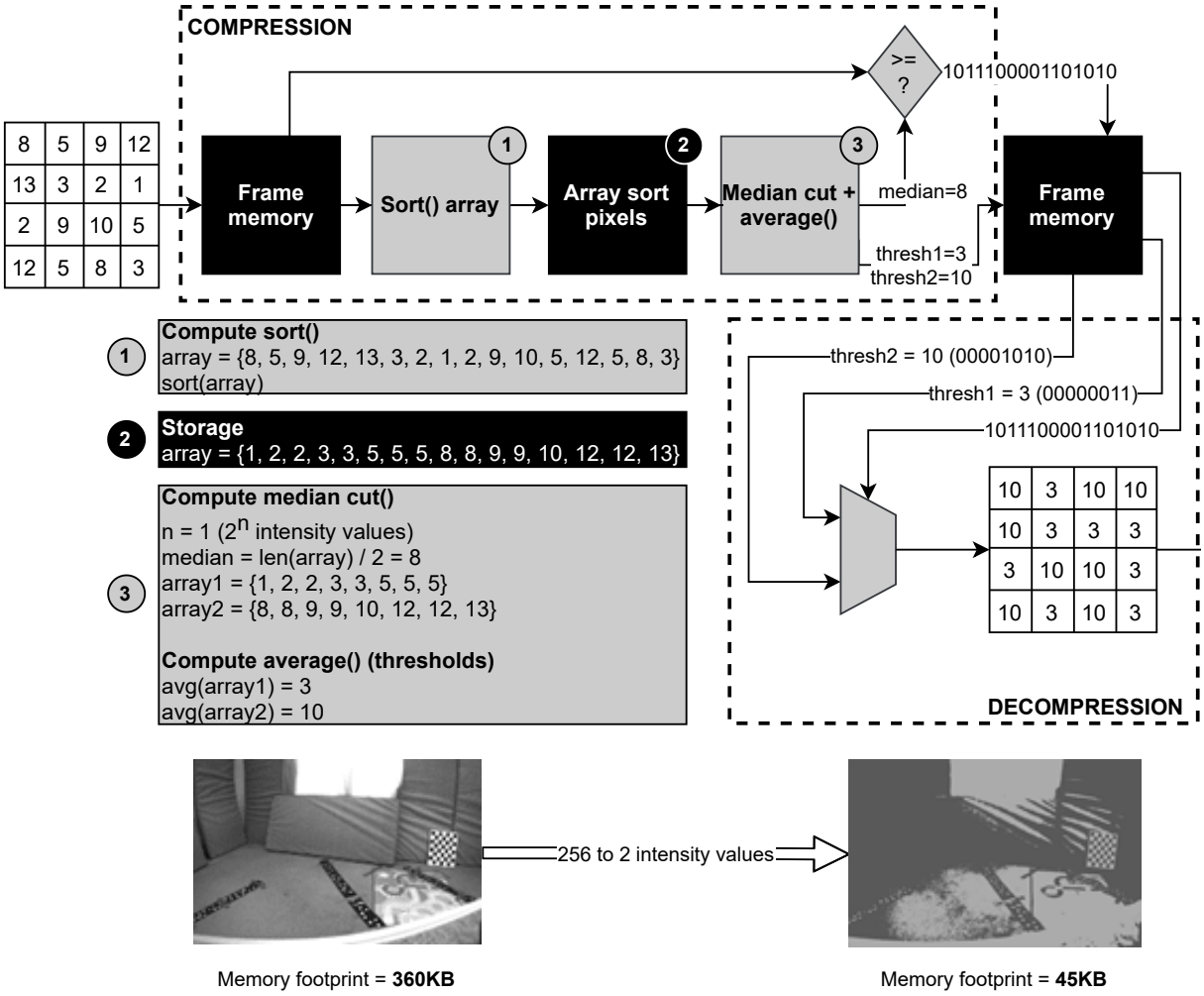


FIGURE 3.3 – Quantification de l'image par la méthode *median cut*. Exemple avec une image d'entrée de taille 4×4.

L'implémentation de la méthode d'un point de vue matériel se caractérise par une phase de compression et une phase de décompression. La figure 3.3 illustre la procédure de quantification avec un exemple d'image de taille 4×4 pour une représentation à 1b par pixel. La compression de l'image est répartie en trois étapes. 1/ Le calcul du tri rend l'utilisation de la mémoire tampon d'image obligatoire. 2/ Les pixels triés sont stockés en mémoire permettant de calculer la médiane du tableau. 3/ Une fois la médiane calculée qui correspond à la valeur '8', le tableau est séparé en deux sous-tableaux pour une représentation à 1b par pixel. La moyenne des valeurs d'intensités de chaque sous-tableau est calculée et correspond aux seuils de quantification avec *thresh1=3* et *thresh2=10* codés en 8b. La valeur de la médiane est comparée aux pixels d'origine afin d'obtenir une mémoire d'image binaire. Si la valeur d'intensité du pixel est supérieure ou égale à la médiane, le bit est à '1', sinon à '0'. A partir de l'image binaire, la phase de décompression permet d'attribuer la valeur de *thresh1* ou *thresh2* aux pixels dont le bit est à '1' ou '0' respectivement. Ainsi, la méthode *median*

cut permet de quantifier l'image avec la représentation à 1b par pixel tout en gardant la dynamique de l'image et permet d'obtenir une empreinte mémoire de 45Ko comparée à l'image de base ayant une empreinte mémoire de 360Ko.

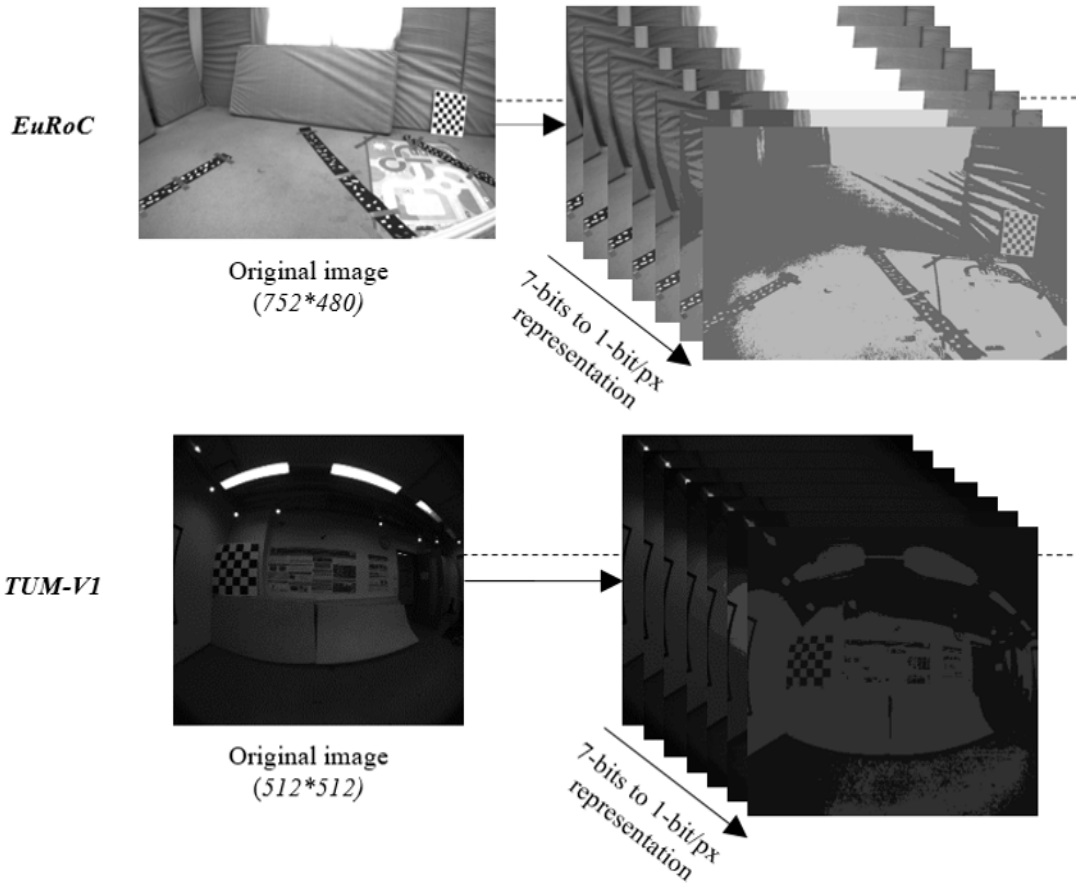


FIGURE 3.4 – Résultats visuels de la quantification *median cut* pour les images de la base de donnée Euroc (Burri et al., 2016) et TUMVI (Schubert et al., 2018). L'image originale est quantifiée pour une représentation allant de 7-bits/px à 1-bit/px. La dernière représentation montre l'image contenant deux valeurs d'intensités maximales.

Les images générées par la méthode sont montrées dans la figure 3.4 en prenant en compte les images de la base de donnée Euroc (Burri et al., 2016) et TUMVI (Schubert et al., 2018). La sortie de l'algorithme est un tableau à deux dimensions (2D) représentant les pixels associés aux valeurs candidates du pixel (soit la valeur moyenne de chaque sous-tableau). Les valeurs stockées dans le tableau 2D sont la position de la valeur du pixel candidat dans le tableau. Cette méthode représente les pixels d'entrée en $log_2 n$ bit(s). La réduction de données atteint un pourcentage maximum de 87% pour la représentation à 1 bit par pixel (bpp) avec une résolution d'image de 752×480.

Algorithm 2 Fonction de quantification par bloc

$block = N \times N$ ▷ N = 4 ; 8 or 16
for chaque pixel(i, j) dans le bloc $N \times N$ **do**
  Trouver les valeurs $max()$ et $min()$
  $thresh = (max + min)/2$ ▷ Définir le seuil
  if $value(i, j) >= thresh$ **then**
    $value(i, j) = thresh$
  else
    $value(i, j) = min$

DR avec Block-wise

La deuxième méthode de quantification utilisée est l'approche *block-wise*. Cette technique permet de quantifier l'image d'entrée par bloc de $N \times N$ pixels, N correspondant à la taille du bloc. L'algorithme 2 décrit la procédure de quantification. Pour chaque bloc de pixels, les valeurs maximales et minimales des valeurs d'intensités sont utilisées pour calculer le seuil défini par $(max + min)/2$. Ce seuil est nécessaire pour définir les nouvelles valeurs d'intensités de l'image quantifiée. Les travaux de la thèse prennent en compte trois tailles de blocs différentes : 4×4, 8×8 et 16×16 ce qui permet de quantifier l'image à 2b, 1.25b et 1.06b par pixel respectivement.

La méthode *block-wise* est utilisée, car elle apporte deux différences majeures par rapport au *median cut*. La première est que les opérations à réaliser facilitent l'implémentation matérielle. En effet, elles se composent du calcul de maximum et minimum qui facilite l'utilisation du traitement en flux. La deuxième est la représentation de l'image quantifiée avec l'ajout de bruit en raison de la quantification par bloc. Les images générées par la méthode sont montrées dans la figure 3.5 en prenant en compte les images de la base de données Euroc (Burri et al., 2016) et TUMVI (Schubert et al., 2018). Les images illustrent le bruit ajouté par la méthode de quantification. Plus la taille du bloc est grand, plus la perte d'information de l'image de base est élevée. Quel est l'impact sur la qualité des points d'intérêts extraits ? Quel gain apporte la méthode sur la chaine de traitement ? Les travaux de la thèse permettent de répondre aux deux questions qui représentent l'apport de l'optimisation de la donnée et à quel prix sur les performances de l'algorithme de localisation et de reconstruction 3D en temps-réel.

La table 3.1 montre le pourcentage de réduction de donnée pour chaque approche de quantification qui est calculé par $nb_{px} * n/8$, avec nb_{px} le nombre de pixels de l'image. Avec une résolution d'image de 752×480, la réduction de donnée pour chaque configuration de quantification par bloc correspond à 75%, 84% et 87% pour les tailles de bloc 4, 8 et 16 respectivement.

Discussions et avantages

Les algorithmes SLAM requièrent une gestion de la mémoire dans les fonctions du *front-end* pour stocker les images d'entrées (Zhang et al., 2017b). L'avantage des méthodes de quantifications étudiées dans cette contribution est de pouvoir considérablement quantifier l'image en perdant un minimum d'information utile tout en minimisant l'impact sur la robustesse et la précision du SLAM. Dans le tableau 3.1, le pourcentage de réduction de données des images avec une quantification *mediant cut* (med$X$) et *block-wise* (block $N \times N$) est montré. On remarque une quantification allant

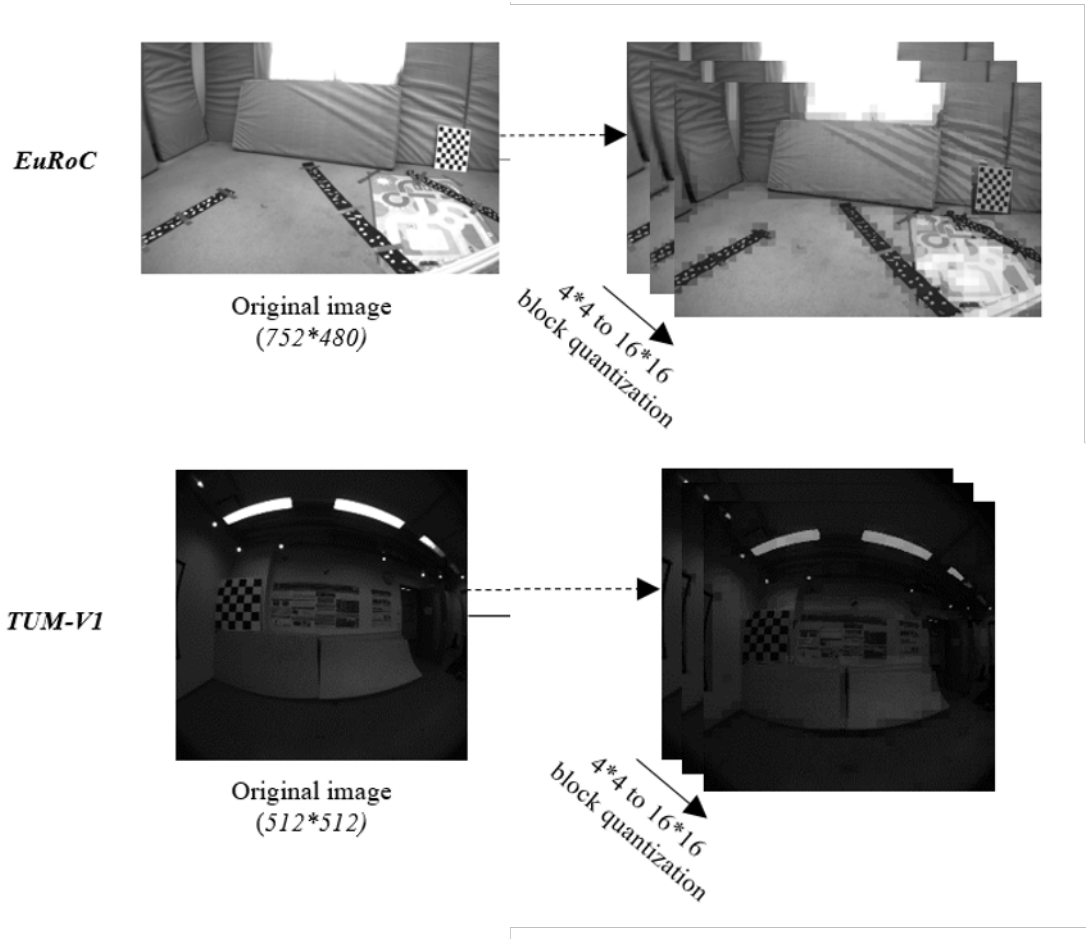


FIGURE 3.5 – Résultats visuels de la quantification *block-wise* pour les images de la base de donnée Euroc (Burri et al., 2016) et TUMVI (Schubert et al., 2018). L'image originale est quantifiée par bloc de pixels allant d'une région 4×4, 8×8 à 16×16.

jusqu'à 87% de réduction de données permettant : 1/ la réduction de l'empreinte mémoire des images, 2/ la réduction de la bande passante en sortie du capteur et 3/ une réduction du nombre de calculs à effectuer lors de traitement des images des fonctions de la partie *front-end* du SLAM.

Dans le système complet SLAM présenté dans la section 3.2.2, un des principaux points de discussion concernant cette étude sur la quantification est la qualité de reconstruction 3D que l'on obtient. En effet, cette dernière repose sur la précision de l'estimation de profondeur des images. Avec la méthode de quantification *median cut*, plus la réduction de donnée est importante, de moins en moins de textures sont présentes à l'image ce qui détériore la qualité de l'estimation de profondeur. Quelle est la précision suffisante requise de profondeur pour la reconstruction 3D ? Le manuscrit répond à cette question dans le chapitre 5 qui présente les résultats de performance de la chaine de traitement SLAM à partir de la méthodologie établie et détaillée dans le chapitre 4.

TABLE 3.1 – Pourcentage de réduction de données suivant la méthode de quantification *median cut* (med$X$) et *block-wise* (block $N \times N$) avec l'image d'entrée de résolution 752×480.

Quantization method	Memory footprint (KB)	Data reduction (%)
8b/px	361	0
med7	316	12
med6	271	25
med5	226	37
med4	180	50
med3	135	62
med2	90	75
block 4×4	90	75
block 8×8	56	84
med1	45	87
block 16×16	45	87

3.3.2. Contrôle du flux d'images par le filtrage adaptatif

La chaine de traitement SLAM mise en place dans la thèse prend en compte le capteur d'image et la centrale inertielle. La réduction de la dynamique permet d'impacter les caractéristiques du capteur d'image par la quantification des pixels. Bien que le nombre de calculs à effectuer est réduit, le capteur injecte les données à un flux constant. Cela implique une consommation de mémoire, de bande passante et augmente la complexité calculatoire pour un traitement en temps-réel. Peut-on réduire cette quantité de donnée en entrée de l'algorithme ? Le SLAM dépend fortement du mouvement du système par la gestion de *keyframes* qui sont générées suivant le calcul de parallaxe des images consécutives. L'utilisation des mesures inertielles permet d'avoir cette information de mouvement et de changement de position du système. En prenant en compte ces données, le flux d'images peut être injecté en fonction du mouvement du système. Dans les cas d'usages du drone et du casque à réalité mixte où les mouvements sont fréquents et brusques, les *keyframes* sont alors générées non plus à 5-6 images par seconde, mais à partir de l'image injectée qui dépend du mouvement de système.

L'objectif de cette contribution est de définir une méthode d'injection des images en entrée permettant de réduire le nombre d'images à traiter sans dégrader la précision de localisation. Le filtrage adaptatif (AF) est une approche permettant de réduire la quantité de donnée en réduisant le flux d'images en entrée en fonction du mouvement du système avec un faible impact sur la qualité de la trajectoire reconstruite par le SLAM.

Stratégie de décimation pour contrôler le flux d'images

L'approche de filtrage adaptatif est utilisée comme une stratégie de décimation pour contrôler le flux d'images en entrée d'algorithmes SLAM comme il est montré dans la Figure 3.6. Le flux d'entrée est modulé grâce aux mesures inertielles issues de l'*inertial measurements unit* (IMU) de la manière suivante : plus la vitesse mesurée est faible, plus le nombre d'images injectées est réduit. Le traitement des données inertielles requis pour déterminer le déplacement du système est basé sur les vitesses angulaires et les accélérations linéaires. Le gyroscope est utilisé pour connaître les rotations

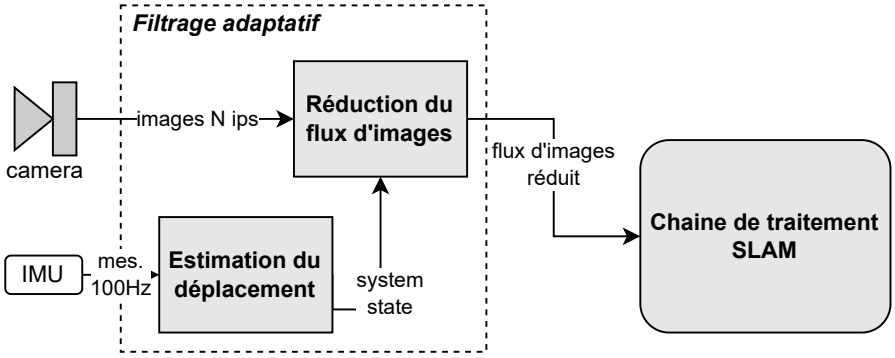


FIGURE 3.6 – Méthode de réduction de données avec le filtrage adaptatif pour les algorithmes SLAM. Le flux d'images d'entrée à N ips est contrôlé à partir de l'état du système mesuré par les données inertielles. Plus il y a de mouvement, plus le flux en entrée du SLAM se rapprochera du débit d'entrée initiale issue de la caméra. Moins il y a de mouvement, plus le débit sera réduit.

du système par la norme de l'amplitude de la vitesse angulaire :

$$\Omega_k = \sqrt{w_{x,k}^2 + w_{y,k}^2 + w_{z,k}^2} \tag{3.1}$$

avec $w_{x,y,z}$, la vitesse angulaire sur l'axe x, y et z à chaque échantillon k. Pour le filtrage adaptatif, la quantité de donnée entre deux images consécutives est prise en compte pour analyser le déplacement du système. La moyenne de la norme de la vitesse angulaire est calculée de la manière suivante :

$$S_G = \frac{1}{N} \sum_{I_{i-1}<k<I_i} \Omega_k \tag{3.2}$$

avec N, le nombre de données inertielles entre deux images consécutives I_i et I_{i-1} et Ω_k la norme de la vitesse angulaire.
Le traitement des données d'accélérations linéaires entre images consécutives prend en compte l'accumulation des mesures d'accélération. Elle permet de déterminer les phases de translation et est décrite de la manière suivante :

1. Suppression de la contribution de gravité : durant la phase statique d'initialisation, l'accélération du système est mesurée et les données acquises seront déduites durant la phase d'acquisition.
2. Suppression de la mesure de bruit : calcul de la moyenne de valeurs d'accélération sur une fenêtre glissante N_{acc}.
3. Estimation de la vitesse : calcul de la moyenne S_A des valeurs entre deux images consécutives basées sur N_{acc}.

La manière dont la gravité est filtrée des mesures d'accélérations est faite de manière naïve où l'on considère le système statique. La suppression de la contribution de gravité g permet de connaître seulement l'accélération dynamique du système sur les trois axes x, y et z. Dans un système réel, l'accéléromètre de l'IMU présente des données bruitées. La suppression de la mesure de bruit suit la méthode d'accumulation M sur les trois axes où la taille de la fenêtre glissante prise en compte

correspond à n échantillons :

$$M = \frac{1}{N_{acc}} \sum_{k=k-n}^{k} A_k - g \tag{3.3}$$

avec k l'échantillon de la mesure d'accélération A. L'équation 3.3 montre notamment la suppression de la contribution de gravité g pour chaque mesure d'accélération sur les axes x, y et z.

Lors de tests effectués dans le cadre d'un stage sur le développement de l'algorithme, il a été remarqué que les mesures d'accélérations capturées lorsque le système était au repos présentaient de nombreuses fluctuations pouvant perturber l'exploitation des données acquises. C'est pour cette raison que ces états de repos E_{repos} sont déterminés de la manière suivante sur les trois axes :

$$E_{repos} = \begin{cases} 1 & \text{si} \sum_{k=k_{last}-n_{repos}}^{k_{last}} M_k < 0.01 \text{ et } n_{repos} > 100 \\ 0 & \text{sinon} \end{cases} \tag{3.4}$$

avec k_{last} le dernier échantillon dans l'état de repos, n_{repos} le nombre d'échantillons considéré M_k la valeur de l'accumulation de l'échantillon k.

L'estimation de la vitesse est donc basée sur l'accumulation des mesures d'accélérations sur les trois axes entre deux images consécutives correspondant à la moyenne basée sur N_{acc} :

$$S_A = \frac{1}{N_{acc}} \sum_{I_{i-1}<k<I_i} M_k - M_{E_repos} \tag{3.5}$$

avec M_k la valeur moyenne de l'accumulation d'échantillon des mesures d'accélération entre deux images consécutives I_{i-1} et I_i et M_{E_repos} la moyenne des valeurs d'accélérations de l'état de repos détaillé dans l'équation 3.4.

A partir des vitesses calculées du gyroscope et de l'accéléromètre, les mouvements du système sont détectés par S_G et S_A respectivement. La méthode de filtrage adaptatif permet de sélectionner les images en fonctions du mouvement du système. Plus le mouvement est important, plus le nombre d'images sélectionnées est élevé. Pour cela, plusieurs heuristiques sont utilisées.

Heuristiques utilisées

La sélection des images d'entrée est déterminée par l'amplitude des signaux S_G et S_A. Deux heuristiques sont utilisées pour déterminer la quantité d'images à utiliser à partir des mouvements du système. La première consiste à fixer des seuils constants, la seconde rend ces seuils adaptatifs par rapport aux mouvements du système.

Le déplacement rotatif du système correspondant à la vitesse angulaire présente un changement de scène plus important qu'un mouvement translatif (c'est-à-dire vitesse linéaire). L'heuristique à seuil constant présente trois seuils pour la moyenne de la norme de la vitesse angulaire S_G et un seuil pour l'estimation de la vitesse linéaire S_A. Les trois seuils Th_{SGt} utilisés pour S_G permettent de déclencher les images à une fréquence variable selon l'amplitude de la vitesse de rotation du système. La fréquence de déclenchement des images de $Th_{SG3} > Th_{SG2} > Th_{SG1}$. Si les valeurs d'amplitudes dépassent un seuil défini comme constant, alors un mouvement important est considéré comme étant détecté pour injecter les images. La figure 3.7 montre le flux d'images adaptatif en rapport aux mesures inertielles

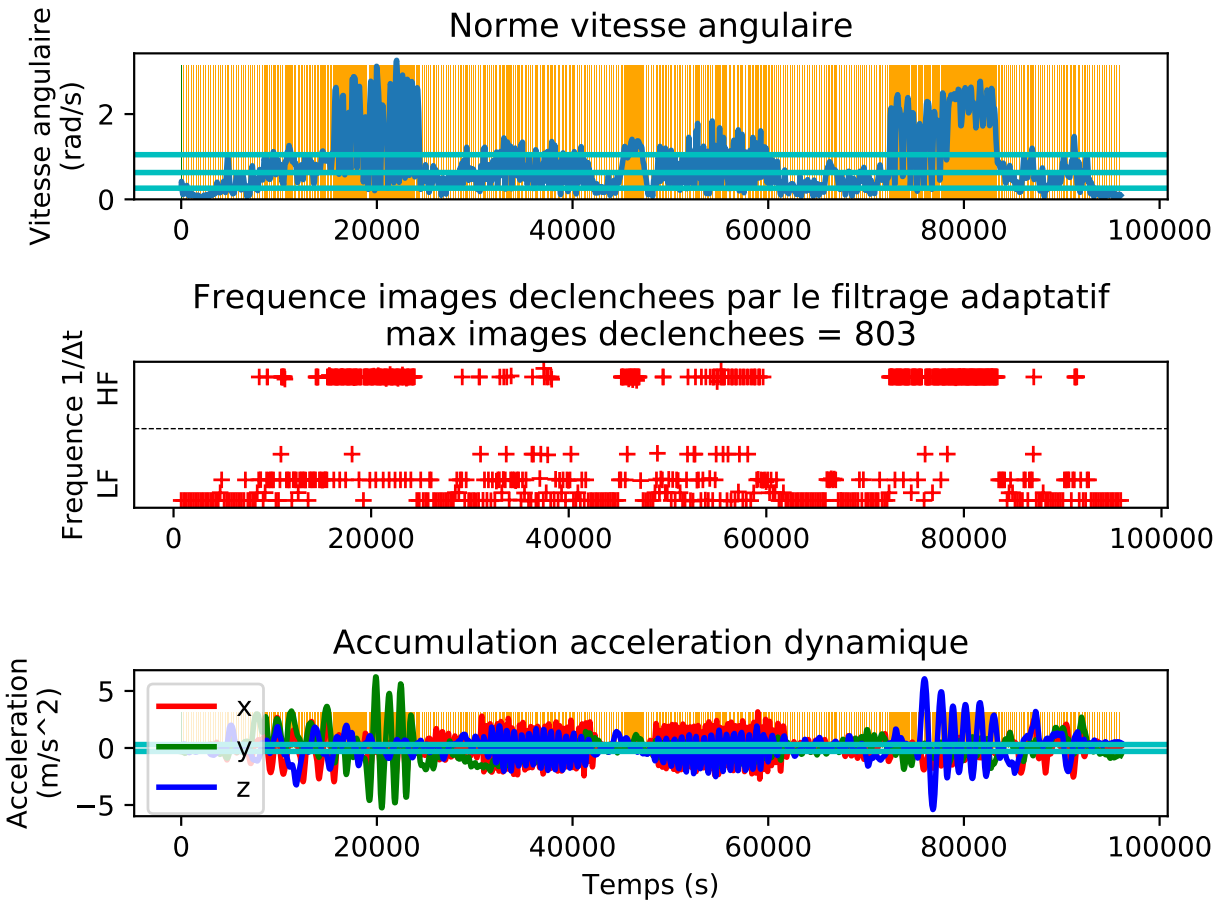


FIGURE 3.7 – Résultat du filtrage adaptatif avec l'heuristique correspondant aux seuils constants sur la séquence *corridor4* de la base de donnée TUMVI (Schubert et al., 2018) illustrés par les barres bleues horizontales. La figure du dessus illustre la vitesse angulaire du système en fonction du temps. La figure du milieu montre la fréquence des images déclenchées en fonction du temps. La ligne en pointillé représente la fréquence moyenne. Les zones de déclenchement à haute fréquence (HF) sont indiquées au-dessus de cette ligne, tandis que les zones à basse fréquence (LF) sont indiquées en dessous. La figure du dessous illustre l'accélération linéaire sur les trois axes x, y et z. Plus le système se déplace rapidement, plus le flux d'image déclenché est important et est illustré par les barres verticales jaunes.

avec la sélection d'images basée sur la norme de la vitesse angulaire et l'accumulation des mesures d'accélération pour la séquence *corridor4* de la base de données TUMVI (Schubert et al., 2018). Les barres horizontales bleues correspondent aux seuils de déclenchement constants de l'image en fonction du mouvement détecté par l'IMU. Dans le cas de mouvement significatif, la sélection des images est déclenchée et identifiée par les barres verticales jaunes. La figure du milieu montre la fréquence des images déclenchées en fonction du temps. La fréquence est calculée par $1/\Delta t$ avec Δt correspondant au temps entre deux acquisitions consécutives. La ligne en pointillé représente la fréquence moyenne. Les zones de déclenchement à haute fréquence (HF) sont indiquées au-dessus de cette ligne, tandis que les zones à basse fréquence (LF) sont indiquées en dessous. Ces zones correspondent aux endroits

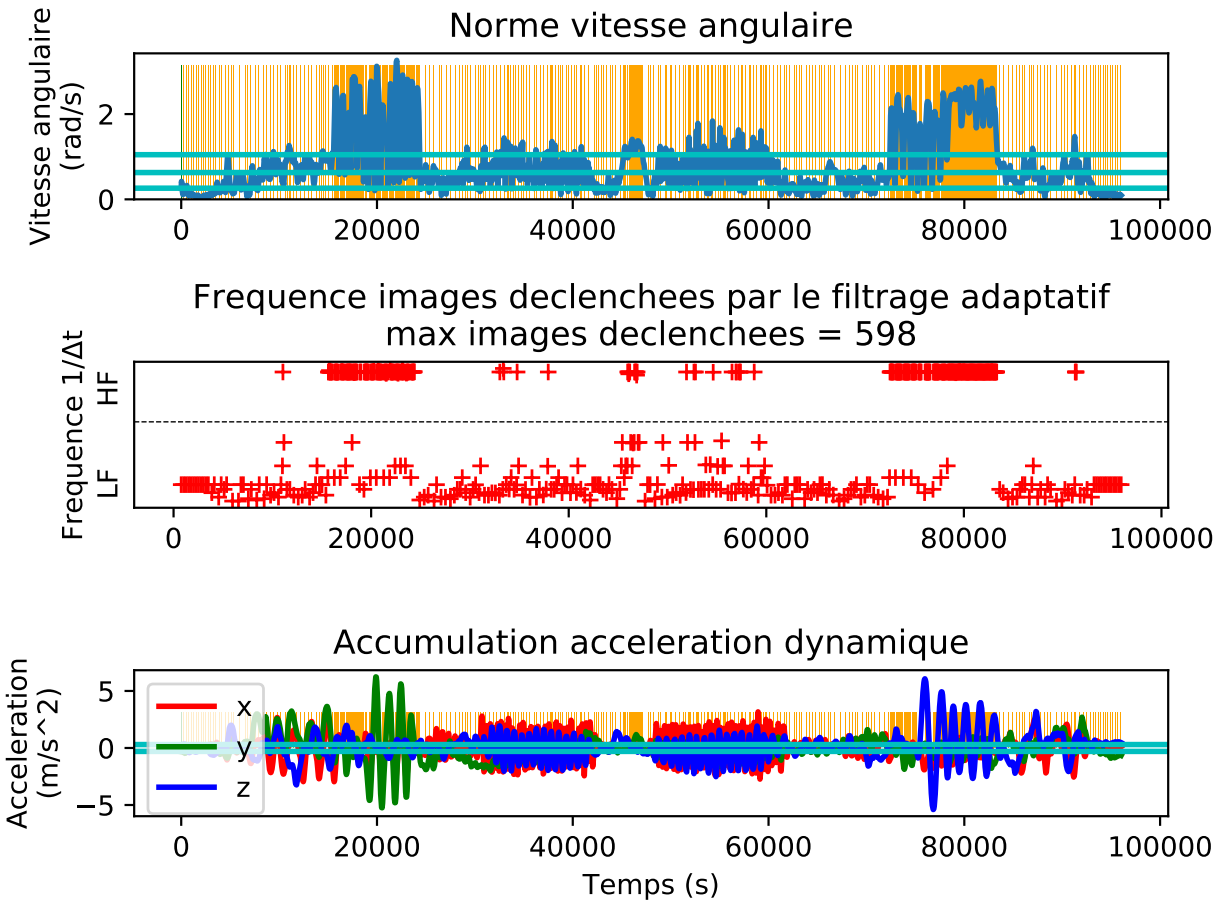


FIGURE 3.8 – Résultat du filtrage adaptatif avec l'heuristique correspondant aux seuils adaptatifs sur la séquence *corridor4* de la base de donnée TUMVI (Schubert et al., 2018). La figure du dessus montre la vitesse angulaire du système en fonction du temps. La figure du milieu montre la fréquence des images déclenchées en fonction du temps. La ligne en pointillé représente la fréquence moyenne. Les zones de déclenchement à haute fréquence (HF) sont indiquées au-dessus de cette ligne, tandis que les zones à basse fréquence (LF) sont indiquées en dessous. La figure du dessous montre l'accélération linéaire sur les trois axes x, y et z. L'heuristique adaptative déclenche les images en fonction du seuil $adapt_{threshold}$ où la sélection est plus tardive qu'avec les seuils constants. Cela rend l'heuristique plus sélective et réduit le nombre d'images à injecter en fonction des mouvements du système.

où le système se déplace de manière significative dans le cas HF ou lorsque le déplacement est faible dans le cas LF, comme le montrent les mesures de vitesse angulaire.

Une deuxième heuristique est proposée avec des seuils adaptatifs. Ces derniers s'adaptent au signal S_G et permet de définir un nouveau seuil $adapt_{threshold}$ suivant le mode de déclenchement des images (déclenchement rapide, medium et lent) déterminé à partir du mouvement rotatif du système. Le résultat est montré dans la Figure 3.8. Cette deuxième heuristique produit un déclenchement des images qui s'adapte au mouvement du système et présente une sélection plus sélective des images à injecter. En effet, on s'intéresse ici principalement au signal S_G et la sélection des images se fait plus tardivement qu'avec les seuils constants à cause de la nouvelle prise en compte du seuil $adapt_{threshold}$.

La figure qui met en évidence la fréquence d'images déclenchées par le filtrage adaptatif montre que les zones de déclenchement illustrées par les croix rouges sont moins fréquentes qu'avec l'heuristique constante de la figure 3.7. Le nombre d'images déclenchées est réduit de 26% par rapport au résultat du filtrage adaptatif avec les seuils constants.

Discussions et avantages

TABLE 3.2 – Nombre d'images déclenchées par les deux heuristiques du filtrage adaptatif (seuils constants et adaptatifs). Les séquences utilisées sont issues de la base de donnée (Burri et al., 2016) et (Schubert et al., 2018) pour *V*, MH** et *room*, corridor** respectivement.

seq.	référence 20 ips	AF const. thresh.	AF adapt. thresh.
V101	2912	595 (4 ips)	448 (3 ips)
V102	1710	510 (6 ips)	291 (3 ips)
V103	2149	770 (7 ips)	469 (4 ips)
V201	2280	465 (4 ips)	352 (3 ips)
V202	2348	752 (6 ips)	452 (4 ips)
MH01	3682	706 (4 ips)	589 (3 ips)
MH02	3040	598 (4 ips)	511 (3 ips)
MH03	2700	556 (4 ips)	420 (3 ips)
MH04	2033	399 (4 ips)	329 (3 ips)
MH05	2273	444 (4 ips)	383 (3 ips)
room1	2821	1960 (14 ips)	1660 (12 ips)
room2	2882	1459 (10 ips)	975 (7 ips)
room3	2821	1408 (10 ips)	993 (7 ips)
room5	2847	1986 (14 ips)	1761 (12 ips)
room6	2636	1102 (8 ips)	672 (5 ips)
corridor1	5990	3050 (10 ips)	2306 (8 ips)
corridor2	6772	3264 (10 ips)	2332 (7 ips)
corridor3	5802	3716 (13 ips)	2919 (10 ips)
corridor4	1927	809 (8 ips)	604 (6 ips)
corridor5	5914	2760 (9 ips)	1990 (7 ips)

La méthode de filtrage adaptatif présente de nombreux avantages dans le but de réduire la complexité calculatoire des algorithmes SLAM. Dans un premier temps, la Table 3.2 montre que l'on réduit considérablement le nombre d'images dans une séquence enregistrée à 20 images par seconde. Les séquences *Vicons (V), Machine Hall (MH)* correspondent à la base de donnée Euroc (Burri et al., 2016) capturée avec l'utilisation d'un drone (MAV) et présente des séquences plus ou moins difficiles (01 : facile, 02 : médium, 03 : difficile) avec des caractéristiques incluant du flou, des changements de luminosité et des mouvements rapides du drone. Les séquences *room, corridor* font parties de la base donnée TUMVI (Schubert et al., 2018) enregistrée à l'aide d'un système de vision tenu à la main sur de longues séquences, des changements de luminosités et des mouvements rapides. L'heuristique

constante et adaptative du filtrage adaptative permettent de réduire en moyenne la quantité d'image pour Euroc de 94% et 83.11% respectivement et de 46.76% et 59.88% respectivement pour la base de donnée TUMVI. La Table 3.2 montre également le nombre de ips correspondant pour chaque séquence enregistrée à 20 ips constant. Bien que le filtrage adaptatif permet d'avoir un taux d'image par seconde variable, ce taux est une moyenne sur toute la séquence et permet d'avoir une comparaison de flux d'images par rapport à la référence. Le nombre de ips est radicalement diminué et les difficultés des séquences sont mis en évidence. Là où le filtrage adaptatif réduit le flux de près de 5 fois dans la base de donnée Euroc en moyenne, on réduit le flux en moyenne d'environ 2 fois pour la base de donnée TUMVI qui présente plus de variation de mouvements durant de longues séquences.

