Monocular SLAM densification for 3D mapping and autonomous drone navigation

Yassine Habib

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE – IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Signal, Image et Vision*

Par
Yassine HABIB

Monocular SLAM densification for 3D mapping and autonomous drone navigation

Thèse présentée et soutenue à IMT Atlantique, Brest, le 15 Février 2024
Unité de recherche : Lab-STICC, CNRS UMR 6285, team RAMBO
Thèse N° : 2024IMTA0390

Rapporteurs avant soutenance :

Pascal VASSEUR Professeur, Universités de Picardie Jules Verne, MIS
Vincent CREUZE Professeur, Polytech Montpellier, LIRMM

Composition du Jury :

Président :	David FILLIAT	Professeur, ENSTA Paris, U2IS
Examinateurs :	Eva ARTUZI	Docteure, Naval Group Research, CEMIS
	Cédric DEMONCEAUX	Professeur, Universités de Bourgogne, ICB
	Vincent CREUZE	Professeur, Polytech Montpellier, LIRMM
Dir. de thèse :	Cédric BUCHE	Professeur des Universités, ENIB / Naval Group Pacific
Encadrant :	Panagiotis PAPADAKIS	Maître de Conférences HDR, IMT Atlantique

Invité(s) :

Tawsif GOKHOOL Docteur, Mob-Energy
Tiago GONÇALVES Ingénieur de recherche, Thales

ACKNOWLEDGEMENT

"All I know is that I know nothing"

— Socrates

This quote reflects a common feeling that I have felt at several points throughout this PhD. It emphasizes the importance of being able to question things, because research and innovation, which are evolving at an incredible pace, will always push back established boundaries.

I would like to express my deepest gratitude to my supervisors, Cédric LE BARZ, Panagiotis PAPADAKIS, Cédric BUCHE, Antoine FAGETTE, and Tiago GONÇALVES, for their invaluable support in completing this thesis under challenging circumstances, including the epidemic, expatriation, and managing different time zones. Each of you has contributed uniquely with your experience and expertise, providing guidance and insight over the past three years (and a little more).

A special thanks goes to Cédric LE BARZ, who recruited me at Thales in 2016 and introduced me to research and, more specifically, to the fields of Vision, Robotics, and AI. Your support in initiating and building this thesis, despite my overweening ambitions, has been invaluable. You have made a singular contribution to my career. Tiago, your bravery and willingness to pick up the torch along the way, despite the reputation that precedes me, deserves the highest praise. You have given me insightful advice, brought new perspectives to my work, and supported me mentally during tough times. Thierry LAMARQUE ... my apologies for keeping you on your toes; I do not think I have ever given you much of a break. I think you have been well served in terms of unexpected situations you have had to deal with. But I am so grateful for your support and to Virginie SAMSON for helping me to realize my expatriation project in Canada. My ambitions were great, but it was the support of both of you that really enabled me to achieve them. To my office mates Rémi, David.A., and Tom, thank you for the engaging and passionate discussions. More broadly, to Jean-Emmanuel, Stéphane, Andreina, David.H, and all my colleagues in the VAR lab, thank you for all the precious moments we have shared.

Antoine, after Singapore, this time you welcomed me at the beginning of winter in Montreal at Thales Research & Technology. Thank you for taking time out of your busy schedule for me, both before and after my visit. I really appreciated your way of seeing things and your informal approach. Ola, I sincerely thank you for our enriching discussions, during which you guided me through various research avenues with your expertise; I only wish I had more time to further explore them. Many thanks also to TDS and especially to the cortAIx team for your warm welcome. It was an enlightening experience that gave me valuable insights into your culture and working methods.

I would like to thank you again Panagiotis, you have also supervised this thesis from the beginning. You taught me a lot about research methodology and scientific rigor. Your support was especially crucial during the various stages of writing, times when I felt somewhat left on my own. Your dedication, especially in the final stages of the thesis, was greatly appreciated. Cédric BUCHE, despite the distance and the time difference that separates us from Australia, you have remained involved. Sometimes you had to take the bad role, but your demands and your frankness are unique and necessary. You pushed me to do better and maximize my chances of success. Lastly, a warm thought to my colleagues in the RAMBO team and the Info department, especially Jérôme, Mihai, Armelle, Hajer, Nacer, Gaëlic, Ikram and Coraline.

Last but not least, thank you, Mom! This PhD is also yours, for all you have sacrificed to bring me to this point. I am grateful to my family, Asmaa, Amine, and Inès, for supporting me in your own way. Rachelle, you have helped me more than you think throughout this journey, not only by improving my English, but also by reviewing some of my writings. Finally, special thanks to my close friends, who have been a constant source of joy; Samy and Yasmine, for your invaluable help during my stay in Canada, and Mehdi, for all the enriching discussions and scientific debates.

PREAMBLE

This thesis was conducted under the academic supervision of IMT Atlantique in Brest, specifically within the RAMBO team, which is part of the Lab-STICC (UMR CNRS 6285) doctoral school. This work was supervised by the Thales group at the ThereSIS (Thales European REsearch center for Security & Information Systems) research center in Palaiseau, France, and Thales Research & Technology in Montreal, Canada. This thesis benefited from a CIFRE grant from the National Association for Research and Technology (ANRT) under agreement N°2019/1877.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Résumé en Français

Introduction

Historiquement, le domaine de la robotique a été d'un intérêt particulier pour les applications de sûreté, de sécurité et de sauvetage en raison de l'utilité des robots qui peuvent remplacer ou compléter les humains dans les opérations dangereuses, atteindre des endroits inaccessibles, manipuler des objets avec précision et collecter des informations. Ils sont ainsi particulièrement appréciés pour les applications de recherche et de sauvetage en milieu urbain (USAR). Dans ce genre de situation d'urgence, la cartographie 3D de l'environnement et la navigation autonome sont des éléments cruciaux des missions de reconnaissances. Ils permettent aux opérateurs d'optimiser leur déploiement en localisant des zones de dangers, en planifiant leur itinéraire, et en identifiant des zones prioritaires d'interventions.

L'utilisation des drones aériens, et plus précisément des quadricoptères, présente un avantage majeur pour ces cas d'usage. Ce type de plateforme est moins coûteuse, plus légère, facilement déployable, et permet une plus grande manœuvrabilité, notamment en environnement intérieur. Cependant, le contrôle de ces drones requiert généralement la mobilisation d'un pilote expérimenté. Ainsi, la navigation autonome pourrait permettre de rendre plus accessible l'utilisation des drones pour ce type d'usage. Cela soulagerait également la charge cognitive des opérateurs en leur permettant de se concentrer davantage sur l'analyse des scènes explorées et le contrôle de plusieurs drones. La navigation autonome requiert trois fonctionnalités clés : la **localisation**, la **cartographie**, et la planification de trajectoire. Cette thèse aborde essentiellement les deux premières fonctionnalités. En effet, la localisation est nécessaire pour estimer la trajectoire du drone et la pose des capteurs vis-à-vis de la scène visionnée. La cartographie permet quant à elle de construire une représentation virtuelle de l'environnement où le drone navigue. Pour permettre la navigation et l'évitement d'obstacles, la carte reconstruite doit être dense, métrique (c'est-à-dire avec l'échelle absolue), et construite en temps-réel.

Le SLAM (Simultaneous Localization And Mapping) est l'approche la plus pertinente pour cette tâche. Tandis que plusieurs capteurs peuvent être considérés, les travaux de

cette thèse se restreignent à l'utilisation d'une caméra monoculaire et d'une centrale inertielle (IMU). En effet, cette combinaison de capteurs est relativement simple à mettre en œuvre et permet de minimiser les contraintes classiques en embarqué (taille, poids, consommation et coûts). L'utilisation d'une centrale inertielle accroît également la robustesse des systèmes de SLAM et permet d'obtenir une information métrique. Toutefois, l'incapacité à percevoir la profondeur à partir d'une caméra monoculaire, aussi connue sous le nom d'ambiguïté d'échelle, rend l'initialisation du SLAM plus difficile et la cartographie éparse.

La recherche sur le SLAM est assez mature, et a jusqu'à lors été majoritairement concentrée sur la robustesse et la précision de la tâche de localisation. Les méthodes de SLAM visuel conventionnelles à l'état-de-l'art sont capables de précisément estimer la trajectoire et de construire une carte non-dense en temps-réel. Leur robustesse et précision ont été démontrées sur plusieurs jeux de données d'évaluations. Cependant, elles n'ont pas été testées de façon exhaustive sur des conditions plus spécifiques, communes à la navigation de drone, comme les vitesses extrêmes ou les grands changements de luminosité, et particulièrement en configuration monoculaire.

Récemment, il y a eu un intérêt grandissant pour la densification du SLAM monoculaire grâce à l'émergence de la prédiction de profondeur monoculaire basée sur l'apprentissage profond. Ces approches reposent majoritairement sur des méthodes de SLAM monoculaire pures sans information métrique. Ainsi, elles se concentrent sur la reconstruction 3D précise sous forme de modèle 3D dense, généralement avec une échelle relative. Quelques techniques denses et métriques ont été proposées grâce à l'utilisation du SLAM monoculaire-inertiel, ce qui a permis d'obtenir les meilleures performances dans ce domaine. Toutefois, la plupart de ces travaux utilisent une stratégie multi-vues et de grosses architectures de réseaux de neurones, ce qui limite leur potentielle application sur des systèmes embarqués. De plus, l'utilisation de l'apprentissage supervisé contraint également les capacités de généralisation de ces méthodes sur de nouveaux environnements.

Dans le cadre de cette thèse, nous nous concentrons sur la navigation des drones en intérieur, où une représentation 3D grossière mais métrique est suffisante pour éviter les obstacles. Cette tâche ne requiert pas de modèle 3D de haute résolution. En effet, une carte d'occupation sous forme de voxel est suffisante et optimale. Cette contrainte, moins stricte, ouvre la voie à des approches plus légères pour obtenir une cartographie 3D dense, métrique et en temps réel en densifiant le SLAM monoculaire-inertiel. Ainsi, l'objectif de cette thèse est la cartographie 3D de scènes intérieures à partir d'un drone léger équipé

d'une caméra monoculaire dans le cadre d'opérations USAR. Nous concentrons notre étude sur la densification du SLAM monoculaire-inertiel dans ce contexte spécifique et adressons les questions de recherches suivantes :

— **Dans quelle mesure le SLAM visuel peut-il être appliqué à la navigation des drones dans des conditions difficiles ?**
Dans le but d'identifier une méthode de base pour nos travaux, nous présentons une évaluation de plusieurs algorithmes de SLAM visuel à l'état-de-l'art dans des conditions de hautes vitesses et de changements de luminosité importants.

— **Comment réaliser une densification du SLAM monoculaire à l'échelle métrique en temps réel sur un système embarqué ?**
Nous introduisons un processus permettant de densifier un système de SLAM monoculaire-inertiel grâce à la prédiction de profondeur monoculaire. Le système ayant pour objectif la reconstruction d'une carte 3D dense et métrique sous forme de voxel favorisant de futures applications de navigation et d'évitement d'obstacle depuis une plateforme embarquée.

— **Comment combiner la profondeur métrique et éparse du SLAM pour retrouver l'échelle absolue d'une carte de profondeur dense prédite ?**
Intégré dans le processus précédent, nous proposons une approche découplée permettant de retrouver l'échelle absolue d'une carte de profondeur dense prédite par un réseau de neurones profond à partir des estimations de profondeurs du SLAM, tout en minimisant l'impact sur le SLAM de base utilisé. La carte de voxels est construite de façon itérative à partir des cartes de profondeurs corrigées, et une fusion multi-vues volumétrique.

État de l'art en matière de densification du SLAM

Le SLAM visuel conventionnel repose essentiellement sur des fondations de la géométrie multi-vue. Il est généralement représenté sous la forme d'un problème d'optimisation de graphe de pose, où le nombre de variables croit avec le nombre de points dans la carte. Etant donné les objectifs initiaux de localisation précise et de temps-réel, la majorité des travaux ont adoptés une cartographie non-dense. Ce choix ayant permis d'adresser la plupart des défis associés à la localisation avec l'implémentation de méthodes plus robustes (ORB-SLAM 3 [1], Basalt [2]), démontrés sur des jeux de données standards (EuRoC [3], TUM-VI [4], KITTI [5]). Concernant les méthodes denses, elles ont jusqu'alors étaient

principalement limitées par leur complexité, notamment en configuration monoculaire.

Toutefois, l'émergence récente de l'apprentissage profond a permis un regain d'intérêt pour le SLAM dense monoculaire. En effet, l'approche typique pour densifier le SLAM consiste à intégrer un réseau de neurones profond pour prédire une carte de profondeur dense. Jusqu'à présent, l'objectif principal de ces recherches a été le reconstruction 3D dense sous forme de modèle 3D de haute qualité. Ainsi, la plupart des solutions reposent sur un SLAM monoculaire pur, sans échelle absolue. Aussi, alors que plusieurs métriques d'estimation de profondeur sont utilisées, il n'y a pas de processus et de métriques d'évaluation standardisées. Etant basé sur l'apprentissage profond, ces méthodes requièrent des ressources de calculs conséquentes qui semblent peu adaptées aux applications embarquées. De plus, l'utilisation de l'apprentissage supervisé implique également une limitation des capacités de généralisation. Aujourd'hui, CodeMapping [6] et CodeVIO [7] sont les méthodes à l'état-de-l'art, basées sur un SLAM monoculaire-inertiel et atteignant d'excellent résultats de cartographie. Cependant, leur code n'a pas été rendu publique.