Un des points à discuter concernant cette méthode est la manière dont le changement de scène est géré, par exemple la variation de luminosité alors que le système reste statique. Dans ce cas, le filtrage adaptatif doit permettre de capturer une image même si aucun mouvement n'est détecté. C'est pour cette raison que l'algorithme développé met en place un signal de déclenchement à une fréquence minimale f_{min} défini pour injecter les images à f_{min} ips permettant d'avoir une image injectée même si le système reste statique. Ce signal est montré dans les Figures 3.7 et 3.8 par les barres vertes verticales présentes au début de la séquence. En effet, les base de données prise en compte contiennent de nombreuses variations de mouvements, c'est pour cette raison que ce déclenchement "préventif" au changement de luminosité est montré seulement lorsque les données issues de l'IMU ne présente aucun mouvement significatif permettant de déclencher les images.

4 - Méthodologie de recherche pour l'analyse et l'impact des mécanismes proposés

Sommaire

Ce chapitre détaille la méthodologie employée dans le but d'étudier et de valider l'impact des contributions du chapitre 3 sur les algorithmes SLAM.

Le chapitre est réparti en trois sections : la section 4.1 présente et détaille en profondeur les jeux de données, les métriques et les outils utilisés durant la validation de nos contributions sur les algorithmes ainsi que les métriques de performance en termes de précision et de consommation mémoire des algorithmes SLAM. La section 4.2 détaille les méthodes SLAM qui ont été identifiées et sélectionnées pour faire partie de l'étude. Pour terminer, la section 4.3 explique la méthodologie commune des expérimentations réalisées suivant les contributions de la thèse.

4.1. Jeux de données, outils et métriques utilisés

4.1.1. Jeux de données

Les travaux de la thèse se concentrent sur deux cas d'usages : le drone et le casque à réalité mixte. Là où le drone peut embarquer une carte de style RaspberryPi, ou NVIDIA Jetson TX2, le casque a des contraintes matérielles plus importantes. Par ces deux cas d'usages, on couvre non seulement plusieurs contraintes matérielles et de consommation d'énergie, mais également des mouvements qui sont différents l'un de l'autre et contraignants pour les algorithmes de localisation en temps réel.

Le cas d'usage du drone est couvert par le jeu de donnée Euroc (Burri et al., 2016) qui est considéré comme la base de données de référence pour les algorithmes SLAM avec 11 séquences qui ont été enregistrées par le drone équipé d'un capteur visuel-inertiel. Les images ont été capturées à 20 images par seconde (ips) avec une résolution WVGA de 752×480. La centrale inertielle utilisée fournit des données à 200Hz (ADIS16488). La base de donnée contient 11 séquences enregistrées dans trois endroits différents : *Vicon room 1 (V1)*, *Vicon room 2 (V2)* et *Machine Hall (MH)*. La table 4.1 décrit les caractéristiques de chaque séquence. Elles fournissent une longueur de trajectoire allant de 36.5m pour *V201* à 130.9m pour *MH03*. Les séquences sont classifiées par ordre de difficulté, allant de facile à difficile avec différentes caractéristiques. Des mouvements lents et une scène claire pour les faciles et des mouvements rapides et une forte variation de luminosité pour les difficiles. Euroc reste encore aujourd'hui une référence pour la comparaison des algorithmes de localisation temps-réel bien que les séquences deviennent plus simples pour les méthodes SLAM dont la problématique de

TABLE 4.1 – Caractéristiques des séquences de la base de données Euroc.

Nom seq.	Catégory	Image (#)	Longueur, Durée	Caracteristiques
V101	easy	2912	58.6 m, 144 s	slow motion, bright scene
V102	medium	1710	75.9 m, 83.5 s	fast motion, bright scene
V103	difficult	2149	79 m, 105 s	fast motion, motion blur
V201	easy	2280	36.5 m, 112 s	slow motion, bright scene
V202	medium	2348	83.2 m, 115 s	fast motion, bright scene
V203	difficult	1922	86.1 m, 115 s	fast motion, motion blur
MH01	easy	3682	80.6 m, 182 s	good texture, bright scene
MH02	easy	3040	73.5 m, 150 s	good texture, bright scene
MH03	medium	2700	130.9 m, 132 s	fast motion, bright scene
MH04	difficult	2033	91.7 m, 99 s	fast motion, dark scene
MH05	difficult	2273	97.6 m, 111 s	fast motion, dark scene

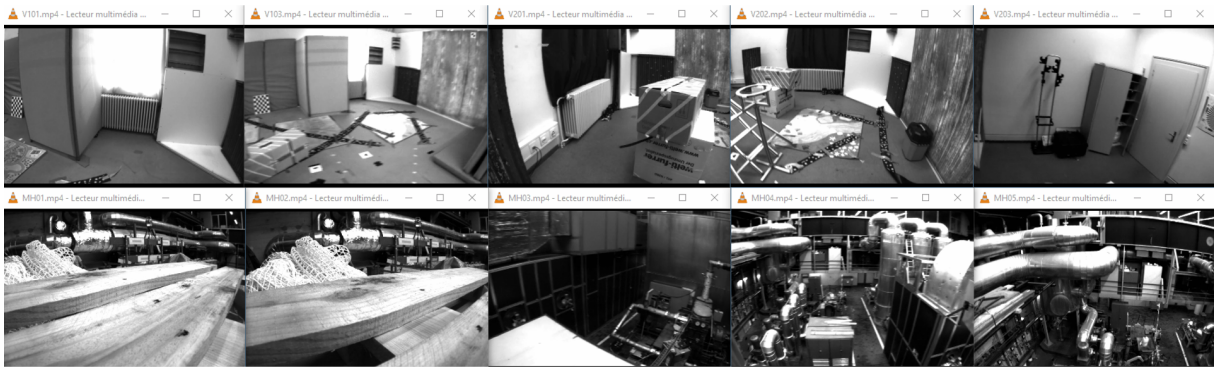
précision a évoluée.

C'est aussi pour cette raison que la base de donnée TUMVI (Schubert et al., 2018) a été utilisée. Ce jeu de données représente le cas d'usage du casque RA/RV avec une caméra tenue à la main et dont les mouvements sont brusques sur de longues séquences. La base TUMVI fournit 28 séquences enregistrées avec la caméra *fisheye* en configuration stéréovision avec une résolution de 512×512 à 20 ips et l'ajout de données inertielles. La centrale inertielle fournit les données à 200Hz (BMI160). Contrairement à Euroc où la vérité terrain est disponible sur l'entièreté de la trajectoire, TUMVI fournit la référence seulement au début à la fin des séquences dans la pièce *room*. La table 4.2 détaille les caractéristiques de la base de données. Deux environnements sont utilisés dans les travaux : *room* où les séquences sont enregistrées dans une seule pièce avec une longueur de trajectoire allant de 67m à 146m et *corridor* où les trajectoires sont longues et passent par plusieurs bureaux avec une longueur allant de 114m à 322m. Les trois principales caractéristiques qui rendent cette base de donnée difficile pour les algorithmes SLAM sont : 1/ les mouvements brusques et rapides de la caméra, 2/ la faible luminosité des séquences et 3/ les longues séquences de *corridor* qui passent plusieurs fois aux endroits déjà visités.

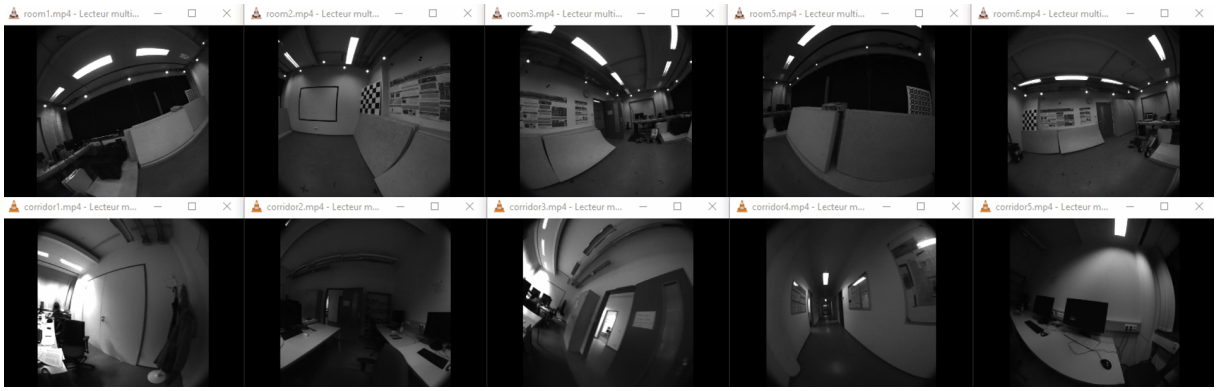
TABLE 4.2 – Caractéristiques des séquences de la base de données TUMVI.

Nom seq.	Image (#)	Longueur
room1	2821	146 m
room2	2882	142 m
room3	2821	135 m
room4	2228	68 m
room5	2847	131 m
room6	2636	67 m
corridor1	5990	305 m
corridor2	6772	322 m
corridor3	5802	300 m
corridor4	1927	114 m
corridor5	5914	270 m

La figure 4.1 illustre une capture d'écran des séquences utilisées pour Euroc 4.1a avec les séquences *Vicons* et *Machine Hall* dans la première et deuxième rangée respectivement et TUMVI 4.1b avec les séquences *room* et *corridor* dans la première et deuxième rangée respectivement. Les algorithmes de localisation temps-réel sont évalués sur les bases de données qui représentent un défi de localisation. Les méthodes SLAM prennent en entrée les données issues des bases de données comprenant les images et les mesures inertielles et donnent en sortie l'estimation de la trajectoire parcourue par la séquence. Les expérimentations visent à évaluer la précision de cette trajectoire par différentes métriques grâce aux outils détaillés dans la section 4.1.2.



(a)



(b)

FIGURE 4.1 – Séquences utilisées des bases de données Euroc (a) et TUMVI (b).

4.1.2. Outils et métriques

Trois principaux outils ont été utilisés pour acquérir les résultats de précision de la localisation, la reconstruction 3D et la consommation mémoire : evo (Grupp, 2017), Open3D (Zhou et al., 2018) et *heaptrack* (Heaptrack) respectivement. Ces outils permettent de calculer les métriques de précision, de robustesse et de consommation pour évaluer les algorithmes SLAM par rapport aux contributions proposées.

L'outil *evo* est utilisé pour trois raisons : 1/ utilisation de l'outil dans l'état de l'art (Rosinol et al., 2020), 2/ possibilité de calculer plusieurs métriques de précision et 3/ plusieurs formats d'export du fichier de la trajectoire estimée (Euroc, rosbag, etc.). C'est une bibliothèque pour évaluer la qualité de la trajectoire estimée des algorithmes SLAM. L'outil évalue la précision de localisation au travers de deux principales métriques : *Absolute Trajectory Error* (ATE) et *Relative Pose Error* (RPE). L'ATE est la métrique de référence pour le SLAM dont le but est de calculer la précision de la trajectoire de manière globale. Elle est majoritairement utilisée dans la littérature et consiste en deux étapes, l'alignement de la trajectoire estimée vers la vérité terrain et le calcul de l'erreur par l'erreur quadratique moyenne (RMSE) (Zhang and Scaramuzza, 2018). Là où l'ATE sert à obtenir une précision globale, la métrique RPE donne l'information sur la précision locale, notamment pour évaluer la dérive de la trajectoire estimée. Les travaux réalisés dans la thèse ont pris en compte la métrique ATE pour

la facilité de calcul et de comparaison des résultats contrairement au RPE qui est plus complexe à calculer et dont la qualité de l'estimation est moins évidente à comparer (Zhang and Scaramuzza, 2018).

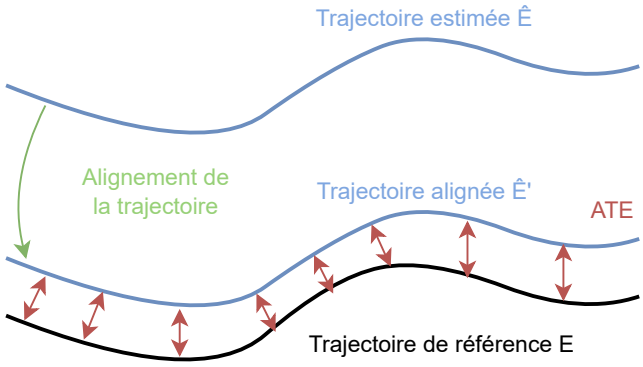


FIGURE 4.2 – Calcul de la mesure de précision ATE à partir de la trajectoire estimée $\hat{E}$ et la trajectoire de référence E (Zhang and Scaramuzza, 2018).

La figure 4.2 montre les deux étapes de la mesure de l'ATE entre la trajectoire estimée et la vérité terrain. La première consiste à aligner les deux trajectoires par le calcul de la transformation S de la trajectoire estimée $\hat{E}$ grâce à la méthode Umeyama (Umeyama, 1991). Cela permet de placer dans le même espace que la trajectoire de référence E. Plusieurs types d'alignement sont possibles. Dans le cas où l'algorithme utilise la caméra monoculaire où le facteur d'échelle n'est pas disponible, l'alignement permet d'effectuer une rotation, translation et une mise à l'échelle de $\hat{E}$ vers E (Sim(3)). Dans notre cas, nous utilisons les images en stéréovision et la centrale inertielle, l'échelle est alors disponible. L'alignement Umeyama réalisée est en SE(3) avec la rotation et la translation. La deuxième étape consiste à quantifier l'erreur de localisation. Pour cela l'erreur quadratique moyenne (RMSE) est utilisée prenant en compte chaque pose estimée et la pose de référence correspondante, comme décrit dans l'équation 4.1.

$$ATE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|E_i - S\hat{E}_i\|^2} \tag{4.1}$$

Un des principaux avantages de l'ATE est qu'elle résume en une seule métrique la précision de la trajectoire estimée ce qui facilite la comparaison entre les méthodes de l'état de l'art. Nous avons remarqué durant les expérimentations que l'ATE peut être faussée lorsque la trajectoire estimée ne couvre pas la même longueur que la trajectoire de référence. Par exemple, la figure 4.3 montre la trajectoire estimée par le SLAM KimeraVIO (Rosinol et al., 2020) sur la séquence *MH03* d'Euroc par l'erreur de translation en mètres. Plus l'erreur est faible et tend vers la couleur bleue, plus la trajectoire est précise. L'ATE RMSE de la trajectoire est de 0.16m. Dans plusieurs cas, lorsque la décimation naïve de la réduction de donnée proposée dans la deuxième contribution du chapitre 3 est appliquée, l'estimation ne couvre pas toute la trajectoire comme c'est illustré dans la figure 4.4. Pourtant, l'ATE RMSE de la trajectoire en question est de 0.02m ce qui est nettement plus précis

que le résultat issu de la trajectoire de la figure 4.3. Cela s'explique par le fait que l'algorithme SLAM n'a retourné que quelques poses à la fin de son exécution qui ont une correspondance par rapport à la vérité terrain et dont l'erreur de localisation est faible. Ce qui n'est pas pris en compte dans le calcul est le pourcentage de la trajectoire reconstruite par rapport à la référence.

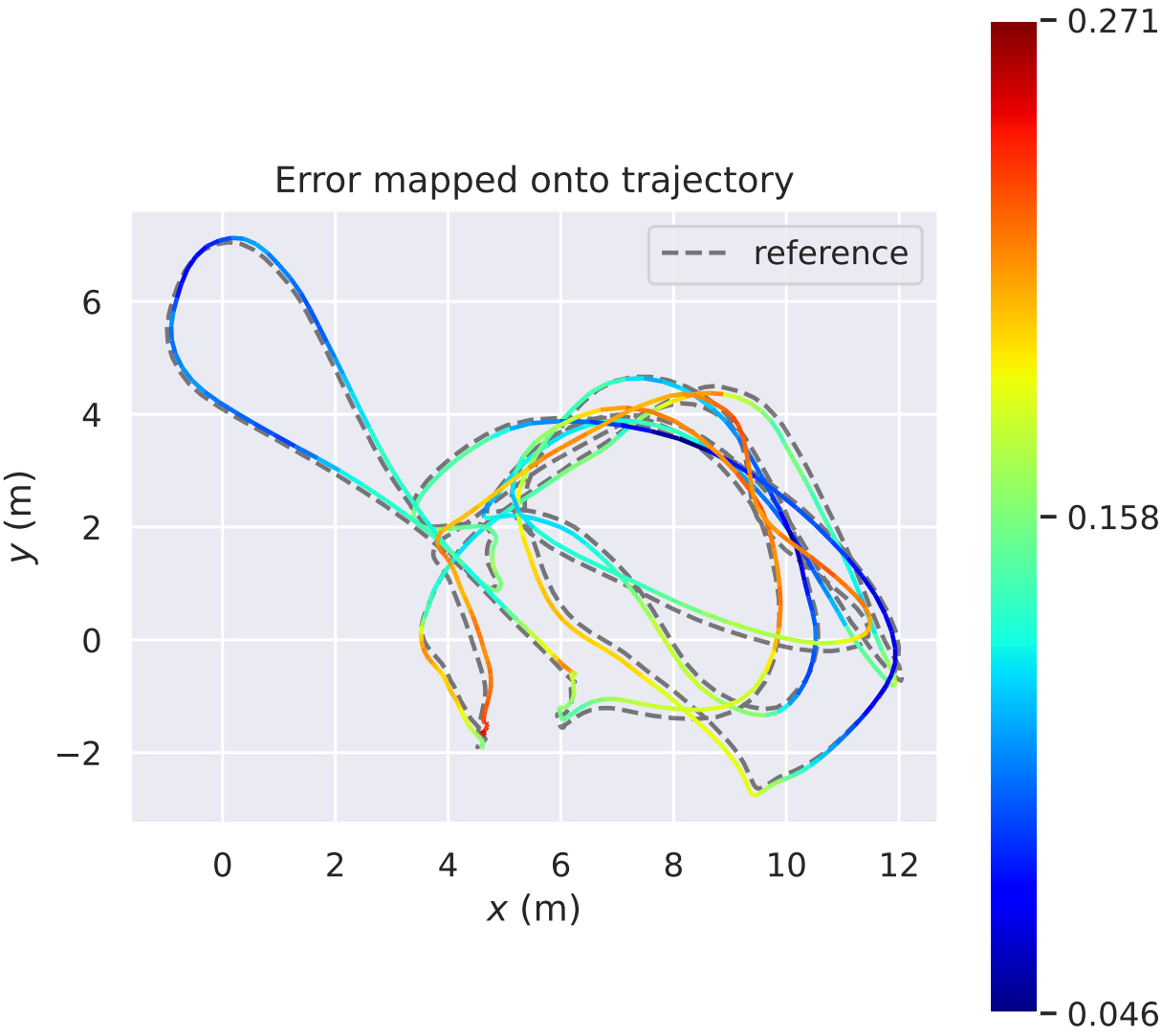


FIGURE 4.3 – Trajectoire estimée de l'algorithme KimeraVIO (Rosinol et al., 2020) sur la séquence MH03 d'Euroc codée en couleur selon l'erreur de translation.

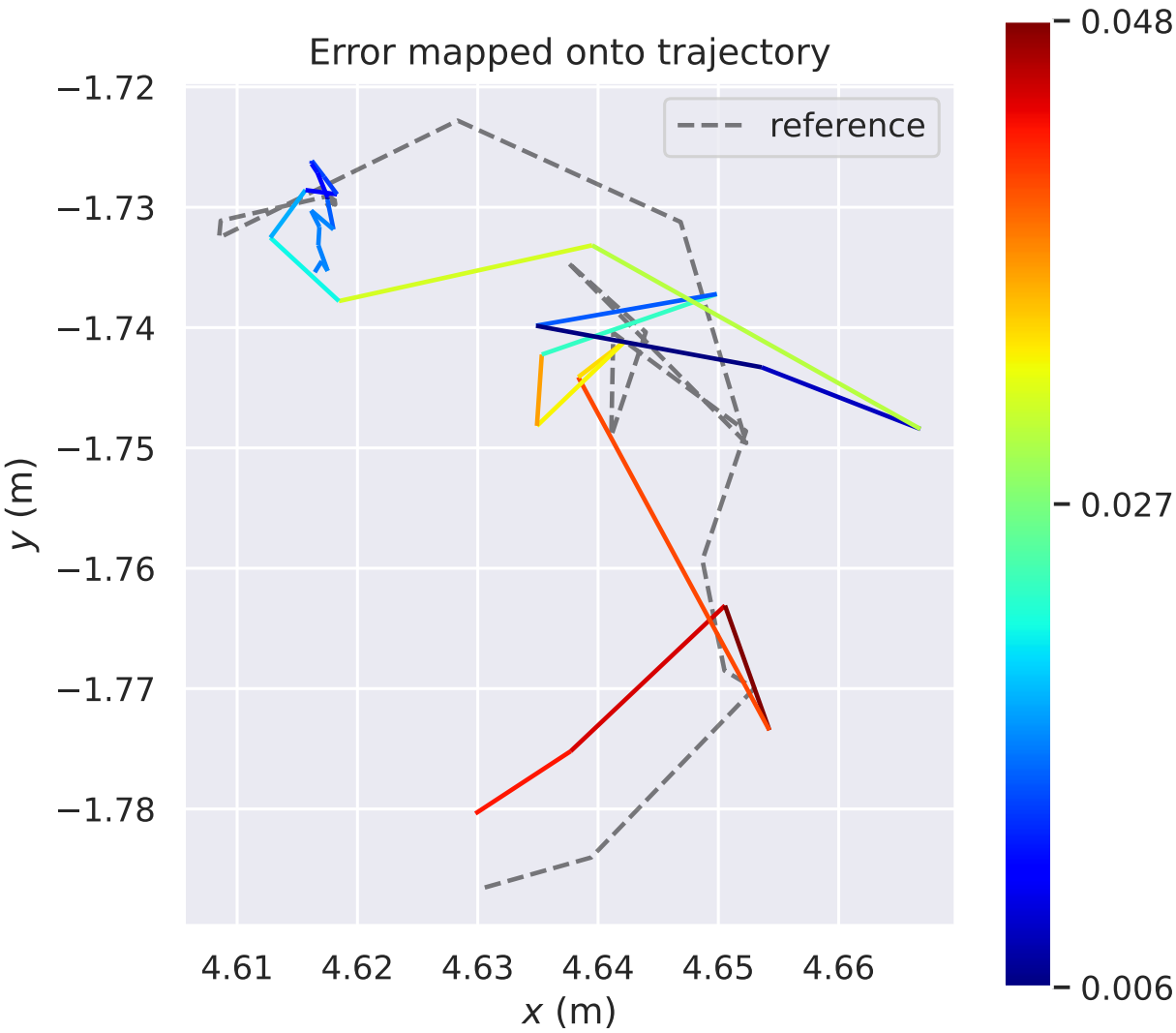


FIGURE 4.4 – Trajectoire estimée de l'algorithme KimeraVIO (Rosinol et al., 2020) sur la séquence MH03 d'Euroc dont la donnée d'entrée est décimée naïvement, codée en couleur selon l'erreur de translation.

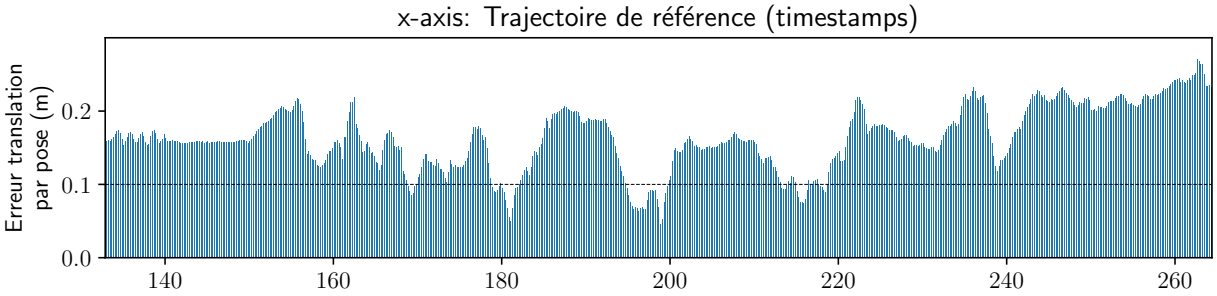


FIGURE 4.5 – Erreur de translation par pose en mètres par rapport à la trajectoire de référence en fonction du temps de la trajectoire estimée par KimeraVIO sur la séquence MH03 d'Euroc.

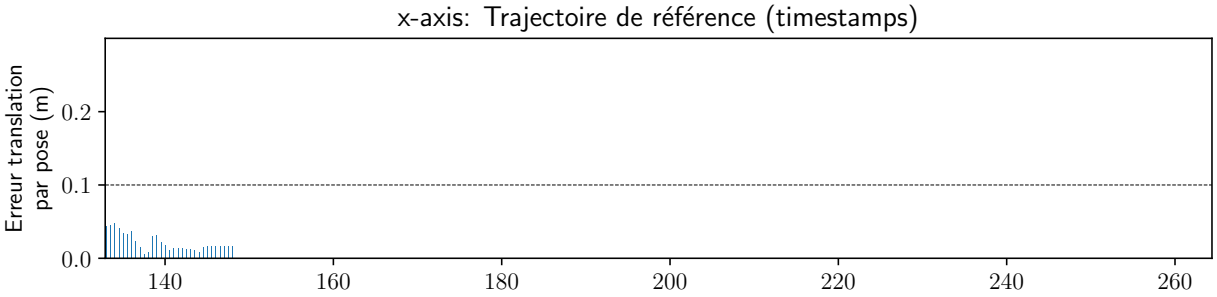


FIGURE 4.6 – Erreur de translation par pose en mètres par rapport à la trajectoire de référence en fonction du temps de la trajectoire estimée par KimeraVIO sur la séquence MH03 d'Euroc à partir de la décimation naïve des données d'entrées.

La métrique de confiance ATE a été mise en place pour vérifier deux points : 1/ l'erreur de translation par pose en fonction de la trajectoire de référence et 2/ le pourcentage de reconstruction de la trajectoire estimée par rapport à la référence. Afin de valider la métrique de confiance, les résultats des deux exemples précédents sont pris en compte dans les figures 4.5 et 4.6 respectivement. La figure 4.5 montre l'erreur de translation par pose en mètres par rapport à la trajectoire de référence en fonction du temps. L'ATE RMSE est bien de 0.16m avec une distance parcourue de 124m ce qui correspond à 100% de la distance parcourue de la trajectoire de référence. La figure 4.6 montre que la trajectoire estimée parcourt seulement 0.2% de la trajectoire de référence avec une précision de 0.02m. Ce pourcentage est pris en compte dans l'analyse des résultats qui sert à définir si la métrique calculée est une valeur aberrante ou non. Dans ce cas, la valeur de précision est retirée ou gardée dans les résultats.

Le deuxième outil utilisé porte sur l'analyse de la reconstruction 3D. Les performances sont évaluées qualitativement et quantitativement. L'analyse qualitative utilise la reconstruction basée voxel et permet d'identifier la précision visuelle de la reconstruction par les trous, les bruits, les gains par rapport à la référence. Afin de mesurer avec une métrique la précision de la reconstruction 3D, le nuage de points généré à partir de la reconstruction voxel (Oleynikova et al., 2017) est utilisé et traité via l'outil Open3D (Zhou et al., 2018).

A partir du nuage de points de référence et celui estimé par la reconstruction 3D, l'algorithme ICP (*Iterative Closest Point*) (Besl and McKay, 1992) est exécuté afin d'aligner les deux représentations

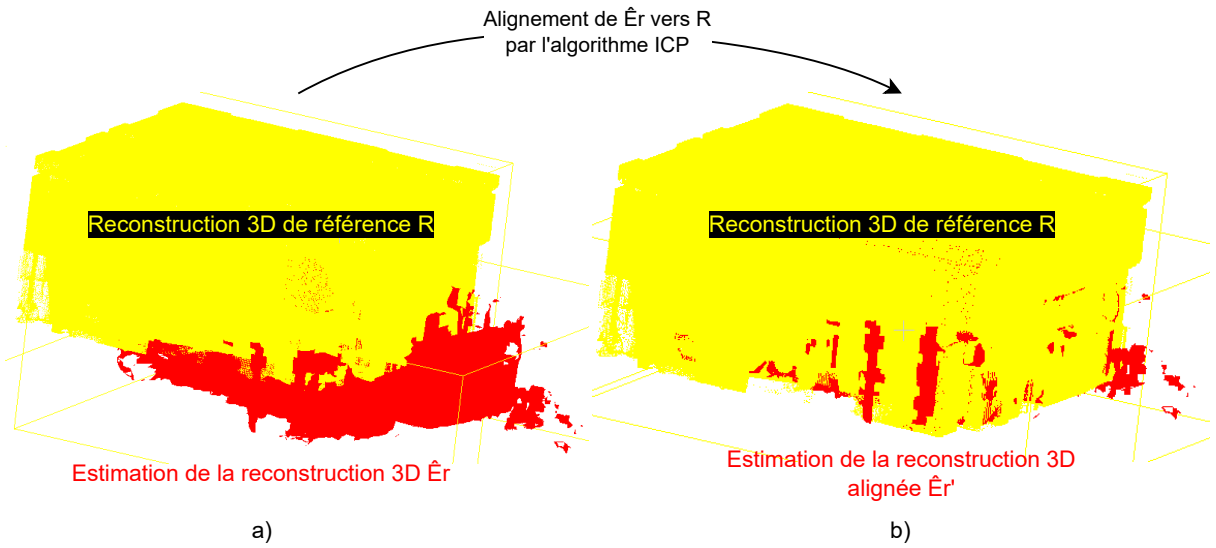


FIGURE 4.7 – Alignement de la reconstruction 3D estimée en rouge vers la reconstruction de référence en jaune avec l'algorithme ICP grâce à l'outil Cloudcompare (Cloudcompare.org). a) représente les deux reconstructions non alignées et b) la reconstruction $\hat{E}_r$ alignée à la référence R.

de la scène dans un même repère. Le calcul de l'alignement utilise la matrice de transformation entre l'estimation de la reconstruction et la référence. Les deux nuages de points respectivement avant et après le recalage ICP sont représentés dans les figures 4.7a et 4.7b avec en jaune la reconstruction et référence et en rouge l'estimation. L'algorithme ICP vise à minimiser la fonction de l'équation 4.2 pour calculer l'alignement approprié.

$$E(\mathbf{T}) = \sum_{(s,n)\in C} \|s - \mathbf{T}e\|^2 \tag{4.2}$$

Pour cela, l'algorithme exécute les deux étapes suivantes :

1. Les points correspondants entre le nuage de points estimé N et celui de référence S sont établis dans $C = (s, e)$. La matrice de transformation $\mathbf{T}$ permet d'aligner les deux représentations 3D.
2. La matrice de transformation $\mathbf{T}$ est mise à jour en minimisant $E(\mathbf{T})$ de l'équation 4.2.

Plusieurs itérations sont effectuées sur ces deux étapes permettant d'affiner l'alignement entre les deux nuages de points. Dans la figure 4.7, 30 itérations ont été réalisées par l'algorithme avec un seuil ICP de 1.0m. A partir de l'alignement des reconstructions, trois métriques sont calculées, *fitness* qui mesure le ratio entre le nombre de correspondances et le nombre de points 3D de l'estimation, la précision RMSE qui est calculée à partir de la distance moyenne entre les points les plus proches dans la reconstruction de référence et la complétude qui mesure à quel point la scène est complètement reconstruite par rapport à la référence (Merrell et al., 2007). La complétude est mesurée grâce à l'outil CloudCompare (Cloudcompare.org) en suivant la méthodologie suivante :

1. Génération de 10^3 points/m^2 de la reconstruction 3D estimée
2. Alignement ICP avec la reconstruction de référence

3. La reconstruction est convertie en une grille de cellules de taille $N \times N$
4. La complétude est mesurée et correspond au pourcentage de cellules où au moins un point 3D y est présent

La précision RMSE en complément de la complétude permet d'avoir un indice de confiance sur la qualité de la reconstruction 3D. En effet, la métrique de précision va prendre en compte seulement les points où il y a la correspondance entre l'estimation et la référence. Ainsi, la métrique RMSE peut donner une bonne précision sans pour autant avoir d'information sur la quantité de points reconstruits par rapport à la référence. La complétude apporte cette information en pourcentage.

TABLE 4.3 – Latence et pic de la mémoire tampon d'images de l'algorithme VINSFusion avec et sans l'outil *heaptrack* pour l'analyse de la consommation mémoire. La latence correspond à la mesure du temps entre l'image injectée et la pose estimée.

	latence (ms)			pic buffer image (#)
	front-end	back-end	total	
sans *heaptrack*	28.10	32.59	60.59	4
avec *heaptrack*	84.07	493.18	577.25	1300

Le dernier outil utilisé est *heaptrack* (Heaptrack) pour l'analyse de la consommation mémoire en Mo des éléments alloués dynamiquement dans la RAM, le tas. Cela représente la majeure partie de la mémoire allouée dans le SLAM dû au traitement temps-réel *multi-threading*. Nous avons remarqué durant les expérimentations que l'outil ajoute une surcouche non négligeable à l'algorithme analysé. La table 4.3 montre l'exécution du SLAM VINSFusion (Qin et al., 2019) avec et sans l'utilisation de *heaptrack*. Avec un flux des images d'entrées à 20 ips et une latence totale de 60.59ms sans l'utilisation de *heaptrack*, l'algorithme tourne en temps-réel avec un pic d'utilisation de la mémoire tampon de 4 images. L'analyse de la consommation mémoire avec l'outil augmente significativement la latence de bout en bout du SLAM à 577.25ms avec un pic d'utilisation mémoire tampon de 1300 images. Lorsqu'une mémoire tampon d'images utilisée en FIFO (*First In, First Out*) est implémentée, l'analyse faite par l'outil est faussée. Dans le contexte embarqué, le fait que *heaptrack* augmente le temps d'exécution de l'algorithme permet d'avoir une émulation des performances du SLAM avec du matériel à ressources restreintes. C'est dans ce cas de figure que les expérimentations se placent pour représenter l'exécution des algorithmes sur du matériel embarqué où les ressources sont limitées.

4.2. Algorithmes SLAM utilisés

Le choix des méthodes SLAM à utiliser dans la thèse pose une problématique en raison du large spectre de techniques analysées dans l'état de l'art. Les approches directes représentent un avantage en termes de robustesse, mais la complexité calculatoire augmente significativement en raison de la prise en compte de tous les pixels de l'image dans la chaine de traitement. Cette technique ne permet pas d'assurer un traitement temps réel sur plateforme embarquée où les ressources de calculs sont limitées. C'est pour cette raison que nous nous concentrons sur les approches indirectes qui exploitent les points d'intérêts de l'image pour atteindre un compromis raisonnable entre l'exécution temps-réel et la précision du système. Les méthodes Kimera (Rosinol et al., 2020), ORBSLAM3

(Campos et al., 2021), VINSFusion (Qin et al., 2019) et OpenVINS (Geneva et al., 2020) sont les quatre algorithmes retenus pour l'étude de la précision et la robustesse de la localisation temps-réel. Ces méthodes couvrent le spectre des techniques utilisées dans l'état de l'art et constituent des implémentations de référence, que ce soit dans le *front-end* (FE) et le *back-end* (BE). La table 4.4 détaille les différentes techniques utilisées qui ont été prises en compte comme critères de sélection. Ces méthodes sont sélectionnées pour leur précision de localisation, leur complexité de calcul et les techniques employées du FE à l'optimisation locale et/ou globale des positions 3D (Engels et al., 2006). Le FE traite les images d'entrée et permet d'obtenir une première estimation du mouvement. Les méthodes selectionnées sont basées sur le suivi de points d'intérêts (*feature tracking* (FT) (yves Bouguet, 2000)) et sur la détection et la description des points d'intérêts (*feature detection and description* (FD) (Rublee et al., 2011)). Afin de fusionner les données visuelles et inertielles et optimiser les états générés par le *front-end*, deux approches ont été identifiées. L'approche basée filtrage et le *factor graph* avec un *back-end* (BE) séparé (Gui et al., 2015). Les quatre méthodes SLAM choisies ont déjà été portées sur des plateformes matérielles embarquées.

TABLE 4.4 – Critères de sélection et performance des algorithmes SLAM identifiés sur les bases de données Euroc * et TUMVI †.

Méthodes	Critères de sélection			Perf. (flux 20 ips)	
	FE	BE	Implementation HW	précision (m)	conso. (Mo)
KimeraVIO	FT	factor graph	Jetson TX2 (20 fps) (Rosinol et al., 2021)	*[0.05-3.49]	*1700
ORBSLAM3	FD	factor graph	RaspberryPi 3B+ (6.11 fps) (Aldegheri et al., 2019)	*[0.01-0.10]	*724.25
			Jetson Nano (9.64 fps) (Silveira et al., 2020)	†[0.01-0.05]	†829.19
VINSFusion	FT	factor graph	UP Board,	*[0.12-0.54]	*1330
			ODROID XU4 (7 fps) (Delmerico and Scaramuzza, 2018)	†[0.06-0.87]	†989.55
OpenVINS	FT	filtering	Jetson TX2 (35.3 fps),	*[0.05-0.22]	*45.18
			Jetson Nano (27.5 fps) (Merrill et al., 2021)	†[0.06-0.83]	

KimeraVIO (Rosinol et al., 2020) est le module VIO de l'architecture Kimera basé sur l'approche de pré-intégration des mesures inertielles (Forster et al., 2017a). La méthode inclut le suivi de points d'intérêts (FT) et est développée avec un BE séparé basé sur le *factor graph*. Avec le module d'optimisation du graphe responsable de la détection et de la fermeture de boucle, la méthode a des capacités SLAM. La plateforme Jetson TX2 a été utilisée dans plusieurs travaux pour évaluer les performances de KimeraVIO sur le GPU embarqué (Rosinol et al., 2021; Jeon et al., 2021). L'algorithme atteint une performance d'estimation de pose allant jusqu'à 20 ips. Les performances obtenues sur Euroc montrent que la méthode a une précision allant de 0.05m à 3.49m. KimeraVIO atteint un pic de consommation mémoire mesuré à 1700 Mo en utilisant l'outil *heaptrack*. L'algorithme est le plus complexe des quatre choisis d'un point de vue calculatoire et compréhension algorithmique.