La prédiction de profondeur monoculaire a donc été un levier majeur pour la densification. Ce sujet de recherche a été très largement adressé, en particulier pour les applications de voitures autonomes. A partir d'une image monoculaire, une carte de profondeur dense est prédite à un facteur d'échelle près. La principale approche repose sur l'apprentissage supervisé, où un modèle est entraîné à partir d'une grande quantité de données annotées. Toutefois, la collecte d'images avec la vérité-terrain de profondeur est très difficile. La principale alternative utilise l'apprentissage supervisé. Basée sur la géométrie multi-vue, il est possible d'entraîner le modèle sans vérité-terrain, à partir de plusieurs images d'une même scène sous différents points de vue et où les poses relatives sont connues. Cette approche, typiquement moins précise, tend à fournir de meilleure capacité de généralisation. PackNet-Sfm [8] est l'une des meilleures de cette catégorie, qui prétend même prédire l'échelle absolue grâce à une supervision de la vélocité. Très récemment, l'intégration des Transformers sur les approches supervisées a permis d'améliorer la précision mais surtout les capacités de généralisation. ZeroDepth [9] est l'une de ces méthodes qui atteint aujourd'hui les meilleures performances.

Pour nos expérimentations, nous avons choisi d'utiliser plusieurs bases de données. EuRoC [3] est un choix classique contenant des séquences collectées dans des scènes intérieures relativement petites. UZH-FPV [10] est un jeu de données contenant des images capturées depuis un drone à haute vitesse et contenant des mouvements agressifs. HILTI [11] est une base de données collectées dans des environnements intérieurs assez grands,

et contenant quelques séquences avec de forts changements de luminosité. Pour évaluer nos travaux, nous utilisons les métriques classiques d'estimation de pose et d'estimation de profondeur.

Du SLAM épars au SLAM dense pour la navigation de drones en intérieur

Le SLAM est devenu la méthode privilégiée pour effectuer les tâches de localisation et de cartographie essentielles à la navigation autonome des drones. Toutefois, lorsqu'elle repose sur une configuration monoculaire-inertielle, elle se limite à des méthodes éparses moins complexes pour atteindre une performance en temps réel.

Architecture du système de navigation

Alors que la cartographie dense tend à complexifier les solutions, nous proposons une approche découplée visant à minimiser l'impact sur la vitesse du SLAM épars de base. Le système proposé repose sur une architecture classique où les différentes tâches sont parallélisées sur CPU. Ainsi, plusieurs études ont analysé les performances de ces algorithmes sur des plateformes embarquées, démontrant leur efficacité. En particulier, on observe l'utilisation négligeable du GPU par ces algorithmes. De plus, les systèmes embarqués modernes montrent une amélioration constante des capacités de leurs processeurs graphiques pour l'apprentissage profond au fil des ans. Tous ces éléments nous confortent dans l'idée que l'utilisation d'un réseau de neurones peut être envisagée pour la densification du SLAM monoculaire et de futures applications de navigation de drone.

L'architecture proposée consiste donc à ajouter un processus de densification de la carte qui fonctionne à la fréquence des images clées. Cette fréquence est plus faible car les images clées sont sélectionnées lorsqu'un changement de point de vue suffisant est détecté. Pour chaque image clée, un réseau neuronal prédit une carte de profondeur dense, qui est ensuite combinée avec les points triangulés par le SLAM pour améliorer sa précision et en particulier pour assurer une échelle métrique. Enfin, la carte des voxels est mise à jour par projection de rayons en utilisant la carte de profondeur corrigée et la pose estimée par le SLAM. L'utilisation d'une grille d'occupation sous forme d'une carte de voxels est optimale pour sa construction et sa mise à jour, son stockage, et pour l'évitement d'obstacles.

Pour la cartographie 3D, une estimation robuste et précise de la trajectoire est es-

sentielle, car les points triangulés sont projetés à partir des poses estimées de la caméra. Il est donc essentiel pour notre système de sélectionner une méthode de SLAM de base suffisamment performante. C'est pourquoi nous présentons une évaluation exhaustive des performances de localisation de plusieurs algorithmes de SLAM visuel à l'état-de-l'art. Nous avons choisi ORB-SLAM 3 MI (Monoculaire-Inertiel) parmi les méthodes monoculaires conventionnelles car elle les surpasse de loin. Pour référence, nous avons également étudié les solutions Basalt [2], Kimera [12] et ORB-SLAM 3 [1] en configuration stéréo-inertielle.

Évaluation du SLAM épars

Toutes ces méthodes ont déjà été testées par leurs auteurs sur la base de données EuRoC [3]. Nous avons donc d'abord vérifié si nous pouvions reproduire correctement les résultats publiés. Dans l'ensemble, nous avons obtenu des résultats très similaires, à l'exception de Kimera, qui s'est avérée incorrecte sur quelques séquences difficiles avec des conditions d'éclairage très faibles. Sur ces scènes particulières, nous avons pu observer la résilience des autres méthodes dans de telles circonstances. Cela est principalement dû à leurs procédures respectives de détection et d'appariement des points caractéristiques, qui sont beaucoup plus robustes. D'autre part, nous avons également observé un retard dans l'initialisation d'ORB-SLAM 3, en particulier sur les scènes difficiles.

Le jeu de données suivant, UZH-FPV [10], contient des séquences avec des mouvements extrêmement rapides. Il a été collecté dans deux configurations différentes, l'une avec la caméra orientée vers l'avant et l'autre vers le bas. La première configuration correspond mieux à notre contexte d'application. Basalt ayant déjà été configuré par ses auteurs pour ces données, elle atteint une excellente robustesse et de très bons résultats compte tenu de la vitesse et des longues trajectoires impliquées. Cependant, l'estimation de l'échelle dans de telles conditions présente quelques lacunes. De façon générale, ORB-SLAM 3 obtient de meilleurs résultats, en particulier dans la configuration monoculaire-inertielle. En revanche, nous n'avons pas pu obtenir de résultats satisfaisants pour Kimera, qui a eu des difficultés à estimer l'échelle ou certains mouvements. Globalement, les séquences considérées contenaient des flous de mouvement et des décalages importants dans les images successives. Pour référence, nous avons également analysé les séquences avec la caméra pointant vers le sol. Si les résultats pour Basalt sont cohérents avec nos observations précédentes, les résultats pour Kimera et ORB-SLAM 3 ne sont pas concluants. En réalité, ORB-SLAM 3 n'a pas réussi à s'initialiser ou a perdu le fil dans la plupart des

séquences. Ces scènes présentent des complexités supplémentaires, telles que la prédominance de textures uniformes.

Enfin, nous avons utilisé la base de données HILTI [11], qui contient de grandes scènes d'intérieur. Il s'agit notamment de scènes comportant de longs couloirs et de grands changements d'éclairage. Malheureusement, nous n'avons pas été en mesure de présenter des résultats pour les solutions stéréo-inertielles. Cependant, nous avons observé d'excellentes performances pour ORB-SLAM 3 MI dans la plupart des séquences, même dans des conditions d'éclairage difficiles.

Nos expériences montrent que ORB-SLAM 3 dans une configuration monoculaire-inertielle émerge comme une base prometteuse pour la navigation des drones. Elle a démontré des performances compétitives sur l'ensemble de données sélectionné, avec de grande vitesse et dans des conditions d'illumination variables, surpassant même parfois les méthodes stéréo-inertielles. Cependant, la solution requiert un paramétrage minutieux, une calibration précise, et rencontre des difficultés sur les images sans textures. Cela tend à créer des retards d'initialisation, notamment lors de faibles mouvements. Basalt a démontré une grande robustesse dans toutes les séquences, mais a eu du mal à estimer avec précision l'échelle et a parfois mal estimé les rotations dans des scénarios de mouvement agressifs, ce qui a entraîné une dérive notable dans les estimations de certaines trajectoires. Kimera a montré des difficultés à estimer avec précision les différents mouvements, bien que ses résultats pourraient être améliorés avec un meilleur ajustement des paramètres.

Densification du SLAM monoculaire

En s'appuyant sur le système présenté précédemment et sur la méthode de SLAM sélectionnée (ORB-SLAM 3 [1]), nous introduisons un processus de récupération d'échelle qui combine les résultats du SLAM épars avec la carte de profondeur dense prédite par un réseau de neurones profond. Ce travail est évalué et comparé aux méthodes existantes. En outre, une présentation des résultats initiaux de la construction de la carte de voxels est fournie pour illustrer la mise en œuvre complète du système proposé.

Estimation de profondeur dense et métrique découplée : récupération d'échelle

Dans le système proposé, nous incorporons un modèle à l'état-de-l'art pour prédire une carte de profondeur dense à partir d'une image monoculaire. PackNet-Sfm [8] est un modèle entraîné sur des scènes extérieures, tandis que ZeroDepth [9] fut également en-

traîné sur de petits espaces intérieurs. Notre cas d'utilisation comporte de grands environnements intérieurs. Bien que ces méthodes affirment estimer l'échelle absolue et affichent des capacités de généralisation prometteuses, étant donné que le changement de domaine avec nos données peut être significatif, nous nous attendons à ce que l'échelle ne soit pas exactement métrique, mais au moins consistante.

Par conséquent, dans une deuxième phase, nous proposons de récupérer l'échelle métrique à partir de la profondeur éparse triangulée par le SLAM. Notre approche repose sur l'hypothèse d'échelle consistante, ce qui signifie que la carte de profondeur dense est prédite jusqu'à un facteur d'échelle globale. La plupart des travaux connexes utilisent l'approche d'échelle médiane pour retrouver le facteur global. Ces méthodes s'appuient sur les points de profondeur de la vérité terrain, généralement des nuages de points LiDAR. Dans notre contexte, nous disposons uniquement des points triangulés par le SLAM dans une quantité extrêmement faible. La méthode de la médiane devient donc moins fiable pour les petits ensembles de données où la variance peut être élevée et les valeurs aberrantes plus impactantes.

Par conséquent, pour tenir compte du faible nombre de points et de leur variabilité potentielle, nous suggérons d'estimer le facteur d'échelle global en minimisant l'erreur quadratique relative entre les points estimés par le SLAM et les points correspondants prédits par le réseau de neurones. Notre proposition repose sur l'observation que les points estimés par le SLAM sont relativement précis, avec une faible erreur de reprojection moyenne, ce qui en fait une bonne source pour retrouver le facteur d'échelle. L'efficacité de notre approche pour la récupération d'échelle métrique est démontrée à travers une analyse quantitative et qualitative des résultats. Bien que notre solution ne surpasse pas les meilleures méthodes de densification du SLAM monoculaire, elle s'en approche très fortement en s'appuyant sur une procédure moins complexe. D'autre part, nous avons également observé certaines limites des modèles de prédiction de profondeur, avec des résultats parfois bruités ou inconsistants, sur des objets fins ou lorsque le changement de domaine semblait trop important.

Construction de la carte de voxels

L'approche présentée plus tôt repose sur l'utilisation d'une vue unique. Pour la cartographie 3D, l'utilisation de techniques multi-vues permet de grandement améliorer les résultats. Une première catégorie d'approches consiste à fusionner plusieurs images d'une même scène sous différents points de vue afin d'améliorer la prédiction de carte de pro-

fondeur. L'autre catégorie s'appuie sur la fusion de cartes de profondeur pour améliorer la qualité de la carte 3D reconstruite. Ici, contrairement à la majorité des travaux existants pour la densification du SLAM monoculaire, nous ne souhaitons pas reconstruire un modèle 3D de haute qualité. Au contraire, nous choisissons d'intégrer une solution existante de construction de carte de voxels par projection de rayons. Cette technique exploite des cartes de profondeur dense dont la pose est connue afin de construire la carte de façon itérative. Lors de la projection de rayon, chaque voxel traversé est mis à jour, en ajustant son poids et sa distance à la surface la plus proche. Il en résulte une fusion volumétrique des cartes de profondeurs.

Bien que des méthodes d'évaluation de carte de voxels existent, aucune n'a encore été appliquée au SLAM monoculaire. Nous avons donc évalué qualitativement notre chaîne complète de cartographie 3D sous forme de voxels en la comparant à une méthode de référence basée sur la vision stéréo.

Nous expérimentons notre système sur les données EuRoC [3] pour une cartographie relativement fine, avec une largeur de voxels de 100 mm. Globalement, notre approche démontre une capacité à reconstruire une carte 3D dense de la pièce, avec l'échelle métrique, et sans vérité terrain. En comparaison avec les résultats de la référence stéréo, notre méthode semble couvrir davantage d'espace. Toutefois, notre carte est plus grossière et certains détails ne sont pas correctement cartographiés, en particulier sur les contours et les objets minces. Cela est principalement dû aux prédictions bruitées de la carte de profondeur et au fait que nous nous appuyons sur un nombre plus faible d'images, ce qui fait que certains endroits sont moins couverts. Ainsi, cela traduit une forte dépendance de notre système sur la qualité de la prédiction de profondeur.