ORBSLAM3 (Campos et al., 2021) est un système SLAM visuel-inertiel multi-carte qui est basé sur la détection et la description de points d'intérêts (FD). L'approche repose sur l'estimation maximum a posteriori (MAP) et utilise comme BE une optimisation par *factor graph*. La version 2 de la méthode SLAM (Mur-Artal and Tardós, 2017) a été implémentée sur RaspberryPi 3B+ et sur les plateformes embarquées NVIDIA Jetson (Silveira et al., 2020; Aldegheri et al., 2019). Elle atteint une

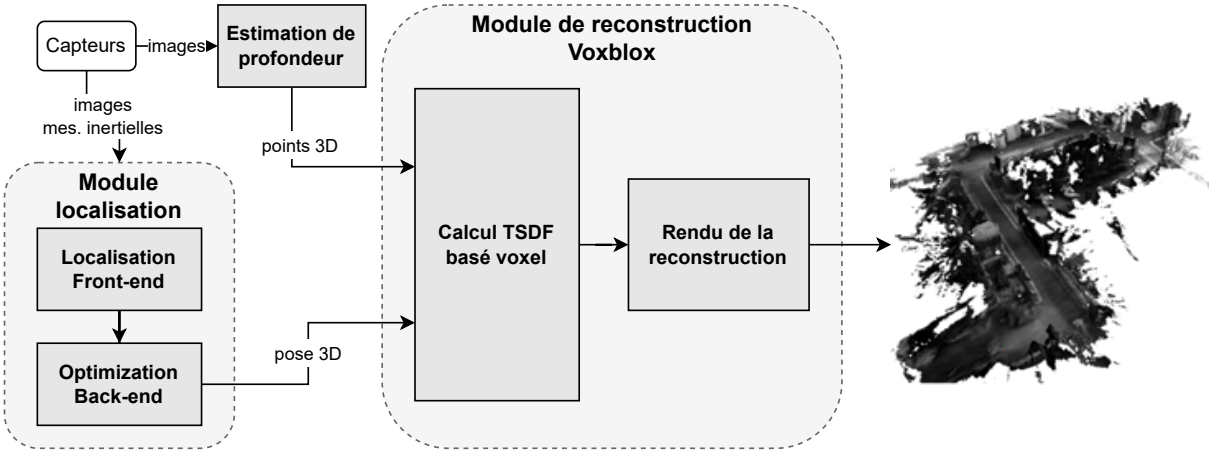


FIGURE 4.8 – Méthode de reconstruction 3D basée voxel, Voxblox, prenant en entrée les poses 3D du module de localisation et les points 3D dense de l'estimation de profondeur de l'image.

performance moyenne de respectivement 6.11 ips et 9.64 ips sur Pi 3B+ et Jetson Nano. En termes de précision, ORBSLAM3 permet d'avoir une erreur de localisation maximum de respectivement 0.10m et 0.05m pour Euroc et TUMVI. La consommation mémoire mesurée montre que la méthode a un pic de consommation de 724.25 Mo et 829.19 Mo respectivement. Les performances sur plateformes embarquées montrent que ORBSLAM3 ne tourne pas en temps-réel en raison du coût conséquent des calculs d'optimisations intégrés dans la chaîne de traitement.

La méthode VINSFusion (Qin et al., 2019) est basée sur VINSMono (Qin et al., 2018) et utilise l'algorithme FT avec une optimisation *factor graph* pour générer les poses du système de navigation visuel-inertiel. L'évaluation des performances (Delmerico and Scaramuzza, 2018) montre que la méthode s'exécute à 7 ips sur les plateformes embarquées UpBoard et ODROID XU4. VINSFusion a une plage de précision allant de 0.12m à 0.54m pour Euroc et 0.06 m à 0.87 m pour TUMVI. Le pic de consommation mémoire mesuré avec l'outil *heaptrack* atteint respectivement 1330 Mo et 989.55 Mo. VINSFusion est un très bon exemple de méthodes qui peuvent tourner sur plateforme embarquée en temps-réel. La complexité algorithmique est faible et la mémoire tampon est implémentée pour éviter que des images soient perdues lors de l'exécution. Dans un contexte embarqué avec des ressources restreintes, la principale problématique est de se passer ou de minimiser l'espace mémoire utilisé. VINSFusion atteint des performances de précision qui sont comparables à l'état de l'art.

OpenVINS (Geneva et al., 2020) est également basé sur FT pour le premier étage de la chaîne de traitement et diffère des autres algorithmes en utilisant le filtre de Kalman *multi-state constraint* (MSCKF) (Mourikis and Roumeliotis, 2007) pour l'estimation des poses. L'approche OpenVINS a été utilisée avec les plateformes NVIDIA Jetson et atteint en moyenne une performance temps-réel de 35.3 ips et 27.5 ips sur les GPU embarqués TX2 et Nano respectivement (Merrill et al., 2021). OpenVINS est similaire à VINSFusion en termes de précision avec une erreur de localisation maximale de 0.22m et 0.83m pour les deux bases de données utilisées. Seule la consommation mémoire sur Euroc a pu être mesurée avec un pic de 45.18 MB.

L'étude de la précision de localisation est effectuée à partir des quatre algorithmes SLAM décrits dans les paragraphes précédents. L'algorithme de reconstruction 3D utilisé est Voxblox (Oleynikova

et al., 2017). La méthode permet d'obtenir une représentation volumétrique de la scène en prenant en compte l'information dense de la profondeur de l'image en entrée et les poses 3D générées par les méthodes de localisation comme c'est illustré dans la figure 4.8. La méthode Voxblox a été

TABLE 4.5 – Performance de l'algorithme Voxblox avec les deux approches d'intégration TSDF, la fusion et la rapide (Oleynikova et al., 2017).

Taille voxel (m)	Integration	TSDF (ms)	Conso. RAM (MB)
0.20	Fusion	56	**49**
	Rapide	(/2.80) **20**	(×1.26) 62
0.05	Fusion	112	**144**
	Rapide	(/4.87) **23**	(×1.06) 153
0.02	Fusion	527	**609**
	Rapide	(/8.36) **63**	(×1.11) 675

identifiée et sélectionnée grâce au schéma de *hashing* qui a été utilisé dans plusieurs recherches pour minimiser la consommation mémoire et la charge CPU de la reconstruction 3D avec une complexité $O(1)$ (Oleynikova et al., 2017; Klingensmith et al., 2015; Muglikar et al., 2020). La méthode propose deux approches d'intégration TSDF, la fusion et la rapide. La table 4.5 montre que la méthode rapide permet d'avoir une application qui tourne jusqu'à 8 fois plus rapidement que la méthode de fusion pour une taille de voxel de 0.02m lors du calcul d'intégration TSDF tout en ayant une consommation RAM similaire. La méthode rapide est prise en compte avec une taille de voxel de 0.10m et une longueur de rayon de 5m qui correspond à la portée maximale de la projection du rayon pour chaque pixel lors du rendu de la reconstruction 3D. Le chapitre 2 a mentionné que l'approche Voxblox est l'une des rares méthodes à avoir été évaluée sur système embarqué Asctec Firefly Intel i7 2.1 GHz CPU avec une performance d'environ 4 ips (Oleynikova et al., 2020).

4.3. Methodologie employée

Afin d'évaluer la robustesse et la précision de la localisation des algorithmes SLAM identifiés et sélectionnés dans la section 4.2, une chaîne de traitement a été mise en place à partir des contributions proposées et est illustrée dans les figures 4.9 et 4.10. Pour chacune des contributions l'approche naïve est utilisée à des fins de référence. Pour la réduction de la dynamique de l'image dans la figure 4.9, les images alimentent les méthodes de réduction de la dynamique, la naïve, la *median cut* et la *block-wise*. Dans la figure 4.10, l'approche naïve correspond au filtrage constant (CF) qui est utilisée pour comparaison avec l'approche de filtrage adaptatif proposée. Les images sont injectées dans le module de réduction de donnée avec le flux de base 20 ips. Le filtrage constant décime les données d'entrée à un flux constant sans prendre en compte le mouvement du capteur. Pour notre comparaison, nous avons réduit le flux de référence 20 ips à 10, 7, 5 et 2 ips pour garder seulement 1 image sur 2, 3, 4 et 10 respectivement.

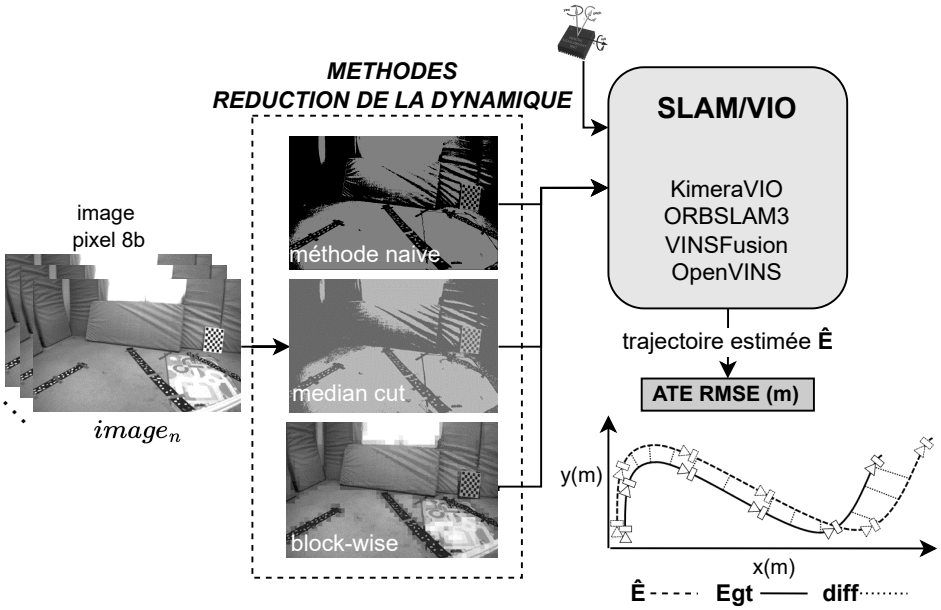


FIGURE 4.9 – Méthodologie employée pour évaluer les performances des algorithmes SLAM à partir du mécanisme de réduction de la dynamique de l'image proposé.

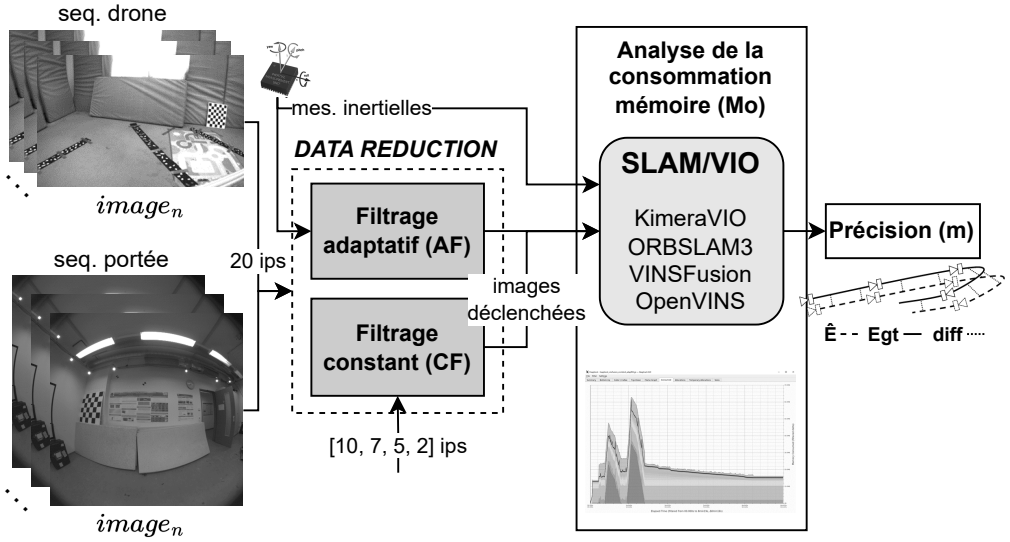


FIGURE 4.10 – Méthodologie employée pour évaluer les performances des algorithmes SLAM à partir du mécanisme de filtrage adaptatif proposé pour le contrôle du flux d'images d'entrée.

TABLE 4.6 – Moyenne du flux d'images injectées sur toutes les séquences Euroc et TUMVI grâce au contrôle du déclenchement d'images par la méthode de filtrage constant (CF) et adaptative (AF) avec les heuristiques *const.* et *adapt.*.

seq.	référence 20 ips	CF 50% 10 ips	CF 33% 7 ips	CF 25% 5 ips	CF 10% 2 ips	AF const. thresh.	AF adapt. thresh.
V101	20	10	7	5	2	4.0	3.0
V102	20	10	7	5	2	6.0	3.4
V103	20	10	7	5	2	7.2	4.4
V201	20	10	7	5	2	4.0	3.0
V202	20	10	7	5	2	6.4	3.8
MH01	20	10	7	5	2	3.8	3.2
MH02	20	10	7	5	2	4.0	3.4
MH03	20	10	7	5	2	4.2	3.2
MH04	20	10	7	5	2	4.0	3.2
MH05	20	10	7	5	2	4.0	3.4
room1	20	10	7	5	2	13.8	11.8
room2	20	10	7	5	2	10.2	6.8
room3	20	10	7	5	2	10.0	7.0
room5	20	10	7	5	2	14.0	12.4
room6	20	10	7	5	2	8.4	5.0
corridor1	20	10	7	5	2	10.2	7.8
corridor2	20	10	7	5	2	9.6	6.8
corridor3	20	10	7	5	2	12.8	10.0
corridor4	20	10	7	5	2	8.4	6.2
corridor5	20	10	7	5	2	9.4	6.8

Les méthodes SLAM sont utilisées en configuration stéréovision et centrale inertielle pour toutes les séquences à l'exception de VINSFusion pour la base de donnée TUMVI. La configuration monoculaire et centrale inertielle est utilisée en raison des paramètres de calibrations manquants. Les paramètres des méthodes SLAM ont été modifiés de la manière suivante : les configurations par défauts ont été utilisées pour ORBSLAM3, VINSFusion et OpenVINS. Afin de comparer KimeraVIO avec ces algorithmes, trois paramètres ont été modifiés. Le *autoInitialize* est mis à *true* pour initialiser la chaîne de traitement à partir des mesures inertielles au lieu de la vérité terrain fournit par la base de donnée. Les paramètres *deterministic_random_number_generator* et *ransac_randomize* à *false* et *true* respectivement afin de ne pas pénaliser l'algorithme lorsque plusieurs exécutions sur la même séquence sont effectuées. La robustesse des algorithmes est évaluée en fonction de ces exécutions avec en entrée les images quantifiées issues des jeux de données. Ces derniers sont détaillés dans la section 4.1.1. Chaque séquence des deux bases de données Euroc et TUMVI ont été pré-traitées avec les contributions de la thèse, soit les méthodes de la réduction de la dynamique et avec le contrôle du flux des données par le filtrage adaptatif. Les bases de données pré-traitées sont stockées à des fins d'expérimentations pour évaluer le fonctionnement des mécanismes développés pour les algorithmes SLAM. Chacune des séquences est alors dupliquée N fois suivant les pas de quantification et les seuils du contrôle de flux correspondant respectivement à la réduction de la dynamique et au filtrage adaptatif. Ces séquences sont ensuite injectées en entrée des méthodes SLAM. Concernant l'approche

de réduction de la dynamique de l'image, les résultats d'erreur de localisation sont obtenus avec l'exécution de 50 itérations des méthodes non déterministes pour les trois approches de quantification par séquence, qui donne un total d'exécution de 118800 itérations. En effet, nous avons 50 exécutions pour les 18 méthodes de quantification (8 *median cut*, 3 *block-wise* et 7 *naïve*), 22 séquences (Euroc et TUMVI) et 4 méthodes SLAM (KimeraVIO, ORBSLAM3, VINSFusion, OpenVINS) agrémentées de 2 variations dont le paramètre de fermeture de boucle est activé (KimeraVIO-lc, VINSFusion-lc). Ces itérations servent à obtenir l'évaluation de la robustesse de chaque algorithme SLAM avec la réduction de la dynamique de l'image en entrée. Concernant l'approche du filtrage adaptatif, la médiane de 5 itérations est prise en compte dans les résultats. Le prétraitement des séquences avec cette approche montre que le nombre total d'images injectées dans les méthodes SLAM est considérablement réduit pour les approches de réductions de données CF et AF détaillé dans la table 4.6. Les colonnes du filtrage adaptatif AF représentent la moyenne du flux d'images injectées sur toute la séquence. Cela sert de comparaison par rapport à la méthode naïve CF, mais n'est pas représentatif du nombre d'images que l'on obtient lors de l'exécution. En effet, lorsqu'il n'y a pas de mouvements, aucune image n'est déclenchée et lorsqu'il y a du mouvement, le nombre d'images déclenchées est lié aux données de la centrale inertielle qui mesure à quel point le mouvement est rapide, comme c'est décrit dans le chapitre 3.

L'évaluation de la trajectoire par rapport à la vérité terrain est basée sur la métrique de l'ATE RMSE détaillée dans la section 4.1.2. Les résultats bruts sont filtrés de deux manières consécutives : 1/ les résultats qui ne respectent pas la métrique établie par la mesure de confiance ATE sont considérés comme des valeurs aberrantes et ne sont pas pris en compte. 2/ La distribution des résultats sur 50 itérations est prise en compte par un ajustement gaussien des résultats. La figure 4.11 montre que la probabilité d'avoir le résultat autour de 1250m est inférieur à 0.05%. Cette mesure est alors considérée comme aberrante et est filtrée des résultats finaux. La figure 4.12 montre la distribution gaussienne des résultats que l'on a sur un jeu de 50 itérations. Dans les deux cas de figures, les métriques qui sont au-delà de ± 6 fois la déviation standard sont enlevées. Cette méthodologie permet d'obtenir les résultats comparables qui montrent la robustesse des algorithmes. Le chapitre 5 détaille, en plus des résultats filtrés, le taux d'erreur et de valeurs aberrantes qui ont été détectées pour chaque algorithme SLAM et chaque jeu de données.

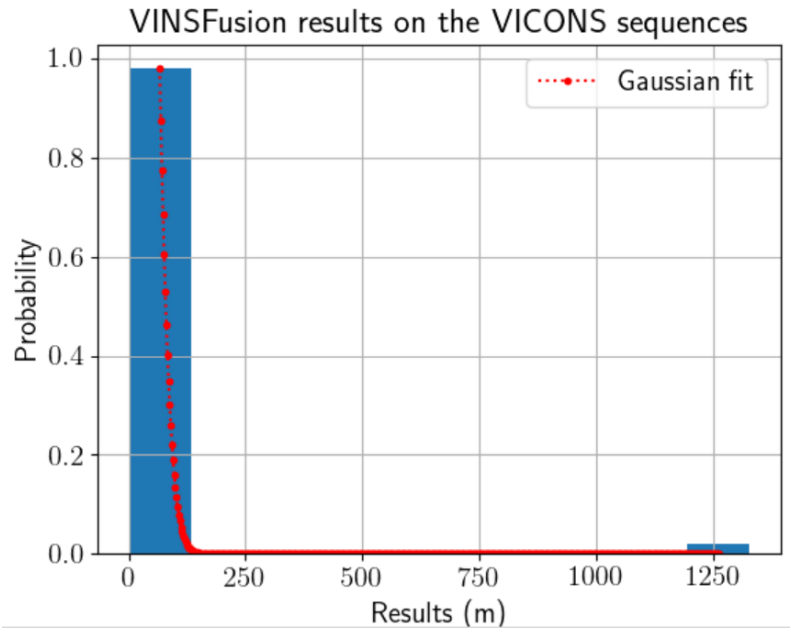


FIGURE 4.11 – Filtrage des résultats aberrants basé sur la déviation standard de la distribution gaussienne. Exemple des résultats de précision obtenus avec l'algorithme VINSFusion.

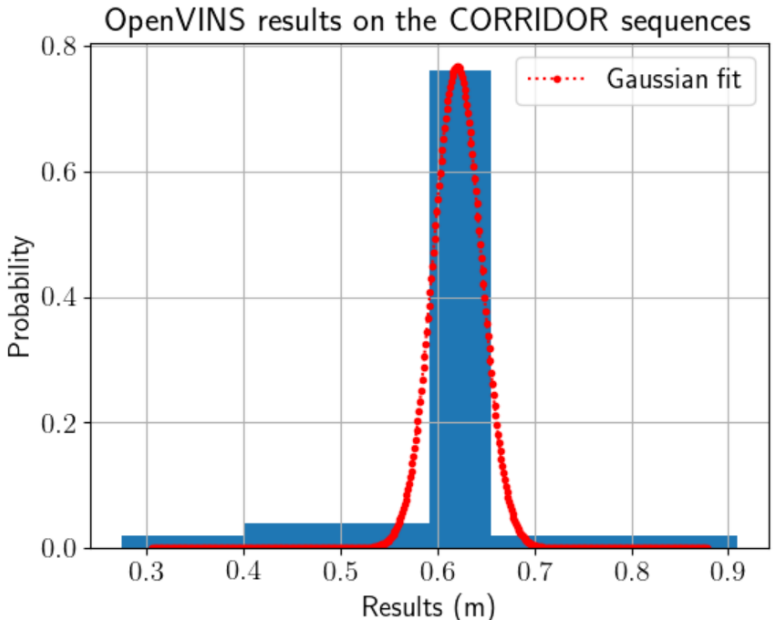


FIGURE 4.12 – Filtrage des résultats aberrants basé sur la déviation standard de la distribution gaussienne. Exemple des résultats de précision obtenus avec l'algorithme OpenVINS.

L'impact des contributions sur la réduction de la donnée est également évalué avec la consommation mémoire des méthodes SLAM. Pour cela, l'analyse est faite en émulant les performances des méthodes SLAM sur système embarqué par deux approches : 1/ la réduction du temps d'exécution grâce à l'outil *heaptrack* détaillé dans la section 4.1.2 et par l'ajout de pause dans l'algorithme permettant d'obtenir une exécution de 100ms à 200ms par pas de 20ms grâce à la librairie standard C++. 2/ Par l'augmentation du flux d'entrée. Par exemple, avec un flux d'entrée de 20 ips, nous émulons les performances des méthodes SLAM confronté à un flux d'entrée de 2 à 5 fois plus rapide. Ces approches ont été mises en place, car l'outil *heaptrack* ne permet pas de contrôler le temps d'exécution de l'algorithme, ce qui rend l'analyse moins pertinente. Ces deux approches émulent les performances des algorithmes SLAM sur du matériel à ressources restreintes, l'impact des contributions sur la consommation mémoire est mesuré sur un large spectre de systèmes embarqués, allant du moins restrictif au plus restrictif.

Le chapitre 5 détaille et discute des expérimentations réalisées allant de la localisation à la reconstruction 3D. La reconstruction 3D est évaluée sur le système complet avec les contributions indépendantes l'une de l'autre et enfin avec l'association des deux contributions, la méthode de quantification associée avec la méthode de filtrage adaptatif. L'évaluation est faite de manière quantitative avec les métriques détaillées dans la section 4.1.2 et qualitative, soit la qualité de la reconstruction 3D sur la séquence *V101* de la base de donnée Euroc. Les expérimentations menées sur la reconstruction 3D ont permis de mettre en place la chaine de traitement complète allant du capteur à la reconstruction voxel. L'implémentation de ce système basé sur le logiciel ROS met en avant des problèmes de synchronisation entre les différents algorithmes utilisés. L'exécution de l'estimation de profondeur est plus rapide que l'estimation de la pose issue des méthodes SLAM. Cela ne permet pas d'avoir une reconstruction 3D complète de la scène en raison des pertes de données lors de l'exécution. Les jeux de données qui ont été pré-traités avec les contributions proposées sont exécutés avec un flux d'entrée divisé par 2 pour résoudre efficacement le problème de synchronisation.

5 - Etude et validation des performances avec les mécanismes d'optimisation

Sommaire

Les contributions de la thèse du chapitre 3 décrivent les mécanismes d'optimisation des données dans le but de réduire le nombre d'opérations effectuées dans la chaîne de traitement qui est présentée dans le chapitre 2. Ce chapitre 5 présente les résultats des performances du SLAM avec ces contributions. Dans les sections 5.1 et 5.2, la précision, la robustesse et l'impact mémoire du module de localisation des méthodes SLAM sont détaillées. La section 5.3 présente et discute les résultats apportés par les méthodes proposées pour le système complet allant vers la reconstruction 3D d'un point de vue qualitatif et quantitatif. L'analyse des gains apportés par les mécanismes d'optimisation et les efforts restants à fournir sur l'implémentation matérielle des contributions sont discutés dans la section 5.4.

Les expérimentations menées sont effectuées à partir des deux bases de données Euroc (Burri et al., 2016) et TUMVI (Schubert et al., 2018) pour représenter les cas d'usages portant sur les drones et sur les capteurs portés, par exemple le casque à réalité augmentée/virtuelle. C'est à partir de ces bases de données que 11 séquences ont été utilisées pour chaque cas d'usage : *Vicons* et *Machine Hall* pour Euroc et *Room* et *Corridor* pour TUMVI. Ces séquences contiennent des scénarios plus ou moins difficile pour les algorithmes de localisation temps-réel. Par exemple, des mouvements rapides et brusques, de longues trajectoires, des changements de luminosités. L'analyse des résultats se concentre davantage sur la précision et la robustesse des algorithmes sur les séquences les plus difficiles.

TABLE 5.1 – Taux d'échec moyen en pourcentage (%) des algorithmes SLAM pour chacunes des séquences utilisées sur 50 itérations.

	ORBSLAM3	KimeraVIO	VINSFusion	OpenVINS
Vicons	3.09	0.42	2.35	0.59
Machine Hall	19.52	0.36	2.78	0.61
Room	1.22	-	0.77	1.28
Corridor	1.02	-	0.96	1.71

5.1. Réduction de la dynamique de l'image

5.1.1. Précision et robustesse des algorithmes SLAM

Un des principaux objectifs est d'étudier la robustesse des algorithmes avec la quantification sur différentes séquences. C'est pour cette raison que l'on détaille dans cette section les performances des SLAM évoluant dans des environnements différents avec des mouvements qui varient suivant le type de système d'acquisition. Le comportement des méthodes de localisation est détaillé pour chaque configuration et une discussion globale sur les choix de quantification est donnée à la fin de la section.

Afin de définir la méthode de quantification, l'erreur de localisation pour chaque algorithme SLAM/VIO est affiché comme étant une fonction de la méthode de réduction de donnée. Il est indiqué dans le chapitre 4 que l'on utilise deux bases de données, Euroc et TUMVI. Elles sont représentées par les séquences *Vicons (V)*, *Machine Hall (MH)* pour Euroc et *Room*, *Corridor* pour TUMVI. Chacune des figures ci-dessous montrent la variabilité de l'erreur de localisation ce qui représente la robustesse de chaque algorithme à la quantification de l'image d'entrée à partir de la méthodologie établie dans le chapitre 4. Les résultats sont filtrés afin d'enlever les valeurs aberrantes dont le taux d'échec en pourcentage est illustré dans la table 5.1 pour les méthodes SLAM en fonction de chaque séquence.

Dans les expérimentations menées, trois configurations sont représentées pour chaque séquence par trois figures avec la référence notée 8bpp pour 8-bits/pixels (px) : la première est la méthode naïve qui est dénotée 7bpp à 1bpp. La deuxième correspond aux algorithmes SLAM sans la fermeture de boucle avec la méthode de quantification *median cut* et *block-wise* qui sont dénotées respectivement *med7* à *med1* pour une représentation 7-bits/px à 1-bit/px et *blockN* avec N, la taille du bloc de pixels. La dernière correspond aux algorithmes SLAM avec la fermeture de boucle suivant les méthodes de quantification *median cut* et *block-wise*. Les méthodes de quantification sont définies comme étant la plage de réduction de la dynamique et sont triées dans l'ordre croissant basé sur le pourcentage de réduction de donnée. Pour chaque figure, les boîtes à moustache correspondent aux résultats des séquences des bases de référence. Ainsi, la figure 5.1 montre pour chaque algorithme SLAM les résultats d'erreur de localisation en mètres pour chaque séquence *Vicons* avec les quantifications naïves de 7bpp à 1bpp. Les séquences faciles correspondent à *V101, V201*, moyennes à *V102, V202* et difficiles à *V103, V203*. Les résultats illustrent que la séquence *V203* est difficile pour KimeraVIO (5.1a) de 7bpp à 3bpp. Cela est dû à une détection de points d'intérêts aberrants plus importants. La variation d'erreur de localisation diminue fortement pour 2bpp et 1bpp où les points détectés sont plus fiables. Les résultats avec la quantification naïve permet d'établir la première conclusion

de cette étude. La quantification de l'image n'impacte pas la précision de localisation jusqu'à la représentation à 3b par pixel. Pour les séquences de la base TUMVI, l'impact est plus important pour l'algorithme VINSFusion dans la figure 5.2b où l'erreur de localisation augmente significativement à partir de 5b par pixel. Cela est notamment dû au fait que VINSFusion sur la base de données TUMVI est exécutée avec la configuration monoculaire et inertielle contrairement à la stéréovision utilisée avec les données des séquences Euroc. La quantification naïve dans la figure 5.2a montre que la référence d'ORBSLAM3 obtient une erreur de localisation inférieure à 0.1m pour toutes les séquences jusqu'à 2bpp contrairement aux méthodes sans fermeture de boucle dans 5.2b et 5.2c. Cela est dû à l'optimisation globale présente dans ORBSLAM3 lorsqu'une fermeture de boucle est détectée. Bien que cette fonction obtienne une localisation précise, elle est coûteuse en mémoire et en calcul comme c'est expliqué dans le chapitre 2.

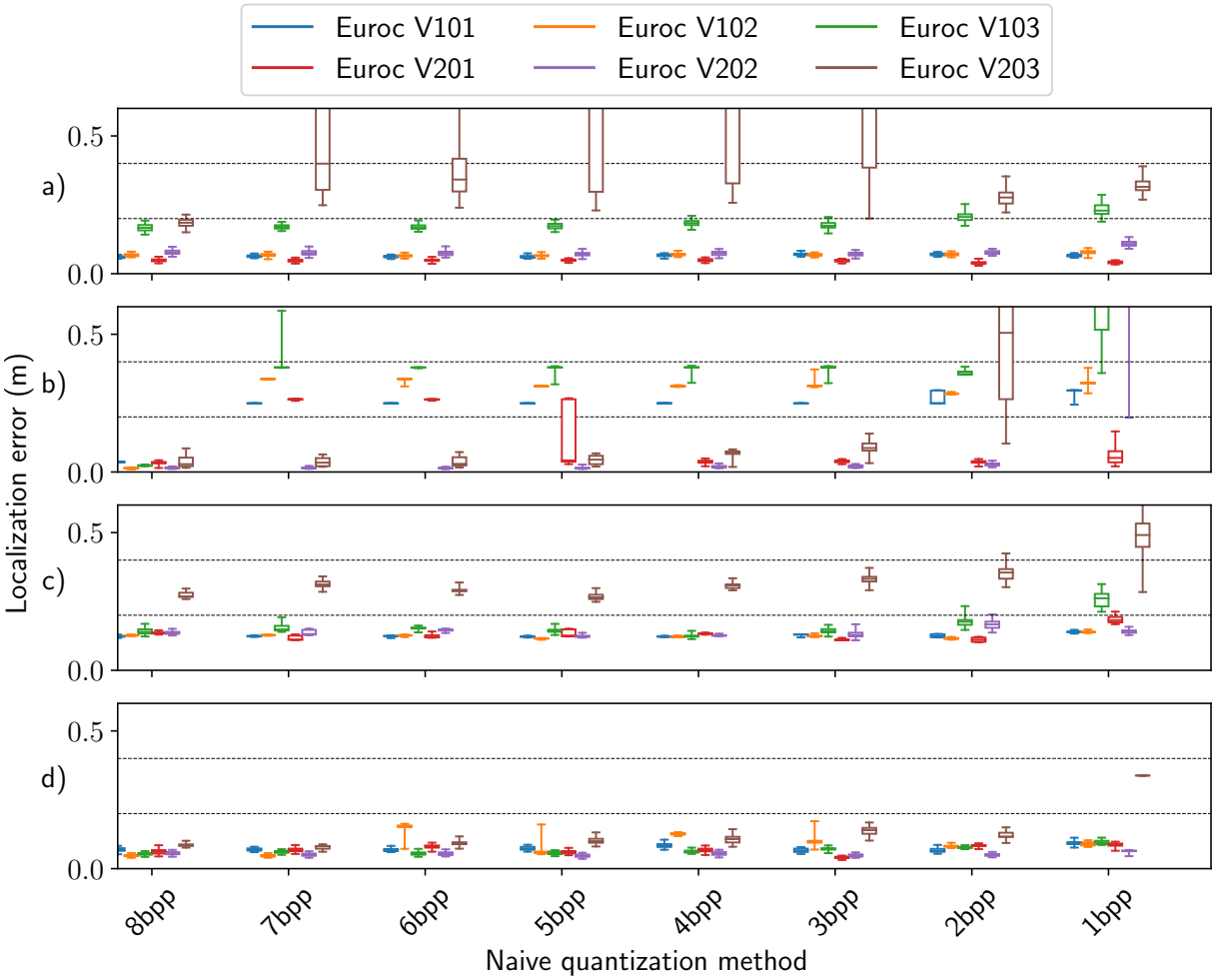


FIGURE 5.1 – Variabilité de l'erreur de localisation des algorithmes SLAM à partir des séquences *Vicons* de la base de donnée Euroc en fonction de la quantification d'image naive (NV) de 8b à 1b avec a) KimeraVIO, b) ORBSLAM3, c) VINSFusion, d) OpenVINS.

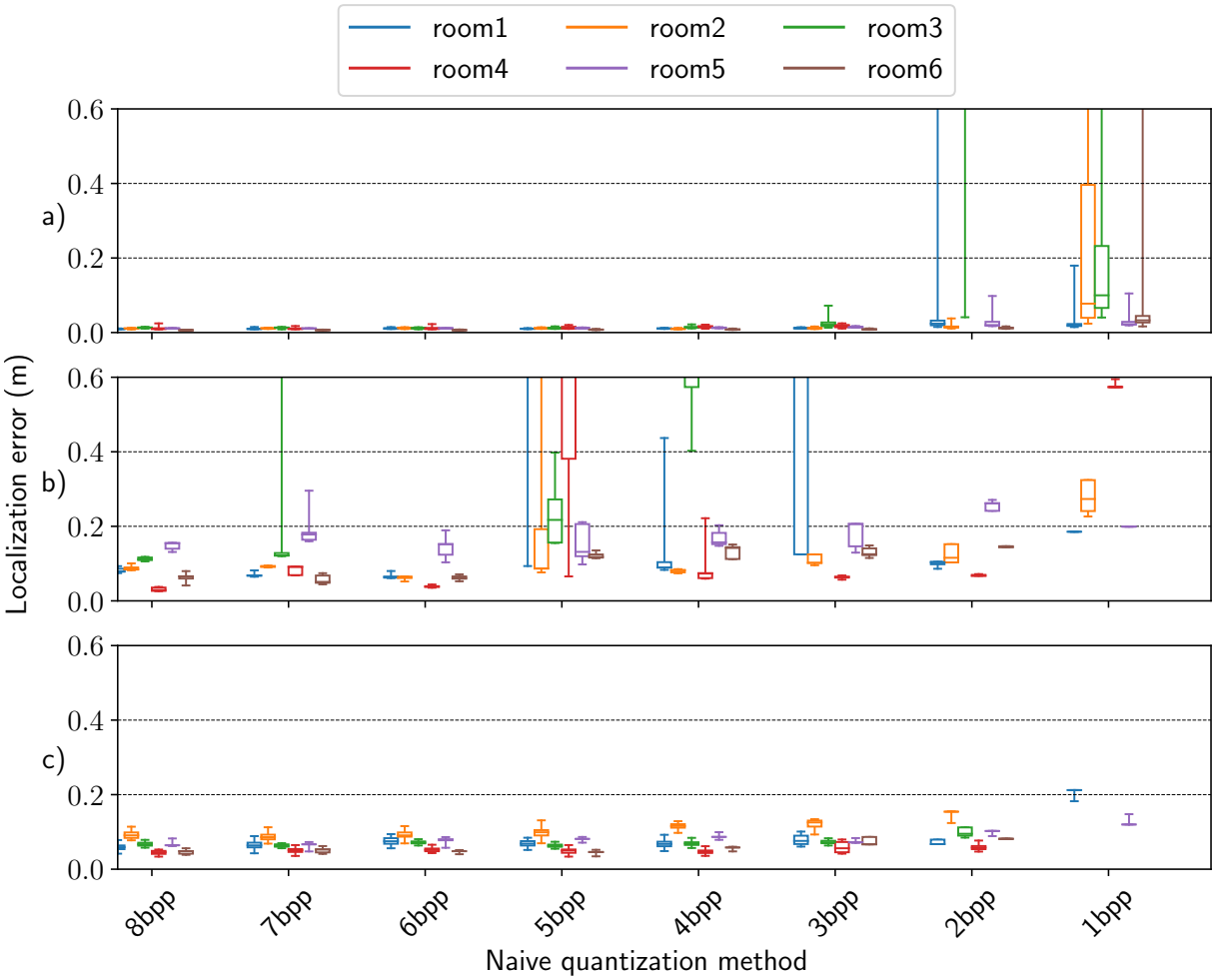


FIGURE 5.2 – Variabilité de l'erreur de localisation des algorithmes SLAM à partir des séquences *Room* de la base de donnée TUMVI en fonction de la quantification d'image naive (NV) de 8b à 1b avec a) ORBSLAM3, b) VINSFusion, c) OpenVINS.

Cette première analyse avec la méthode de référence naïve confirme l'intérêt de réduire la plage d'encodage des pixels tout en obtenant une précision de localisation similaire à la référence qu'est l'image en pleine précision 8b jusqu'à un certain seuil de quantification.