La construction itérative et la fusion multi-vues volumétrique doivent permettre de mitiger les effets des prédictions bruitées ou aberrantes. L'utilisation de voxels plus grands (200 mm) permet d'atténuer ces effets en augmentant le nombre de points les plus proches qui sont moyennés. Néanmoins, il en résulte une carte plus grossière et un espace navigable plus restreint.

Conclusion et perspectives

Cette thèse a exploré la densification du SLAM monoculaire pour la cartographie 3D en temps réel, dense et métrique, cruciale pour la navigation autonome des drones dans les environnements intérieurs, en particulier dans des scénarios USAR. En relevant les défis de l'ambiguïté d'échelle et de la complexité inhérente au SLAM monoculaire, nous

proposons un système découplé qui combine le SLAM épars avec l'estimation monoculaire de la profondeur pour obtenir une reconstruction 3D dense et métrique sous forme de voxels, pensée pour de futures applications en temps réel sur des systèmes embarqués.

Les approches existantes reposent principalement sur des schémas multi-vues complexes ou nécessitent la vérité terrain de profondeur pour l'ajustement de l'échelle. Alors que notre méthode exploite les ressources GPU pour la densification sans pénaliser la fréquence du SLAM sous-jacent. Elle applique également une nouvelle procédure de récupération d'échelle permettant d'ajuster les cartes de profondeur prédites par le réseau de neurones à l'aide de la profondeur éparse estimée par le SLAM. Grâce à une analyse comparative dans des scénarios difficiles spécifiques aux drones, nous avons choisi ORB-SLAM 3 [1] comme méthode de base pour sa robustesse, malgré certaines limitations dans l'initialisation de la centrale inertielle. Les fonctions de récupération d'échelle et de cartographie en voxel de notre système ont été validées en intégrant différent modèles d'estimation de profondeur monoculaire à l'état-de-l'art. Nos expérimentations ont permis d'apprécier l'efficacité de notre approche qui a fourni des résultats prometteurs pour la navigation en temps réel des drones en intérieur. Cependant, elles ont également révélé une forte dépendance à la qualité des méthodes de SLAM et de prédiction de profondeur dense sous-jacentes.

Ces travaux ont vocation à être améliorés, en commençant par une évaluation quantitative plus approfondie. Cela requiert notamment la mise à disposition de nouveaux jeux de données d'évaluation et d'apprentissage comportant des vérités terrains de qualité. L'évaluation de la carte de voxels reconstruite à l'aide de métriques pertinentes permettrait de mieux mesurer les performances de notre approche. Une analyse détaillée des temps d'exécution de la chaîne complète sur plateformes embarquées est également nécessaire pour juger de l'applicabilité de la solution sur des systèmes réels. L'utilisation d'une approche couplée est une autre perspective de recherche qui permettrait d'exploiter les capacités des réseaux de neurones pour modéliser la relation entre une image monoculaire et les estimations du SLAM, afin de directement prédire une carte de profondeur dense et métrique. D'autre part, la construction de la carte de voxels pourrait être améliorée en ajustant la fonction de projection de rayons afin qu'elle considère l'incertitude des prédictions pour pondérer l'influences des points et minimiser la propagation des erreurs. Finalement, la carte 3D dense reconstruite pourrait être exploitée afin d'améliorer les performances de localisation du SLAM, notamment pour l'initialisation des points et la robustesse sur les images sans texture.

Table of Contents

GLOSSARY

ATE Absolute Trajectory Error. A metric used to measure the absolute accuracy of SLAM trajectory estimation. ATE quantifies the difference between the estimated and ground truth trajectory. 80

BA Bundle Adjustment. A technique in computer vision and photogrammetry used to refine the parameters (such as camera poses and 3D points) of a 3D reconstruction model to minimize the error between the observed and projected 2D points in images. It is commonly employed in Structure-from-Motion (SfM) and multi-view stereo (MVS) pipelines to improve the accuracy of 3D reconstructions from multiple images. 56

CNN Convolutional Neural Network. A type of deep neural network designed for processing structured grid data, such as images and video. CNNs are commonly used in image recognition and computer vision. 69

CVAE Conditional Variational Auto-Encoder. An extension of the standard VAE that incorporates conditional variables into the architecture. This enables the model to generate data conditioned on certain attributes, making it more versatile for tasks like conditional data generation, where the output is guided by specific input conditions. 70

DNN Deep Neural Network. A type of artificial neural network with multiple layers (deep architecture). DNNs are often used in machine learning and deep learning tasks, including image and speech recognition. 43

ESDF Euclidean Signed Distance Function. A variant of the signed distance function that calculates the Euclidean distance of each point in the map to the closest obstacle. This function is particularly useful for path planning and navigation tasks in robotics. 94

GNSS Global Navigation Satellite System. A satellite-based navigation system that provides geolocation and time information to GPS receivers. Examples include GPS (Global Positioning System) and GLONASS. 34

GPU Graphics Processing Unit. A specialized electronic circuit designed to accelerate the processing of images and videos. GPUs are commonly used in graphics rendering and machine learning tasks. 38

IMU Inertial Measurement Unit. A device that measures and reports a body's specific force, angular rate, and sometimes the magnetic field surrounding the body. IMUs are often used in robotics, navigation, and control systems. 37

LiDAR Light Detection and Ranging. A remote sensing technology that uses laser light to measure distances and create detailed 3D maps of objects and environments. LiDAR is used in various applications, including autonomous vehicles and mapping. 40

MDE Monocular Depth Estimation. A computer vision task that involves estimating the depth or distance of objects in a scene using a monocular camera. Monocular depth estimation is important for various applications, including autonomous navigation and 3D scene understanding. 43, 47

RMSE Root Mean Square Error. A statistical measure of the difference between predicted values and observed values. RMSE is commonly used to evaluate the accuracy of predictive models, with lower RMSE values indicating better model performance. 81

RPE Relative Pose Error. A metric used to measure the relative accuracy of pose estimation in computer vision and robotics. RPE quantifies the difference in pose between consecutive time steps or between two poses in a sequence. 80

SLAM Simultaneous Localization and Mapping. A technique used in robotics and computer vision to create and maintain a global map of an environment while simultaneously tracking the location of a sensor within that environment. 28, 39, 41

SoM System on Module. A small, integrated computing module that contains essential components of a computer system, often used in embedded and IoT (Internet of Things) applications. 38

TSDF Truncated Signed Distance Function. A representation in 3D reconstruction and mapping, where each voxel in a grid contains a truncated distance to the nearest surface. This function is useful in scenarios like robotic mapping and 3D scene reconstruction. 67, 94, 124

UAV Unmanned Aerial Vehicle. A type of aircraft that operates without a human pilot onboard. UAVs are commonly used for various applications, including aerial surveillance, photography, and remote sensing. 31

UGV Unmanned Ground Vehicle. A robotic vehicle that operates on land without the need for a human driver. UGVs are used in tasks such as autonomous exploration, reconnaissance, and search and rescue missions. 36

USAR Urban Search and Rescue. A specialized task that involves the location, extraction, and medical treatment of individuals trapped in urban or disaster-stricken environments. USAR teams often use advanced technology and equipment, including robots. 31

VAE Variational Auto-Encoder. A type of generative model that leverages deep learning and variational Bayesian methods to learn latent representations of input data. Unlike traditional auto-encoders, VAEs introduce probabilistic constraints on the encoding process, enabling them to generate new data samples that resemble the input data. 70

VIO Visual-Inertial Odometry. An extension of VO technology that combines visual and inertial sensor data to estimate the motion and position of a device or vehicle. The use of an IMU increases robustness and accuracy and also provides metric scale. 39

ViT Vision Transformer. A type of neural network model that applies the transformer architecture, originally designed for natural language processing tasks, to image analysis. ViTs divide an image into patches and process these through a series of self-attention mechanisms, enabling the model to focus on different parts of the image and understand the global context better. 76

VO Visual Odometry. A method used to estimate the motion and trajectory of a sensor or vehicle by analyzing sequential images. It typically builds a local map to determine relative motion, focusing on local consistency. 21, 63

List of Figures

List of Tables

Chapter 1

Introduction

The use of drones has attracted considerable interest in recent years, be it for personal or professional use. Their ease of use and deployment make them indispensable resources for safety, security, and rescue. Despite advances in drone control that facilitate their operation, safe navigation in the presence of obstacles is not always guaranteed, especially in indoor environments. Therefore, it is still necessary to mobilize a certified pilot for each intervention.

In this context, autonomous or semi-autonomous navigation is a key functionality that is highly desired in this sector. In parallel, 3D mapping can facilitate navigation whilst offering insights into the structure of the operational theatre. Several approaches exist to map the environment, plan the drone trajectory, and avoid obstacles. In practice, however, they are complex to implement, especially onboard, with limited resources, or in highly adverse conditions. Improving 3D perception in an embedded context would therefore enhance operational control of one or more Unmanned Aerial Vehicles (UAVs) in first responder or military operations and provide additional intelligence for intervention.

1.1 Specifics of autonomous indoor drone navigation

Historically, the field of Robotics has been of particular interest for safety, security, and rescue applications due to the expendability of robots that can replace or complement humans in hazardous operations, reach inaccessible places, handle objects with precision, and collect information [13], [14].

In particular, Urban Search And Rescue (USAR) applications are of particular interest. In the context of a specialized emergency response, the focus is on rescuing individuals trapped or injured in urban environments following disasters or emergencies. USAR teams, comprising trained experts and specialized equipment aim to locate survivors, provide medical assistance, and ensure their safe evacuation from collapsed structures or hazardous situations. Collaborating with various agencies, USAR operations require specialized skills

and technological support, including drones, to overcome challenges such as unstable structures, limited access, and time-sensitive searches, all in the pursuit of saving lives and providing critical assistance during crisis situations. Several studies reviewed these aspects of the applications of robots and especially drones for USAR [13]–[17].

Safety and security use cases may involve tactical criteria, such as discretion or high resilience, that impose additional constraints. Autonomous navigation and 3D mapping in military drones can serve surveillance and reconnaissance missions [13], [18]–[20]. They can provide perimeter security and patrolling, support search and rescue operations by covering large areas to locate personnel, facilitate route clearance in hazardous areas, and enhance counter-IED operations by identifying and eliminating potential threats. Overall, this contributes to improving tactical flexibility, situational awareness, and risk reduction in military operations.

The French group Thales is a major actor in various fields, including aerospace, defense, and security. The company, which is supporting this research, has been involved in several works related to drones, illustrating their usefulness in various domains. To address the potential risks associated with drone operations, Thales has developed a comprehensive guideline to help operators assess risk and comply with European regulations [21]. Products in the Thales ScaleFlyt line [22] are designed to enhance operations safety with features such as antijamming and geocaging. The BlueSwarm system introduces a novel counter-UAV approach to automatically and autonomously manage drone threats by detecting intrusions and executing countermeasures in real-time [23]. Besides its existing solutions for long-range surveillance [20], Thales is also exploring smaller drones, especially for first responder applications, as demonstrated by several research projects [24]–[27]. A notable project closely related to this thesis is Prometheus (illustrated in Figure 1.1), which focuses on autonomous drones for underground exploration and mapping, essential for ensuring safety in underground environments such as railway systems [28], [29].

In fact, among robotic platforms, drones were quickly adopted for their ease of deployment, low cost, and versatility [17], [30]. Nevertheless, in the beginning, controlling a drone was not a trivial task and required a trained pilot. Since then, navigation assistance has reached a level of maturity where drones (especially multi-rotor drones) have become widely available in the consumer market. Manufacturers, such as DJI and Parrot, have released several devices allowing anyone to capture high-quality video footage [31]. Yet, as demonstrated in [30], indoor navigation remains challenging because the number of

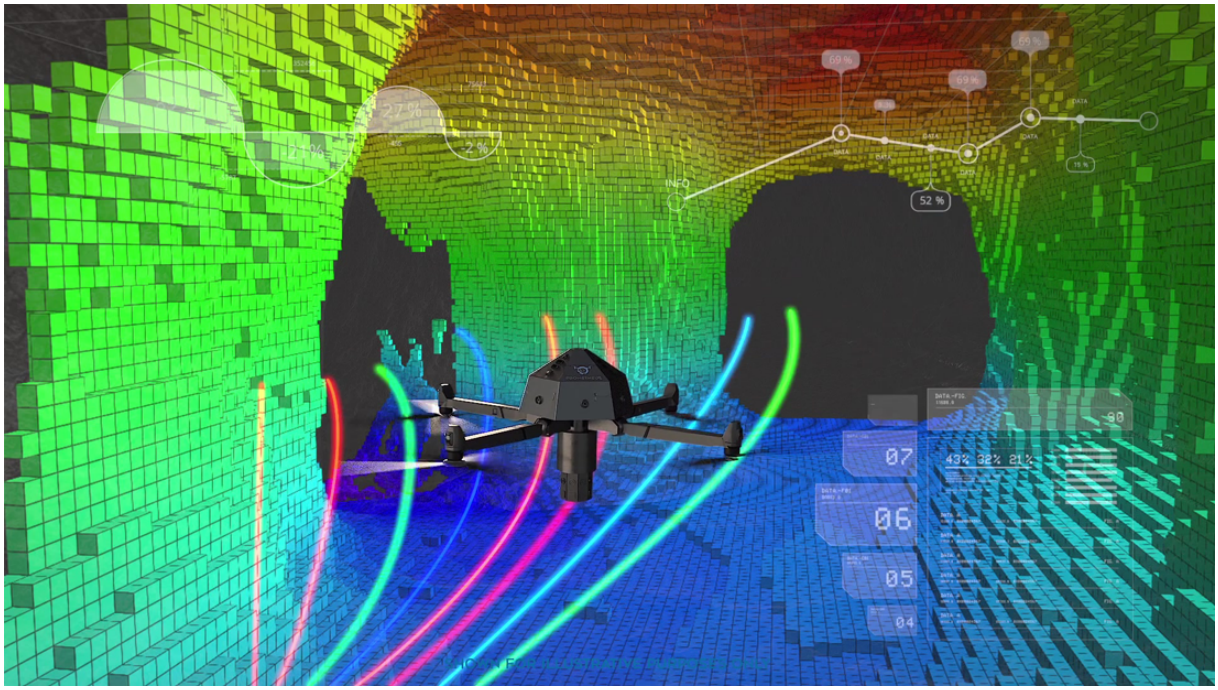


Figure 1.1 – Underground exploration illustration from the Prometheus project [28].

obstacles is higher, and free space is even more limited. In fact, as flight safety cannot be ensured at all times, regulations in several countries strictly limit the use of drones to specific isolated outdoor areas or specific usage [13], [31]. However, the missions of first responders often involve working in hazardous indoor environments. For this purpose, they require a certified pilot to be fully dedicated to the piloting. By enabling autonomous or semi-autonomous navigation, the operator would be able to control multiple drones or focus on scene analysis [17].

Such a navigation system requires three key functionalities: **localization**, environment **mapping**, and **path planning**. Determining the localization is fundamental for other tasks. It can be estimated using various types of sensors, the accuracy of which is highly dependent on the environment and the type of movement. The mapping task consists in building a virtual representation of the environment in which the drone navigates. Depending on the application, the map can be local or global, represented in 2D or 3D with different levels of density (Figure 1.2), be estimated with an absolute scale or up to a scale factor, be dynamic, or include only static obstacles. Finally, path planning consists of computing the best trajectory to reach a destination considering the estimated localization in the reconstructed map. Thus, it is highly dependent on the first two functionalities and

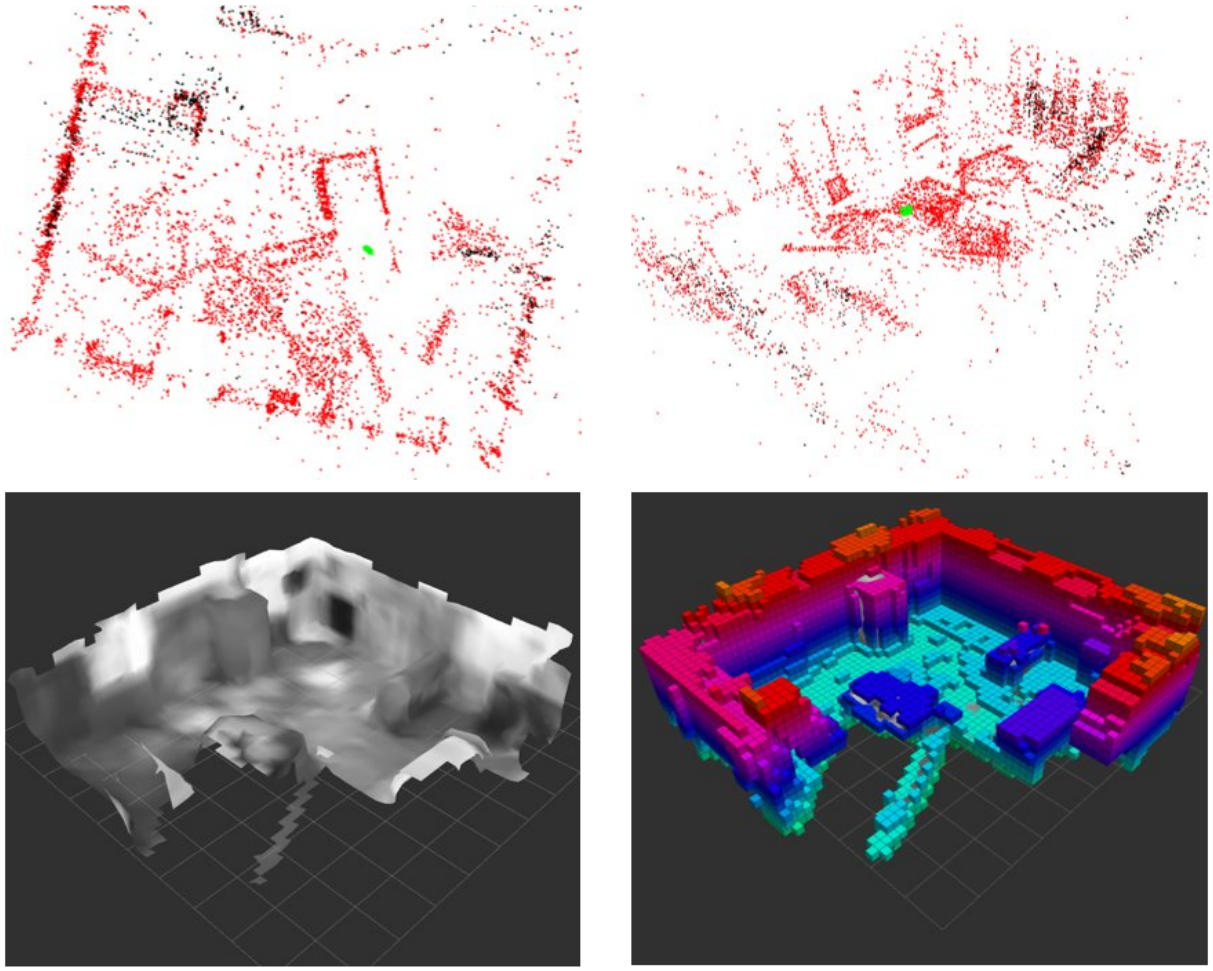


Figure 1.2 – Various map representation: 2D points (top-left), 3D point cloud (top-right), 3D mesh (bottom-left), voxels (bottom-right).

takes into account multiple criteria such as safety, speed, and length. This thesis will specifically focus on the tasks of localization and environment mapping.

Localization

A typical use case is the localization of a robot that may be affected by extrinsic or intrinsic limitations. Regarding intrinsic constraints, some applications prevent the use of some sensors, typically active sensors such as LiDARs or RGB-D cameras, because of energy consumption or discretion criteria. As for today, most systems rely on extrinsic sources such as Global Navigation Satelite System (GNSS)-based localization which is not guaranteed indoors, underground, or in zones of conflict. In fact, satellite reception

cannot be assured in all areas, or the signal can even be spoofed. Moreover, the granularity of this kind of geo-localization system is not adapted for indoor navigation and obstacle avoidance. The review [32] provides a comprehensive comparison of various indoor and outdoor localization technologies. For indoor environments, alternatives to GNSS include solutions based on Wireless Access Point (WAP), Bluetooth Low Energy (BLE), Radio Frequency Identification (RFID), or Ultra-wideband (UWB). While some of these systems give a higher precision, sometimes for a higher cost, they all involve a strong external dependency for the robotic platform. In contrast, during a rescue or military operation, an autonomous system capable of operating even in the absence of external sources is preferred and sometimes required. Thus, localization from passive sensors is of high interest for the exploration of unknown buildings or undergrounds by robots or first responders.

Mapping

On the other hand, mapping is also a critical function for rescue operations. For instance, in the context of first responders' intervention, the autonomous exploration and real-time mapping of a dangerous collapsed building is crucial. Indeed, in this scenario, the structure of the surrounding environment is a priori unknown and irregular, so identifying hazards is critical before sending rescuers [15], [16]. First responders are required to do a fast site survey to build a map of the building and gather as much information as possible. To this end, fly assistance or even autonomous navigation is essential. It allows the operator to fly the drone in difficult lighting conditions, narrow paths, avoid obstacles, and focus on inspecting visual images. Indeed, manual control of a remote platform is generally based on images from an onboard camera with a limited field of view, which strongly affects the pilot's perception of the environment [30], [33]. Building a 3D map can provide the operator with a view including the surroundings of the platform [33]–[35], or allow an autopilot system to assist the navigation by avoiding obstacles [17], [36], [37]. In addition, the mapping of the area of operation can be used to efficiently schedule the intervention. It allows to locate victims and plan the optimal itinerary to rescue them. In practice, the choice of a representation for a reconstructed 3D map is essential. In fact, maintaining a global map in large-scale environments requires memory and algorithm optimization as computational load and memory grow with the size of the map [38]. Dense point clouds or high-resolution 3D meshes provide a superior level of precision, whereas obstacle avoidance for drone indoor navigation might not require such precision. In such cases, a coarse reconstruction such as a voxel [1]map (Figures 1.1 and 1.2) could be

sufficient as long as it provides metric and dense structure [39]–[41].

Selecting a drone for indoor navigation

Multiple types of robotic platforms are considered for USAR, surveillance, or reconnaissance operations [13], [14]. Robots enable the replacement of vehicles, airplanes, or helicopters for both indoor and outdoor missions. Without pilots onboard, these platforms can be controlled remotely and achieve the same missions with reduced risk. They are available in multiple sizes, powered either thermally or electrically, and capable of carrying a significant payload. Unmanned ground vehicles (UGVs) offer high payload capacity and long endurance but are less maneuverable. UAVs may be a preferable alternative, but despite their improved agility, they require more advanced piloting skills. The most common categories of drones are fixed-wing, multi-rotor, and single-rotor, whose characteristics are compared in [42] and summarized below.

Fixed-wing drones have a design resembling traditional airplanes. They are rather expensive but highly efficient in terms of endurance and range. They always fly forward and are also very fast, but this makes them more constrained in their movements. Fixed-wing drones can cover large areas in a single flight making them popular for long-range surveillance, aerial mapping, and environmental research.

Single-rotor drones, similar to traditional helicopters, are typically powered by gas, giving them extended autonomy and higher payload capacity. Their aerodynamic design and larger rotor have benefits in hovering, stability, and wind resistance. However, their mechanical complexity requires more maintenance, increases cost, and makes them harder to use. These drones are suited for tasks requiring long endurance, large payloads, and stable flight, such as heavy lifting, surveillance, search and rescue.

Multi-rotor drones are known for their versatility, agility, and precise control, making them ideal for close-quarters operations. They range down to very small sizes, are cost-effective, and are the easiest alternative to operate. However, being primarily battery-powered limits their payload capacity and flight duration. These drones are widely used for aerial photography, surveillance, 3D mapping, or entertainment. Their ability to hover and maneuver in confined spaces makes them suitable for indoor and urban environments.

Therefore, small multi-rotor drones are particularly attractive indoors, as they generally offer greater mobility than UGVs. Small drones, also known as Micro Aerial Vehicles

1. A voxel, or volumetric pixel, is a three-dimensional analog of a pixel. It represents a value on a regular grid in 3D space and is often used in computer graphics for 3D modeling and visualization.

Figure 1.3 – Quadrotor UAVs equipped with an embedded system and various sensors for the BlueSwarm system [23] (left) and the TeamAware project [25] (right).

(MAVs), have been widely used in the literature for indoor navigation, especially quadrotors [17], [30], [43]. However, MAVs usually have higher power constraints than UGVs as most of the energy is consumed by the motors. Thus, the choice of payload is essential and must consider the volume, weight, and power consumption of each device.

Payload content

Depending on mission requirements, a drone should be configured with an appropriate payload. Two examples are shown in Figure 1.3. The first example shows a drone suitable for outdoor use, able to carry about 1 kg of payload and larger sensors for 20 to 30 minutes. The second example features a smaller drone designed for indoor use that can carry up to 700 g of payload for about 10 minutes, illustrating the significant volume and capacity constraints. Thus, a standard drone setup includes a flight controller, wireless communications, GNSS, and an Inertial Measurement Unit (IMU). The additional equipment is selected to perform the mission as efficiently as possible while minimizing Size, Weight, Power, and Cost, commonly referred to as SWaP-C constraints.

In the case of autonomous navigation, and more specifically real-time 3D mapping, calculations must be carried out onboard to control the drone with minimum latency [43]. Therefore, an embedded system with sufficient computing resources is required. Systems

such as Odroid, NUC, or UP Board have been benchmarked for applications on drones [44]. However, when using visual sensors, and by extension computer vision algorithms, having a GPU is preferable, if not essential. The NVIDIA Jetson series is particularly adapted for this need [45], [46]. In a small format, such Systems on Modules (SoMs) feature a GPU and a range of hardware accelerators for Deep Learning (DL), camera acquisition, and video streaming.