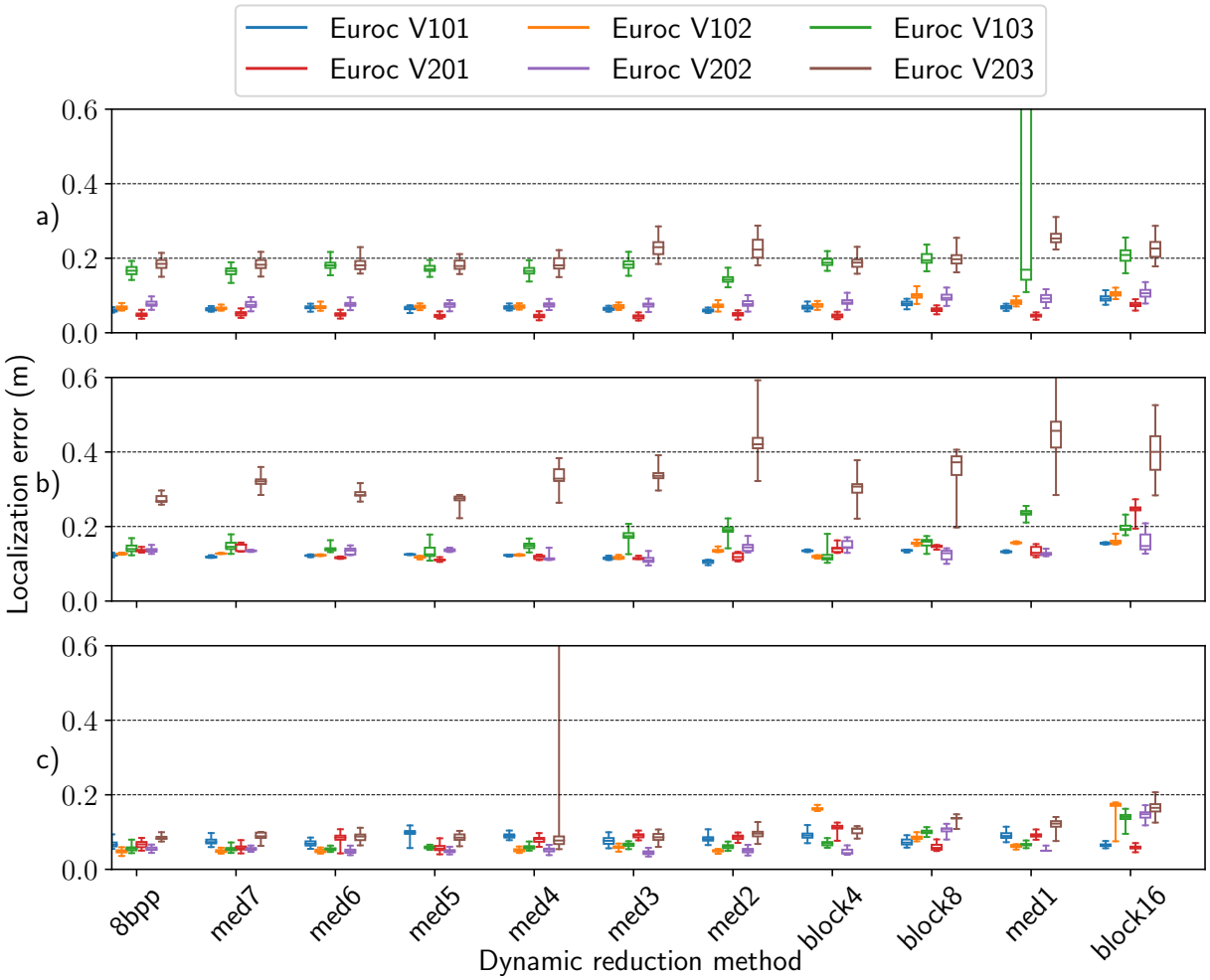


FIGURE 5.3 – Variabilité de l'erreur de localisation des algorithmes SLAM à partir des séquences *Vicons* de la base de donnée Euroc en fonction de la quantification d'image *median cut* (medX) et *block-wise* (blockN) avec a) KimeraVIO, b) VINSFusion, c) OpenVINS.

La Figure 5.3 montre les résultats de localisation pour les méthodes SLAM sans fermeture de boucle avec la réduction de la dynamique *median cut* et *block-wise*. Les trois algorithmes 5.3a 5.3b 5.3c présentent des résultats robustes par rapport à la référence *8bpp* pour chaque méthode de quantification allant jusqu'à une représentation à 1-bit/px qui est dénotée *med1*. Pour KimeraVIO (5.3a) avec la séquence *V103* en verte, l'erreur de localisation de la configuration *med1* est plus importante que celle à *8bpp*. La précision reste autour de 0.2m avec la méthode *block16* qui présente un pourcentage de réduction de donnée similaire à *med1* qui est de 87%. C'est une séquence difficile qui présente très peu de texture lorsque la représentation de l'image est à *med1*. La quantification *block16* permet à l'algorithme de détecter plus de points d'intérêts exploitables comme l'indique la

variation d'erreur. Les résultats de VINSFusion (5.3b) montrent que la variabilité de l'erreur s'accentue pour la majorité des séquences avec les images d'entrées quantifiées, notamment pour la séquence difficile *V203*. Les séquences faciles ne sont pas impactées par la réduction de donnée. La même analyse est appliquée à OpenVINS qui obtient une précision accrue pour chaque pas de quantification. On voit par cette figure que les séquences *Vicons* ne présentent pas de difficultés pour les algorithmes SLAM sans fermeture de boucle avec les méthodes de quantification utilisées. Cette expérimentation montre que la méthode *median cut* fournit une meilleure robustesse et précision sur les réductions de dynamiques basses par rapport à la quantification naïve de l'image.

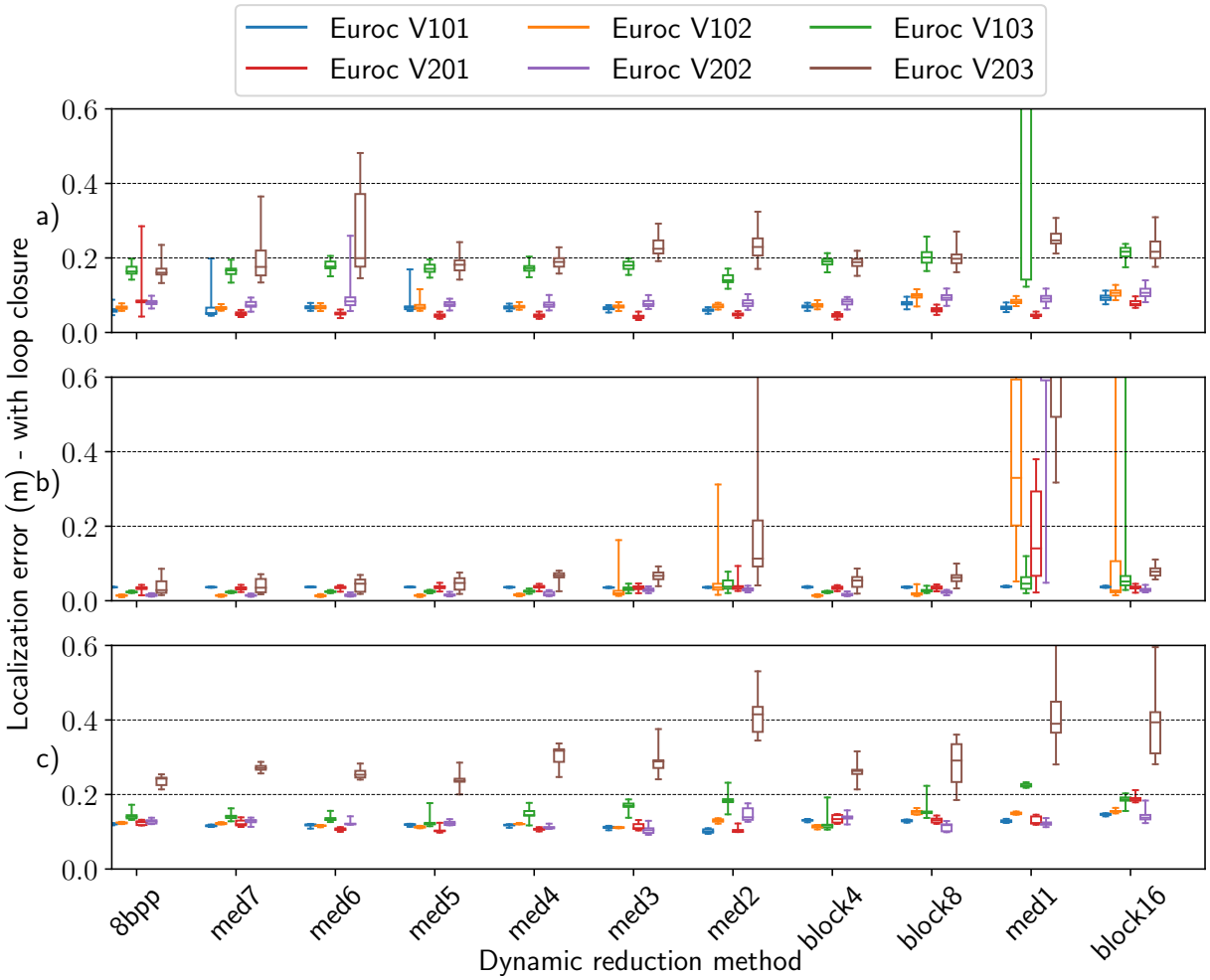


FIGURE 5.4 – Variabilité de l'erreur de localisation des algorithmes SLAM avec fermeture de boucle à partir des séquences *Vicons* de la base de donnée Euroc en fonction de la quantification d'image *median cut* (medX) et *block-wise* (blockN) avec a) KimeraVIO-lc, b) ORBSLAM3, c) VINSFusion-lc.

La figure 5.4 montre la variabilité de l'erreur de localisation pour les algorithmes SLAM avec le paramètre de fermeture de boucle activé (lc) pour KimeraVIO-lc 5.4a, ORBSLAM3 5.4b et VINSFusion-lc 5.4c. La variation de l'erreur de localisation est plus importante avec le paramètre de fermeture de boucle sur la séquence *V203* dans la figure 5.4a allant jusqu'à la représentation *med6* qu'avec

la méthode SLAM sans fermeture de boucle dans la figure 5.3a. La méthode VINSFusion-lc fournit des résultats robustes jusqu'à la quantification *med2* et pour la configuration *block8* sauf pour la séquence *V203* dont l'erreur de localisation varie considérablement en fonction de la quantification de l'image. La méthode ORBSLAM3 (5.4b) est robuste à la quantification *median cut* jusqu'à *med3*. La méthode SLAM basée sur la détection et la description de points d'intérêts est plus robuste comparé aux méthodes basées sur la technique du *tracking KLT* employée pour les algorithmes des figures 5.4a et 5.4c grâce à la description binaire des points d'intérêts (Rublee et al., 2011).

Les figures 5.3 et 5.4 montrent que la précision des algorithmes se dégrade progressivement jusqu'à une représentation à 2b/px avec la quantification *med2*. Les SLAM sont plus robustes avec la quantification *block4* qui permet d'obtenir une précision similaire à la référence 8b tout en quantifiant l'image avec une réduction de donnée de 75%. Cela s'explique par la représentation plus nette des textures de l'image en minimisant le bruit rajouté et qui n'impacte pas la qualité de l'estimation de localisation des algorithmes utilisés. Le choix de la méthode de réduction de la dynamique dépend fortement de la méthode SLAM et de la difficulté de la base de données. La configuration *block4* et *med4* proposent un compromis entre la réduction de la donnée d'entrée de 75% et 50% respectivement et la précision qui est reste similaire à la référence.

Les résultats des séquences *Machine Hall* de la base de données Euroc sont présentés dans les figures 5.5 et 5.6 pour les algorithmes SLAM avec et sans fermeture de boucle respectivement. La figure 5.5 montre que la robustesse des algorithmes n'est pas impactée par la quantification de l'image. La méthode KimeraVIO (5.5a) présente peu voir aucune différence par rapport à la référence 8bpp. La même analyse s'applique pour VINSFusion (5.5b) où l'on voit également que la précision avec la séquence MH04 en rouge est similaire à la référence pour la configuration *block16* alors que pour un pourcentage de réduction de donnée similaire, la méthode *med1* ne fournit pas de précision inférieure à 0.6m. La séquence MH04 est également celle qui présente des résultats qui varient considérablement pour OpenVINS dans la figure 5.5c. La quantification permet d'obtenir une meilleure précision que la référence jusqu'à la quantification *med5* et *block4* pour la méthode *block-wise*.

Comme pour les séquences *Vicons*, la figure 5.6 montre que la variation d'erreur de localisation est plus importante avec la fermeture de boucle pour KimeraVIO (5.6a). Plus on quantifie l'image d'entrée, plus l'algorithme est robuste. Globalement, les algorithmes SLAM sont robustes à la quantification de donnée avec la base de données Euroc. Les méthodes de localisation sont robustes à une quantification allant jusqu'à 75% de réduction de donnée, ce qui correspond à la quantification *block-wise* avec une fenêtre de taille 4×4. La taille de fenêtre 8×8 peut également être considérée, mais la figure 5.5c montre que l'erreur de localisation augmente significativement pour la configuration *block8* sur la séquence *MH04*.

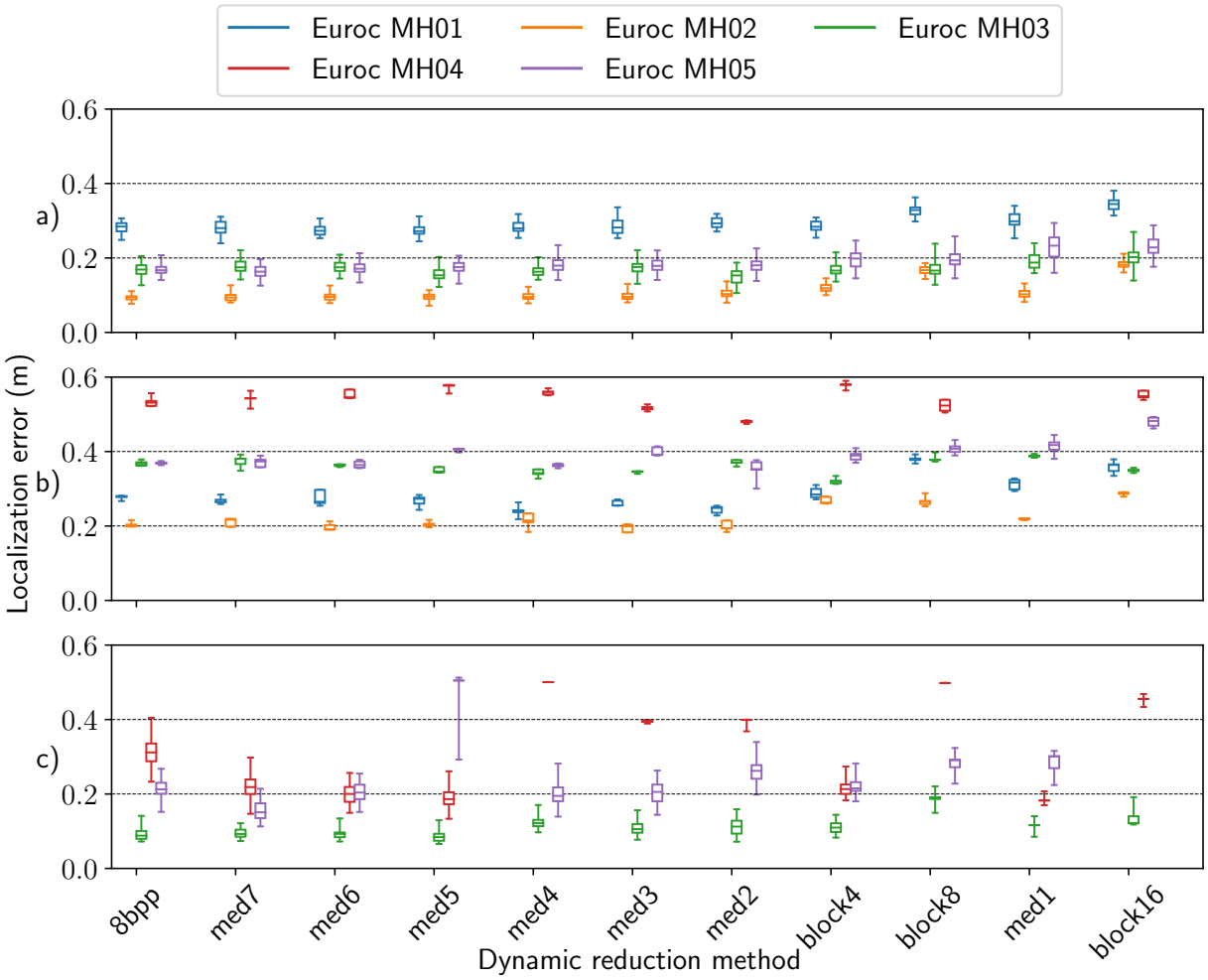


FIGURE 5.5 – Variabilité de l'erreur de localisation des algorithmes SLAM à partir des séquences *Machine Hall* de la base de donnée Euroc en fonction de la quantification d'image *median cut* (medX) et *block-wise* (blockN) avec a) KimeraVIO, b) VINSFusion, c) OpenVINS.

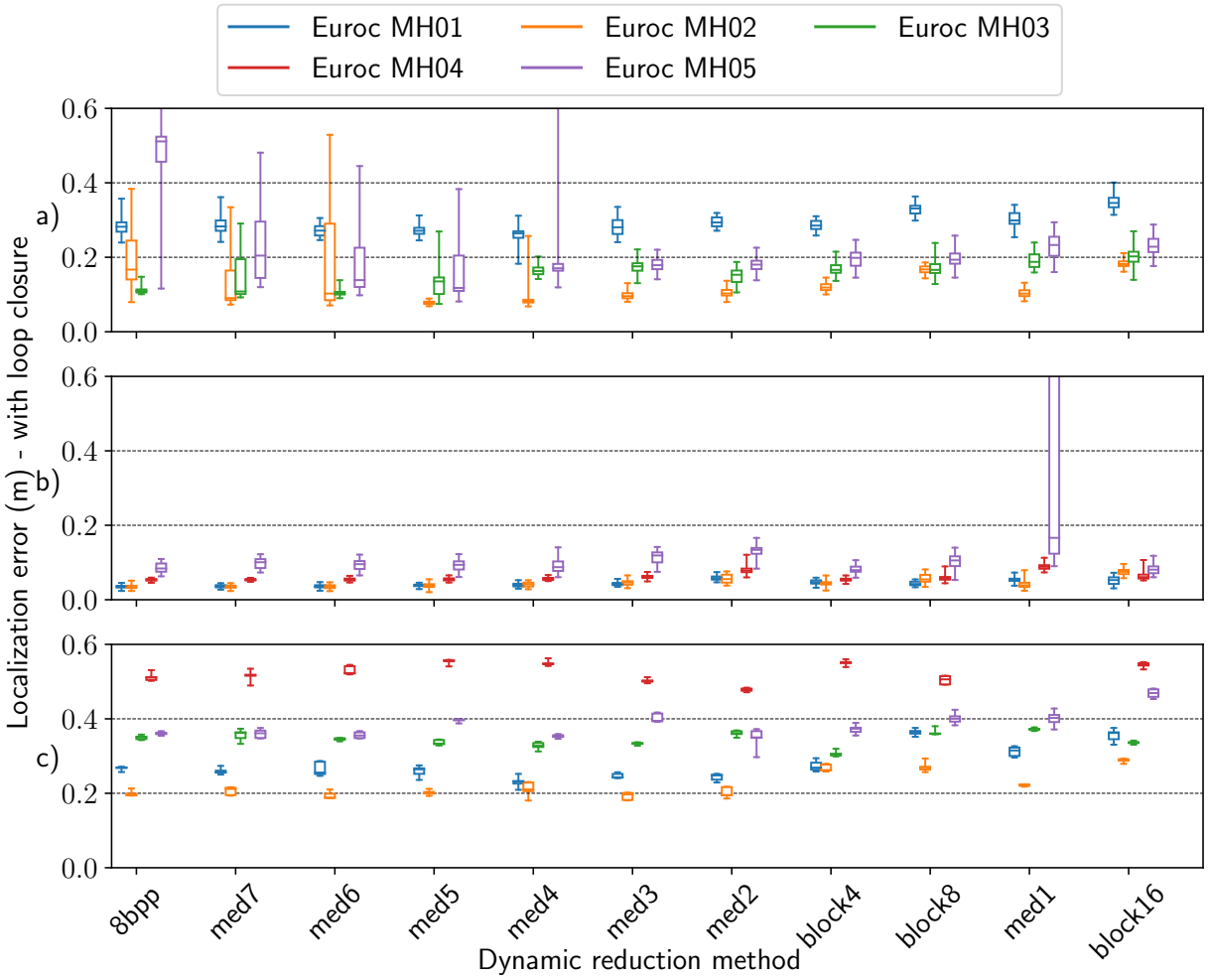


FIGURE 5.6 – Variabilité de l'erreur de localisation des algorithmes SLAM avec fermeture de boucle à partir des séquences *Machine Hall* de la base de donnée Euroc en fonction de la quantification d'image *median cut* (medX) et *block-wise* (blockN) avec a) KimeraVIO-lc, b) ORBSLAM3, c) VINSFusion-lc.

La suite des expérimentations concerne la base de données TUMVI qui est plus difficile que Euroc pour deux raisons : 1/ les séquences sont capturées par une caméra tenue à la main ce qui accentue fortement les mouvements et 2/ les séquences *corridor* sont longues dans le temps et l'erreur de localisation des SLAM s'accumule de pose en pose ce qui peut engendrer une localisation approximative. Les figures 5.7 et 5.8 présentent les résultats des algorithmes SLAM dans les séquences *room*.

La figure 5.7a montre à quel point les séquences sont difficiles pour la configuration monoculaire et inertielle de VINSFusion lorsque l'image d'entrée est quantifiée. L'impact de la quantification est notamment présente pour la séquence *room3* qui illustre une variabilité d'erreur importante dès la quantification *med7*. Le paramètre de fermeture de boucle activée pour l'algorithme présente les mêmes résultats dans la figure 5.8b. ORBSLAM3 est robuste à la quantification de l'image jusqu'à une représentation à 2 bit par pixel avec *med2*. L'algorithme présente de grandes variations d'erreur lorsque l'image est quantifiée à 1b par pixel.

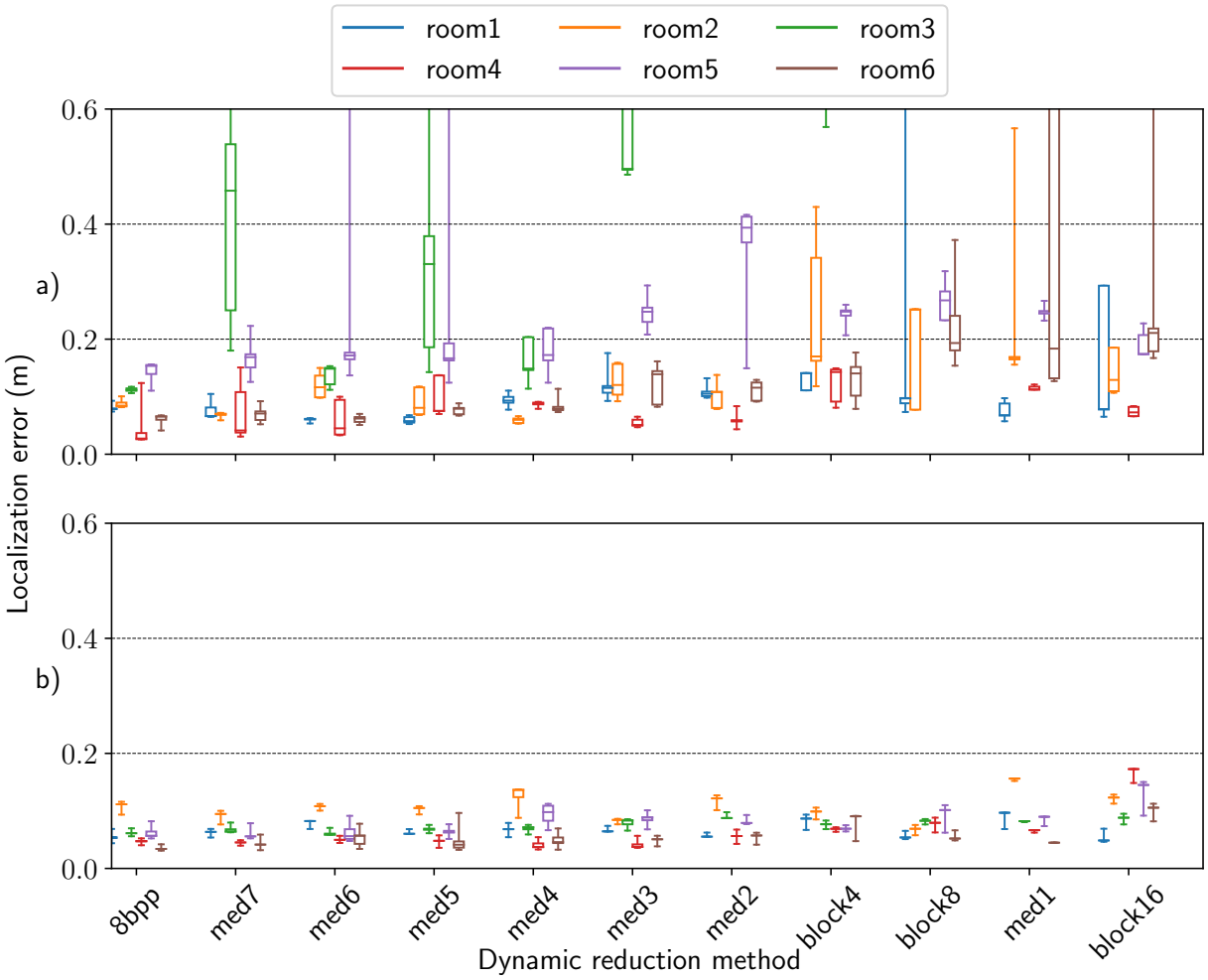


FIGURE 5.7 – Variabilité de l'erreur de localisation des algorithmes SLAM à partir des séquences *Room* de la base de donnée TUMVI en fonction de la quantification d'image *median cut* (medX) et *block-wise* (blockN) avec a) VINSFusion, b) OpenVINS.

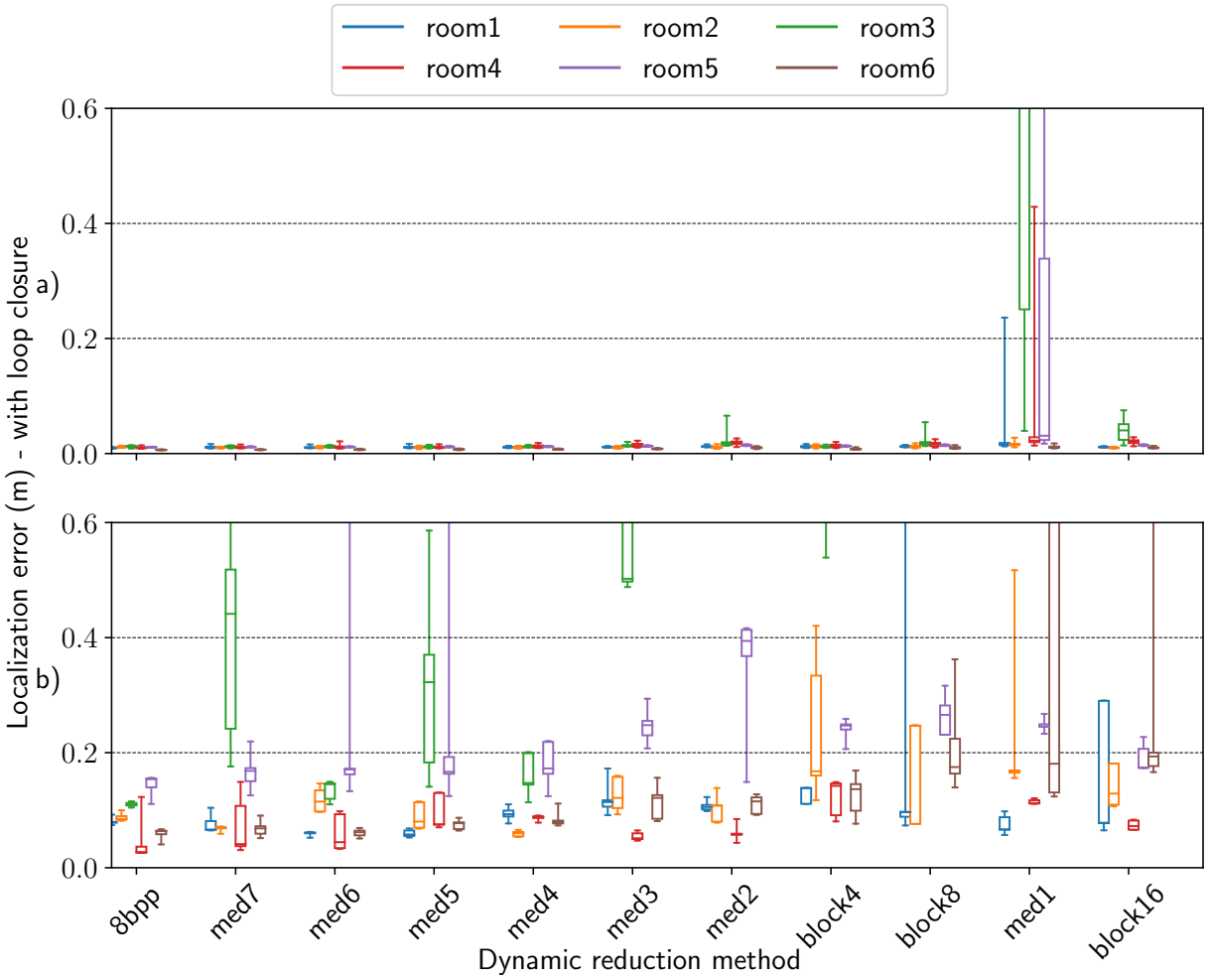


FIGURE 5.8 – Variabilité de l'erreur de localisation des algorithmes SLAM avec fermeture de boucle à partir des séquences *Room* de la base de donnée TUMVI en fonction de la quantification d'image *median cut* (medX) et *block-wise* (blockN) avec a) ORBSLAM3, b) VINSFusion-lc.

Les séquences *corridor* présentent plus de variation d'erreur de localisation pour chaque algorithme SLAM dans les figures 5.9 et 5.10. La figure 5.10a montre qu'avec la méthode *median cut*, ORBSLAM3 est robuste jusqu'à la représentation à 4b par pixel, soit *med4*. Dans la même figure, on remarque que l'algorithme VINSFusion (5.10b) avec le paramètre de fermeture de boucle activé permet de réduire l'erreur de localisation par rapport à la référence pour une représentation *med6*. Cette même analyse s'applique pour OpenVINS dans la figure 5.9b où l'on obtient une meilleure précision pour les séquences *corridor4-5* avec une quantification allant jusqu'à *block8*.

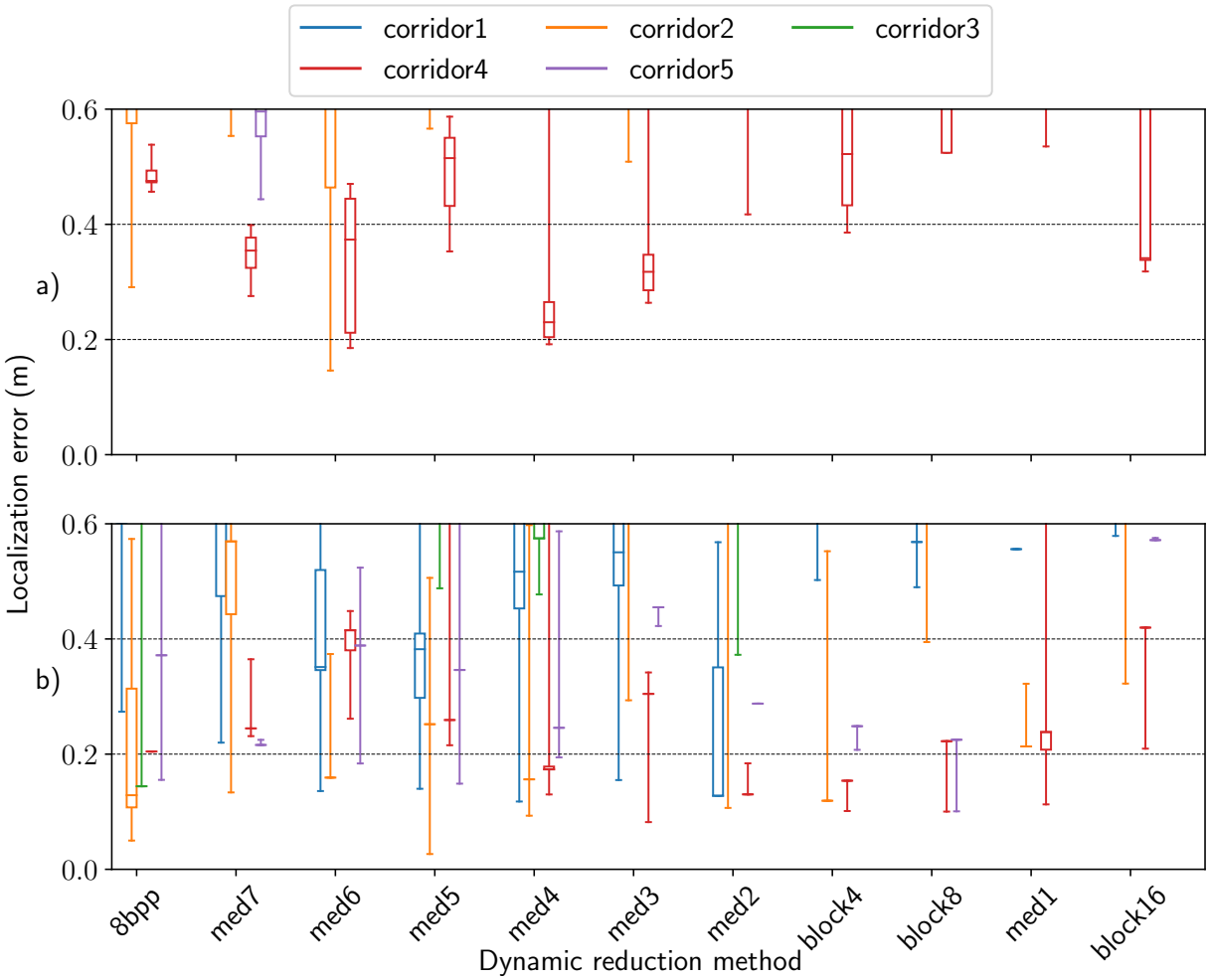


FIGURE 5.9 – Variabilité de l'erreur de localisation des algorithmes SLAM à partir des séquences *Corridor* de la base de donnée TUMVI en fonction de la quantification d'image *median cut* (medX) et *block-wise* (blockN) avec a) VINSFusion, b) OpenVINS.

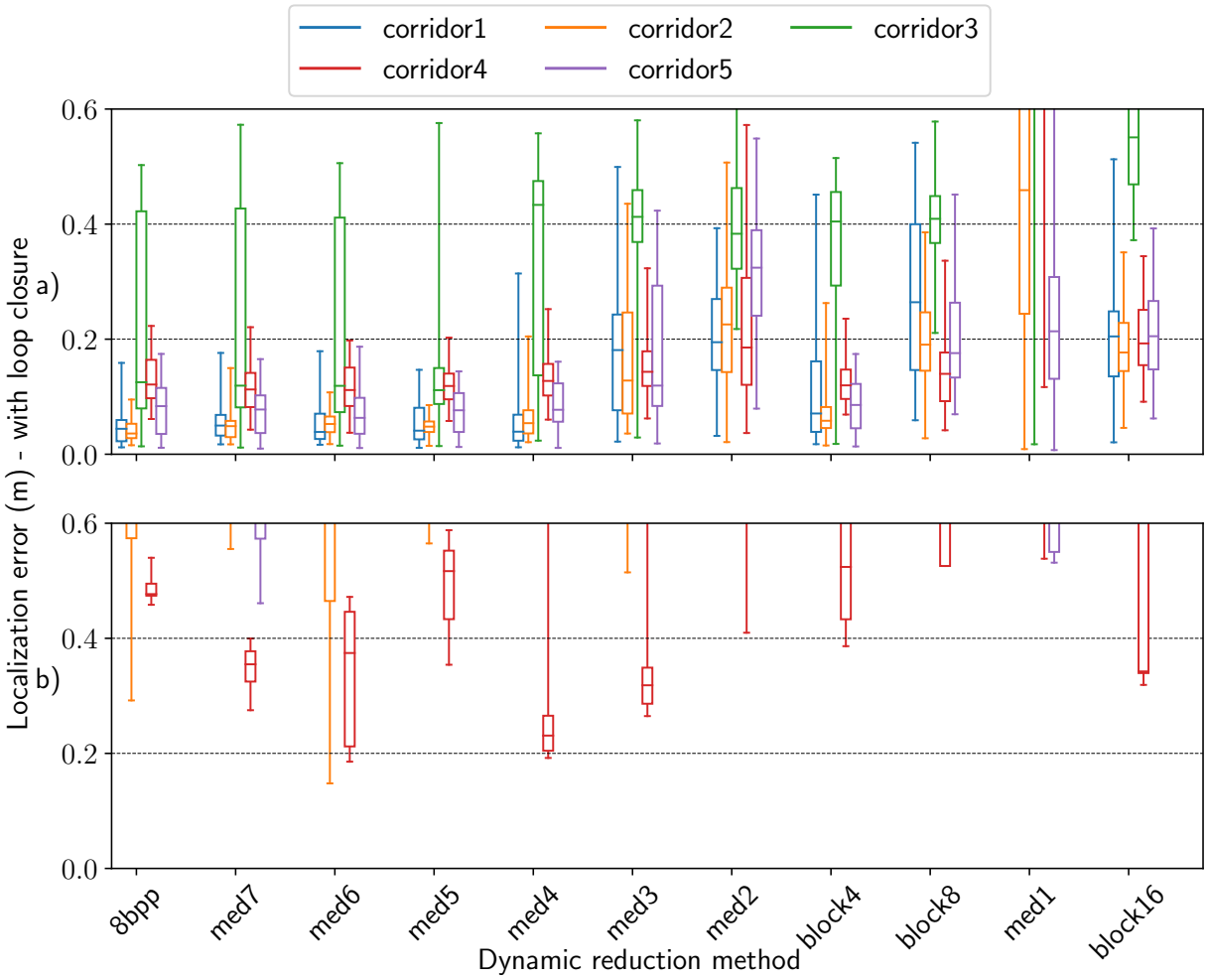


FIGURE 5.10 – Variabilité de l'erreur de localisation des algorithmes SLAM avec fermeture de boucle à partir des séquences *Corridor* de la base de donnée TUMVI en fonction de la quantification d'image *median cut* (medX) et *block-wise* (blockN) avec a) ORBSLAM3, b) VINSFusion-lc.

Les expérimentations menées montrent que les algorithmes SLAM sont robustes à la quantification de l'image d'entrée suivant le cas d'usage applicatif. Cela oblige à ajuster les paramètres de quantification pour l'utilisation de méthodes de localisation temps-réel pour les drones (Euroc) et pour les capteurs tenus à la main (TUMVI). Les différentes analyses détaillent que l'image peut être quantifié jusqu'à une représentation à 4b avec la technique *med4* pour le cas d'usage du casque RA/RV et *block4* pour le cas d'usage du drone ce qui représente une réduction de donnée de 50% et de 75% respectivement.

La figure 5.11 montre l'erreur relative de localisation par rapport à la référence en fonction des techniques de quantification utilisées avec les séquences difficiles de *Vicons* 5.11a et de *Machine hall* 5.11b. Dans 5.11a, la configuration *block4* confirme que la précision est similaire voire meilleure que la référence avec un gain d'environ 5%. Dans 5.11b, la précision avec cette méthode de quantification reste compris dans l'intervalle $\pm$20% d'erreur relative par rapport à la référence. La figure montre le gain de 10% de précision avec la configuration *med2*. Cependant, cette quantification n'est pas celle choisie pour le cas d'usage du drone, car l'erreur de précision relative dans la figure 5.11a augmente significativement avec la séquence difficile de *Vicons*. Quant au cas d'usage correspondant au casque RA/RV avec la caméra tenue à la main, le choix de la méthode de quantification *med4* correspond au compromis à réaliser en termes d'erreur de précision relative illustrée dans les résultats pour le cas d'usage de la base de donnée TUMVI. La figure 5.11c montre que l'erreur relative pour les séquences *room* ne fait qu'augmenter avec la quantification de l'image. Les séquences *corridor* dans la figure 5.11d s'avèrent plus propice puisque l'on peut quantifier la donnée jusqu'à *med3* tout en gagnant en précision de 10% par rapport à la référence et qui permet d'avoir une réduction de donnée de 62%. En regardant dans le détail la robustesse des algorithmes sur les séquences *corridor*, on remarque que la quantification *med4* est le choix le plus pertinent, comme c'est illustré dans la figure 5.9b. La technique qui permet d'obtenir une précision similaire à la référence tout en assurant une robustesse pour les séquences *room* et *corridor* est la réduction de la dynamique par la représentation de l'image à 4b par pixel, soit *med4*.