Several sensors have been studied for drone indoor navigation or 3D mapping [17], [47]–[49]. While active sensors can typically measure depth directly, they tend to be bulky, expensive, and power-hungry. In contrast, passive sensors, which generally have better SWaP-C properties, present a greater challenge in deriving depth information from their output. Thus, the choice of sensor setup depends on the intended application. This thesis aims to provide drone autonomous navigation and 3D mapping capabilities for USAR scenarios. As we focus on small drone applications, energy efficiency is a major criterion for extending mission duration. Therefore, the ideal setup would rather include passive sensors [47], and preferably a monocular camera and an IMU when considering all the SWaP-C constraints [17], [30], [43]. However, this choice comes with inherent challenges in depth perception. For this reason, the following section briefly explores possible solutions for achieving dense 3D mapping in drone indoor navigation scenarios.

1.2 3D mapping for UAV navigation

In this work, we focus on the development of dense, metric[1], and real-time 3D mapping for autonomous UAV navigation, as accurate localization and environment mapping are essential for autonomous flight. While some approaches are able to build a dense and metric 3D map from sensor data, many are not optimized for real-time applications on small drones, particularly in challenging real-world scenarios like USAR.

Real-time localization and 3D mapping

Localization is a critical part that allows environment mapping and path planning. Typically, Visual Inertial Odometry (VIO) and Simultaneous Localization And Mapping (SLAM) address this problem [30]. VIO is dedicated to tracking the robot by building a short-term local map, which requires less computation. SLAM, on the other hand, builds and maintains a long-term global map, allowing for place recognition, better accuracy, and robustness, but using significantly more resources. Nowadays, VIO is usually preferred for real-time robot localization, whenever there is no specific need for mapping. SLAM is preferred when accurate global positioning is required, which is ensured by the loop closure procedure and maintaining a global map [50].

Another challenge is to create a dense and metric 3D map in real-time. The expected map should be metric in order to keep the drone at a safe distance from potential obstacles and retrieve a precise location of places and hazards. The map should also be sufficiently dense for navigation, especially in narrow areas and to detect thin obstacles. Finally, this mapping procedure should enable live operation by being executed in real-time. Structure from Motion (SfM) provides an offline solution to reconstruct a high-resolution 3D mesh. It performs a 3D reconstruction by processing together all the images acquired by the drone, and the estimated model is not guaranteed to be metric [51]. This method is too complex and cannot be considered for real-time mapping applications [52]. On the other hand, SLAM is an online method that works in real-time, which generally favors computation efficiency over map density. Finally, while SLAM has reached an increased level of technological maturity, its use has still not been extensively tested for some challenging drone navigation conditions, especially for monocular setups.

1. In computer vision and photogrammetry, metric scale refers to the scale at which real-world measurements are accurately represented in images or data. Achieving metric scale is essential for accurate 3D reconstruction and mapping.

Active vs passive sensors for SLAM

SLAM techniques can be classified according to the sensor types they employ. Active sensors emit signals or energy and measure the response to gather information about the environment. In contrast, passive sensors capture ambient energy to gather information about their surroundings. Below, the primary sensors used in SLAM are presented and their essential properties are summarised.

LiDAR (Light Detection And Ranging): These sensors emit laser beams and measure the time it takes for the beams to reflect off surfaces and return. This data is used to create detailed 3D point clouds of the environment. While they typically have poor SWaP-C properties, lighter versions have been introduced, but these tend to offer lower quality measurements [43].

RGB-D (Red, Green, Blue - Depth) Cameras: This type of camera combines traditional RGB color imagery with depth information. It uses structured light or time-of-flight technology to estimate depth, resulting in colored 3D point clouds. These sensors provide both visual and depth data, but they typically have higher power consumption and limited depth range, which restricts their usage to indoor applications.

Event Cameras: These novel sensors (also known as Dynamic Vision Sensor - DVS) do not capture traditional frames but instead detect changes in intensity over time. They are very sensitive to motion, offer high temporal resolution, and have a particularly low power consumption. Event cameras are especially suited for high-speed and dynamic scenes, such as fast drone maneuvers or tracking fast-moving objects. They also have a higher dynamic range making them more robust to high illumination changes and low lighting conditions. However, their weaknesses include the inability to provide data in the absence of motion, limited availability, higher cost, and the need for specialized algorithms to process their output.

Stereo Cameras: Comprising two cameras mounted at a fixed distance from each other, known as the baseline, stereo cameras derive depth information by analyzing the disparities between corresponding points in the images from both cameras. Nevertheless, this sensor type requires precise calibration, and the depth range is limited by the baseline distance, which also affects the overall size of the camera. Additionally, handling dual image streams from stereo cameras increases computational load and power consumption.

Monocular Cameras: Consisting of a single lens, these cameras capture 2D images and are commonly used for navigation and perception tasks. Although lacking inherent depth perception capabilities, they are preferred for their light weight, low cost, and low

power usage. Additionally, they are relatively easy to calibrate and have been widely studied in computer vision, making them a practical choice for a variety of applications.

IMU: This electronic device integrates multiple sensors to measure orientation, velocity, and acceleration. It typically includes an accelerometer that measures linear accelerations and a gyroscope that measures angular rates. Some IMUs also feature a barometer to measure atmospheric pressure, enhancing altitude estimation and overall accuracy. IMUs are prone to measurement noise and sensor bias, resulting in drift over time, particularly for lower-quality and affordable models. This drift can significantly impact the long-term accuracy of the data they provide, necessitating frequent recalibration or fusion with other data sources to maintain reliable measurements.

Sensors	**Pros**	**Cons**
LiDAR	• Accurate 3D point cloud • Long-range sensing • Suitable for outdoor	• Expensive • Relatively bulky and heavy • Power consumption • Vulnerable to weather conditions
RGB-D Camera	• Measure depth • Easier SLAM initialization • Dense 3D maps	• Limited outdoor use • Complex calibration • Power consumption • Interference with active sensors
Event Camera	• Low latency • High dynamic range • Suitable for fast motion tracking	• Price and availability • Requires specialized algorithms • No data without motion
Stereo Camera	• Depth estimation by stereo matching • Easier SLAM initialization	• Extrinsic calibration • Range limited by the baseline • More data to process
Monocular Camera	• Inexpensive • Small and lightweight • Easy calibration • Low power consumption	• Scale ambiguity • Complex SLAM initialization • No mapping for pure rotations
IMU	• Inter-frame motion estimation • Low latency and high frequency • Metric information for monocular SLAM	• Drift over time (sensor biases) • Visual-inertial calibration • Synchronization

Table 1.1 – Comparison of sensors for visual SLAM

These sensors have been extensively reviewed in the literature [30], [47]–[49] and their characteristics for SLAM applications are summarised in the table 1.1. LiDARs and RGB-D cameras have proven to be effective in producing 3D maps in various works [53]–

[57]. Nonetheless, when considering SWaP-C constraints alongside the discrete operation requirement, these active sensors are incompatible with real-time application on small lightweight drones.

Stereo cameras enable dense depth estimation via stereo matching, thus enabling some SLAM algorithms to reconstruct dense and metric 3D maps [12]. Nevertheless, in the context of the aforementioned limitations, stereo cameras are notable for their relatively high power consumption compared to other passive camera options.

In recent years, event cameras have attracted increasing interest for drone applications. A comprehensive survey [58] provides a detailed review of the literature in this field. In particular, it highlights several successful implementations of event-based SLAM, especially in scenarios involving high-speed motion and challenging lighting conditions. Furthermore, it covers depth estimation with event cameras, in which both monocular [59] and stereo configurations [60] have shown promising results. However, the major limitation of event cameras is their inability to provide data in the absence of motion or in a textureless environment. Given these aspects and the previously mentioned drawbacks, using this type of sensor for dense mapping presents significant challenges. Hence, while event cameras fall outside the primary focus of this work, their potential integration with visual cameras represents an exciting avenue for future research.

Nowadays, monocular cameras are commonly used for SLAM [48]. Such systems extract visual features such as keypoints and edges in successive frames to track camera movements and derive depth information by triangulating these features across multiple views. Nevertheless, the lack of direct depth perception results in scale ambiguity. Despite this, monocular SLAM offers advantages such as simplicity, easy calibration, and minimal hardware requirements.

IMU data are essential for the accurate estimation of drone motion between camera frames, especially in situations with indistinct visual features, motion blur, or uniform image textures. Furthermore, inertial measurements provide metric information that is crucial for scale recovery in monocular setups. Nonetheless, effective integration of these different sensors requires precise temporal synchronization and accurate calibration between the camera and the IMU.

Thus, visual SLAM combining a monocular camera and an IMU provides an optimal balance between lightweight hardware, cost, energy efficiency, and real-time performance. Previous investigations have explored monocular VIO or SLAM systems using other sensor configurations in scenarios involving high-speed motion or variable lighting conditions

[44], [61], [62]. Although monocular-inertial SLAM has reached a mature stage [38], its effectiveness in such challenging drone navigation contexts has not been fully explored yet. Moreover, while existing real-time methods deliver robust camera positioning, they mostly produce sparse 3D maps. Consequently, the densification of monocular SLAM has emerged as a significant area of interest, which is essential in our context of indoor drone navigation and obstacle avoidance.

Densifying monocular-inertial SLAM for drone navigation

Although monocular cameras are appropriate for real-time SLAM, their inherent depth ambiguity poses practical limitations. Initially, geometric methods for dense monocular SLAM were too complex and were consequently sidelined in favor of sparse methods prioritizing localization robustness [38]. Nevertheless, with the evolution of robust sparse SLAM methods, recent research has explored monocular SLAM densification by leveraging advances in deep learning. The most effective approach integrates monocular depth estimation (MDE), which predicts a dense depth map from a single frame, into the SLAM process. In particular, MDE has seen significant advances in recent years, largely driven by the capabilities of Deep Neural Networks (DNNs) [63].

These approaches have primarily been applied in purely monocular SLAM settings without metric data. Some works have attempted either to infer metric depth maps [64] or to predict depth up to a scale factor [65], [66]. A few techniques have been developed to construct dense and metric maps through the use of monocular-inertial SLAM, resulting in the best performance in this domain [6], [7]. These advancements are notably attributed to the implementation of multi-view strategies, which enhance accuracy but also increase complexity. Most of the existing methods in this area use large DNN architectures, which may limit their potential application in embedded systems. Additionally, these techniques are often tailored to specific benchmarks due to their reliance on supervised learning approaches, whose primary focus is to achieve precise 3D reconstructions.

In contrast, this thesis focuses on drone indoor navigation, where a coarse 3D representation is sufficient for obstacle avoidance accuracy, and does not require high-precision 3D mapping. This loosened constraint paves the way for more lightweight approaches to achieve dense, metric, real-time 3D mapping by densifying monocular-inertial SLAM.

1.3 Objective of this thesis

The scope of this thesis is the 3D mapping of indoor scenes from a lightweight drone equipped with a monocular camera in the context of USAR and military operations. The built 3D map can be a coarse representation like a voxel map, but it needs to be dense, metric, and done in real-time. This enables autonomous navigation of the drone and provides the operator with a fast 3D reconstruction of the terrain. We focus our study on the densification of monocular-inertial SLAM in this specific context.

Contributions

To what extent can state-of-the-art visual SLAM be applied to drone navigation in challenging conditions?
We propose a workflow based on monocular-inertial SLAM for 3D mapping which requires a baseline method adapted to our context. Therefore, the initial inquiry that we address revolves around the applicability of current advances in visual SLAM to the domain of drone navigation, particularly in challenging scenarios. In practice, indoor drone movements often involve motion blur, low lighting, or high illumination changes. These disturbances can have a significant impact on the robustness of the algorithms. Numerous methodologies have been developed, typically evaluated against established SLAM benchmarks, some of which include drone-acquired data under standard conditions. Other studies have evaluated visual SLAM algorithms for indoor navigation of drones and robots, focusing primarily on RGB-D and Stereo cameras, while others have become obsolete or less relevant.

Our contribution, published in [67], is to identify a representative, state-of-the-art monocular baseline method, and evaluate it under conditions that include low illumination, illumination changes, and high-speed motions. We assess the level of robustness of the chosen baseline by illustrating its ability to provide accurate global positioning in real-time in all of the specified scenarios. We also note limitations in its IMU initialization procedure, which requires specific conditions. Our study enables us to conclude that the selected baseline is well-suited for providing metric scale localization and sparse 3D mapping in real-time, necessary for our workflow.

How can we perform monocular SLAM densification with metric scale in real-time on an embedded system ?
We introduce a pipeline in [67] designed for real-time processing on an embedded system,

aiming to densify monocular-inertial SLAM by capitalizing on monocular depth estimation and constructing a dense and metric voxel map. Alternative approaches focus on either improving robustness and accuracy or reconstructing a dense 3D mesh with detailed precision. Our objective is to use monocular SLAM densification for drone navigation.

To achieve this, instead of using point clouds or 3D meshes, we opt for creating an occupancy map using voxels, as it is ideal for our purpose. This particular structural choice is notable for its potential memory-efficient storage, and its ability to be efficiently constructed and updated incrementally through raycasting techniques. By partitioning space into a grid, each voxel can efficiently replace a full set of points, indicating whether a region is occupied or not. Additionally, most related works employ a pure monocular setup, constructing a map with relative scale. In contrast, we use a monocular-inertial setup for SLAM, allowing us to estimate the absolute scale. The resulting sparse depth, being metric, is leveraged to correct the predicted dense depth map.

How can we combine SLAM sparse and metric depth to recover the absolute scale of a predicted dense depth map?
Expanding upon this pipeline, we conceived and developed a loosely coupled approach that integrates sparse yet metric SLAM depth data with the dense depth map obtained from MDE. Most of the works on monocular SLAM densification employ a supervised approach with restricted generalization abilities. They commonly rely on complex architectures or use a pure monocular baseline without metric information. These solutions typically aim at improving the robustness in textureless scenes or performing precise 3D reconstruction. As a result, their primary concern is generally neither real-time processing nor metric scale.

Our methodology presented in [68] involves scale recovery from SLAM sparse landmarks estimated via multi-view geometry. Notably, we have proposed a decoupled framework that minimizes the impact on the underlying SLAM baseline frame rate. We evaluate our approach against relevant works using depth estimation metrics and provide both quantitative and qualitative assessments using specific indoor datasets. Furthermore, leveraging the resulting scaled dense depth map, we employ raycasting to construct and continually update a voxel map, achieving multi-view volumetric depth refinement. We demonstrate the capabilities of our framework to build a voxel map suitable for drone navigation and show its limitations through a qualitative analysis.

Outline of the thesis

Following this introduction, the thesis will be structured into the following sections:

— **Chapter 2** reviews the current landscape in Visual SLAM. We analyze conventional techniques, observing the shift from dense methods to efficient and robust sparse approaches that can carry out real-time execution. Furthermore, we delve into the field of monocular SLAM densification, emphasizing the application of DNNs to predict dense depth maps from single RGB images. In this context, we explore recent enhancements in the area of monocular depth estimation. The chapter also introduces key metrics and datasets necessary for evaluating SLAM and MDE.

— **Chapter 3** introduces our proposal for a pipeline to achieve real-time, dense, and metric 3D mapping through monocular-inertial SLAM. In this chapter, we explain the pipeline's architecture and the reasoning behind our design choices, with a particular focus on their practical applications in obstacle avoidance and autonomous drone navigation. Additionally, we selected a state-of-the-art SLAM baseline, justifying our decision with benchmarking against challenging conditions.

— Within **Chapter 4**, we present a method for determining the absolute scale of a dense depth map inferred via deep learning by utilizing the sparse depth data from SLAM. Our approach is evaluated and compared with related research. Furthermore, we explore the implementation of grouped raycasting techniques to construct a voxel map while carrying out multi-view volumetric refinement. The implementation results of the complete framework are presented and analyzed.

— In the final chapter, **Chapter 5**, the initial part summarizes our contributions in addressing our research questions, while the second part explores potential directions for enhancing these concepts.

Chapter 2

State of the art in monocular SLAM densification

In recent years, SLAM has emerged as a fundamental technology in the field of robotics and computer vision. With the rapid advancement of visual sensors and computational capabilities, monocular SLAM has garnered significant attention thanks to its simplicity and applicability in a wide range of real-world scenarios. However, conventional monocular SLAM techniques are facing challenges in accurately reconstructing dense and detailed maps from a single camera in real-time, limiting their potential in drone autonomous navigation applications.

This chapter begins by defining the mathematical basis for visual SLAM, providing a foundation for understanding the core principles of the field. Next, we review the main methods of conventional visual SLAM and trace their evolution over time. In the sequel, we examine the latest progress in densifying monocular SLAM, highlighting limitations, and exploring contemporary techniques that aim to produce more informative and robust maps. The quest for densification involves harnessing the power of DNNs for depth prediction from a single image. Hence, we also examine the present progress of research in the MDE field. Finally, we recall the standard metrics and notations employed within these specific domains, along with the common benchmarks used to evaluate them.

Ultimately, the synthesis of the state-of-the-art in densifying monocular SLAM will serve as a solid foundation for the subsequent chapters of this thesis. Building on this knowledge, we propose new approaches and conduct a series of extensive experiments to address existing challenges and contribute to the advancement of monocular SLAM and its implications for autonomous drone navigation.

2.1 Theoretical primer for Visual SLAM

In this section, we present basic mathematical definitions and notations that will be used throughout this thesis. For clarity, we will use boldface to represent vectors and matrices. We review camera principles and introduce selected camera models that define the relationship between the 3D scene and the resulting camera image. Additionally, we give a brief overview of 2-view epipolar geometry for use in our specific context. Following this, we describe the operational model of an IMU to better understand the strengths and limitations of this sensor, particularly the effects of noise and biases. Finally, we briefly outline the prevailing formulation of the SLAM problem today and describe the process of joint estimation of camera poses and 3D landmarks' positions.

2.1.1 Camera modeling

Rigid body transformation

The main distinction between linear and affine transformations is their handling of translations. Linear transformations do not include translations and strictly preserve linearity and the origin. In contrast, affine transformations involve both linear operations and translations, allowing for a wider range of transformations that include linear transformations as a subset.

In reality, nearly all object transformations will involve both linear (such as rotation) and translation operations. Therefore, a coordinate transformation becomes affine rather than linear when considering a complete rigid-body motion. To address this, we typically use homogeneous coordinates, which involve adding a scalar coordinate to express translations as linear transformations. Therefore, in the case of 3D geometry, we would represent a point as $[X, Y, Z, 1]^T$ and a vector as $[X, Y, Z, 0]^T$, where $(X, Y, Z) \in \mathbb{R}^3$.

A rigid body motion is represented by a transformation in $SE(3)$, the Special Euclidean group in three dimensions. The rigid body transformation $\mathbf{T}$ can be expressed as a rotation $\mathbf{R}$ in $SO(3)$, the Special Orthogonal group, and a translation $\mathbf{t} \in \mathbb{R}^3$ and formulated by a homogeneous 4×4 matrix:

$$\forall \mathbf{T} \in SE(3), \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

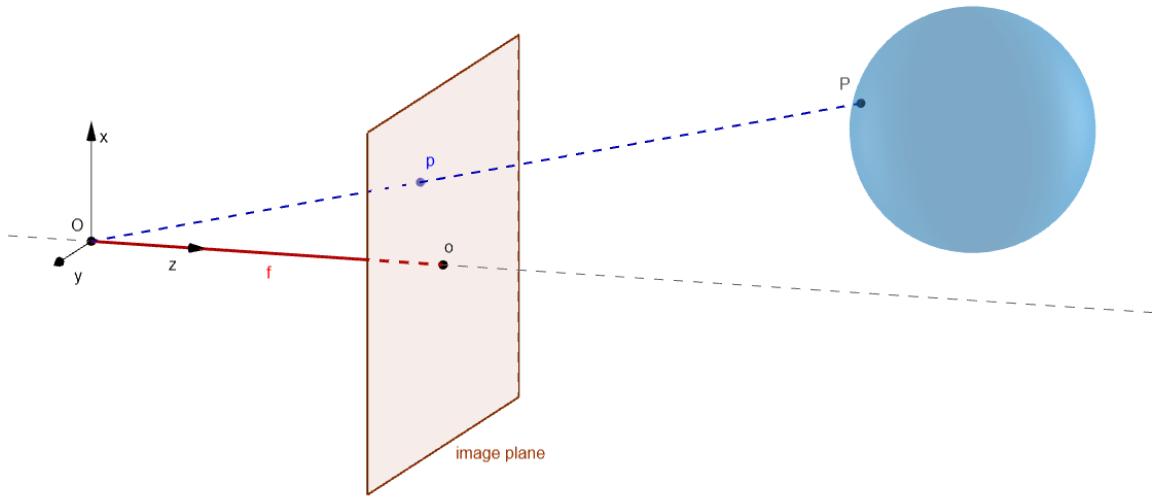


Figure 2.1 – Frontal pinhole imaging model, adapted from [69]: The image plane is located at the focal length f from the camera center **O**, with the optical axis intersecting at the principal point **o**. The pixel **p** represents the projection of a 3D point **P** onto the image.

Camera models

A camera is an optical device consisting of a lens and a sensor that captures image intensities from a scene. The sensor is arranged in a grid pattern where each cell accumulates light intensity during a given exposure period. The scene observed by the camera through the lens is projected onto a 2D surface known as the image plane, which is located at a distance f from the center of the camera (see Figure 2.1), termed as focal length.

A camera model is a mathematical representation that describes how a camera captures the 3D world and projects it onto a 2D image. It involves the camera's intrinsic parameters (related to its optics and sensor) and extrinsic parameters (related to its position and orientation in 3D space). Intrinsic parameters are usually provided by the manufacturer or can be estimated via a process known as calibration. For a detailed explanation of the camera calibration procedure, readers can refer to Section 6.5 in [69]. The pinhole camera model is a standard model used in computer vision and computer graphics which simplifies the camera optical system to a single point called "pinhole". In this model, light rays from the scene pass via the pinhole and yield inverted images onto the camera image sensor.

Let's define a 3D point $\mathbf{P} = [X, Y, Z, 1]^T$ in the camera frame, whose coordinates in the world reference frame are denoted by $\mathbf{P}_W = [X_W, Y_W, Z_W, 1]^T$. The Euclidean

transformation $\mathbf{T}_{C,W} \in SE(3)$ represents the transformation from the camera frame to the world frame, which corresponds to the pose[1] of the camera frame $\mathbf{C}$ in the world frame. The inverse of this transform, $\mathbf{T}_{C,W}^{-1} = \mathbf{T}_{W,C}$ is defined by the rotation $\mathbf{R}_W$ and translation $\mathbf{t}_W$. The projection of $\mathbf{P}$ onto the image plane is denoted by the pixel $\mathbf{p} = [u, v, 1]^T$ and can be expressed by the following equations, where $\lambda \in \mathbb{R}^*$ is a scalar:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x s_x & f_y s_\theta & o_x \\ 0 & f_y s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_W & \mathbf{t}_W \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{2.2}$$

$$\lambda \mathbf{p} = \mathbf{K}\mathbf{\Pi}\mathbf{T}_{W,C}\mathbf{P}_W$$

Here, $\mathbf{\Pi}$ is the standard projection matrix. The camera extrinsic parameters are represented by the rigid body transformation $\mathbf{T}_{C,W}$, which transforms the coordinates of $\mathbf{P}_W$ to the camera frame, giving $\mathbf{P} = \mathbf{T}_{W,C}\mathbf{P}_W$. The matrix $\mathbf{K}$ comprises the camera intrinsic parameters and is commonly referred to as the calibration matrix: f_x and f_y are the focal lengths along the $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ axes, (o_x, o_y) is the principal point coordinates (image coordinates of the optical center), (s_x, s_y) is the inverse size of a pixel and s_θ is the skew factor. In most cameras, pixels are square or nearly square, resulting in equal focal lengths ($f_x = f_y = f$) and a negligible skew factor ($s_\theta \approx 0$). Assuming that the camera frame coincides with the world frame (or that the pose $\mathbf{C}$ is known), Equation (2.2) becomes:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f s_x & 0 & o_x \\ 0 & f s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.3}$$

$$\lambda \mathbf{p} = \mathbf{K}\mathbf{\Pi}\mathbf{P} = \pi(\mathbf{P})$$

Here, π is the function projecting a 3D point onto pixel coordinates in the image plane. This expression can be further developed to:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = Z \begin{bmatrix} f s_x X + o_x \\ f s_y Y + o_y \\ 1 \end{bmatrix} \tag{2.4}$$

1. A pose represents the position and orientation of an object or a coordinate frame in 3D space, typically consisting of a translation vector and a rotation matrix or quaternion.

This expression exposes the monocular depth ambiguity. Specifically, as the depth Z is not observable, it is replaced by a positive scalar λ (where $\lambda = Z > 0$), since we consider the points in front of the camera:

Camera calibration is a process used to determine the intrinsic parameters of a camera. If required, its extrinsic parameters are also determined, especially in scenarios such as camera-IMU calibration to determine the camera pose relative to the IMU frame. A standard practice in robotics is to assume that the body frame of the drone is aligned with the IMU frame. Camera-IMU calibration allows for aligning the motion data from the IMU with the visual data from the camera, which helps to accurately predict the camera's motion between frames. Regarding the intrinsic parameters, the pinhole model does not account for lens radial distortion, which is prevalent in real-world cameras, especially those with a wide field of view. To address this, more complex camera models, such as the radial-tangential, the fisheye [70] (also known as equidistant), or the double sphere [71] models, are also used for more accurate representations of camera optics.

In practice, calibration parameters correct lens distortion and allow the conversion of 2D image points into normalized image coordinates. Despite the inherent depth ambiguity in monocular vision, a camera model with calibrated parameters expresses the relationship between image pixels and corresponding points in the scene, up to a scale factor. As we will see, this relationship can be exploited in multi-view geometry, in particular for the triangulation of 3D points.