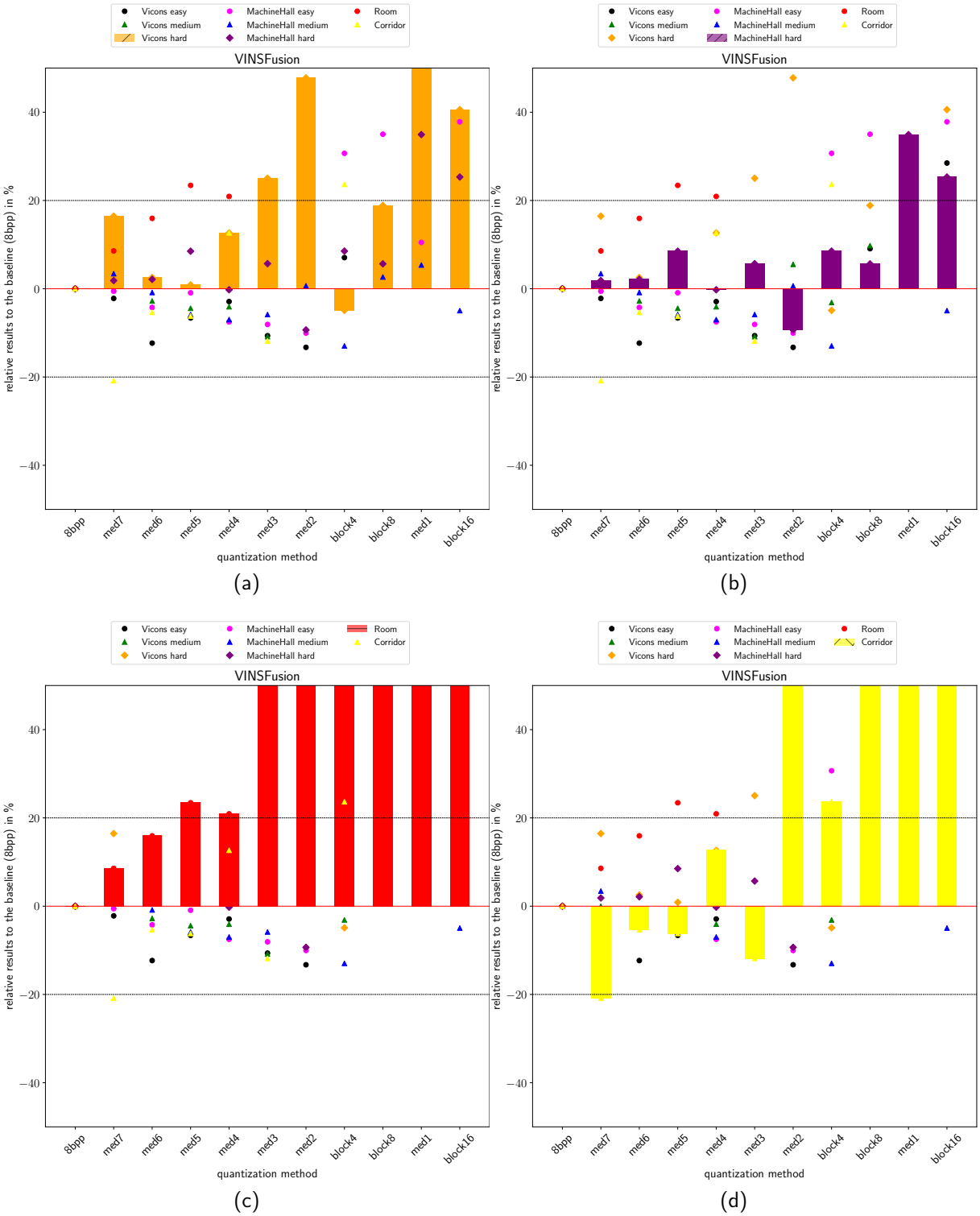


FIGURE 5.11 – Erreur relative de localisation de VINSFusion par rapport à la référence 8b en fonction des techniques de quantification utilisées sur les séquences difficiles *Vicons* (a), *Machine Hall* (b) et *Room* (c), *Corridor* (d).

5.1.2. Analyse approfondie sur les fonctions SLAM

Les expérimentations ont montré l'impact de la réduction de la dynamique sur les méthodes SLAM en termes de précision de localisation. Quel est le gain d'un point de vue matériel pour ces algorithmes de localisation en temps-réel ? La réduction de l'encodage des bits dans le calcul du traitement de l'image permet de réduire le nombre d'opérations à réaliser. La quantification permet également de gagner en espace mémoire lorsqu'il y a l'utilisation de mémoires tampons pour les images que ce soit pour stocker l'intégralité des pixels ou seulement des lignes de pixels. Pour approfondir cette analyse, les expérimentations menées n'ont pas été concluantes en raison de l'outil utilisé. L'étude de la consommation mémoire a été réalisée grâce à l'outil *heaptrack* qui ne permet pas d'avoir une analyse fine des opérations réalisées pour traiter l'image en entrée. Cela vient notamment du fait que la majorité des algorithmes utilisent la librairie OpenCV qui donne une abstraction des traitements effectués sur l'image. L'analyse théorique n'a pas pu être démontrée en pratique et fait partie des perspectives de recherches concernant cette contribution. Cela nécessite une analyse de la manière dont l'algorithme de localisation est implémenté pour savoir comment la réduction de la dynamique impacte les différents étages du SLAM et comment l'erreur est propagée dans la chaine de traitement. Cette étude permettrait d'identifier quelles sont les étapes spécifiques des algorithmes qui sont impactées et à quel point elles le sont.

5.1.3. Robustesse de l'extraction des caractéristiques avec la quantification

En théorie, les étapes directement impactées par la quantification sont les fonctions de traitement des pixels, comme l'extraction de caractéristiques de l'image et la mise en correspondance des données visuelles. Les travaux de l'état de l'art les plus alignés avec la contribution proposée concernent l'optimisation utilisée pour réduire l'empreinte mémoire (Suleiman et al., 2019) et la quantification logarithmique ou par gradient (Christie et al., 2020). Cette dernière ne réduit pas l'encodage des pixels en dessous de 5 bits sans perdre en performance et sans augmenter le nombre de points d'intérêts à extraire comme c'est détaillé dans le chapitre 2.3. Le mécanisme développé impacte tous les étages de la chaine de perception tout en assurant une erreur de localisation faible pour deux cas d'usages différents, contrairement aux travaux de l'état de l'art qui prennent en considération le drone avec l'image en pleine précision pour la fonction d'extraction de caractéristiques. La mise en place de la méthode *median cut* pour la quantification de l'image réduit l'encodage des pixels jusqu'à 2 bits pour certaines séquences du jeu de données *Machine Hall* d'Euroc sans augmenter le nombre de points à détecter et quantifie l'image de manière uniforme sans perdre en précision. Les expérimentations identifient également la quantification optimale suivant les cas d'usage en comparant la méthode *median cut* et *block-wise*. Suivant toutes les séquences utilisées, la quantification 4 bits *med4* obtient le meilleur compromis pour le cas d'usage du casque à réalité mixte avec la base de données TUMVI. L'utilisation d'un drone avec le jeu de données Euroc permet de quantifier fortement l'image avec la méthode *block4* qui représente une réduction de données à 2 bits/pixel. Pourquoi le mécanisme de réduction de la dynamique fonctionne sur les algorithmes SLAM ?

La qualité de la trajectoire reconstruite dépend fortement de la qualité des points d'intérêts extraits de l'image. Les algorithmes utilisés emploient deux techniques différentes, ORB pour la mise en correspondance et Shi-Tomasi pour le suivi KLT. Les méthodes se basent sur une différence de gradients comparée à un seuil pour définir si un point est d'intérêt ou non, ce qui correspond à une

détection binaire. La mise en place du descripteur BRIEF pour ORB accentue la robustesse de la mise en correspondance puisqu'elle évite que le descripteur binaire soit sensible au bruit. En perdant de l'information par la quantification de l'image grâce à la méthode *median cut*, l'extraction des caractéristiques se concentre sur les zones d'intérêts de l'image. On réduit également la quantité de valeurs aberrantes lors de la mise en correspondance des données visuelles extraites. Autrement dit, lors de la quantification, les données pertinentes de l'image sont mises en évidences, ce qui permet d'extraire des points d'intérêts de meilleure qualité et ainsi d'obtenir une association des données robuste. Si on perd trop d'information dans l'image avec une quantification à 2b/pixel ou 1b/pixel, on perd de la précision dans l'extraction des caractéristiques ce qui engendre une perte de précision de l'algorithme. Dans le cas de la méthode *block-wise*, du bruit est ajouté à l'image qui impacte les performances de localisation lorsque la taille du bloc de quantification est élevé (exemple : taille de 16×16 pixels). Le compromis entre la perte de performance et la réduction des données est atteinte avec une taille de bloc par 4×4 pixels et une erreur de localisation faible.

5.2. Contrôle du flux d'images par le filtrage adaptatif

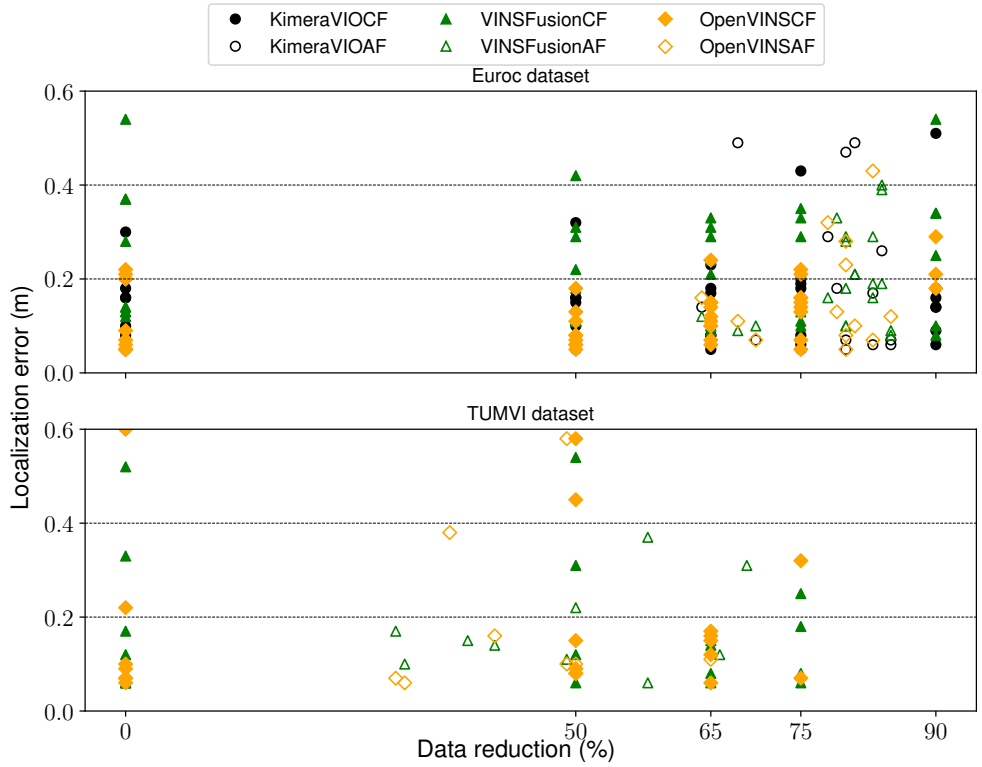
La section 5.1 a montré les résultats des expérimentations pour démontrer l'impact de la contribution proposée sur la réduction de la dynamique de l'image. Dans cette analyse, seule l'image est prise en compte pour réduire la quantité de données à injecter dans la chaine de traitement. Les algorithmes SLAM utilisés prennent en compte les capteurs d'images et la centrale inertielle. Le mécanisme de filtrage adaptatif proposé comme étant la deuxième contribution de la thèse permet de prendre en compte les données inertielles dans le but d'optimiser le nombre d'opérations à effectuer dans la chaine de traitement SLAM et ainsi de réduire les besoins en ressources sur la manière dont est injectée l'image.

5.2.1. Précision de l'estimation de trajectoire

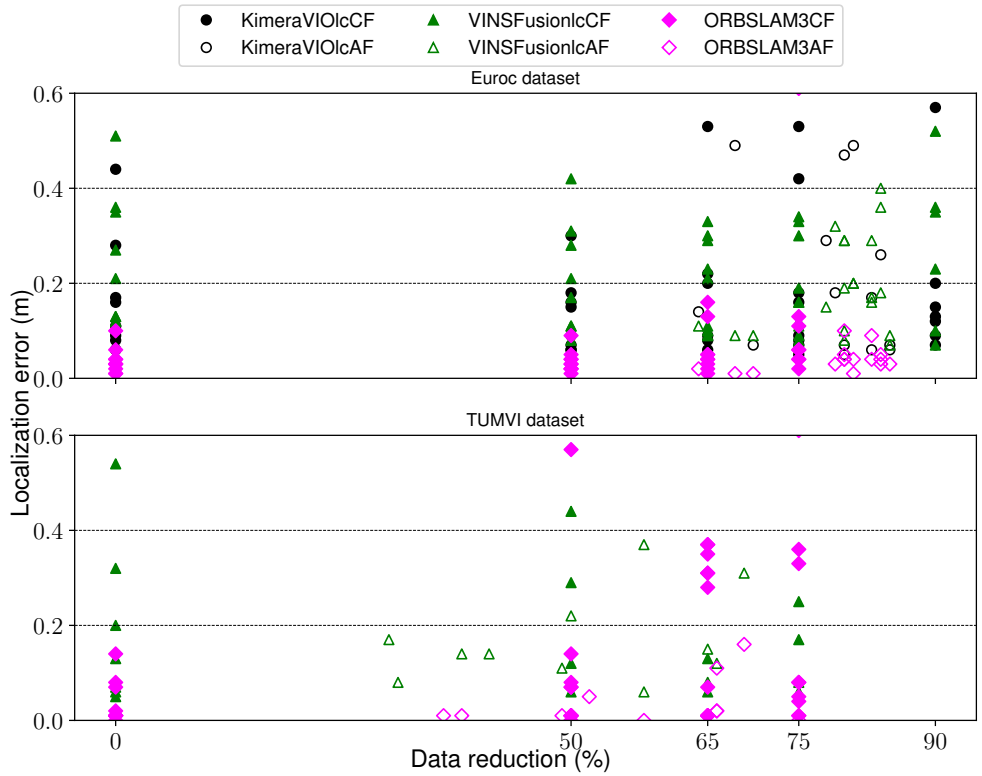
Afin d'évaluer l'impact du filtrage adaptatif sur la précision des trajectoires estimées, l'erreur de localisation a été extraite à partir du traitement naif par le filtrage constant du flux (CF) et par le filtrage adaptatif (AF) des séquences de Euroc et TUMVI. Pour gérer la variabilité de l'erreur dû à la nature des méthodes SLAM non déterministe, la précision médiane sur cinq exécutions de la trajectoire estimée est utilisée. La méthode KimeraVIO n'a pas pu être exécuté sur la base de donnée TUMVI en raison de problèmes de compatibilité des données de calibration.

Les figures 5.12a et 5.12b mettent en perspective la précision de localisation des trajectoires estimées par les méthodes SLAM en relation avec le pourcentage de réduction de données pour toutes les séquences utilisées.

La figure 5.12a illustre les résultats avec les méthodes SLAM sans la fermeture de boucle et avec la fermeture de boucle dans la figure 5.12b. Tous les résultats sont regroupés dans cette figure, que ce soit concernant les séquences et les heuristiques utilisées pour l'AF. La première conclusion importante est détaillée avec les résultats obtenus par la méthode à filtrage constante (CF) avec le pourcentage de réduction de données à 50%, 65%, 75% et 90%. La réduction de la quantité de donnée par la décimation à flux constant pour les méthodes SLAM permet d'obtenir des résultats relativement similaires à la référence (0% de réduction de donnée). Avec la base de donnée Euroc, la meilleure précision est atteinte avec la réduction du flux à 5-7 images par seconde (ips), ce qui correspond à 65-75% de réduction de donnée pour l'approche CF. Les résultats avec l'approche AF qui sont illustrés par les marqueurs vides montrent qu'il est possible de considérablement réduire la quantité d'images injectée avec la réduction de donnée entre 75% et 90% tout en fournissant des erreurs de localisation basses pour la majorité des méthodes SLAM. La figure illustre également les résultats avec la base de donnée TUMVI dans le graphique du bas. Malgré la difficulté plus importante de la base de donnée, l'approche AF permet de réduire la quantité de donnée injectée tout en gardant la qualité de reconstruction comparable à la référence. La réduction est moins importante que celle obtenue avec la base de donnée Euroc, soit une réduction d'environ 50% au lieu d'environ 75% pour Euroc.



(a)



(b)

FIGURE 5.12 – Erreur de localisation des algorithmes SLAM sans (a) et avec (b) fermeture de boucle en fonction du pourcentage de réduction de données par la décimation des images sur toutes les séquences d'Euroc et de TUMVI.

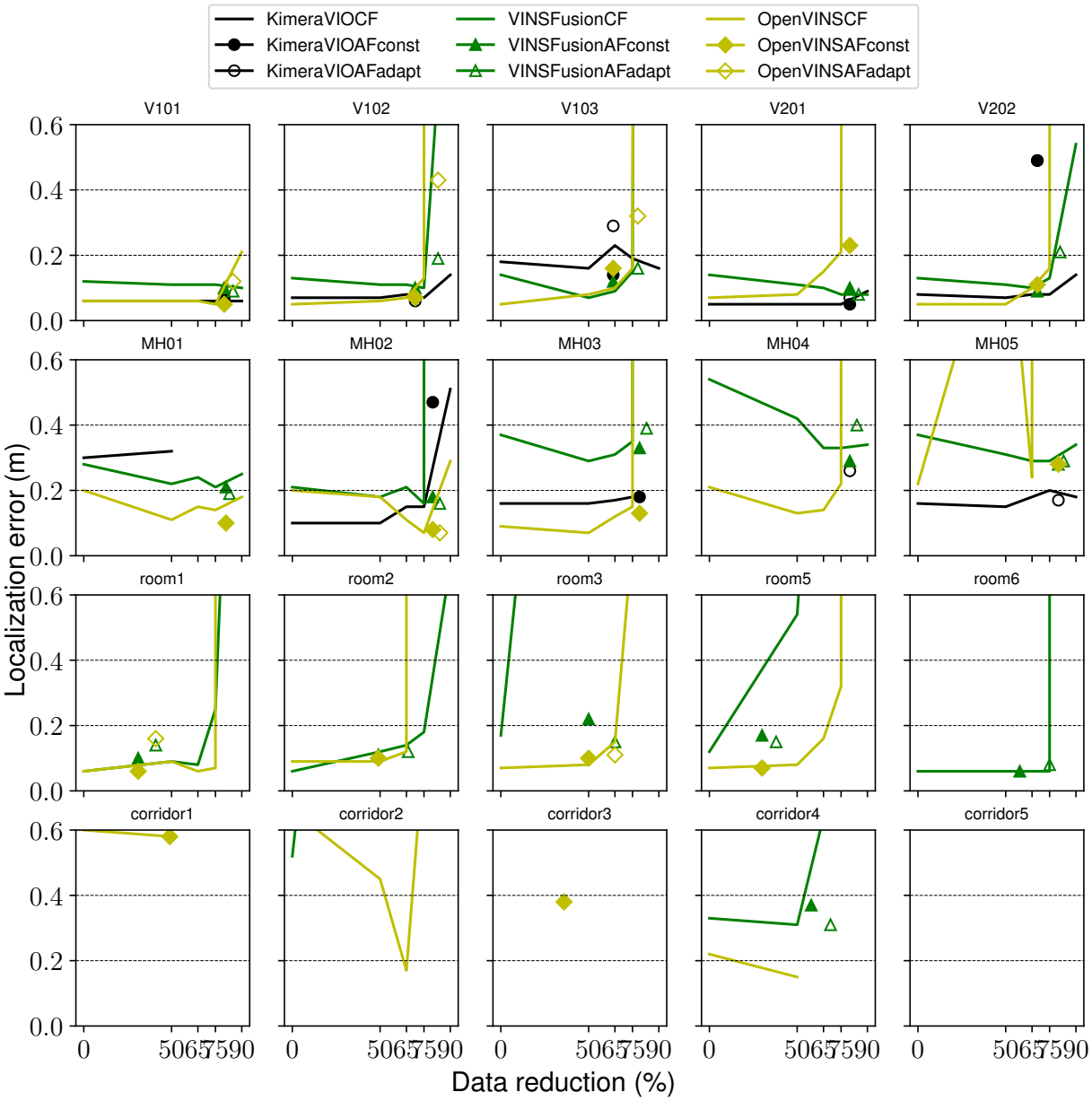


FIGURE 5.13 – Erreur de localisation des algorithmes SLAM sans fermeture de boucle en fonction du pourcentage de réduction de donnée

Les figures 5.13 et 5.14 montrent l'impact du filtrage adaptatif sur toutes les séquences d'Euroc et TUMVI avec les algorithmes respectivement sans et avec la fermeture de boucle activée. Les courbes sans marqueurs correspondent aux résultats de précision issus de la référence à filtrage constant du flux d'entrée (CF). Les marqueurs pleins et vides correspondent aux résultats de précision issus de la contribution proposée avec le filtrage adaptatif et l'utilisation des heuristiques constantes et adaptatives respectivement. Lorsque le résultat d'une des deux heuristiques n'est pas visible sur le graphique, cela signifie que l'algorithme a échoué à se localiser avec cette configuration. L'heuristique

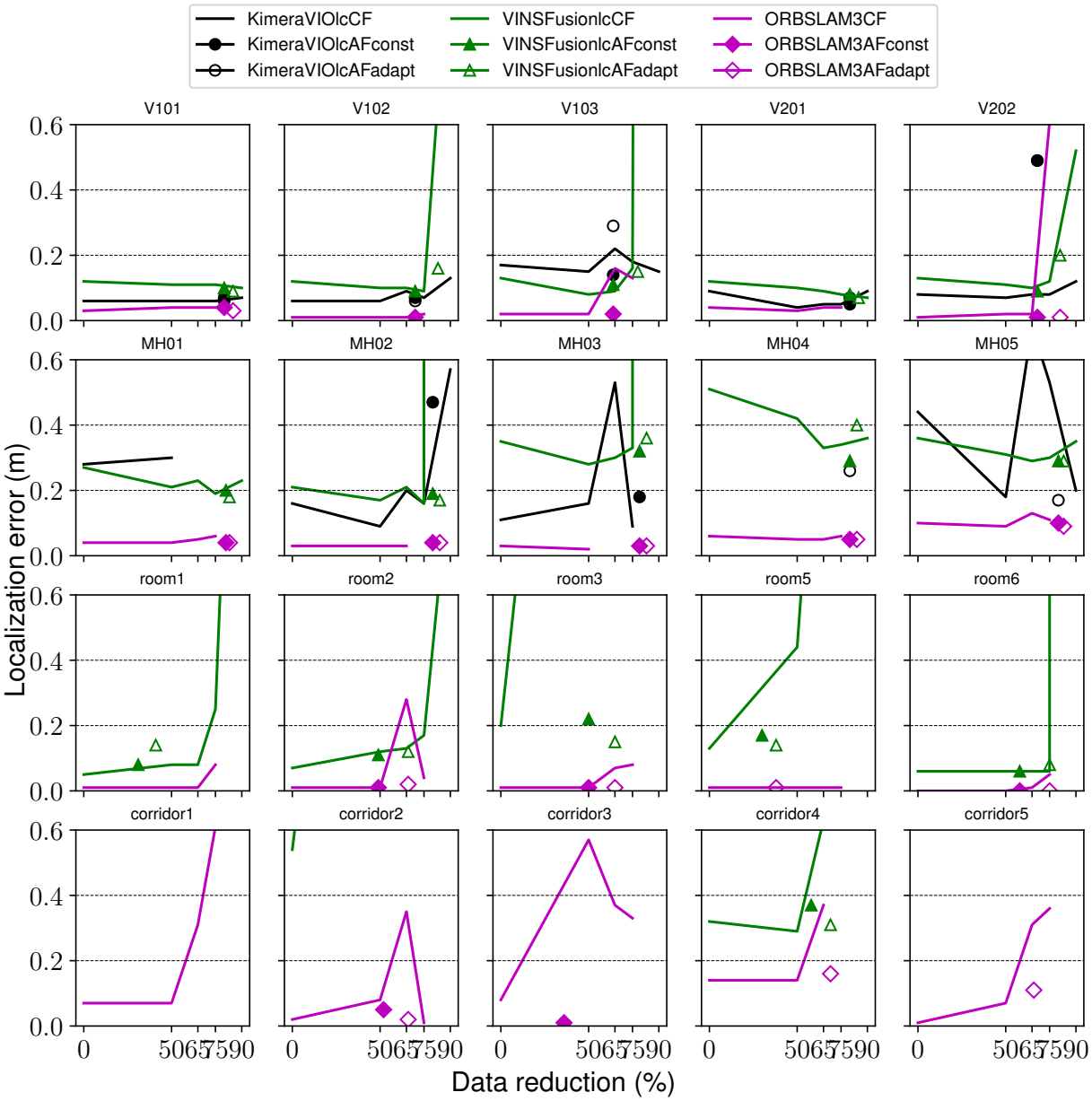


FIGURE 5.14 – Erreur de localisation des algorithmes SLAM avec fermeture de boucle en fonction du pourcentage de réduction de donnée.

de l'AF adaptative apporte un pourcentage de réduction de donnée plus important que l'AF constante. Les résultats suivent la méthodologie employée et détaillée dans le chapitre 4. Ainsi, certains marqueurs manquent aux graphiques. Par exemple, la séquence *V103* et l'algorithme ORBSLAM3AF de la figure 5.14 illustre seulement la précision pour l'heuristique constante de l'AF. Cela signifie que les résultats avec l'heuristique adaptative de l'AF ne correspondent pas aux critères établis dans la méthodologie et sont considérés comme des résultats aberrants.

Dans l'ensemble, les résultats montrent que les algorithmes ne sont pas robustes à une décimation

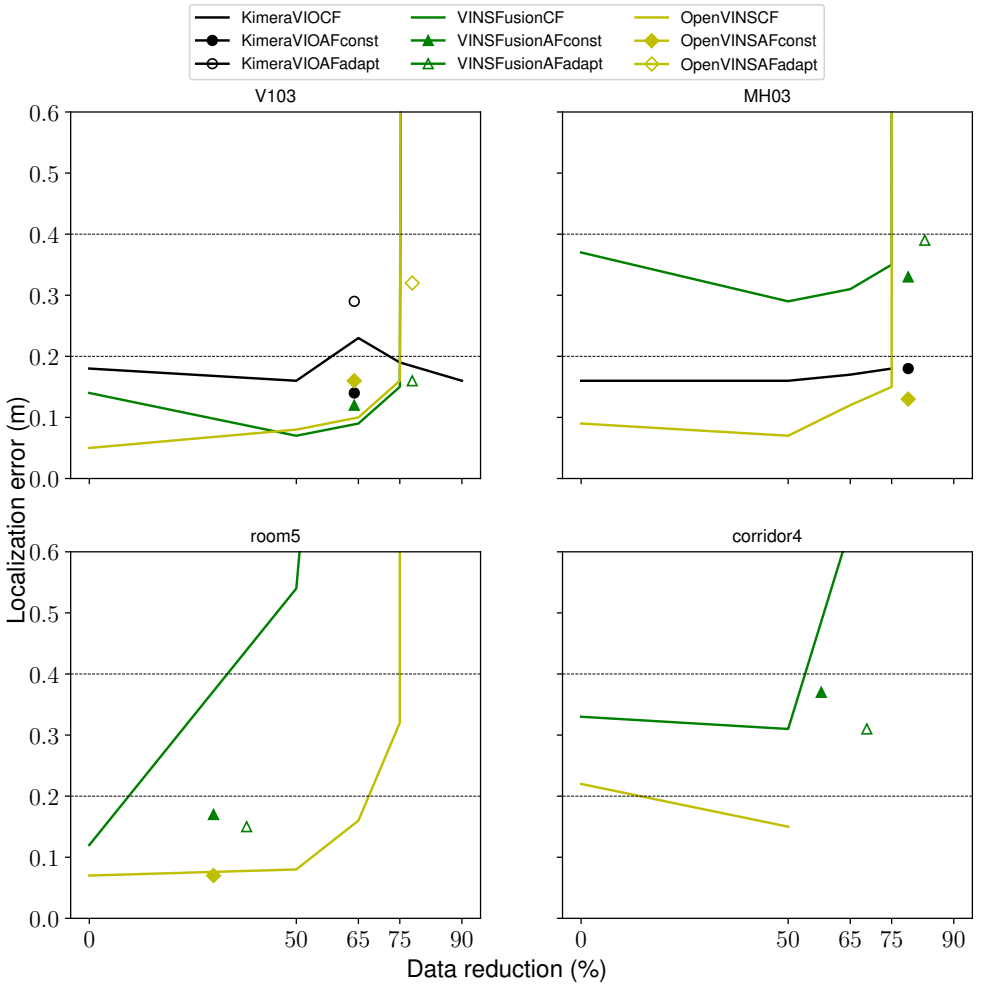


FIGURE 5.15 – Erreur de localisation des algorithmes SLAM sans fermeture de boucle en fonction du pourcentage de réduction de donnée sur les séquences difficiles.

constante du flux d'images allant jusqu'à 75%. Le filtrage adaptatif garantit une localisation précise avec une réduction de données supérieure à 80%, comme le montre le graphique *V102* de la figure 5.13 avec l'algorithme OpenVINS et VINSFusion. En comparant les résultats sans et avec la fermeture de boucle, peu de différences sont visibles avec les algorithmes KimeraVIO et VINSFusion (en noir et vert respectivement). La précision de localisation est meilleure de quelques millimètres, comme le montre la séquence *MH01*, ce qui est négligeable.

Dans le but d'obtenir une analyse plus fine des résultats, les quatre séquences les plus difficiles parmi les cas d'usage étudiés sont sélectionnées dans les figures 5.15 et 5.16. Ces séquences représentent la complexité des bases de données en termes de vitesse de déplacement du système, les changements de luminosité et la longueur de la trajectoire.

La figure 5.15 montre que l'AF réduit la quantité de donnée d'entrée tout en fournissant une précision similaire à la décimation du flux d'images CF correspondant avec la séquence *V103*. Les résultats de OpenVINS avec l'heuristique adaptative de l'AF au-delà de 75% démontre la robustesse

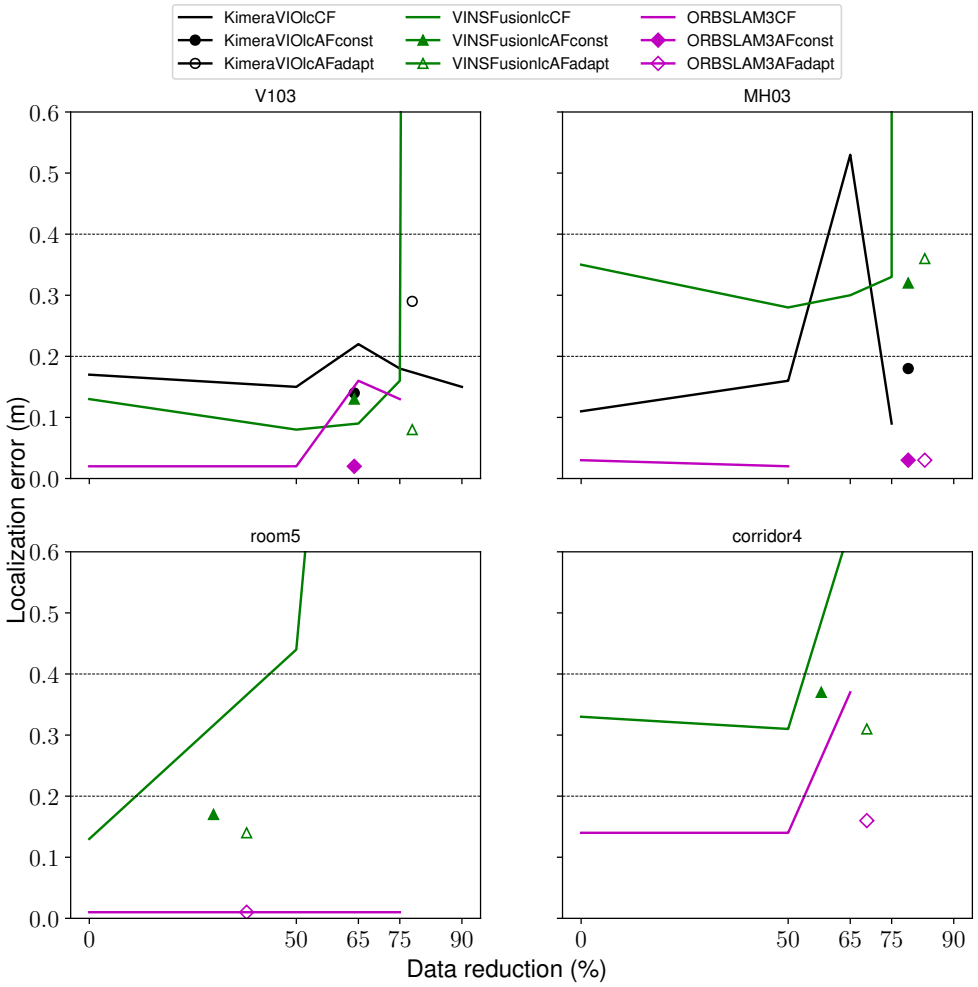


FIGURE 5.16 – Erreur de localisation des algorithmes SLAM avec fermeture de boucle en fonction du pourcentage de réduction de donnée sur les séquences difficiles.

de l'approche de filtrage comparé à la méthode naïve CF. Dans la séquence *MH03*, la même analyse s'applique pour VINSFusion. Pour les séquences *room5* et *corridor4*, la vérité terrain est seulement disponible pour le début et la fin de la trajectoire. Le filtrage des données injectées fournit des résultats plus sensibles que ceux pour la base de donnée Euroc. Pour VINSFusion, la précision du SLAM avec l'AF est nettement meilleure qu'avec CF. Dans la figure 5.16, ORBSLAM3 fournit une meilleure précision que les autres algorithmes SLAM avec la fermeture de boucle. Au-delà du filtrage constant de 50%, ORBSLAM3 échoue à estimer la trajectoire du système pour la séquence *MH03*. On voit l'impact du filtrage adaptatif qui permet à l'algorithme de s'initialiser correctement et d'assurer une reconstruction précise de la trajectoire tout en réduisant considérablement le nombre d'images à traiter.

5.2.2. Analyse de la consommation mémoire

Dans le contexte embarqué, l'optimisation de la consommation mémoire est cruciale pour minimiser le coût de l'empreinte de l'algorithme sur la plateforme matérielle embarquée. Dans ce cadre, l'analyse des performances SLAM sur système embarqué est émulée en logiciel par deux manières : 1/ réduction du temps d'exécution de l'algorithme, 2/ augmentation du flux d'entrée des images du SLAM.

La première émulation de l'algorithme consiste à utiliser deux techniques. La première utilise l'outil d'analyse mémoire *heaptrack* qui réduit considérablement le temps d'exécution du SLAM. Par exemple, VINSFusion a une latence moyenne mesurée de 60.77ms avec la séquence *V103* de la base de donnée Euroc. Lorsque la couche *heaptrack* est ajoutée, cette latence est augmentée à 577.25ms. La latence est définie par le temps entre l'injection de l'image d'entrée dans la chaine de traitement et l'estimation de pose en sortie. La deuxième technique consiste à maitriser la latence en augmentant le temps d'exécution de l'algorithme de 100ms à 200ms par incrémentation de 20ms.

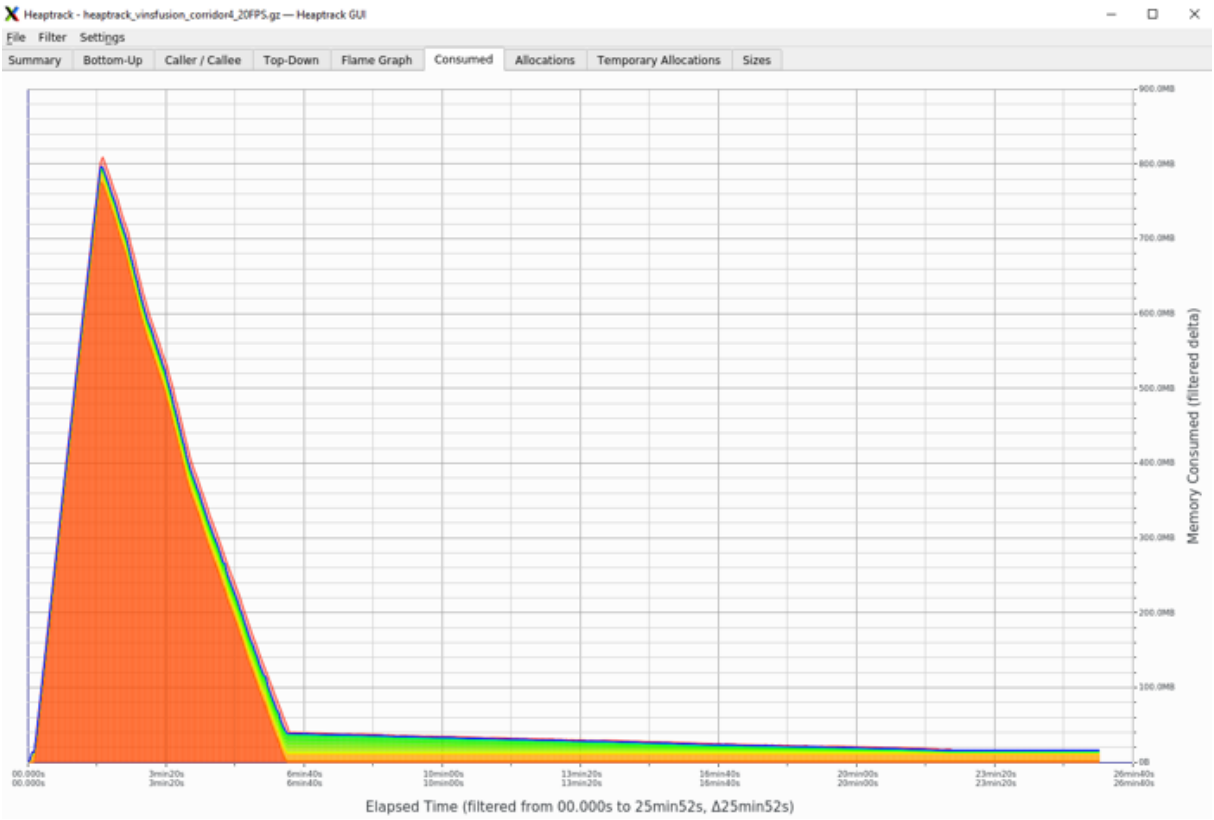


FIGURE 5.17 – Consommation de mémoire de l'algorithme VINSFusion sur *corridor4* avec le flux d'entrée constant à 20 ips.

Les figures 5.17, 5.18, 5.19 montrent les résultats de la consommation de mémoire avec l'outil d'analyse mémoire *heaptrack* pour VINSFusion sur la séquence *corridor4* de la base de donnée TUMVI. Cette séquence a été choisie pour visualiser clairement l'impact de la réduction de données avec la contribution de l'AF sur la consommation de la mémoire. L'étude a été réalisée avec l'algorithme sans

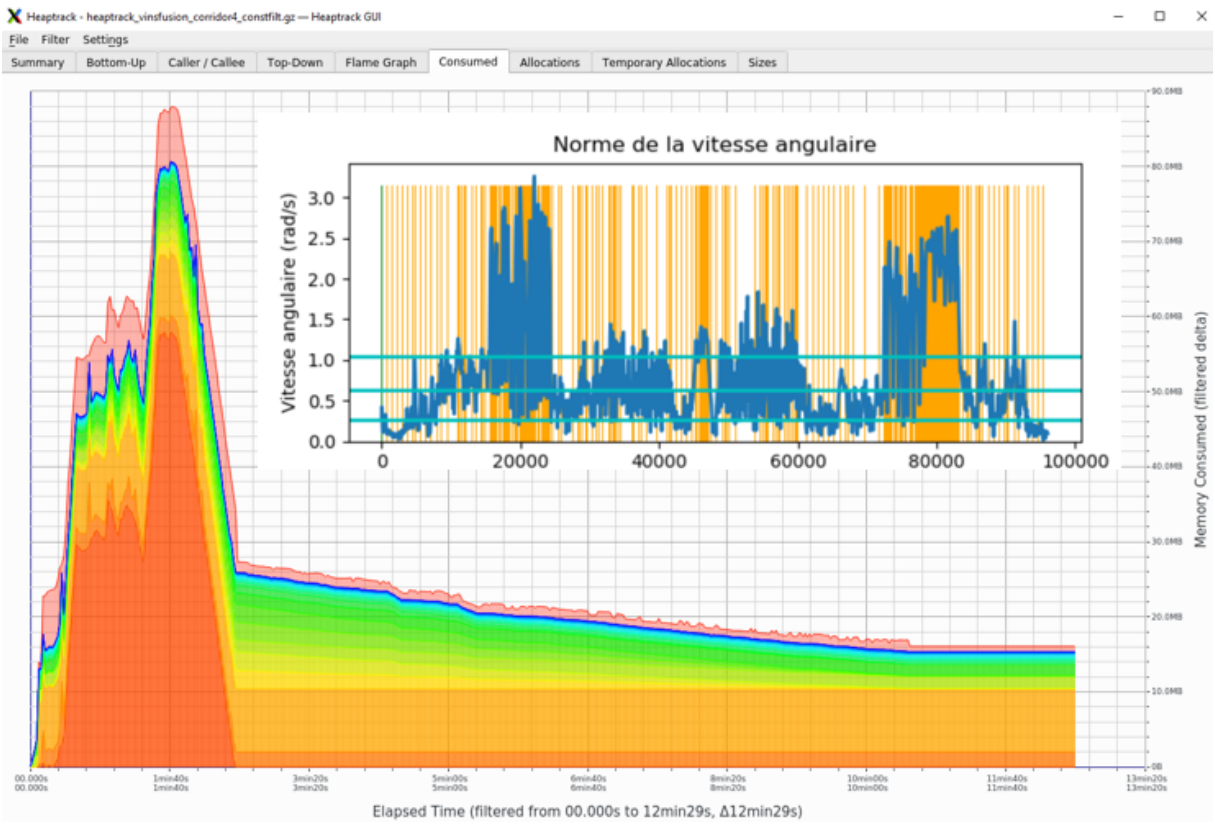


FIGURE 5.18 – Consommation de mémoire de l'algorithme VINSFusion sur *corridor4* avec le flux d'entrée adaptatif par l'heuristique AFconst.

modification, qui inclut une mémoire tampon des images d'entrées. La figure 5.17 illustre le pic de consommation qui atteint 800Mo à cause de la mémoire tampon d'images pour l'entrée du flux à 20 ips. Les résultats montrent l'impact du filtrage adaptatif avec l'heuristique constante et adaptative dans les figures 5.18 et 5.19 respectivement. Le pic de consommation est réduit à 80Mo et 60Mo respectivement. L'utilisation de la méthode de réduction des données permet à la méthode SLAM d'être exécutée en temps-réel lorsque aucun mouvement significatif n'est détecté, ce qui résulte en une basse fréquence du flux d'entrée des images. Lorsque le système est en mouvement, la bande passante d'entrée augmente. Ainsi, la mémoire tampon de l'image se remplit et se vide en FIFO (*First In, First Out*). La sélection des images est augmentée à des moments spécifiques de la séquence et est illustrée par les graphes de la norme de la vitesse angulaire détaillés dans le chapitre 3. Ces moments spécifiques correspondent au mouvement significatif du système et qui correspond au remplissage de la pile d'images injectées en entrée de la chaine de traitement.

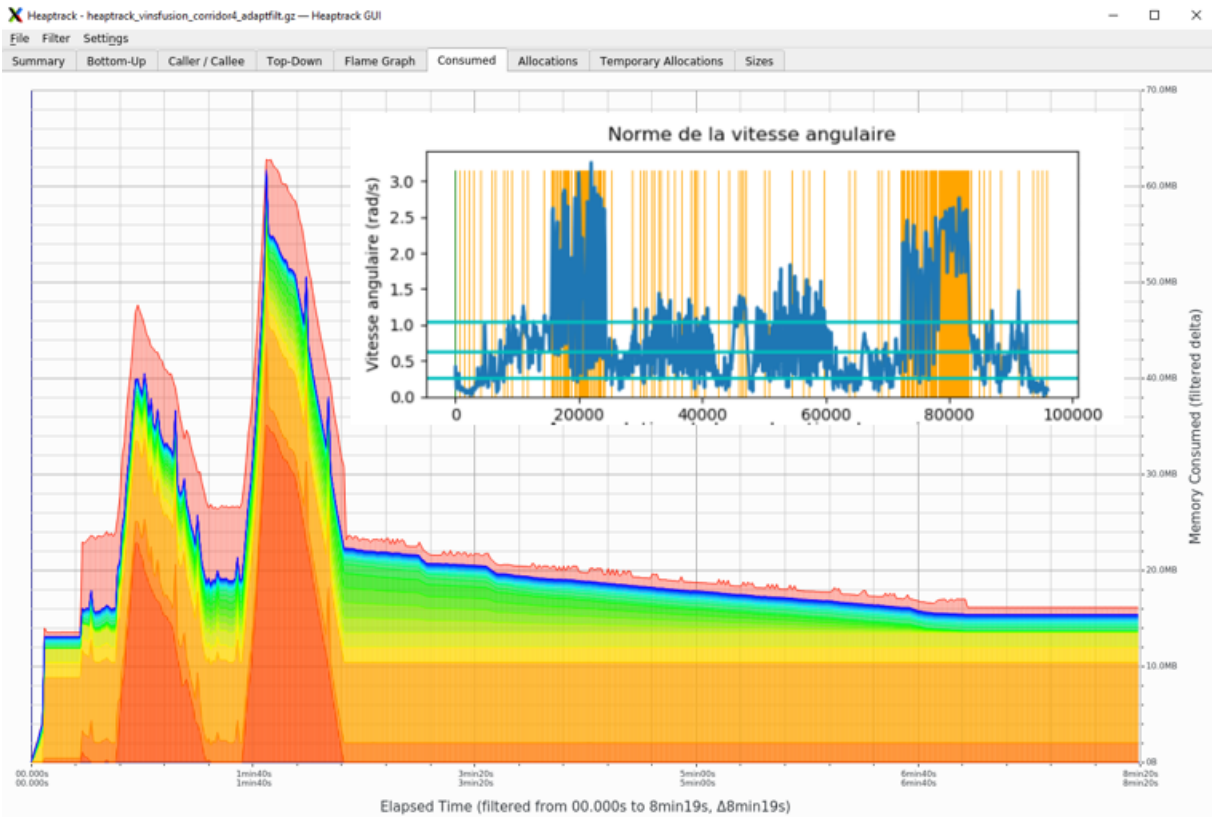


FIGURE 5.19 – Consommation de mémoire de l'algorithme VINSFusion sur *corridor4* avec le flux d'entrée adaptatif par l'heuristique AFadapt.

la figure 5.20 montre les résultats sous forme d'histogrammes de la taille de la mémoire tampon d'images présente dans VINSFusion avec la deuxième technique de réduction du temps d'exécution de l'algorithme. Pour cela, le temps d'exécution de la partie du traitement d'images qui inclut le *tracking KLT* a été augmenté de 100ms à 200ms avec une incrémentation de 20ms. Cela correspond à 6 analyses qui sont réparties par ligne. La première colonne correspond aux résultats de références sans modification des données. La deuxième et la troisième colonne correspondent aux résultats avec les heuristiques de l'AF constante et adaptative respectivement. Les graphes sont des histogrammes, ils montrent en abscisse la taille du pic de la mémoire tampon des images et en ordonnée l'occurrence des pics pour l'exécution de VINSFusion sur la séquence *corridor4*. La figure 5.20a montre que VINSFusion stocke en mémoire un maximum de près de 1000 images. La taille de la mémoire tampon est considérablement réduite pour l'AF constante dans la figure 5.20b et de plus de 10 fois avec l'heuristique adaptative 5.20c. L'occurrence du nombre d'images stockées montre que l'exécution en temps réel est atteint pour la majorité de la séquence où aucune image n'est stockée dans la mémoire tampon contrairement à la référence qui durant l'exécution a plus de 140 fois près de 1000 images dans la mémoire. Lorsque le temps d'exécution est augmenté à 200ms dans la dernière ligne du graphique, cette analyse n'est plus pertinente. La figure 5.20q et 5.20r montre que l'algorithme a besoin de stocker en majorité 120 à 180 images bien que cette contribution réduit considérablement l'utilisation de la mémoire tampon des images d'entrée par rapport à la référence 5.20p.

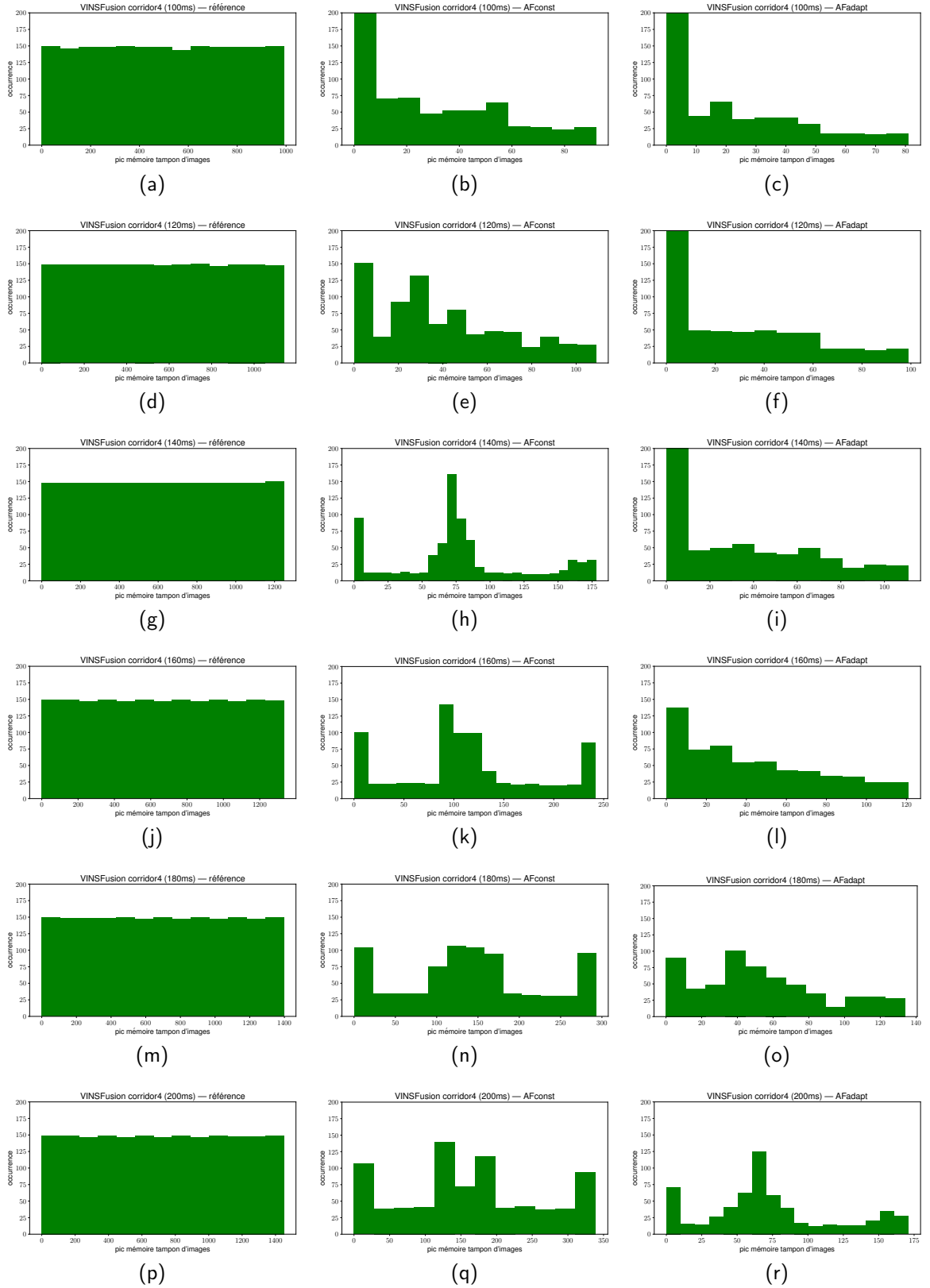


FIGURE 5.20 – Pic de la mémoire tampon d'images avec l'émulation de l'exécution sur système embarqué par la réduction du temps d'exécution avec de la gauche vers la droite : référence, AFconst, AFadapt et de la première à la dernière ligne : exécution à 100ms, 120ms, 140ms, 160ms, 180ms, 200ms.

La deuxième manière d'émuler les performances du SLAM sur système embarqué consiste à augmenter le flux d'entrée des images. La figure 5.21 montre le flux de base à 20 images dans la première ligne augmenté par 2, 3, 4 et 5 pour les lignes suivantes. L'histogramme est le même que l'analyse précédente sur l'augmentation du temps d'exécution. La première ligne montre que les résultats avec l'AF dans 5.21b et 5.21c réduit le nombre d'occurrences où 1 image se trouve dans la mémoire tampon dû à la décimation du nombre de données. La dernière ligne où le taux du flux d'entrée est multiplié par 5 montre l'impact du filtrage adaptatif par rapport à la référence. Dans 5.21n, le pic de la mémoire tampon est divisé par 10 et la plus forte occurrence du nombre d'images dans la mémoire reste inférieur à 10 images par rapport à la référence 5.21m.

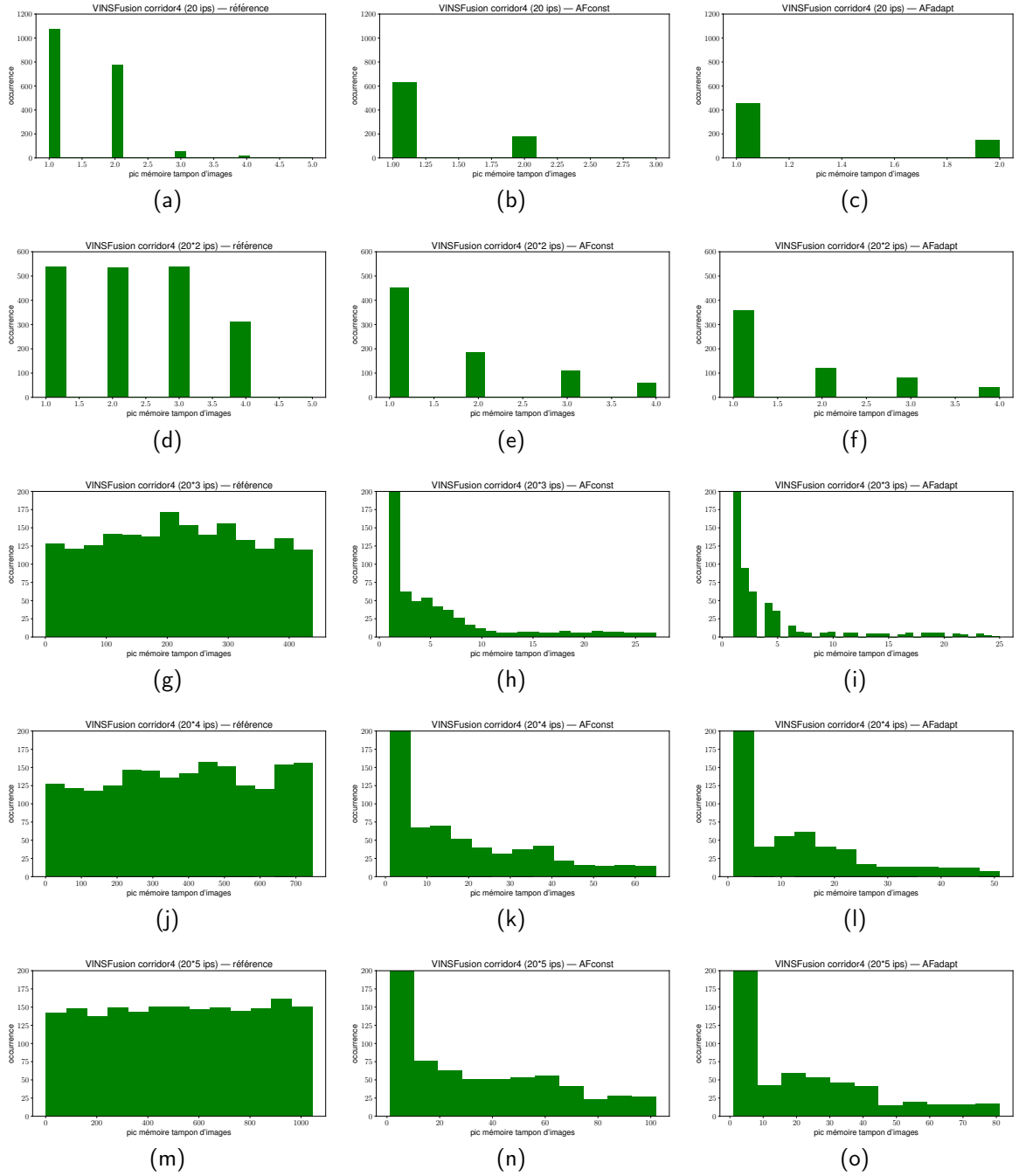


FIGURE 5.21 – Pic de la mémoire tampon d'images avec l'émulation de l'exécution sur système embarqué par l'augmentation du flux d'entrée. De la gauche vers la droite : référence, AFconst, AFadapt et de la première à la dernière ligne : exécution à 20 ips, 20*2 ips, 20*3 ips, 20*4 ips, 20*5 ips.

5.2.3. Discussion sur l'impact du filtrage adaptatif

Les expérimentations ont montré que le filtrage a un impact significatif sur la consommation mémoire de l'algorithme SLAM tout en assurant une erreur de localisation faible. Les travaux de l'état de l'art les plus alignés avec le mécanisme proposé concernent la méthode de sélection adaptative de *keyframes* (Piao and Kim, 2019). Le filtrage adaptatif répond à plusieurs limitations identifiées dans la partie 2.3 : 1/ les paramètres qui sont mis en place pour ajuster les seuils de déclenchement des images restent fixes selon les jeux de données. Cela réduit la complexité du filtrage adaptatif bien que ces paramètres peuvent être ajustés pour adapter au mieux le cas d'usage ciblé. 2/ Le temps d'exécution du filtrage adaptatif est négligeable par rapport à l'ensemble de la chaine de traitement. L'algorithme assure un traitement temps-réel tout en réduisant les ressources matérielles utilisées et impacte peu la qualité de performance des méthodes de perception. Dans certaines séquences, le mécanisme apporte une meilleure précision de la trajectoire reconstruite grâce à une sélection des images plus précise qui réduit la quantité de données aberrantes à filtrer. Pourquoi le contrôle du flux fonctionne pour les algorithmes SLAM ?

Les algorithmes introduisent différentes conditions pour déterminer si une image est une *keyframes* ou non. La condition principale porte sur le nombre de points correspondant entre la dernière KF et l'image actuelle. Si ce nombre est inférieur à un seuil prédéfini, l'image est considérée comme une *keyframe*. La précision du SLAM dépend alors fortement du mouvement du système. L'approche mise en place assure que chaque image injectée dans la chaine de traitement a une parallaxe suffisante avec la dernière image pour générer une nouvelle KF. Autrement dit, lorsque le système se déplace rapidement, l'AF injecte plus d'images que si le système se déplace lentement pour éviter que l'algorithme de localisation se perde tout en réduisant la quantité de données à traiter.

5.3. Etude sur le système vers la reconstruction 3D

Les sections 5.1 et 5.2 montrent l'impact des contributions sur la partie localisation du système hétérogène. Les conclusions montrent que le SLAM est robuste et précis par rapport à la référence avec des données d'entrée réduites de 50% à 75% pour la quantification de l'image. La méthode de quantification la plus appropriée au traitement d'image pour le cas d'usage du drone est la méthode *block-wise* qui réduit la quantité de donnée de manière significative suivant la taille du bloc. Avec un bloc de taille 4×4, la qualité de l'image est fortement réduite avec une réduction de donnée allant jusqu'à 75%. Malgré le bruit présent sur les bords où le gradient est élevé, la précision reste similaire à la référence à 8b par pixels. Elle offre alors la solution idéale pour une implémentation matérielle en flux comme il est indiqué dans la section 3.3 pour avoir un traitement temps-réel à faible complexité. La méthode de quantification *median cut* par la réduction de la dynamique de l'image avec la représentation à 4b est appropriée pour le cas d'usage du casque RA/RV représenté par la base de donnée TUMVI avec les capteurs tenus à la main. Elle permet aux algorithmes SLAM d'être robuste aux mouvements brusques et aux longues séquences tout en réduisant la quantité de donnée à traiter.

Les résultats sur la précision de la localisation et l'impact sur la réduction des besoins en ressources de la méthode de filtrage adaptatif montrent que cette solution s'adapte aux mouvements du système pour réaliser les calculs seulement lorsque des mouvements significatifs sont détectés. Cela permet

au SLAM de traiter un minimum d'images tout en assurant la précision de la *keyframe* générée qui est similaire à la référence, soit lorsque le flux est constant à 20 ips. Cette méthode correspond aux cas d'usages visés dans la thèse, les drones et les casques à réalité RA/RV. En ajoutant les données inertielles au mécanisme d'optimisation avec le filtrage adaptatif, l'heuristique constante réduit la quantité de donnée jusqu'à 75% tout en assurant une précision similaire à la référence pour les séquences difficiles capturées par le drone. Les expérimentations sur le filtrage adaptatif montrent que la technique permet d'obtenir une réduction d'environ 50% de donnée pour les séquences de TUMVI tout en ayant une précision similaire à la référence correspondant à un flux d'entrée constant de 20 images par seconde.

Les résultats confirment que les contributions proposées et détaillées dans le chapitre 3 ont un impact positif sur la réduction des besoins en ressources matérielles pour les algorithmes de type SLAM et notamment sur la partie localisation temps-réel. A quel point peut-on quantifier l'image et réduire le flux d'entrée pour obtenir une précision de la reconstruction 3D de type voxel similaire à la référence ? C'est à cette question que l'on répond dans les sections 5.3.1 et 5.3.2. La section 5.3.1 détaille les expérimentations menées sur la précision de reconstruction obtenue avec chacune des contributions indépendamment de l'autre et la section 5.3.2 évalue le système proposé du chapitre 3 avec l'association des mécanismes d'optimisation, soit la quantification de l'image et l'optimisation du flux d'entrée sur la précision et la qualité de la reconstruction 3D.

5.3.1. Module de réduction des données

L'évaluation de la reconstruction 3D se fait de manière qualitative et quantitative. Les métriques utilisées sont détaillées dans le chapitre 4.1.2. Pour simplifier l'analyse des expérimentations sur la reconstruction 3D avec les méthodes embarquables et dont les contributions de la thèse apportent un gain significatif, seule la méthode VINSFusion (Qin et al., 2019) est utilisée pour obtenir la pose 3D de la caméra associée avec la méthode de reconstruction 3D Voxblox (Oleynikova et al., 2017). Voxblox prend également en entrée les données issues de l'estimation de profondeur de l'image. L'algorithme *block matching* implémenté dans OpenCV est utilisé car il permet d'avoir une exécution en temps-réel de l'estimation de profondeur tout en assurant une précision suffisante pour la reconstruction 3D. Ce calcul est de l'ordre de 27ms en moyenne. Les expérimentations sont effectuées avec la séquence drone *V101* de la base de donnée Euroc. La figure 5.22 montre la référence qualitative avec une dynamique complète de l'image à 8b par pixel et le flux constant à 20 ips. La reconstruction basée voxel en gris génère le maillage surfacique issu de Voxblox et la trajectoire du drone de VINSFusion est illustrée en vert. Les paramètres de l'algorithme Voxblox permettant d'obtenir la reconstruction temps-réel sont utilisés et on remarque plusieurs trous dans la référence. Par rapport à la vérité terrain, la reconstruction a une erreur de 0.49m avec une complétude de 70% détaillé dans la table 5.2 sur la première colonne 8bpp. Ces mesures servent de référence pour l'analyse de l'impact de la réduction de la dynamique et de l'optimisation du flux d'entrée sur la reconstruction 3D.

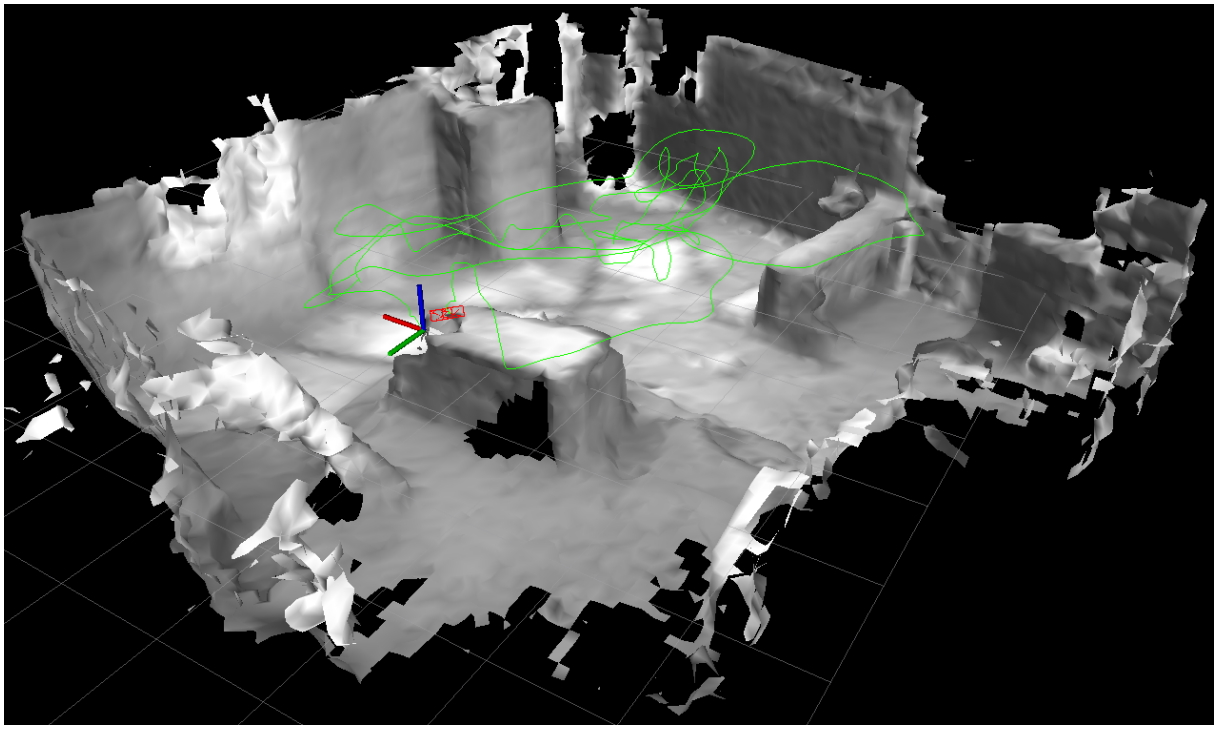


FIGURE 5.22 – Reconstruction 3D de référence avec la précision 8b/px de l'image et le flux constant à 20 ips.

TABLE 5.2 – Précision RMSE, mesure de la zone de recouvrement *fitness* (# de points ayant une correspondance / # de points de l'estimation) et nombre de points ayant une correspondance de l'estimation vers la référence *nb_match* des reconstructions 3D avec le mécanisme de réduction de la dynamique de l'image.

	8bpp	med7	med6	med5	med4	med3	med2	block4	block8	med1	block16
↓ RMSE (m)	0.49	0.51	0.48	**0.46**	0.47	**0.46**	**0.46**	**0.46**	0.47	0.52	**0.46**
↑ fitness (#)	0.42	0.42	0.42	0.42	0.41	0.42	0.42	0.46	0.41	**0.46**	0.45
↑ nb_match (#)	1359462	1361527	1360564	1355738	1311817	1343124	1346655	**1475608**	1321902	1469496	1461792
↑ complétude (%)	70.00	70.91	68.33	66.67	70.00	69.09	67.27	75.00	**83.33**	65.45	67.69

La table 5.2 montre que la reconstruction 3D est plus précise en RMSE avec la quantification de l'image suivant la configuration *block16* avec 0.46m d'erreur. Lorsque l'on met la précision qualitative de la figure 5.23 en perspective à la métrique RMSE, la reconstruction est bruitée et ne permet pas d'avoir une représentation précise des éléments qui entourent le système. La métrique RMSE prend en compte dans le calcul seulement les points 3D ayant une correspondance avec la vérité terrain. Dans ce cadre, la métrique *nb_match* doit être prise en considération. La table 5.2 montre que c'est avec la configuration *block4* que l'on obtient une précision similaire à la configuration *block16* avec le plus haut nombre de point correspondant à la vérité terrain pris en compte dans le calcul de la précision. La figure 5.24 représente l'estimation de la reconstruction 3D avec la quantification d'image par bloc. La représentation 3D permet d'avoir une reconstruction détaillée de la scène où les éléments principaux sont visibles et reconnaissables par rapport à la référence.

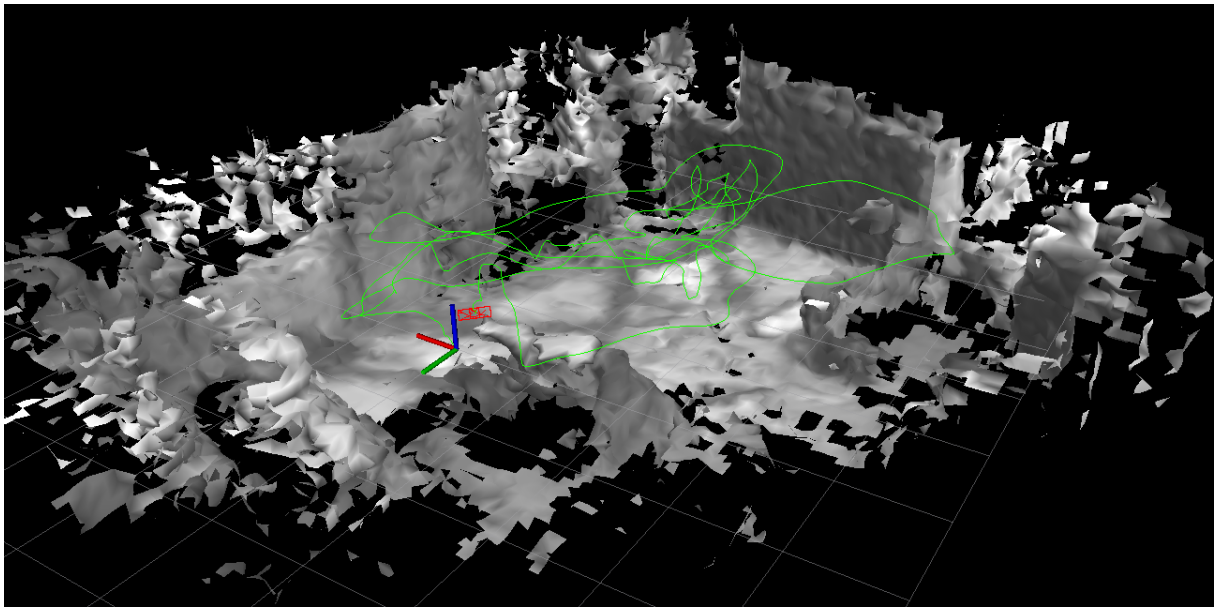


FIGURE 5.23 – Reconstruction 3D avec réduction de la dynamique de l'image par la méthode *block16* et le flux constant à 20 ips.

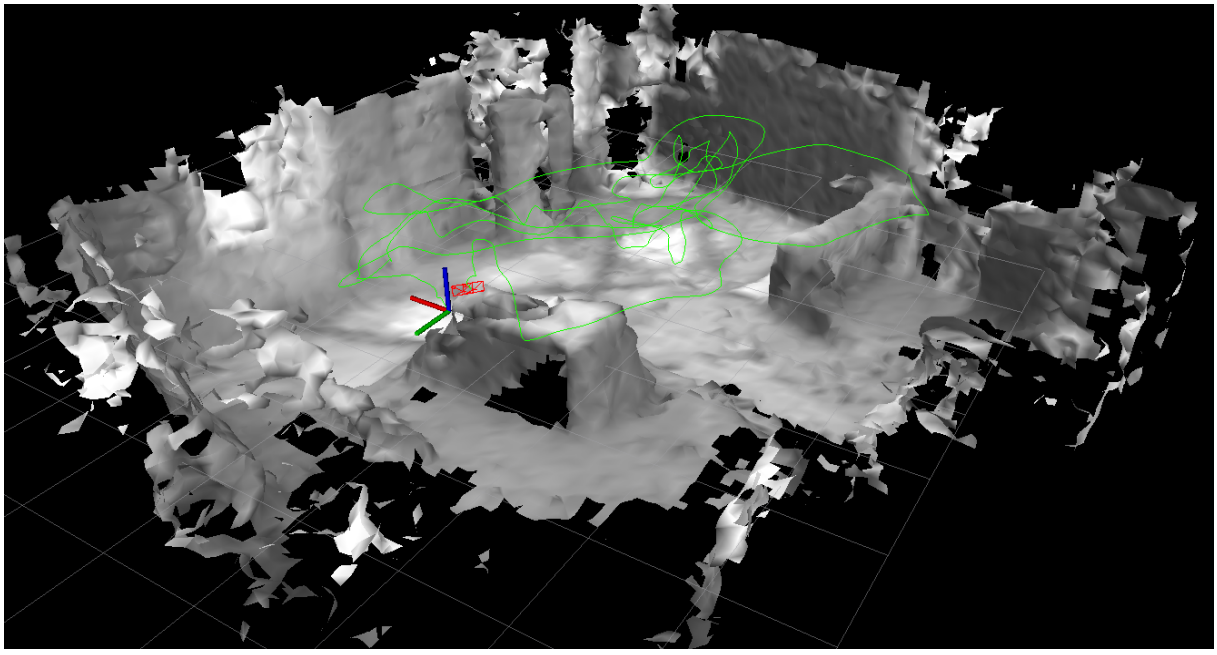


FIGURE 5.24 – Reconstruction 3D avec réduction de la dynamique de l'image par la méthode *block4* et le flux constant à 20 ips.

Quant à la quantification de l'image par la technique de *median cut*, la qualité de la reconstruction 3D est moins bonne comme c'est illustré dans la figure 5.25. La représentation à 1b par pixel montre que la précision de l'estimation de profondeur de l'image impacte directement la qualité de la reconstruction 3D. Avec la quantification *med1* qui représente un pourcentage de réduction de donnée similaire à *block16*, il y a trop peu de texture dans l'image. Cela rend la précision de l'estimation de profondeur de l'image approximative et ne permet pas d'obtenir une représentation 3D de la scène robuste. Malgré ce manque de textures, la reconstruction avec la quantification *median cut* la plus élevée est prometteuse, on retrouve certains aspects de la reconstruction permettant de situer où sont les obstacles dans l'environnement, mais trop faible en complétude pour pouvoir pleinement exploiter la reconstruction 3D. C'est avec la représentation *med4* de l'image que l'on atteint une précision suffisante de la reconstruction tout en réduisant la quantité de donnée à injecter, comme c'est illustré dans la figure 5.26. Par rapport à la référence 8b, tous les éléments pertinents de la scène sont représentés et identifiables pour le système mobile malgré la présence de trous.

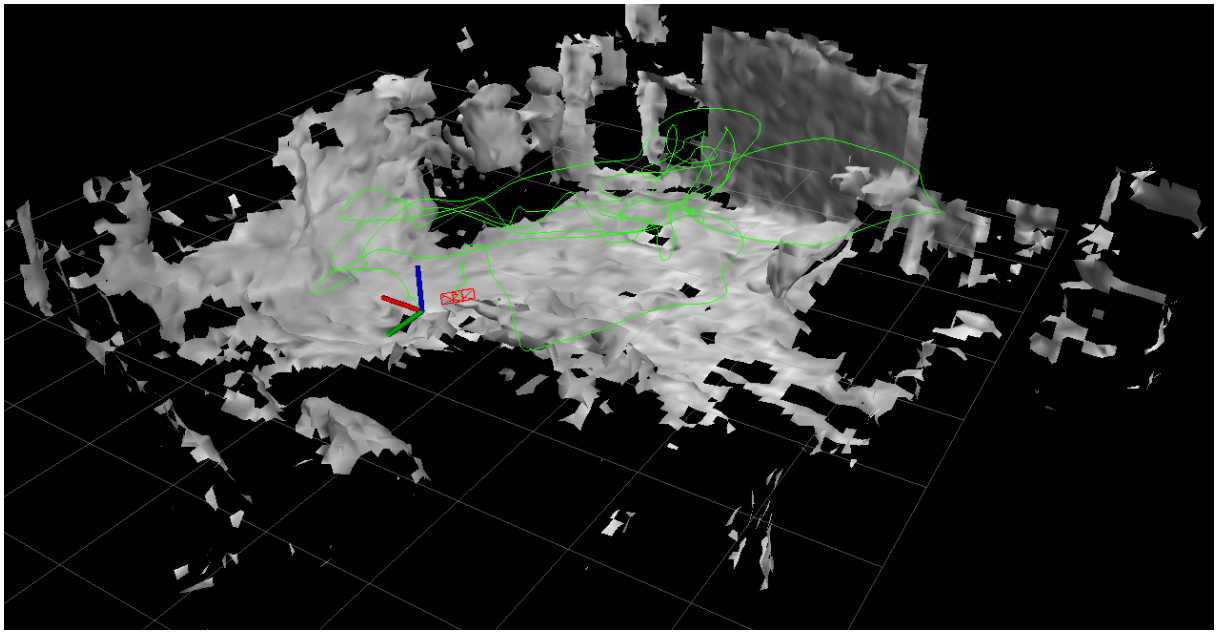


FIGURE 5.25 – Reconstruction 3D avec réduction de la dynamique de l'image par la méthode *med1* et le flux constant à 20 ips.