Epipolar geometry of two views

In the following, we consider a calibrated camera with known intrinsic and extrinsic parameters, the latter implying that the camera frame is finally set to coincide with the world coordinate frame. Thus, we can assume an ideal perspective camera, simplifying Equation 2.3:

$$\lambda \mathbf{p} = \mathbf{\Pi P} \tag{2.5}$$

Epipolar geometry defines the geometric constraints that exist between two cameras observing a 3D scene from different viewpoints or equivalently to a monocular camera capturing two images at successive positions, under the assumption of a static scene. Thus, epipolar geometry can be used to facilitate establishing correspondences between features in different camera frames. By establishing feature correspondences between frames,

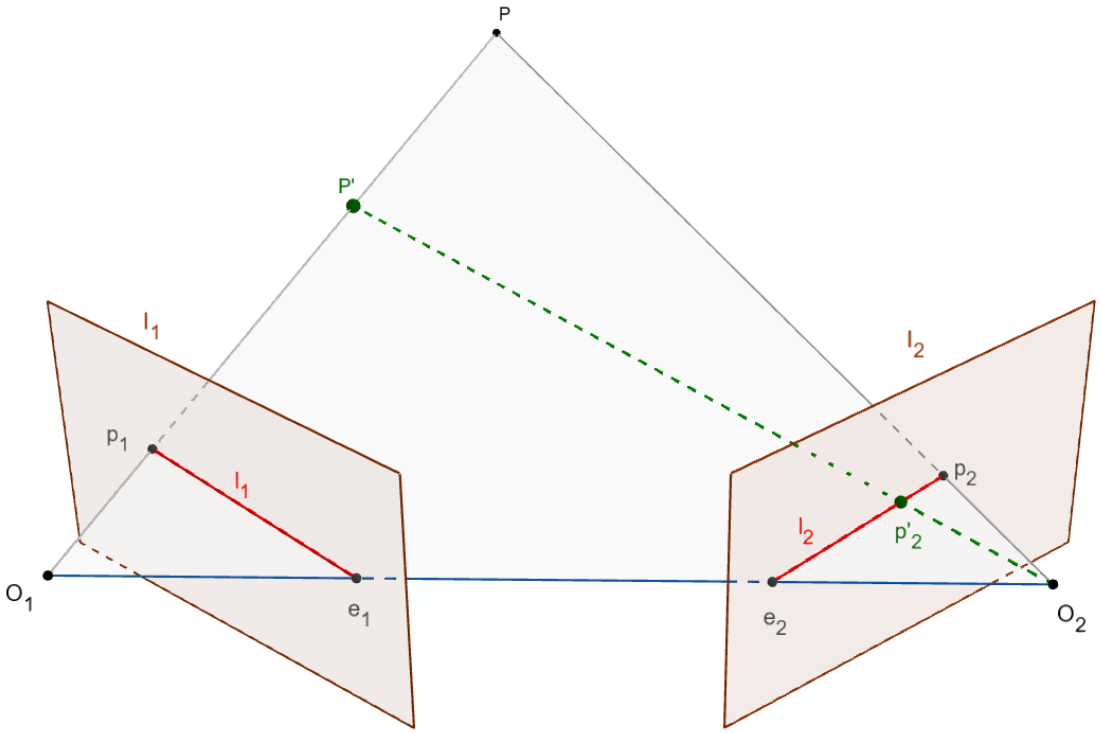


Figure 2.2 – Representation of the epipolar constraint: Two cameras, positioned at $\mathbf{O}_1$ and $\mathbf{O}_2$, capture the same scene and project it onto their respective image planes I_1 and I_2. The points $\mathbf{p}_1$ and $\mathbf{p}_2$ are the projections of the 3D point $\mathbf{P}$. The baseline, which connects the camera centers $\mathbf{O}_1$ and $\mathbf{O}_2$, intersects the image planes at the epipoles $\mathbf{e}_1$, $\mathbf{e}_2$. The epipolar lines $\mathbf{l}_1$, $\mathbf{l}_2$ are formed by the intersection of the epipolar plane $(\mathbf{O}_1, \mathbf{P}, \mathbf{O}_2)$ with the image planes. The point $\mathbf{P}'$ lies on the ray $(\mathbf{O}_1\mathbf{P})$, and its projection $\mathbf{p}'_2$ falls on the epipolar line $\mathbf{l}_2$.

SLAM algorithms can estimate the relative pose between consecutive camera poses.

As shown in Figure 2.2, an epipolar plane is defined by an observed point $\mathbf{P}$ and the centers of the two cameras. The epipolar lines result from the intersection of the epipolar plane with the image planes. Each camera has an epipole located at the point where the baseline, connecting the two camera centers, intersects with the image plane. Thus, the epipolar geometry allows to establish correspondences between points in the two camera images using the following equation:

$$\mathbf{p}_2^T \mathbf{E} \mathbf{p}_1 = 0 \tag{2.6}$$

where $\mathbf{E}$ represents the so-called essential matrix. It is defined by translation $\mathbf{t} \in \mathbb{R}^3$ and rotation $\mathbf{R} \in SO(3)$ describing the transformation between the two cameras. The

skew-symmetric matrix $\hat{\mathbf{t}}$, allowing to represent a vector cross product by ordinary matrix multiplication, allows $\mathbf{E}$ to be expressed as:

$$\mathbf{E} = \hat{\mathbf{t}}\mathbf{R} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.7}$$

A property of the essential matrix is that it projects an image point (or pixel) $\mathbf{p}_1$ from the first camera onto an epipolar line in the second one. Indeed, as illustrated in Figure 2.2, when the depth remains unknown, the image point is projected up to a scale factor thus describing a line. So, once the essential matrix is determined, the search for the pixel $\mathbf{p}_2$ corresponding to the pixel $\mathbf{p}_1$ in the image I_2 is restricted to the epipolar line $\mathbf{l}_2$.

By extension, the Fundamental Matrix describes the same relation without assuming calibrated cameras. It is derived from the essential Matrix and the camera intrinsic matrices $\mathbf{K}$ of the two cameras. It can be expressed as:

$$\mathbf{F} = \mathbf{K}_2^{-T}\mathbf{E}\mathbf{K}_1^{-1} \tag{2.8}$$

An interesting application for epipolar constraints is relative pose recovery between 2 image views. By establishing enough correspondences between the pixels of the two images, it is possible to use the above properties to estimate the essential matrix. Classical techniques such as SIFT [72], FAST [73], or ORB [74] are used to detect and describe highly distinctive features in the image and thus select keypoints. The descriptors ensure that keypoints can be matched robustly across various frames despite challenges presented by changes in illumination, scaling, rotation, and translation. From there, extracting the relative rotation and translation is a straightforward task. The accuracy of the feature matching is critical, as false matches will result in inaccurate pose estimation. Standard algorithms, such as the 8-point [75] or 5-point [76] algorithms, may be considered to estimate the essential matrix.

Error metrics

— Reprojection error:
As defined in the Equation (2.3), the function π projects a 3D point onto the image. Let's define the function g as the reprojection of a 3D point $\mathbf{P}$ based on the camera pose $\mathbf{C}_i \in SE(3)$ at the given time instant i:

$$g(\mathbf{P}, \mathbf{C}_i) = \pi(\mathbf{C}_i^{-1}\mathbf{P}) \tag{2.9}$$

The reprojection error quantifies the difference between the actual positions of 2D image points and their expected positions from the projection of known 3D world coordinates onto the image plane. It is a key metric for evaluating the quality of camera calibration, 3D reconstruction, and structure-from-motion processes. Suppose the 3D coordinates of a point $\mathbf{P}$ are known, and the corresponding pixel $\mathbf{p}$ in the current image has been detected through feature matching based on its position in a previous image. Then, considering all the detected pixels $\mathbf{p}$, the image reprojection error can be defined as the squared error between the detected pixels coordinates and those projected by a camera model:

$$\mathbf{e}_{\text{reproj}} = \sum_{\mathbf{p}} \|\mathbf{p} - g(\mathbf{P}, \mathbf{C}_i)\|^2 \tag{2.10}$$

Typically, the goal is to minimize the reprojection error by adjusting the camera model (intrinsic and extrinsic parameters). Smaller reprojection errors indicate a better fit of the camera model to the observed data, which implies more accurate camera parameter estimation.

— Photometric error:

Let I be an image. Then, the function $I : \mathbb{R}^2 \to \mathbb{R}$ is defined such that for any pixel coordinate $\mathbf{p} \in \mathbb{R}^2$, $I(\mathbf{p}) \in \mathbb{R}$ represents the intensity value at that coordinate. We also define the inverse function of π from (2.3) and (2.4), which backprojects pixel coordinates $\mathbf{p} = [u, v]$, combined with its respective depth d, into 3D space:

$$\pi^{-1}(\mathbf{p}, d) = \begin{bmatrix} d\dfrac{u - o_x}{f s_x} \\ d\dfrac{v - o_y}{f s_y} \\ d \end{bmatrix} \tag{2.11}$$

The photometric error measures the difference in pixel intensities (or colors) between corresponding pixels across two or more images and is relevant to image alignment, where the goal is to infer the relative camera pose between different viewpoints by minimizing the photometric error. Let's consider a monocular camera that captures images of a scene from various overlapping viewpoints. Given a pixel $\mathbf{p}_k$ with depth

d_k in the image I_i, and $\mathbf{T}_{j,i}$ as the relative camera pose of I_i with respect to I_j, the image photometric error is given by:

$$\mathbf{e}_{\text{photo}} = \sum_k \left\| I_i(\mathbf{p}_k) - I_j(\pi(\mathbf{T}_{j,i}\pi^{-1}(\mathbf{p}_k, d_k))) \right\| \tag{2.12}$$

This error is typically computed over all pixels in an image, providing dense alignment and better robustness in featureless regions. Nevertheless, it is vulnerable to illumination changes and occlusions, and may not be as reliable in textureless areas.

2.1.2 IMU modeling

An IMU measures linear acceleration $\vec{\mathbf{a}}$ in $m.s^{-2}$ along three axes via an accelerometer and angular velocity $\vec{\boldsymbol{\omega}}$ in $rad.s^{-1}$ about the same axes via a gyroscope. However, raw data is often subject to bias and sensor noise. The sensor noise is assumed to be Gaussian noise with known covariance matrices ($\mathbf{Q}_a$ and $\mathbf{Q}_\omega$). Bias is an offset and can be static or affected by various factors. In practice, it is modeled as a random walk driven by a white noise process [77], [78]. Noise and bias parameters can be supplied by the manufacturer, but may be refined by calibration procedures for higher accuracy.

To use IMU data for navigation, it is essential to perform integration. Integrating linear acceleration provides estimates of velocity and position while integrating angular velocity produces an estimate of orientation or attitude. Nevertheless, bias compensation and noise filtering are necessary during this process to maintain accuracy.

$$\mathbf{a}_{\text{raw}} = \mathbf{a}_{\text{gt}} + \mathbf{b}_a + \mathbf{n}_a \tag{2.13}$$

$$\boldsymbol{\omega}_{\text{raw}} = \boldsymbol{\omega}_{\text{gt}} + \mathbf{b}_\omega + \mathbf{n}_\omega \tag{2.14}$$

In the above equations, $\mathbf{a}_{\text{raw}}$ and ω_{raw} represent the raw measurements of linear acceleration and angular velocity, respectively. $\mathbf{a}_{\text{gt}}$ and $\boldsymbol{\omega}_{\text{gt}}$ denote the true values without any errors. The terms $\mathbf{b}_a$ and $\mathbf{b}_\omega$ are the biases for the accelerometer and gyroscope, respectively, while $\mathbf{n}_a$ and $\mathbf{n}_\omega$ capture the Gaussian noise associated with each sensor.

IMUs inherently suffer from drift, where small measurement errors accumulate over time, especially during integration, leading to significant deviations in estimated trajectories or orientations. To mitigate this, filtering or fusion techniques such as the Kalman filter or sensor fusion with external measurements such as GPS or visual data are used to

provide more accurate and reliable navigation estimates.

Pre-integration provides a refined approach to handling IMU measurements in vision-inertial systems. Instead of integrating raw IMU data at each time step, which can be computationally demanding and prone to errors, pre-integration aggregates this data over a fixed interval. This approach consolidates the effects of acceleration and rotation over that interval into a single relative motion. This enables a more compact representation and efficient optimization when fusing IMU data with visual observations. As demonstrated in [79], the use of Lie algebra ensures mathematical consistency and stability during these operations, making pre-integration particularly relevant for long-term navigation tasks.

2.1.3 SLAM problem representation

In this part, we explore two fundamental methodologies that have become central to the field of SLAM: Pose Graph Optimization (PGO) and Bundle Adjustment (BA). These approaches build upon the concepts of camera and IMU modeling discussed in earlier sections. We will briefly present how PGO and BA articulate the SLAM problem [80], [81], providing insight into the complexities involved, particularly in the context of real-time dense mapping.

We consider a moving drone equipped with a monocular camera that is rigidly attached to the drone. Consequently, the pose of the drone can be seamlessly related to the pose of the camera. Let us denote the pose of the camera in the world frame at time instant i as $\mathbf{C}_i \in SE(3)$. This yields the trajectory formed by the sequence of camera poses: $\{\mathbf{C}_i\}_{i=1}^{N}$. Similarly, we can define the sequence of captured images as $\{I_i\}_{i=1}^{N}$.