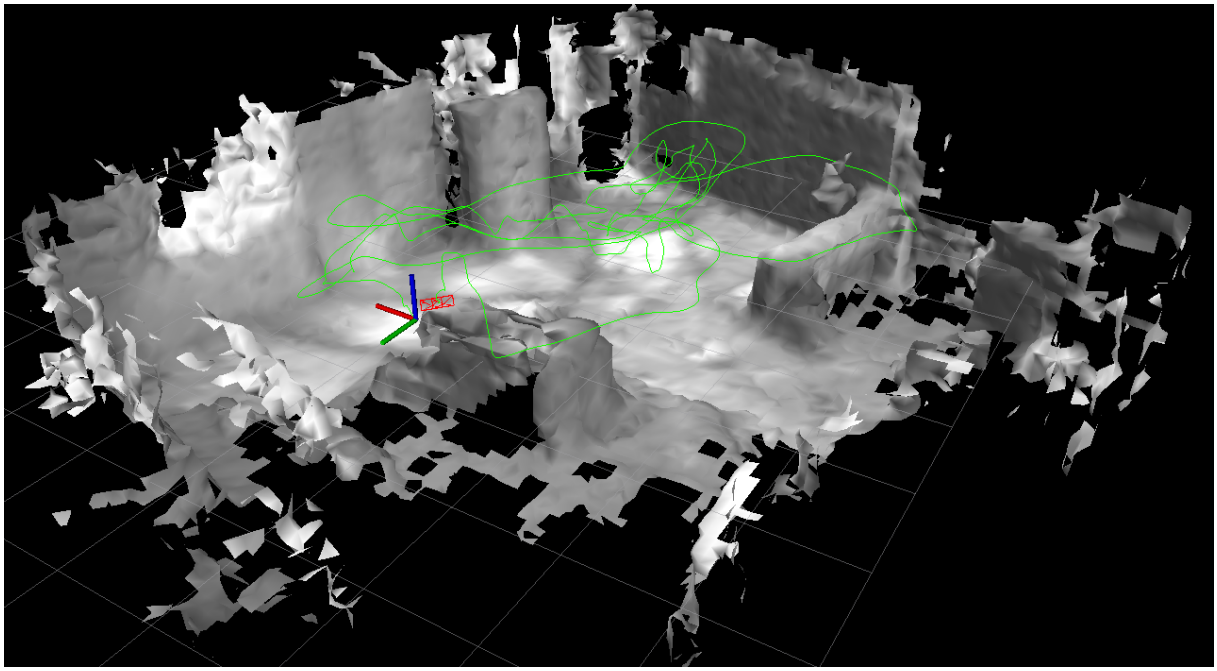


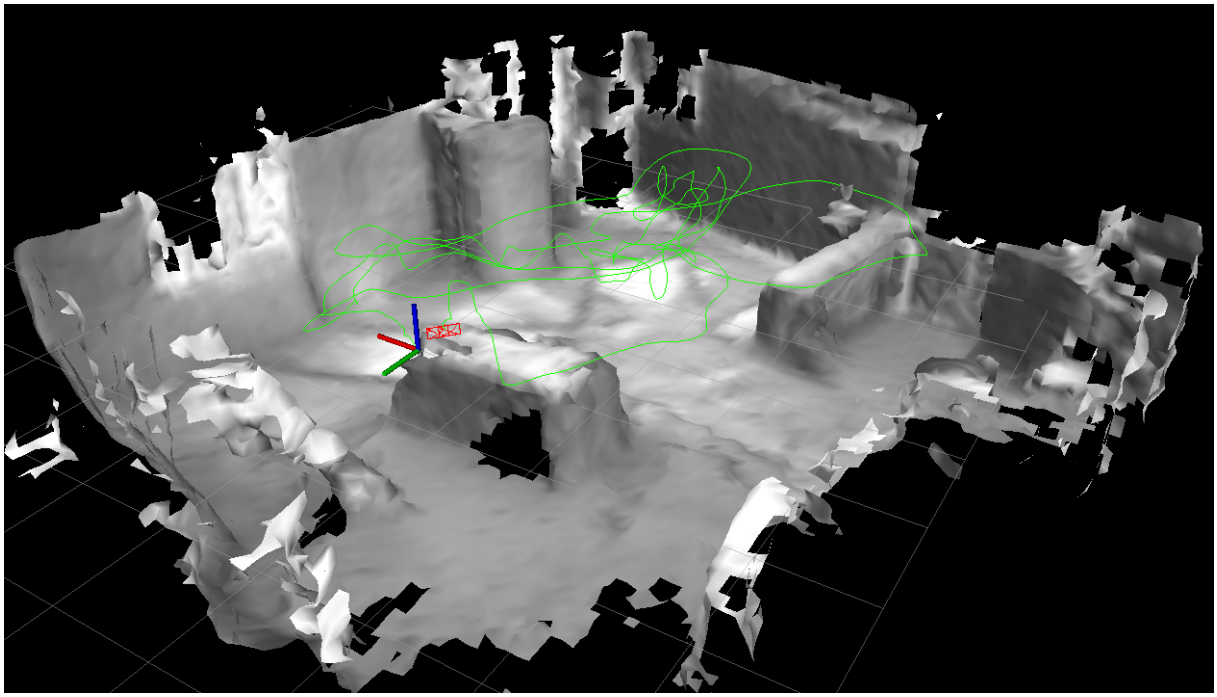
FIGURE 5.26 – Reconstruction 3D avec réduction de la dynamique de l'image par la méthode *med4* et le flux constant à 20 ips.

TABLE 5.3 – Précision RMSE, mesure de la zone de recouvrement *fitness* (# de points ayant une correspondance / # de points de l'estimation) et nombre de points ayant une correspondance de l'estimation vers la référence *nb_match* des reconstructions 3D avec le mécanisme de contrôle du flux d'images par le filtrage adaptatif.

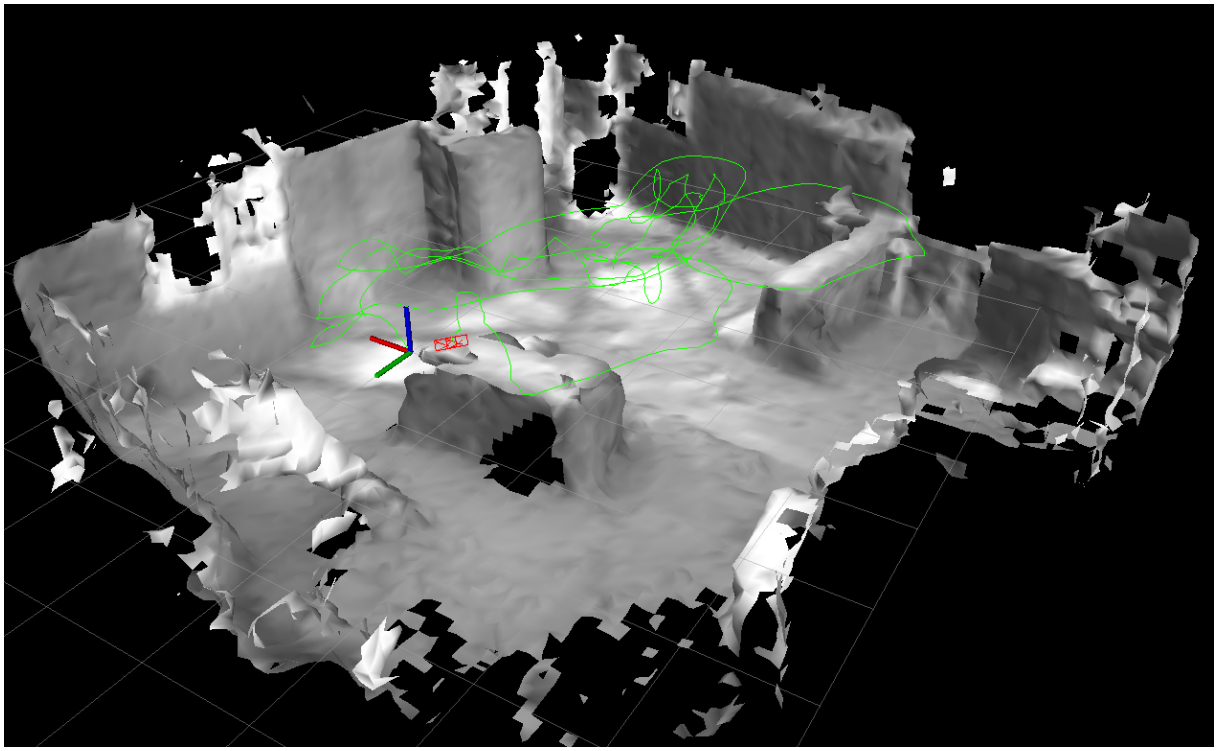
	8bpp	AFconst	AFadapt
↓ RMSE (m)	**0.49**	0.51	**0.49**
↑ fitness (#)	0.42	0.42	**0.43**
↑ nb_match (#)	1359462	1351214	**1379639**
↑ complétude (%)	70.00	**70.91**	67.27

La table 5.3 détaille la précision de la reconstruction 3D avec la technique de filtrage adaptatif où le flux d'entrée est contrôlé en fonction des mouvements du système. La précision RMSE et le nombre de points correspondant montrent que le filtrage adaptatif avec l'heuristique *adapt* obtient une précision similaire à la référence, mais avec une complétude plus faible qu'avec l'heuristique *const*. La figure 5.27 montre la précision qualitative de la reconstruction 3D suivant les deux heuristiques utilisées, constante et adaptative sur les figures 5.27a et 5.27b respectivement. La reconstruction 5.27a est plus complète qu'avec l'heuristique adaptative avec une précision similaire à la référence. Le contrôle du flux d'image n'affecte pas la qualité de la reconstruction 3D contrairement à la réduction de la dynamique de l'image qui influe directement sur la qualité de l'estimation de profondeur en réduisant la complétude de la reconstruction ou en ajoutant du bruit dans la représentation 3D de la scène.

La suite des expérimentations sert à mettre en place le système hétérogène proposé dans le chapitre 3 à partir des mécanismes d'optimisation des données qui apportent le compromis entre la réduction des besoins en ressources matérielles et la précision suffisante pour avoir un système précis et robuste. Les méthodes *med4* et *block4* sont choisies pour le traitement dans le capteur et la méthode de contrôle du flux *AFconst* pour le traitement proche capteur à partir des images quantifiées. La précision qualitative et quantitative de la reconstruction 3D est évaluée à partir du système mis en place sous forme de nœuds avec le logiciel ROS. Le système hétérogène proposé dans les contributions de la thèse est implémenté avec le traitement dans le capteur par la réduction de la dynamique, le traitement proche capteur avec le filtrage adaptatif prenant en entrée les images quantifiées et injecte dans le système SLAM le flux d'images suivant les mouvements du système. Le système SLAM est représenté dans la plateforme embarquée avec le module de localisation où est utilisé VINSFusion, le module d'estimation de profondeur avec l'algorithme *block matching* et le module de reconstruction 3D avec l'algorithme Voxblox.



(a)



(b)

FIGURE 5.27 – Reconstruction 3D avec contrôle du flux d'images par le filtrage adaptatif suivant les deux heuristiques AFconst (a) et AFadapt (b).

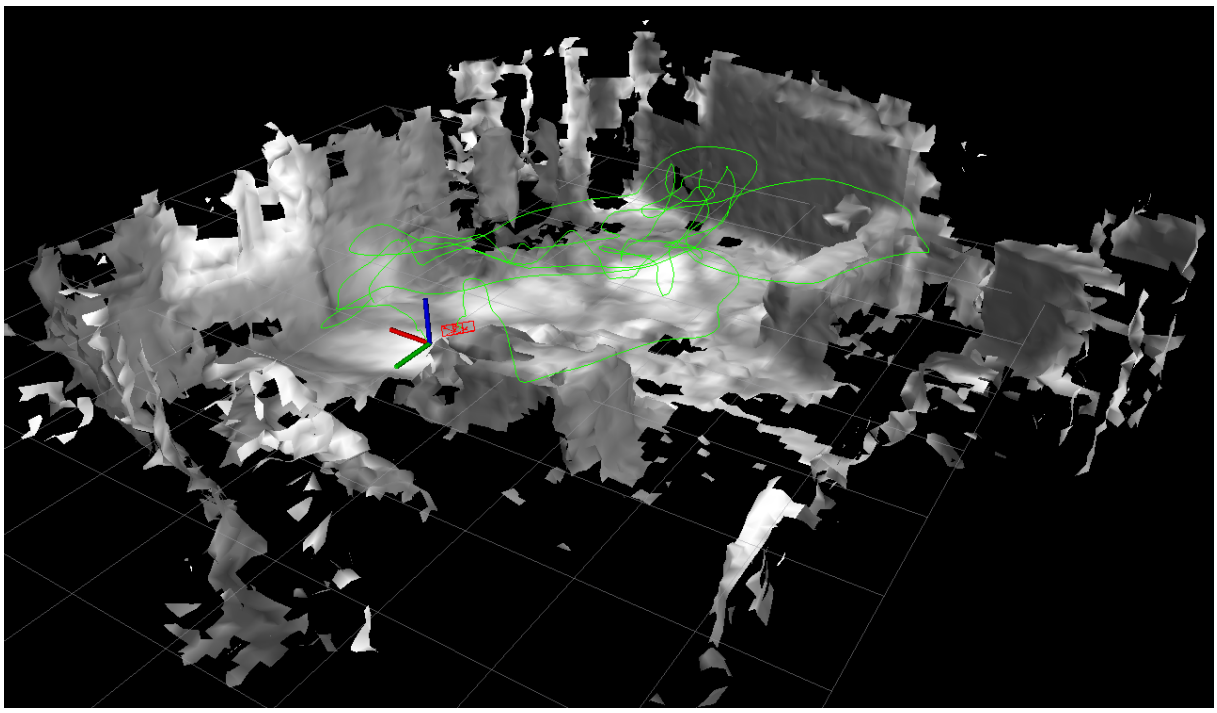
5.3.2. Association des mécanismes d'optimisation des données

L'association des mécanismes d'optimisation des données prend en compte deux configurations exploitables pour les deux bases de données utilisées dans les expérimentations pour la chaine de traitement de la reconstruction 3D. La première se compose de la technique *median cut* avec la représentation 4b de l'image associée avec la méthode de filtrage adaptative suivant l'heuristique constante. La configuration *med4* obtient le meilleur compromis pour les algorithmes SLAM en termes de robustesse, de précision et d'optimisation pour réduire les besoins en ressources pour le cas d'usage lié aux casques à réalité RA/RV et capteurs portés à la main comme dans la base de donnée TUMVI. La deuxième configuration se compose de la technique de quantification *block4* avec la même heuristique constante de l'AF. Cette réduction de la dynamique est adaptée au cas d'usage du drone où les expériementations ont été réalisées avec la base de donnée Euroc.

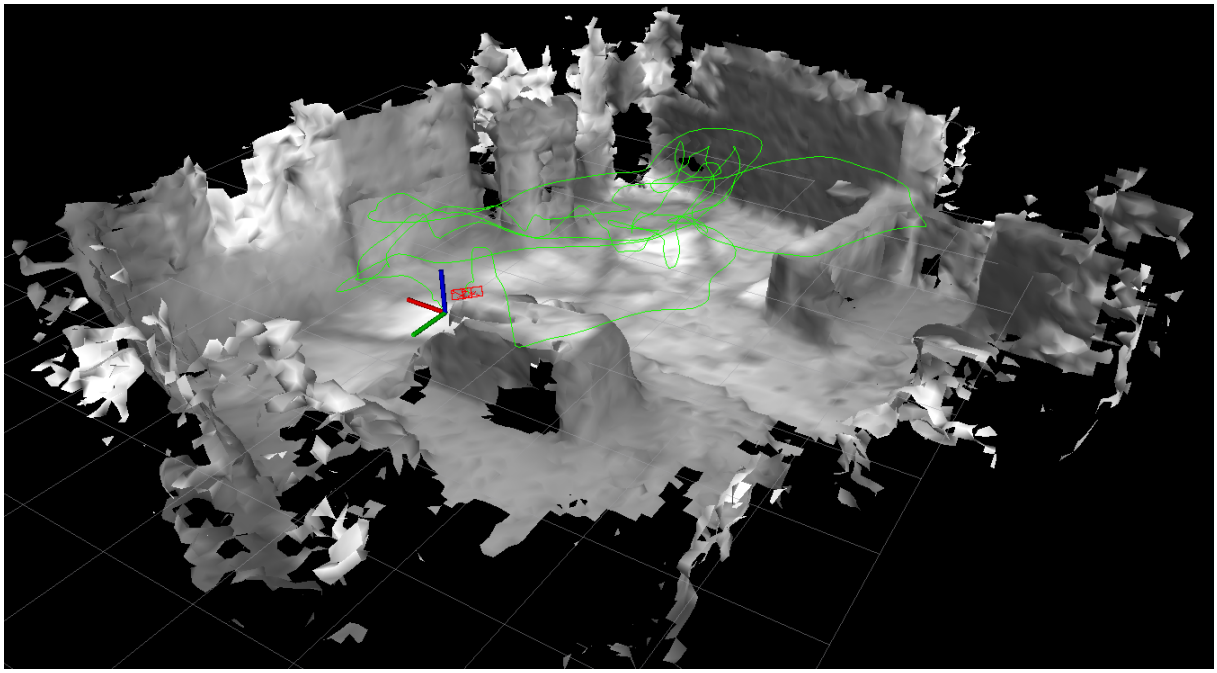
TABLE 5.4 – Résultats des performances de localisation et de reconstruction 3D du système hétérogène en associant les mécanismes d'optimisation des données *med4+AFconst* et *block4+AFconst* pour les cas d'usage TUMVI et Euroc respectivement.

	localisation VINS-Fusion		**reconstruction 3D Voxblox**	
	med4+AFconst	**block4+AFconst**	**med4+AFconst**	**block4+AFconst**
↓ RMSE (m)	0.10	0.10	**0.46**	0.47
↑ fitness (#)	-	-	0.43	0.43
↑ nb_match (#)	-	-	**1375354**	1369531
↑ completude (%)	-	-	68.33	**71.67**

La table 5.4 montre les métriques pour la reconstruction allant de la précision RMSE à la complétude la reconstruction 3D avec l'association des mécanismes d'optimisation d'injection des données. La précision est similaire entre les deux configurations pour la reconstruction Voxblox. C'est la métrique de complétude qui diffère de près de 3%. La figure 5.28 montre la précision qualitative des reconstructions 3D suivant les deux configurations avec *med4+AFconst* dans 5.28a et *block4+AFconst* dans 5.28b. La figure montre bien la différence de complétude entre les deux reconstructions. La configuration *med4* ne permet pas de reconstruire la scène dans sa totalité à cause du manque de texture présent dans l'image d'entrée pour l'estimation de profondeur. Dans la configuration *block4*, la reconstruction est plus complète et plus précise qualitativement où l'on reconnait les éléments pertinents de la scène. Cette reconstruction apporte du bruit dans la représentation de la scène, notamment sur les bords.



(a)



(b)

FIGURE 5.28 – Reconstruction 3D du système mis en place avec la réduction de la dynamique de l'image intégré au capteur et du contrôle du flux à injecter dans le traitement proche capteur avec les méthodes *med4+AFconst* (a) et *block4+AFconst* (b).

Globalement, la reconstruction avec les deux configurations permet d'avoir une représentation globale de la scène avec l'image quantifiée et le flux d'image adaptatif. Les expérimentations montrent un résultat contre-intuitif. En effet, l'estimation de profondeur est le paramètre qui détermine la qualité de la reconstruction 3D. Il y a peu de texture présente dans l'image quantifiée, la qualité de l'estimation de profondeur de l'image est dégradée pour l'algorithme de reconstruction 3D. Bien que les résultats montrent que la représentation perd en précision et en robustesse, le contenu de la scène est présent et exploitable pour des fonctions de plus haut niveau, comme la détection d'obstacles et l'interaction avec les objets de la scène.

5.4. Discussions de l'étude système

Les contributions proposées dans la thèse permettent de gagner sur trois principaux points de la chaine de traitement tout en assurant une exécution en temps-réel et une précision de localisation et de reconstruction 3D similaire à la référence : 1/ la consommation de mémoire, 2/ la bande passante et 3/ la consommation d'énergie.

TABLE 5.5 – Gain en pourcentage moyen de la quantité de *keyframes* pour chacunes des séquences utilisées.

	KimeraVIO	ORBSLAM3	VINSFusion	OpenVINS
Vicons	24.65%	3.47%	73.00%	72.69%
Machine Hall	24.43%	5.59%	80.57%	79.62%
Room	-	2.41%	42.38%	38.59%
Corridor	-	0.00%	54.61%	43.86%

Plusieurs étapes sont prises en compte dans la consommation mémoire des algorithmes SLAM : 1/ le type de points d'intérêts à extraire dans l'image d'entrée, 2/ la description de ces points caractéristiques, 3/ la quantité de points suivis à garder en mémoire, 4/ le nombre de *keyframes* générées, 5/ la densité de la reconstruction 3D et la stratégie d'optimisation du *back-end* mise en place à partir des *keyframes*. L'association des mécanismes d'optimisation mise en place dans la chaine de traitement permet d'influer sur tous ces aspects d'un point de vue mémoire et notamment dans le contexte embarqué. Les expérimentations montrent qu'en émulant les performances des algorithmes sur système à ressources restreintes, l'espace mémoire utilisé est réduit jusqu'à 92% lorsque l'algorithme met en place une mémoire tampon pour le stockage des images. La table 5.5 détaille le gain en pourcentage de la quantité de *keyframes* pour chacune des méthodes SLAM en fonction des bases de données utilisées. C'est avec la méthode VINSFusion que la quantité de pose 3D générée est considérablement réduite jusqu'à 80% de *keyframes* en moins pour les séquences *Machine Hall*. Pour autant, les expérimentations menées avec les images quantifiées et le contrôle du flux d'entrée montrent que la précision de la localisation est similaire à la référence. Les algorithmes SLAM n'ont pas besoin d'un flux constant avec une image en pleine précision. Les informations issues des caméras visibles sont redondantes et réduire cette redondance permet de considérablement optimiser les besoins en ressources des algorithmes.

TABLE 5.6 – Apport des contributions proposées par rapport à la référence (Liu et al., 2019) sur la puissance consommée de l'algorithme SLAM avec la séquence *V101* de la base de donnée Euroc.

Intel i7		ARM Cortex-A9	
référence	contributions	référence	contributions
137 kW	**28 kW**	5 kW	**934 W**

La bande passante est fortement optimisée sur deux points : le calcul de détection de points d'intérêts à partir de l'encodage des pixels réduits jusqu'à une représentation 4b et par l'injection des images dans la chaine de traitement de manière adaptative en fonction du mouvement du système. Le nombre d'opérations effectué est réduit principalement sur le traitement de l'image. Grâce au contrôle du flux d'image le gain est significatif dans le temps où les contributions permettent de traiter moins d'images. La consommation d'énergie est directement réduite par la diminution du nombre d'opérations effectuées. Par exemple, l'algorithme ORBSLAM a une consommation de puissance sur processeur Intel i7 et sur ARM Cortex-A9 de 47W et 1.57W respectivement avec une consommation d'énergie par image de 2519mJ et 875mJ et le temps d'exécution correspondant de 53.6ms et 555.7ms (Liu et al., 2019). En prenant en compte ces données comme référence avec la base de donnée de 2912 images (*Euroc V101*), la puissance consommée à la fin de la séquence correspond à 137kW et 5kW pour i7 et ARM respectivement. La table 5.6 montre un gain d'environ 80% de puissance consommée couvrant toute la séquence du drone avec les contributions proposées où l'on traite seulement 595 images tout en ayant une précision de localisation et de reconstruction similaire à la référence.

Les mécanismes d'optimisation mis en place permettent de contribuer considérablement dans la réduction des besoins en ressource des algorithmes SLAM. Les expérimentations menées montrent que les méthodes développées ont plusieurs limitations. Les résultats varient d'un cas d'usage à un autre ce qui rend les analyses et les conclusions complexes. Par exemple, les deux cas d'usages ciblés dans la thèse correspondant aux deux bases de données Euroc et TUMVI sont différents l'un de l'autre et les expérimentations montrent à quel point les résultats varient pour les séquences plus difficiles que les autres. Dans le cas où le système ne cesse d'être en mouvement, les mécanismes d'optimisation permettent de réduire considérablement la quantité de donnée à injecter. Au contraire, pour un cas d'usage où il y a très peu de mouvement rotatif, voire aucun, la contribution sur la réduction de la dynamique est la seule qui permet de réduire la complexité calculatoire des algorithmes de traitements d'image. Le contrôle du flux dépend fortement du mouvement du système, si aucun mouvement n'est détecté, ce mécanisme n'est pas efficace. Les résultats sur la reconstruction 3D montrent également certaines limitations quant à l'utilisation de la réduction de la dynamique de l'image. Ce mécanisme dégrade fortement la qualité de l'estimation de profondeur de l'image et impacte la précision de l'algorithme de reconstruction 3D. Cela engendre dans la représentation de la scène des manques de textures et du bruit. Ces limitations sont discutées dans le chapitre 6 avec les perspectives sur les propositions de mécanismes d'optimisation et sur le domaine applicatif dans l'objectif de réduire et d'optimiser le nombre d'opérations effectuées pour diminuer la consommation d'énergie des algorithmes SLAM.

6 - Conclusion et perspectives

Conclusion

Les recherches réalisées durant la thèse apportent des éléments de réponse aux deux principales questions : comment réduire les besoins en ressources des fonctions de perception de la chaine de traitement SLAM allant de la localisation à la reconstruction 3D en temps-réel ? Quel est le partitionnement des étages de calculs pour le système hétérogène intégrant les plateformes embarquées allant de la conception matérielle (ASIC, FPGA) à l'implémentation des algorithmes sur COTS (ARM, GPU embarqué) ?

Les contributions proposées répondent à la première question par l'optimisation des données à injecter issues des caméras visibles. A partir de ces mécanismes, un système hétérogène est proposé et discuté en utilisant différentes unités de calculs. L'optimisation des données est réalisée par deux techniques, la réduction de la dynamique de l'image et le contrôle du flux par le filtrage adaptatif. L'impact des méthodes sur les algorithmes de type SLAM est évalué par la précision de la localisation et de la reconstruction 3D basée voxel. L'un des principaux objectifs est alors de déterminer le compromis entre la qualité de l'estimation et la réduction de la consommation des ressources matérielles tout en assurant l'exécution des algorithmes en temps-réel. Les performances varient suivant le cas d'usage visé. La thèse prend en compte les bases de données représentant la trajectoire d'un drone et d'un casque à réalité mixte. Les expérimentations menées montrent que les données pixels peuvent être quantifiées jusqu'à 50% ce qui représente le pixel à 4 bits pour le cas d'usage du casque à réalité mixte et jusqu'à 75% pour le drone ce qui correspond à 2 bits/pixel. L'utilisation de la quantification par la méthode *median cut* réduit l'encodage des pixels jusqu'à 2 bits tout en ayant une erreur de localisation faible pour certaines séquences du jeu de données Euroc. Les performances du système SLAM donnent une précision similaire à la référence qui est 8bits/pixel avec les données quantifiées. Les résultats de ces expérimentations sont justifiés par l'utilisation d'algorithmes de détection de points d'intérêt binaires. En quantifiant l'image de manière optimale, la quantité de données aberrantes est réduite, ce qui entraine une association des données plus robuste. Bien que la précision et la robustesse de la trajectoire suivant la quantification d'image ne soit pas impactée, la qualité de la reconstruction 3D est plus bruitée avec une complétude moins élevée que la référence. Pour autant, le compromis entre les performances matérielles et algorithmiques est atteint lorsque les éléments pertinents de la représentation 3D de la scène sont exploitables pour des fonctions haut-niveaux, comme la détection d'obstacle ou l'interaction avec l'environnement ambiant.

Avec le contrôle du flux d'images, la quantité de données à traiter dépend du mouvement du système et convient à l'architecture des algorithmes SLAM. En effet, la précision du SLAM dépend des optimisations de pose générées par les *keyframes*. Le mécanisme mis en place déclenche un nombre variable d'images en fonction de la vitesse de déplacement du système. Plus la vitesse est élevée, plus le nombre d'images déclenchées est important. Le traitement en temps-réel de la chaine de perception est assuré avec un temps d'exécution négligeable pour l'algorithme d'optimisation, tout en réduisant de manière significative la complexité calculatoire des méthodes de perception. Le filtrage adaptatif décime jusqu'à 80% des images tout en assurant une erreur de localisation et de reconstruction faible

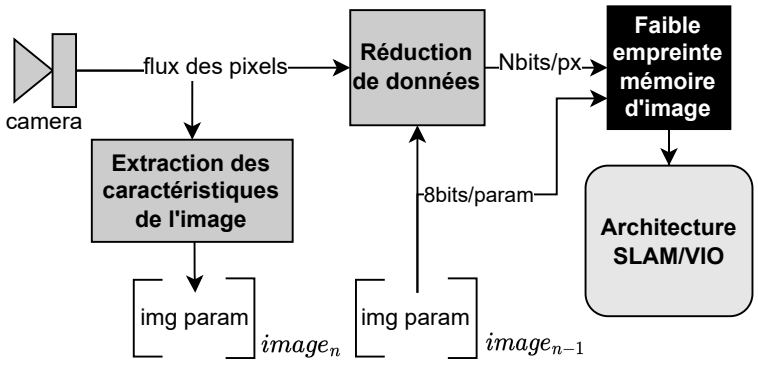


FIGURE 6.1 – Réduction de la dynamique des pixels avec un traitement en flot de données pour les algorithmes SLAM.

et similaire à la référence. L'étude de l'impact mémoire dans le contexte embarqué montre que le pic de consommation est réduit jusqu'à 92%. Le système complet gagne alors sur plusieurs points : 1/ la latence avec la co-conception logicielle/matérielle pour l'implémentation des mécanismes dans le système hétérogène, 2/ sur la consommation mémoire où l'impact se place sur la quantité d'images à traiter, des points d'intérêts qui en ressortent et sur le nombre de *keyframes* générées, 3/ sur la bande passante avec un contrôle du flux d'images variable prenant en compte une quantification allant jusqu'à 2 bits par pixel et 4/ sur la consommation d'énergie qui permet de gagner 80% d'énergie consommée sur la séquence de drone *Euroc V101*.

Perspectives

Les mécanismes proposés ont des limitations qui se caractérisent par deux principaux points, le cas d'usage visé et la qualité de la reconstruction 3D. Ainsi, les travaux apportent plusieurs perspectives :

1/ L'implémentation de la méthode de réduction de la dynamique de l'image pour impacter les caractéristiques du capteur. La technique *median cut* est complexe à implémenter en flux contrairement à la méthode *block-wise* et permet d'obtenir une meilleure précision sur certains jeux de données utilisés. La première perspective a pour but de réduire sa complexité, la perspective est d'appliquer à l'image actuelle les paramètres calculés par l'algorithme dans l'image précédente comme le montre la figure 6.1. Cela réduit l'empreinte mémoire requis par la réduction *median cut* et permet de traiter les pixels en flux. La deuxième perspective vise à rendre la quantification adaptative en fonction de la pertinence des éléments de la scène. Dans le traitement pixel par pixel, la quantification n'est pas fixe, mais adaptative. Plus l'information dans l'image est importante, plus la quantification est élevée. L'un des points potentiel à prendre en compte est d'effectuer les calculs sur les régions où la quantification est élevée sans réaliser de calculs sur les zones de l'image où la quantification est faible. La principale question à aborder sont : comment reconnaitre les éléments pertinents de la scène ?

2/ La technique de contrôle du flux par le filtrage adaptatif (AF) est dépendant du cas d'usage. Par exemple, les drones et les casques à réalité mixte sont des systèmes qui peuvent présenter une forte accélération angulaire. Ils sont donc adaptés à l'utilisation du filtrage adaptatif. Dans d'autres systèmes où les vitesses de rotation sont faibles, l'utilisation de l'AF est limitée. Comment déclencher les images lorsqu'il n'y a pas de mouvements significatifs ? Aujourd'hui, plus il y a de mouvements,

plus il y a d'images déclenchées. La fonction déclenche une image toutes les N images par mesure de sécurité pour assurer le bon fonctionnement du système de localisation temps-réel. L'amélioration de cette fonctionnalité est la principale perspective de recherche sur le contrôle du flux permettant d'être le plus efficace possible suivant les cas d'usages visés.

3/ La mise en place des différentes techniques d'optimisation des données à un impact important sur la consommation des ressources et prend en compte un large spectre de méthodes de l'état de l'art, mais ne prend pas en compte la complexité des fonctions implémentées dans chaque algorithme. Cette perspective se concentre sur la réduction de la complexité calculatoire de la reconstruction 3D. Comment réduire la complexité de la reconstruction 3D basée voxel ? Un des points sur lesquels on peut agir est la taille du voxel qui affecte la granularité du modèle 3D et sa consommation mémoire. Ainsi, la principale perspective est d'avoir une taille de voxel variable. Comme pour la dynamique de l'image, cette fonctionnalité est variable en fonction de la pertinence des éléments, qui est caractérisée par la donnée temporelle ou sémantique. Le cas d'usage est également un facteur important à prendre en compte. La granularité de la reconstruction 3D dépend de son utilisation. Par exemple, une représentation plus fine des éléments de la scène est nécessaire pour l'interaction avec les objets que pour de la détection d'obstacles, où seule la complétude de la reconstruction 3D est importante. La taille variable des voxels doit être définie suivant l'utilisation de la reconstruction 3D.

4/ Pour finir, l'état de l'art évolue rapidement et notamment avec l'intégration des méthodes basées sur les réseaux de neurones qui offrent la possibilité de couvrir une large quantité de données avec une précision élevée, mais avec une complexité calculatoire importante. L'intégration de ces approches dans la chaine de traitement SLAM permet d'améliorer la robustesse des algorithmes de localisation en temps-réel par l'amélioration de la qualité des points d'intérêts extraits mieux distribués et pour la gestion des environnements dynamiques. L'implémentation de la chaine de traitement hybride, associant les modèles géométriques et neuronaux est pertinente et est un axe de recherche à suivre dans le domaine du SLAM embarqué pour réduire le nombre d'opérations sans perdre en précision et en robustesse.

La thèse a permis de contribuer au domaine du SLAM embarqué en réduisant la complexité calculatoire des algorithmes par le biais de deux mécanismes d'optimisation. Les limitations identifiées par les méthodes de l'état de l'art ont été adressées afin de trouver le compromis optimal entre la réduction des données et la précision. Un des concepts connexes auquel les travaux contribuent est le SLAM frugal. Les contributions prennent en considération le principe de frugalité pour assurer un traitement temps-réel et une faible utilisation des ressources matérielles tout en maintenant un système robuste et précis. Les perspectives restent nombreuses d'un point de vue matériel et logiciel et permettent de contribuer à l'état de l'art dont les recherches sont discutées par trois principales questions : comment rendre le SLAM plus robuste et moins coûteux pour une autonomie des systèmes long terme ? A-t-on besoin d'une forte cohérence géométrique ? Quelle est la représentation 3D de l'environnement ambiant idéale pour les systèmes réels ?

Bibliographie

W. N. Greene and N. Roy. Flame : Fast lightweight mesh estimation using variational smoothing on delaunay graphs. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4696–4704, 2017. doi : 10.1109/ICCV.2017.502.

A. Rosinol, M. Abate, Y. Chang, and L. Carlone. Kimera : an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696, 2020. doi : 10.1109/ICRA40945.2020.9196885.

Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3 : An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, pages 1–17, 2021. doi : 10.1109/TRO.2021.3075644.

Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016. doi : 10.1177/0278364915620033. URL http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract.

D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers. The tum vi benchmark for evaluating visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1680–1687, 2018. doi : 10.1109/IROS.2018.8593419.

Z. Zhang and D. Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251, 2018. doi : 10.1109/IROS.2018.8593941.

Cloudcompare.org. Cloudcompare - open source project. https://www.cloudcompare.org.

A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze. Navion : A 2-mw fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones. *IEEE Journal of Solid-State Circuits*, 54(4) :1106–1119, 2019. doi : 10.1109/JSSC.2018.2886342.

J. Delmerico and D. Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2502–2509, 2018. doi : 10.1109/ICRA.2018.8460664.

Olivia Christie, Joshua Rego, and Suren Jayasuriya. Analyzing sensor quantization of raw images for visual slam. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 246–250, 2020. doi : 10.1109/ICIP40778.2020.9191352.

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012. doi : 10.1109/IROS.2012.6385773.

Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox : Incremental 3d euclidean signed distance fields for on-board mav planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

Runze Liu, Jianlei Yang, Yiran Chen, and Weisheng Zhao. Eslam : An energy-efficient accelerator for real-time orb-slam on fpga platform. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367257. doi : 10.1145/3316781.3317820. URL https://doi.org/10.1145/3316781.3317820.

Liangkai Liu, Sidi Lu, Ren Zhong, Baofu Wu, Yongtao Yao, Qingyang Zhang, and Weisong Shi. Computing systems for autonomous driving : State of the art and challenges. *IEEE Internet of Things Journal*, 8(8) :6469–6486, 2020.

Dimitris Chatzopoulos, Carlos Bermejo, Zhanpeng Huang, and Pan Hui. Mobile augmented reality survey : From where we are to where we go. *IEEE Access*, 5 :6917–6950, 2017. doi : 10.1109/ACCESS.2017.2698164.

Andy Couturier and Moulay A. Akhloufi. A review on absolute visual localization for uav. *Robotics and Autonomous Systems*, 135 :103666, 2021. ISSN 0921-8890. doi : https://doi.org/10.1016/j.robot.2020.103666. URL https://www.sciencedirect.com/science/article/pii/S0921889020305066.

Pedro Pinies, Juan D. Tardos, and Jose Neira. Localization of avalanche victims using robocentric slam. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3074–3079, 2006. doi : 10.1109/IROS.2006.282247.

Nakul Garg, Irtaza Shahid, Karthik Sankar, Mallesham Dasari, Ramanujan K Sheshadri, Karthikeyan Sundaresan, and Nirupam Roy. Bringing ar/vr to everyday life - a wireless localization perspective. In *Proceedings of the 24th International Workshop on Mobile Computing Systems and Applications*, HotMobile '23, page 142, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400700170. doi : 10.1145/3572864.3581589. URL https://doi.org/10.1145/3572864.3581589.

Andrew J. Davison. Futuremapping : The computational structure of spatial ai systems. *ArXiv*, abs/1803.11288, 2018.

P. Alliez, Fabien Bonardi, S. Bouchafa, Jean-Yves Didier, H. Hadj-Abdelkader, F. I. I. Muñoz, V. Kachurka, Bastien Rault, M. Robin, and D. Roussel. Real-time multi-slam system for agent localization and 3d mapping in dynamic scenarios. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4894–4900, 2020a.

Pierre Alliez, Fabien Bonardi, Samia Bouchafa, Jean-Yves Didier, Hicham Hadj-Abdelkader, Fernando Israel Ireta Muñoz, Viachaslau Kachurka, Bastien Rault, Maxime Robin, and David Roussel. Indoor Localization and Mapping : Towards Tracking Resilience Through a Multi-SLAM Approach.

In *MED 2020 - 28th Mediterranean Conference on Control and Automation*, pages 465–470, Saint Raphael, France, September 2020b. URL https://hal.inria.fr/hal-02611679.

Ji Zhang and S. Singh. Loam : Lidar odometry and mapping in real-time. In *Robotics : Science and Systems*, 2014.

Viachaslau Kachurka, David Roussel, Hicham Hadj-Abdelkader, Fabien Bonardi, Jean-Yves Didier, and Samia Bouchafa. SWIR Camera-Based Localization and Mapping in Challenging Environments. In *20th International Conference on IMAGE ANALYSIS AND PROCESSING (ICIAP 2019)*, volume 11752 of *Lecture Notes in Computer Science*, pages 446–456, Trento, Italy, September 2019. doi : 10.1007/978-3-030-30645-8_41. URL https://hal.archives-ouvertes.fr/hal-02271971.

Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and D. Scaramuzza. Ultimate slam ? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 3 :994–1001, 2018.

G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. Event-based vision : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. doi : 10.1109/TPAMI.2020.3008413.

Quentin Picard, Stephane Chevobbe, Mehdi Darouich, and Jean-Yves Didier. Image quantization towards data reduction : Robustness analysis for slam methods on embedded platforms. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 4158–4162, 2022a. doi : 10.1109/ICIP46576.2022.9897315.

Quentin Picard, Stephane Chevobbe, Mehdi Darouich, Zoe Mandelli, Mathieu Carrier, and Jean-Yves Didier. Work-in-progress : Smart data reduction in slam methods for embedded systems. In *2022 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, pages 23–24, 2022b. doi : 10.1109/CASES55004.2022.00018.

C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping : Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6) :1309–1332, 2016. doi : 10.1109/TRO.2016.2624754.

Cyrill Stachniss, John J. Leonard, and Sebastian Thrun. *Simultaneous Localization and Mapping*, pages 1153–1176. Springer International Publishing, Cham, 2016. ISBN 978-3-319-32552-1. doi : 10.1007/978-3-319-32552-1_46. URL https://doi.org/10.1007/978-3-319-32552-1_46.

David M. Rosen, Kevin J. Doherty, Antonio Terán Espinoza, and John J. Leonard. Advances in inference and representation for simultaneous localization and mapping. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1) :215–242, 2021. doi : 10.1146/annurev-control-072720-082553. URL https://doi.org/10.1146/annurev-control-072720-082553.

H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping : part i. *IEEE Robotics Automation Magazine*, 13(2) :99–110, 2006. doi : 10.1109/MRA.2006.1638022.

T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam) : part ii. *IEEE Robotics Automation Magazine*, 13(3) :108–117, 2006. doi : 10.1109/MRA.2006.1678144.

Changhao Chen, Bing Wang, Chris Xiaoxuan Lu, Agathoniki Trigoni, and Andrew Markham. A survey on deep learning for localization and mapping : Towards the age of spatial machine intelligence. *ArXiv*, abs/2006.12567, 2020.

Ruihao Li, Sen Wang, and Dongbing Gu. Deepslam : A robust monocular slam system with unsupervised deep learning. *IEEE Transactions on Industrial Electronics*, 68(4) :3577–3587, 2021. doi : 10.1109/TIE.2020.2982096.

Imane Salhi, Martyna Poreba, Erwan Piriou, Valerie Gouet-Brunet, and Maroun Ojail. Chapter 8 - multimodal localization for embedded systems : A survey. In Michael Ying Yang, Bodo Rosenhahn, and Vittorio Murino, editors, *Multimodal Scene Understanding*, pages 199–278. Academic Press, 2019. ISBN 978-0-12-817358-9. doi : https://doi.org/10.1016/B978-0-12-817358-9.00014-7. URL https://www.sciencedirect.com/science/article/pii/B9780128173589000147.

Quentin Picard, Stephane Chevobbe, Mehdi Darouich, and Jean-Yves Didier. A survey on real-time 3D scene reconstruction with SLAM methods in embedded systems. working paper or preprint, November 2021. URL https://hal.science/hal-04201480.

Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping : A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2 :194–220, 2017.

M. Zollhöfer, P. Stotko, Andreas Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. State of the art on 3d reconstruction with rgb-d cameras. *Computer Graphics Forum*, 37, 2018.

Yusra Alkendi, Lakmal Seneviratne, and Yahya Zweiri. State of the art in vision-based localization techniques for autonomous navigation systems. *IEEE Access*, 9 :76847–76874, 2021. doi : 10.1109/ACCESS.2021.3082778.

Mihai Bujanca, Xuesong Shi, Matthew Spear, Pengpeng Zhao, Barry Lennox, and Mikel Lujan. Robust slam systems : Are we there yet ? In *IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS 2021)*. (Accepted/In press), 2021.

Rana Azzam, Tarek Taha, Shoudong Huang, and Yahya Zweiri. Feature-based visual simultaneous localization and mapping : a survey. *SN Applied Sciences*, 2, 02 2020. doi : 10.1007/s42452-020-2001-3.

C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1) :1–21, 2017a. doi : 10.1109/TRO.2016.2597321.

Chris Engels, Henrik Stewénius, and David Nistér. Bundle adjustment rules. In *In Photogrammetric Computer Vision*, 2006.

C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. Svo : Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2) :249–265, 2017b. doi : 10.1109/TRO.2016.2623335.

J. Engel, T. Schöps, and D. Cremers. LSD-SLAM : Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014.

S. Garg, N. Sünderhauf, F. Dayoub, D. Morrison, A. Cosgun, G. Carneiro, Q. Wu, T. J. Chin, I. Reid, S. Gould, P. Corke, and M. Milford. *Semantics for Robotic Mapping, Perception and Interaction : A Survey*. Now Foundations and Trends, 2020.

Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks : A survey. *Robotics and Autonomous Systems*, 66 :86–103, 2015. ISSN 0921-8890. doi : https://doi.org/10.1016/j.robot.2014.12.006. URL https://www.sciencedirect.com/science/article/pii/S0921889014003030.

Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. Does computer vision matter for action ? *Science Robotics*, 4(30) :eaaw6661, 2019. doi : 10.1126/scirobotics.aaw6661. URL https://www.science.org/doi/abs/10.1126/scirobotics.aaw6661.

Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1403–1410 vol.2, 2003. doi : 10.1109/ICCV.2003.1238654.

A. Davison, I. Reid, Nicholas D. Molton, and O. Stasse. Monoslam : Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29 :1052–1067, 2007.

T. Qin, P. Li, and S. Shen. Vins-mono : A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4) :1004–1020, 2018. doi : 10.1109/TRO.2018.2853729.

A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007. doi : 10.1109/ROBOT.2007.364024.

D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6) :756–770, 2004. doi : 10.1109/TPAMI.2004.17.

Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4) :629–642, Apr 1987. doi : 10.1364/JOSAA.4.000629. URL http://www.osapublishing.org/josaa/abstract.cfm?URI=josaa-4-4-629.

Margarita Chli Laurent Kneip and Roland Siegwart. Robust real-time visual odometry with a single camera and an imu. In *Proceedings of the British Machine Vision Conference*, pages 16.1–16.11. BMVA Press, 2011. ISBN 1-901725-43-X. http ://dx.doi.org/10.5244/C.25.16.

Edward H. Adelson, Peter J. Burt, Charles H. Anderson, Joan M. Ogden, and James R. Bergen. Pyramid methods in image processing. 1984.

Hans P. Moravec. Towards automatic visual obstacle avoidance. In *IJCAI*, 1977.

C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. doi : 10.1109/CVPR.1994.323794.

Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33833-8.

G. LoweDavid. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33833-8.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb : An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. doi : 10.1109/ICCV.2011.6126544.

Jean yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.

J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-point ransac for ekf-based structure from motion. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3498–3504, 2009. doi : 10.1109/IROS.2009.5354410.

D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004. doi : 10.1109/CVPR.2004.1315094.

R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam : Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, 2011a. doi : 10.1109/ICCV.2011.6126513.

J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3) :611–625, 2018. doi : 10.1109/TPAMI.2017.2658577.

Daniel Cremers. Direct methods for 3d reconstruction and visual slam. In *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, pages 34–38, 2017. doi : 10.23919/MVA.2017.7986766.

M. Pizzoli, C. Forster, and D. Scaramuzza. Remode : Probabilistic, monocular dense reconstruction in real time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2609–2616, 2014. doi : 10.1109/ICRA.2014.6907233.

J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *2013 IEEE International Conference on Computer Vision*, pages 1449–1456, 2013. doi : 10.1109/ICCV.2013.183.

J. Engel, J. Stückler, and D. Cremers. Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942, 2015. doi : 10.1109/IROS.2015.7353631.

Xiang Dong, Long Cheng, Hu Peng, and Teng Li. Fsd-slam : a fast semi-direct slam algorithm. *Complex & Intelligent Systems*, 2021. doi : 10.1007/s40747-021-00323-y.

Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

J. Leonard, H. Durrant-Whyte, and I.J. Cox. Dynamic map building for autonomous mobile robot. In *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, pages 89–96 vol.1, 1990. doi : 10.1109/IROS.1990.262373.

Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings IROS '91 :IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, pages 1442–1447 vol.3, 1991. doi : 10.1109/IROS.1991.174711.

J. Folkesson and H. Christensen. Graphical slam - a self-correcting map. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 1, pages 383–390 Vol.1, 2004. doi : 10.1109/ROBOT.2004.1307180.

Udo Frese and Lutz Schroder. Closing a million-landmarks loop. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5032–5039, 2006. doi : 10.1109/IROS.2006.282531.

Frank Dellaert and Michael Kaess. Square root sam : Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12) :1181–1203, 2006. doi : 10.1177/0278364906072768. URL https://doi.org/10.1177/0278364906072768.

Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem : A survey. In *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development : Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, page 363–371, NLD, 2008. IOS Press. ISBN 9781586039257.

Jianjun Gui, Dongbing Gu, Sen Wang, and Huosheng Hu. A review of visual inertial odometry from filtering and optimisation perspectives. *Advanced Robotics*, 29 :1289 – 1301, 2015.

Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J. Taylor, and Vijay Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, 3(2) :965–972, 2018. doi : 10.1109/LRA.2018.2793349.

Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. Openvins : A research platform for visual-inertial estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672, 2020. doi : 10.1109/ICRA40945.2020.9196524.

F. Dellaert and M. Kaess. Factor graphs for robot perception. *Found. Trends Robotics*, 6 :1–139, 2017.

F. Dellaert and Others. Georgia tech smoothing and mapping (gtsam), 2019. URL https://gtsam.org/.

Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2 : Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2) :216–235, 2012. doi : 10.1177/0278364911430419. URL https://doi.org/10.1177/0278364911430419.

R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o : A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. doi : 10.1109/ICRA.2011.5979949.

Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 3 2022. URL https://github.com/ceres-solver/ceres-solver.

V. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers. Visual-inertial mapping with non-linear factor recovery. *IEEE Robotics and Automation Letters*, 5(2) :422–429, 2020. doi : 10.1109/LRA.2019.2961227.

Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3) :314–334, 2015. doi : 10.1177/0278364914554813. URL https://doi.org/10.1177/0278364914554813.

D. Galvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5) :1188–1197, 2012. doi : 10.1109/TRO.2012.2197158.

Han Wang, Juncheng Li, Maopeng Ran, and Lihua Xie. Fast loop closure detection via binary content. In *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, pages 1563–1568, 2019a. doi : 10.1109/ICCA.2019.8899937.

Fabian Schenk and Friedrich Fraundorfer. Reslam : A real-time robust edge-based slam system. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 154–160, 2019. doi : 10.1109/ICRA.2019.8794462.

R. Mur-Artal and J. D. Tardós. Orb-slam2 : An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5) :1255–1262, 2017. doi : 10.1109/TRO.2017.2705103.

Thomas Schöps, Torsten Sattler, and M. Pollefeys. Surfelmeshing : Online surfel-based mesh reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42 :2494–2507, 2020.

Simena Dinas and Jose Bañón. A review on delaunay triangulation with application on computer vision. *IJCSE - International Journal of Computer Science and Engineering*, 3 :9–18, 03 2014.

Antoni Rosinol, Torsten Sattler, M. Pollefeys, and L. Carlone. Incremental visual-inertial 3d mesh generation with structural regularities. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8220–8226, 2019.

L. Teixeira and M. Chli. Real-time mesh-based scene estimation for aerial inspection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4863–4869, 2016. doi : 10.1109/IROS.2016.7759714.

Masashi Yokozuka, Shuji Oishi, Simon Thompson, and Atsuhiko Banno. Vitamin-e : Visual tracking and mapping with extremely dense feature points. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9633–9642, 2019.

Enrico Piazza, Andrea Romanoni, and Matteo Matteucci. Real-time cpu-based large-scale three-dimensional mesh reconstruction. *IEEE Robotics and Automation Letters*, 3(3) :1584–1591, 2018. doi : 10.1109/LRA.2018.2800104.

Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry*, 22(1) :21 – 74, 2002. ISSN 0925-7721. doi : https://doi.org/10.1016/S0925-7721(01)00047-5. URL http://www.sciencedirect.com/science/article/pii/S0925772101000475. 16th ACM Symposium on Computational Geometry.

William E. Lorensen and Harvey E. Cline. Marching cubes : A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH 1987*. ACM Press, 1987. doi : 10.1145/37401.37422.

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 303–312, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917464. doi : 10.1145/237170.237269. URL https://doi.org/10.1145/237170.237269.

R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion : Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011b. doi : 10.1109/ISMAR.2011.6092378.

Matthew Klingensmith, Ivan Dryanovski, S. Srinivasa, and J. Xiao. Chisel : Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields. In *Robotics : Science and Systems*, 2015.

S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In *Proceedings Visualization '98 (Cat. No.98CB36276)*, pages 233–238, 1998. doi : 10.1109/VISUAL.1998.745713.

P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2) :239–256, 1992. doi : 10.1109/34.121791.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion : Real-time 3d reconstruction and interaction using a moving depth camera. In *UIST '11 Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, October 2011.

Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6), November 2013. ISSN 0730-0301. doi : 10.1145/2508363.2508374. URL https://doi.org/10.1145/2508363.2508374.

O. Kähler, V. Prisacariu, C. Ren, X. Sun, P. Torr, and D. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics*, 21 :1241–1250, 2015.

Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks : A tutorial and survey. *Proceedings of the IEEE*, 105(12) :2295–2329, 2017.

S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo : Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050, 2017. doi : 10.1109/ICRA.2017.7989236.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet : Learning optical flow with convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015. doi : 10.1109/ICCV.2015.316.

Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Agathoniki Trigoni. Vinet : Visual-inertial odometry as a sequence-to-sequence learning problem. In *AAAI*, 2017.

Chengzhou Tang and Ping Tan. BA-net : Dense bundle adjustment networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1gabhRcYX.

Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J. Davison. Deepfactors : Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters*, 5 :721–728, 2020.

Saba Arshad and Gon-Woo Kim. Role of deep learning in loop closure detection for visual and lidar slam : A survey. *Sensors*, 21(4) :1243, Feb 2021. ISSN 1424-8220. doi : 10.3390/s21041243. URL http://dx.doi.org/10.3390/s21041243.

Xiwu Zhang, Yan Su, and Xinhua Zhu. Loop closure detection for visual slam systems using convolutional neural network. In *2017 23rd International Conference on Automation and Computing (ICAC)*, pages 1–6, 2017a. doi : 10.23919/IConAC.2017.8082072.

Dongjiang Li, Xuesong Shi, Qiwei Long, Shenghui Liu, Wei Yang, Fangshi Wang, Qi Wei, and Fei Qiao. Dxslam : A robust and efficient visual slam system with deep features. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4958–4965, 2020. doi : 10.1109/IROS45743.2020.9340907.

Marc Sons, Christian Kinzig, Dominic Zanker, and Christoph Stiller. An approach for cnn-based feature matching towards real-time slam. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1305–1310, 2019. doi : 10.1109/ITSC.2019.8917293.

Zhilin Xu, Jincheng Yu, Chao Yu, Hao Shen, Yu Wang, and Huazhong Yang. Cnn-based feature-point extraction for real-time visual slam on embedded fpga. In *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 33–37, 2020. doi : 10.1109/FCCM48280.2020.00014.

Qunjie Zhou, Torsten Sattler, and Laura Leal-Taixe. Patch2pix : Epipolar-guided pixel-level correspondences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4669–4678, 2021.

Jiexiong Tang, Ludvig Ericson, John Folkesson, and Patric Jensfelt. Gcnv2 : Efficient correspondence prediction for real-time slam. *IEEE Robotics and Automation Letters*, 4(4) :3505–3512, 2019. doi : 10.1109/LRA.2019.2927954.

Shervin Minaee, Yuri Y. Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. doi : 10.1109/TPAMI.2021.3059968.

Linlin Xia, Jiashuo Cui, Ran Shen, Xun Xu, Yiping Gao, and Xinying Li. A survey of image semantics-based visual simultaneous localization and mapping : Application-oriented solutions to autonomous navigation of mobile robots. *International Journal of Advanced Robotic Systems*, 17(3) : 1729881420919185, 2020. doi : 10.1177/1729881420919185. URL https://doi.org/10.1177/1729881420919185.

Konstantinos-Nektarios Lianos, Johannes L. Schönberger, Marc Pollefeys, and Torsten Sattler. Vso : Visual semantic odometry. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 246–263, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01225-0.

Abel Gawel, Carlo Del Don, Roland Siegwart, Juan Nieto, and Cesar Cadena. X-view : Graph-based semantic multi-view localization. *IEEE Robotics and Automation Letters*, 3(3) :1687–1694, 2018. doi : 10.1109/LRA.2018.2801879.

C. Zhang, L. Chen, and S. Yuan. St-vio : Visual-inertial odometry combined with image segmentation and tracking. *IEEE Transactions on Instrumentation and Measurement*, 69(10) :8562–8570, 2020. doi : 10.1109/TIM.2020.2989877.

Shuhuan Wen, Pengjiang Li, Yongjie Zhao, Hong Zhang, Fuchun Sun, and Zhe Wang. Semantic visual slam in dynamic environment. In *Autonomous Robots*, 2021. doi : https://doi.org/10.1007/s10514-021-09979-4.

Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. Ds-slam : A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1168–1174, 2018a. doi : 10.1109/IROS.2018.8593691.

K. Tateno, F. Tombari, and N. Navab. Real-time and scalable incremental segmentation on dense slam. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4465–4472, 2015. doi : 10.1109/IROS.2015.7354011.

J. McCormac, A. Handa, A. Davison, and S. Leutenegger. Semanticfusion : Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, 2017. doi : 10.1109/ICRA.2017.7989538.

J. Wald, K. Tateno, J. Sturm, N. Navab, and F. Tombari. Real-time fully incremental scene understanding on mobile platforms. *IEEE Robotics and Automation Letters*, 3(4) :3402–3409, 2018. doi : 10.1109/LRA.2018.2852782.

Radu Alexandru Rosu, Jan Quenzel, and Sven Behnke. Semi-supervised semantic mapping through label propagation with semantic texture meshes. *International Journal of Computer Vision*, 128 (5) :1220–1238, jun 2019. doi : 10.1007/s11263-019-01187-z.

M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto. Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery. *IEEE Robotics and Automation Letters*, 4(3) :3037–3044, July 2019. ISSN 2377-3766. doi : 10.1109/LRA.2019.2923960.

G. Narita, T. Seno, Tomoya Ishikawa, and Yohsuke Kaji. Panopticfusion : Online volumetric semantic mapping at the level of stuff and things. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4205–4212, 2019.

Z. Landgraf, F. Falck, M. Bloesch, S. Leutenegger, and A. J. Davison. Comparing view-based and map-based semantic labelling in real-time slam. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6884–6890, 2020. doi : 10.1109/ICRA40945.2020.9196843.

K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam : Real-time dense monocular slam with learned depth prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6565–6574, 2017. doi : 10.1109/CVPR.2017.695.

M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. Codeslam - learning a compact, optimisable representation for dense visual slam. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2560–2568, 2018. doi : 10.1109/CVPR.2018.00271.

Xiaochuan Yin, Xiangwei Wang, Xiaoguo Du, and Qijun Chen. Scale recovery for monocular visual odometry using depth estimated with deep convolutional neural fields. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5871–5879, 2017. doi : 10.1109/ICCV.2017.625.

Yi Li, Chenggang Xie, Huimin Lu, Xieyuanli Chen, Junhao Xiao, and Hui Zhang. Scale-aware monocular slam based on convolutional neural network. In *2018 IEEE International Conference on Information and Automation (ICIA)*, pages 51–56, 2018a. doi : 10.1109/ICInfA.2018.8812582.

Diogo Martins, Kevin Van Hecke, and Guido De Croon. Fusion of stereo and still monocular depth estimates in a self-supervised learning context. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 849–856, 2018. doi : 10.1109/ICRA.2018.8461116.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving ? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. doi : 10.1109/CVPR.2012.6248074.

Jincheng Yu, Feng Gao, Jianfei Cao, Chao Yu, Zhaoliang Zhang, Zhengfeng Huang, Yu Wang, and Huazhong Yang. Cnn-based monocular decentralized slam on embedded fpga. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 66–73, 2020. doi : 10.1109/IPDPSW50202.2020.00019.

Jincheng Yu, Guangjun Ge, Yiming Hu, Xuefei Ning, Jiantao Qiu, Kaiyuan Guo, Yu Wang, and Huazhong Yang. Instruction driven cross-layer cnn accelerator for fast detection on fpga. *ACM Trans. Reconfigurable Technol. Syst.*, 11(3), December 2018b. ISSN 1936-7406. doi : 10.1145/3283452. URL https://doi.org/10.1145/3283452.

Weisong Wen, Yiyang Zhou, Guohao Zhang, Saman Fahandezh-Saadi, Xiwei Bai, Wei Zhan, Masayoshi Tomizuka, and Li-Ta Hsu. Urbanloco : A full sensor suite dataset for mapping and localization in urban scenes. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2310–2316, 2020.

L.-T. Hsu, N. Kubo, W. Chen, Z. Liu, T. Suzuki, and J. Meguro. Urbannav : An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas. In *In Proceedings of the ION GNSS+ 2021*, 2021.

Jinyong Jeong, Younggun Cho, Young-Sik Shin, Hyunchul Roh, and Ayoung Kim. Complex urban dataset with multi-level sensors from highly diverse urban environments. *International Journal of Robotics Research*, 38(6) :642–657, 2019.

Xuesong Shi, Dongjiang Li, Pengpeng Zhao, Qinbin Tian, Yuxin Tian, Qiwei Long, Chunhao Zhu, Jingwei Song, Fei Qiao, Le Song, Yangquan Guo, Zhigang Wang, Yimin Zhang, Baoxing Qin,

Wei Yang, Fangshi Wang, Rosa H. M. Chan, and Qi She. Are we ready for service robots ? the OpenLORIS-Scene datasets for lifelong SLAM. In *2020 International Conference on Robotics and Automation (ICRA)*, pages 3139–3145, 2020.

Nicholas Carlevaris-Bianco, Arash K Ushani, and Ryan M Eustice. University of michigan north campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35 (9) :1023–1035, 2016. doi : 10.1177/0278364915614638. URL https://doi.org/10.1177/0278364915614638.

Andrzej Pronobis and Barbara Caputo. COLD : COsy Localization Database. *The International Journal of Robotics Research (IJRR)*, 28(5) :588–594, May 2009. doi : 10.1177/0278364909103912. URL http://www.pronobis.pro/publications/pronobis2009ijrr.

Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are we ready for autonomous drone racing ? the UZH-FPV drone racing dataset. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

András L Majdik, Charles Till, and Davide Scaramuzza. The zurich urban micro aerial vehicle dataset. *The International Journal of Robotics Research*, 36(3) :269–273, 2017. doi : 10.1177/0278364917702237. URL https://doi.org/10.1177/0278364917702237.

Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L. Schönberger, Pablo Speciale, Lukas Gruber, Viktor Larsson, Ondrej Miksik, and Marc Pollefeys. LaMAR : Benchmarking Localization and Mapping for Augmented Reality. In *ECCV*, 2022.

Selcuk Karakas, Pierre Moulon, Wenqi Zhang, Nan Yang, Julian Straub, Lingni Ma, Zhaoyang Lv, Elizabeth Argall, Georges Berenger, Tanner Schmidt, Kiran Somasundaram, Vijay Baiyya, Philippe Bouttefroy, Geof Sawaya, Yang Lou, Eric Huang, Tianwei Shen, David Caruso, Bilal Souti, Chris Sweeney, Jeff Meissner, Edward Miller, and Richard Newcombe. Aria data tools. https://github.com/facebookresearch/aria_data_tools, 2022.

E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss. ReFusion : 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. 2019. URL https://www.ipb.uni-bonn.de/pdfs/palazzolo2019iros.pdf.

Seonwook Park, Thomas Schöps, and Marc Pollefeys. Illumination change robustness in direct visual slam. In *ICRA*, 2017.

Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera : From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 2021.

A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1524–1531, 2014. doi : 10.1109/ICRA.2014.6907054.

Jakob Engel, Vladyslav C. Usenko, and D. Cremers. A photometrically calibrated benchmark for monocular visual odometry. *ArXiv*, abs/1607.02555, 2016.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012. doi : 10.1109/IROS.2012.6385773.

Senbo Wang, Jiguang Yue, Yanchao Dong, Shibo He, Haotian Wang, and Shaochun Ning. A synthetic dataset for visual slam evaluation. *Robotics and Autonomous Systems*, 124 :103336, 2020. ISSN 0921-8890. doi : https://doi.org/10.1016/j.robot.2019.103336. URL https://www.sciencedirect.com/science/article/pii/S0921889019301009.

Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based visual inertial odometry. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017. doi : 10.1109/cvpr.2017.616. URL https://github.com/daniilidis-group/msckf_mono.

Tong Qin, Shaozu Cao, Jie Pan, and Shaojie Shen. A general optimization-based framework for global pose estimation with multiple sensors. *ArXiv*, abs/1901.03642, 2019.

Yannick Verdie, F. Lafarge, and P. Alliez. Lod generation for urban scenes. *ACM Transactions on Graphics (TOG)*, 34 :1 – 14, 2015.

Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion : Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. Graph.*, 36(3), May 2017. ISSN 0730-0301. doi : 10.1145/3054739. URL https://doi.org/10.1145/3054739.

T. Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion : Dense slam without a pose graph. In *Robotics : Science and Systems*, 2015.

Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3) :80–es, 2007. ISSN 0730-0301. doi : 10.1145/1276377.1276478. URL https://doi.org/10.1145/1276377.1276478.

Sebastian O. H. Madgwick, Andrew J. L. Harrison, and Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7, 2011. doi : 10.1109/ICORR.2011.5975346.

M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 298–304, 2015. doi : 10.1109/IROS.2015.7353389.

R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. Slam++ : Simultaneous localisation and mapping at the level of objects. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, 2013. doi : 10.1109/CVPR.2013.178.

Stefano Aldegheri, Nicola Bombieri, Domenico D. Bloisi, and Alessandro Farinelli. Data flow orb-slam for real-time performance on embedded gpu boards. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5370–5375, 2019. doi : 10.1109/IROS40897.2019.8967814.

O. C. B. Silveira, J. G. O. C. de Melo, L. A. S. Moreira, J. B. N. G. Pinto, L. R. L. Rodrigues, and P. F. F. Rosa. Evaluating a visual simultaneous localization and mapping solution on embedded platforms. In *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*, pages 530–535, 2020. doi : 10.1109/ISIE45063.2020.9152370.

Nathaniel Merrill, Patrick Geneva, and Guoquan Huang. Robust monocular visual-inertial depth completion for embedded systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H.S. Torr. Fast online object tracking and segmentation : A unifying approach. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1328–1338, 2019b. doi : 10.1109/CVPR.2019.00142.

K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi : 10.1109/ICCV.2017.322.

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam : A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5) :1147–1163, 2015. doi : 10.1109/TRO.2015.2463671.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet : A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12) :2481–2495, 2017. doi : 10.1109/TPAMI.2016.2644615.

Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan Yuille. Towards unified depth and semantic prediction from a single image. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2800–2809, 2015. doi : 10.1109/CVPR.2015.7298897.

I. Laina, C. Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. *2016 Fourth International Conference on 3D Vision (3DV)*, pages 239–248, 2016.

T. J. Steiner, R. D. Truax, and K. Frey. A vision-aided inertial navigation system for agile high-speed flight in unmapped environments : Distribution statement a : Approved for public release, distribution unlimited. In *2017 IEEE Aerospace Conference*, pages 1–10, 2017. doi : 10.1109/AERO.2017.7943834.

Helen Oleynikova, Christian Lanegger, Zachary Taylor, Michael Pantic, Alexander Millane, Roland Siegwart, and Juan Nieto. An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments. *Journal of Field Robotics*, 37(4) :642–666, 2020.

Q. Gautier, A. Althoff, and R. Kastner. Fpga architectures for real-time dense slam. In *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, volume 2160-052X, pages 83–90, 2019. doi : 10.1109/ASAP.2019.00-25.

Konstantinos Boikos and Christos-Savvas Bouganis. Semi-dense slam on an fpga soc. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4, 2016. doi : 10.1109/FPL.2016.7577365.

Maria Rafaela Gkeka, Alexandros Patras, Christos D. Antonopoulos, Spyros Lalis, and Nikolaos Bellas. Fpga architectures for approximate dense slam computing. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, Grenoble, France, February 2021.

Taihú Pire, Thomas Fischer, Gastón Castro, Pablo De Cristóforis, Javier Civera, and Julio Jacobo Berlles. S-ptam : Stereo parallel tracking and mapping. *Robotics and Autonomous Systems*, 93 :27–42, 2017. ISSN 0921-8890. doi : https://doi.org/10.1016/j.robot.2017.03.019. URL https://www.sciencedirect.com/science/article/pii/S0921889015302955.

Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan : Dense 3d reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, 2011. doi : 10.1109/IVS.2011.5940405.

Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth : Fast monocular depth estimation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108, 2019. doi : 10.1109/ICRA.2019.8794182.

Skydio 2, 2019. URL https://www.skydio.com/skydio-2.

D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini. A 64-mw dnn-based visual navigation engine for autonomous nano-drones. *IEEE Internet of Things Journal*, 6(5) : 8357–8371, 2019. doi : 10.1109/JIOT.2019.2917066.

E. Terry. Silicon at the heart of hololens 2. In *2019 IEEE Hot Chips 31 Symposium (HCS)*, pages 1–26, 2019. doi : 10.1109/HOTCHIPS.2019.8875669.

Ziyun Li, Y. Chen, Luyao Gong, Lu Liu, D. Sylvester, D. Blaauw, and H. Kim. An 879gops 243mw 80fps vga fully visual cnn-slam processor for wide-range autonomous exploration. *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 134–136, 2019.

S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3923–3929, 2013. doi : 10.1109/IROS.2013.6696917.

Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza. Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle. *J. Field Robot.*, 33(4) :431–450, June 2016. ISSN 1556-4959. doi : 10.1002/rob.21581. URL https://doi.org/10.1002/rob.21581.

Thomas Whelan, John McDonald, Michael Kaess, Maurice Fallon, Hordur Johannsson, and John J. Leonard. Kintinuous : Spatially extended kinectfusion, July 2012.

Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap : An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. doi : 10.1007/s10514-012-9321-0. URL https://octomap.github.io.

F. Steinbrücker, J. Sturm, and D. Cremers. Volumetric 3d mapping in real-time on a cpu. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2021–2028, 2014. doi : 10.1109/ICRA.2014.6907127.

Manasi Muglikar, Zichao Zhang, and D. Scaramuzza. Voxel map for visual slam. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4181–4187, 2020.

J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart. A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 431–437, 2014. doi : 10.1109/ICRA.2014.6906892.

M. Lepecq and M. Darouich. A stream hardware architecture for keypoint matching based on a speculative approach. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2020. doi : 10.1109/ISCAS45731.2020.9180928.

Zhengdong Zhang, Amr Suleiman, Luca Carlone, Vivienne Sze, and Sertac Karaman. Visual-inertial odometry on chip : An algorithm-and-hardware co-design approach. In *Robotics : Science and Systems*, 2017b.

Microsoft. Microsoft hololens v1 and hololens v2, 2016, 2019. URL https://www.microsoft.com/en-us/hololens.

J. H. Yoon and A. Raychowdhury. Neuroslam : A 65-nm 7.25-to-8.79-tops/w mixed-signal oscillator-based slam accelerator for edge robotics. *IEEE Journal of Solid-State Circuits*, 56(1) :66–78, 2021. doi : 10.1109/JSSC.2020.3028298.

L. Millet, S. Chevobbe, C. Andriamisaina, L. Benaissa, E. Deschaseaux, E. Beigne, K. Ben Chehida, M. Lepecq, M. Darouich, F. Guellec, T. Dombek, and M. Duranton. A 5500-frames/s 85-gops/w 3-d stacked bsi vision chip based on parallel in-focal-plane acquisition and processing. *IEEE Journal of Solid-State Circuits*, 54(4) :1096–1105, 2019. doi : 10.1109/JSSC.2018.2886325.

P. Dudek and P. J. Hicks. A general-purpose processor-per-pixel analog simd vision chip. *IEEE Transactions on Circuits and Systems I : Regular Papers*, 52(1) :13–20, 2005. doi : 10.1109/TCSI.2004.840093.

Riku Murai, S. Saeedi, and P. Kelly. Bit-vo : Visual odometry at 300 fps using binary features from the focal plane. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8579–8586, 2020.

Jianing Chen, S. Carey, and Piotr Dudek. Feature extraction using a portable vision system. In *Vision-based Agile Autonomous Navigation of UAVsWorkshop, IEEE/RSJ International Conference on IntelligentRobots and Systems (IROS)*, 2017.

M. J. Ebstyne, F. Schafalitzky, D. Steedly, C. Chan, E. Eade, A. Kipman, and G. Klein. Pose tracking an augmented reality device.

Patrick Hübner, Kate Clintworth, Qingyi Liu, Martin Weinmann, and Sven Wursthorn. Evaluation of hololens tracking and depth sensing for indoor mapping applications. *Sensors*, 20(4), 2020. ISSN 1424-8220. doi : 10.3390/s20041021. URL https://www.mdpi.com/1424-8220/20/4/1021.

K. Khoshelham, H. Tran, and D. Acharya. Indoor mapping eyewear : Geometric evaluation of spatial mapping capability of hololens. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W13 :805–810, 2019. doi : 10.5194/isprs-archives-XLII-2-W13-805-2019. URL https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W13/805/2019/.

Mark Buckler, Suren Jayasuriya, and Adrian Sampson. Reconfiguring the imaging pipeline for computer vision. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 975–984, 2017. doi : 10.1109/ICCV.2017.111.

Georges Younes, Daniel Asmar, Elie Shammas, and John Zelek. Keyframe-based monocular slam : design, survey, and future directions. *Robotics and Autonomous Systems*, 98 :67–88, 2017. ISSN 0921-8890. doi : https://doi.org/10.1016/j.robot.2017.09.010. URL https://www.sciencedirect.com/science/article/pii/S0921889017300647.

Jin-Chun Piao and Shin-Dug Kim. Real-time visual–inertial slam based on adaptive keyframe selection for mobile ar applications. *IEEE Transactions on Multimedia*, 21(11) :2827–2836, 2019. doi : 10.1109/TMM.2019.2913324.

Reza Eyvazpour, Maryam Shoaran, and Ghader Karimian. Hardware implementation of slam algorithms : A survey on implementation approaches and platforms. *Artif. Intell. Rev.*, 56(7) : 6187–6239, nov 2022. ISSN 0269-2821. doi : 10.1007/s10462-022-10310-5. URL https://doi.org/10.1007/s10462-022-10310-5.

Mateusz Wasala, Hubert Szolc, and Tomasz Kryjak. An efficient real-time fpga-based orb feature extraction for an uhd video stream for embedded visual slam. *Electronics*, 11(14), 2022. ISSN 2079-9292. doi : 10.3390/electronics11142259. URL https://www.mdpi.com/2079-9292/11/14/2259.

Fu Li, Shaowu Yang, Xiaodong Yi, and Xuejun Yang. Towards visual slam with memory management for large-scale environments. In Bing Zeng, Qingming Huang, Abdulmotaleb El Saddik, Hongliang Li, Shuqiang Jiang, and Xiaopeng Fan, editors, *Advances in Multimedia Information Processing – PCM 2017*, pages 776–786, Cham, 2018b. Springer International Publishing. ISBN 978-3-319-77383-4.

Iman Firmansyah and Yoshiki Yamaguchi. Fpga-based implementation of the stereo matching algorithm using high-level synthesis. In *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 1–7, 2021. doi : 10.1109/MCSoC51149.2021.00009.

Yeongmin Lee and Hyeji Kim. A high-throughput depth estimation processor for accurate semiglobal stereo matching using pipelined inter-pixel aggregation. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(1) :411–422, 2022. doi : 10.1109/TCSVT.2021.3061200.

Ziyun Li, Qing Dong, Mehdi Saligane, Benjamin Kempke, Luyao Gong, Zhengya Zhang, Ronald Dreslinski, Dennis Sylvester, David Blaauw, and Hun-Seok Kim. A 1920 $\times$ 1080 30-frames/s 2.3 tops/w stereo-depth processor for energy-efficient autonomous navigation of micro aerial vehicles. *IEEE Journal of Solid-State Circuits*, 53(1) :76–90, 2018c. doi : 10.1109/JSSC.2017.2751501.

Paul Heckbert. Color image quantization for frame buffer display. *SIGGRAPH Comput. Graph.*, 16 (3) :297–307, jul 1982. ISSN 0097-8930. doi : 10.1145/965145.801294. URL https://doi.org/10.1145/965145.801294.

Michael Grupp. evo : Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo, 2017.

Qian-Yi Zhou, Jaesik Park, and V. Koltun. Open3d : A modern library for 3d data processing. *ArXiv*, abs/1801.09847, 2018.

Heaptrack. A heap memory profiler for linux. https://github.com/KDE/heaptrack.

S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4) :376–380, 1991. doi : 10.1109/34.88573.

Paul Merrell, Philippos Mordohai, Jan-Michael Frahm, and Marc Pollefeys. Evaluation of large scale scene reconstruction. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007. doi : 10.1109/ICCV.2007.4409218.

Jinwoo Jeon, Sungwook Jung, Eungchang Lee, Duckyu Choi, and Hyun Myung. Run your visual-inertial odometry on nvidia jetson : Benchmark tests on a micro aerial vehicle. *IEEE Robotics and Automation Letters*, 2021.