Pose graph optimization

The PGO is commonly used as a standard representation of the SLAM problem [80], [82], [83]. As illustrated in Figure 2.3, nodes represent the camera poses. Edges between nodes represent the spatial constraints that can arise from sensor measurements. In addition, when the algorithm recognizes a previously visited location, it identifies a loop closure and adds a constraint (an edge) to the graph between non-consecutive nodes. This helps to correct trajectory drift.

In PGO, the objective is to refine the node values so that the implied relative transformations better match the measurements represented by the edges. This is achieved through an alignment process expressed as a cost function, usually applied as a nonlinear

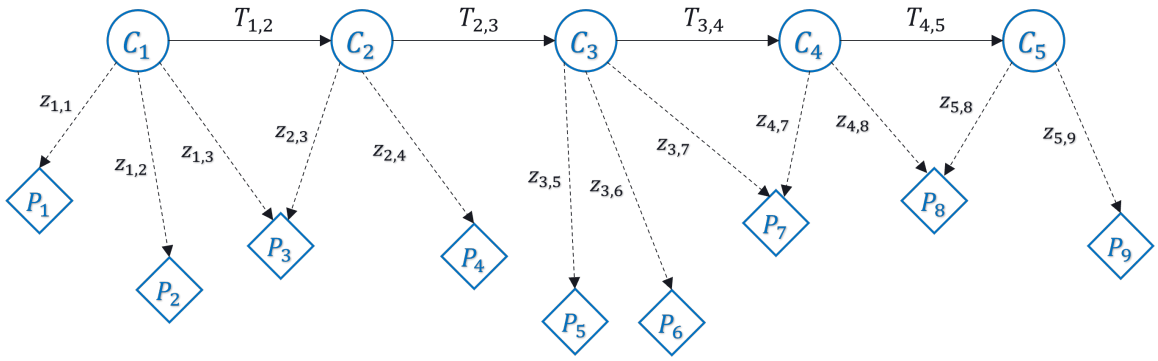


Figure 2.3 – Pose graph representation of the SLAM problem. The graph consists of nodes denoted by $\mathbf{C}_i$, for the drone (or camera) pose, and some variants also include 3D landmarks $\mathbf{P}_k$. Regarding the edges, $\mathbf{T}_{i,j}$ represents the transformation between time instant i and j, while $\mathbf{z}_{i,k}$ is the measurement of the landmark k at time instant i.

least squares problem. Well-known solvers such as Gauss-Newton or Levenberg-Marquardt are used to solve this problem. As a result, the trajectory of the drone is refined and the errors, especially those due to loop closure[1], are distributed over the entire trajectory.

Considering a graph with nodes representing only camera poses, and focusing on camera pose estimation, the associated cost function is:

$$\arg\min_{\mathbf{C}_i,\mathbf{C}_j} \sum_{i,j} \|\mathbf{C}_i - \mathbf{T}_{i,j}\mathbf{C}_j\|^2 \tag{2.15}$$

The relative pose $\mathbf{T}_{i,j}$ is an edge of the graph that represents spatial constraints based on sensor measurements that do not change during optimization. Nodes, represented by camera poses $\mathbf{C}_i$ and $\mathbf{C}_j$, are adjusted to best fit the constraints given by the edges.

Beyond pose graphs, factor graphs represent a more advanced modelization that explicitly represents a probabilistic optimization problem. In this approach, variable nodes represent unknown values, whereas factor nodes are reflective of constraints on subsets of these variables. The edges in this type of graph connect only nodes of different types. This formulation facilitates the use of incremental solvers such as iSAM2 [84], which optimize computational effort by focusing on the graph segments most affected by recent data. Additionally, such models typically allow inertial data to be incorporated into the graph, as well as IMU parameters that are continuously refined [1], [2], [79]. The tutorial

1. A process in SLAM that detects when the robot revisits a previously observed location, allowing for corrections to the estimated trajectory and map by aligning the current observation with the previous one. This reduces drift in long-term navigation.

in [85] provides a comprehensive explanation of factor graphs, including examples of its application in SLAM using the GTSAM toolbox [86].

Bundle Adjustment

Bundle Adjustment is a refined optimization technique commonly used in visual SLAM to simultaneously refine camera poses and 3D landmark positions [50], [81], [83], [87]. Chapter 11 of [69] provides a detailed presentation of the whole process, from feature detection to projective reconstruction. Here we present a brief overview of the optimization problem, particularly in the context of SLAM. Basically, the idea is to minimize the reprojection error (as defined in Equation (2.10)) between observed feature points in the images and their predicted projections from the current state estimate. In the context of SLAM, BA expands on the PGO problem by incorporating landmarks into nodes, thereby providing additional constraints to optimize both the trajectory and the map. This yields the following cost function:

$$\arg\min_{\mathbf{P}_k,\mathbf{C}_i} \sum_{i,k} \left\| \mathbf{p}_i^k - g(\mathbf{P}_k, \mathbf{C}_i) \right\|^2 \tag{2.16}$$

This equation seeks to refine the 3D positions of all landmarks, denoted by $\mathbf{P}_k$ with k indexing over all landmarks, and the camera poses, $\mathbf{C}_i$ with i indexing over all images or time. The variable $\mathbf{p}_i^k$ denotes the observed projection of the k-th landmark in the i-th image. Meanwhile, the function $g(\mathbf{P}_k, \mathbf{C}_i)$ computes the expected projection of that landmark based on the current estimates of its 3D position and camera pose. The objective is to reduce the difference between the observed and predicted projections for all images and landmarks, resulting in an optimized reconstruction of the camera trajectory and scene.

In SLAM, features are detected and tracked across frames, resulting in 2D correspondences. System initialization is an essential initial step, which creates an initial estimate of the relative camera transformations, typically derived from the Essential, Fundamental, or Homography matrices [88], or by using IMU data [89]. Then, based on the epipolar constraints discussed in section 2.1.1, the 3D positions of these features are triangulated. Specifically, a point $\mathbf{P}$ is determined as the intersection of rays ($\mathbf{O}_i\mathbf{p}_i$) projected from different camera views. Considering the inherent uncertainties in this initial triangulation resulting from both pose estimation errors and feature tracking inconsistencies, BA refines the 3D point estimates by simultaneously optimizing camera poses and landmark

positions, thereby improving the accuracy of the triangulated points.

Therefore, considering these approaches, we can see that the construction of a dense 3D map requires the integration of landmarks corresponding to each pixel in the images into the chosen graphical model as new nodes. Such integration results in a significant increase in the underlying BA optimization's complexity. In the following section, we will review various SLAM methods that successfully build on these classical techniques, highlighting their ability to achieve real-time and robust localization performance when constructing a sparse map. Afterward, we will explore novel strategies to achieve dense monocular SLAM.

2.2 Conventional Visual SLAM

The challenges of SLAM and VIO have been effectively addressed for several years now, and their solutions have been widely integrated into industrial products such as warehouse robots, Microsoft HoloLens, Google Tango, and Oculus Quest [38], [90], [91]. However, there are still some challenges, as highlighted in the review by Cadena et al.[38]. Based on that study, long-term data association addresses robustness and scalability, as outliers in data association can severely corrupt the system. Map representation and semantic reasoning optimize map storage and account for scene objects. Deep learning methods tend to replace intermediate stages such as scene depth estimation or inter-frame pose regression. Finally, new sensors are being investigated, such as range, light field, or event-based cameras. All these topics have been motivated by real-world applications in robotics where SLAM can be applied. This section explores conventional SLAM methods relying on passive sensors and relevant to drone navigation. Although these methods can effectively track drone localization, they are limited in achieving dense 3D mapping with monocular setups, a challenge that can now be addressed by leveraging deep learning capabilities.

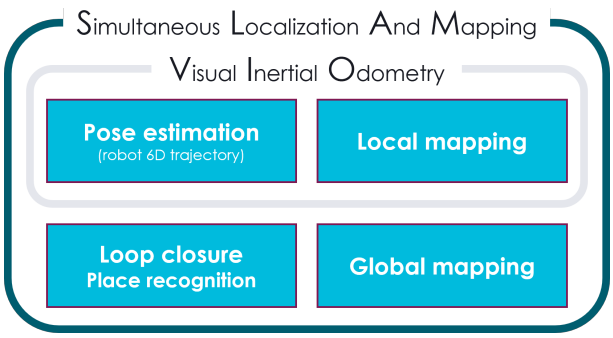


Figure 2.4 – Diagram illustrating the connection between SLAM and VIO: VIO is typically considered as a subset of SLAM, focusing on pose estimation and local mapping, while SLAM extends these functionalities with global mapping and loop closure detection.

Both visual SLAM and VIO estimate camera poses for every frame and triangulate or update the positions of 3D landmarks. However, as illustrated in Figure 2.4, VIO performs data association only over a local map, ensuring local trajectory consistency but leaving accumulated drift uncorrected. In contrast, SLAM carries out long-term data association and thus maintains a global map. When re-observing landmarks, it can correct the accumulated drift through a process called loop closure [50].

In the remainder of this section, we discuss the main categorizations of SLAM approaches and the integration of inertial data providing metric information and enhancing performance. Subsequently, we review some of the reference methods in conventional visual SLAM. Table 2.1 provides a comparison of various SLAM techniques based on their sensor configurations, their ability to estimate the absolute scale and to reconstruct a dense 3D map.

Sensors	**Features**	**Constraints**	**Density**	**Scale**	**Methods**
Mono	• Passive sensor • Cheap • High resolution • Minimal calibration • Classical algorithms	• Scale ambiguity • No mapping under pure rotation • Difficult initialization	• Sparse	Relative	PTAM [92], ORB-SLAM [87], DTAM [93], LSD-SLAM [94], VITAMIN-E [95], ORB-SLAM 3 [1]
Mono + IMU	• Metric measures • IMU high rate • Inter-frame motion estimation	• IMU drift and biases • Visual-Inertial calibration • Synchronization	• Sparse	Absolute	ORB-SLAM VI [96], VINS-Mono [97], OKVIS [98], SVO + GTSAM [79], ORB-SLAM 3 [1]
Stereo	• Depth from stereo vision • Easier SLAM initialization	• More data to process • Extrinsic calibration • Limited range precision due to limited baseline	• Sparse • Dense via stereo matching	Absolute	ORB-SLAM 2 [99], VINS-Fusion [100], OV2SLAM [101], SOFT2 [102], ORB-SLAM 3 [1]
Stereo + IMU	• Robustness • Speed up computation	• Stereo-IMU calibration and synchronization	• Sparse • Dense via stereo matching	Absolute	OKVIS [98], VINS-Fusion [100], Kimera [12], Basalt [2], ORB-SLAM 3 [1]

Table 2.1 – Comparative overview of conventional visual SLAM methods categorized by sensor configuration, highlighting their distinct features, constraints, typical map density, scale estimation capabilities, and notable reference implementations.

Filtering and Keyframe strategies

The seminal methods in this field relied on non-linear filtering, specifically **EKF SLAM** [103] or **Fast SLAM** [104], as formulated and analyzed in [82], [105]. These techniques estimate the state (camera pose and landmarks) in every frame. The process predicts the current state using a motion model and then refines this estimate using image measurements. The updated covariance provides an indication of the estimate's uncertainty. Since SLAM is inherently a non-linear problem, these methods use linearization, which introduces approximation errors. Moreover, because previous states are marginalized, these errors cannot be corrected afterward.

As demonstrated in [81], they are less accurate and less efficient than keyframe-based methods, which process each frame to track the camera pose, and then select and store some of them as keyframes. Camera pose is typically estimated by motion-only BA on

local frames, using fixed landmarks to optimize only the camera pose. Following this, full BA is applied to keyframes to jointly refine poses and landmarks using global map data.

Feature-based and Direct approaches

Two types of approaches prevail in visual SLAM. Feature-based methods mainly use the image reprojection error (cf. eq. (2.10)) as introduced in Section 2.1.3. 2D-to-3D correspondences are obtained by extracting and describing keypoints in the image using a feature detector and descriptor (e.g. **SIFT** [72], **Shi-Tomasi** corners [106], **FAST** [73], **ORB** [74], etc) and then associating them with 3D landmarks and other images using robust feature matching. In contrast, direct methods determine camera pose through image alignment, minimizing the photometric error (2.12). They compare pixel intensities across the image, using an initial depth estimate for alignment, and jointly adjust depth and camera pose. The density of the resulting map reconstruction depends on the area of the image processed: dense (full image), semi-dense (high gradient regions), and sparse (small pixel set). As a result, direct methods are slower than feature-based methods because they process a much larger amount of data. However, they are more resilient to motion blur or textureless regions.

Visual Inertial Odometry

As previously mentioned, monocular cameras suffer from depth ambiguity. Pairing them with an IMU helps resolve the missing scale information and improves the overall state estimation. Moreover, IMU sensors have a much higher output rate (1 kHz and above) compared to RGB cameras (30-60 Hz). Consequently, many VIO methods derive inter-frame relative motion by integrating or pre-integrating inertial data between frames. In practice, initializing the IMU parameters is a critical task whose complexity depends on the quality of the sensor. Thus, accurate synchronization and calibration between the IMU and the camera is essential.

The literature identifies two types of VIO approaches: loosely coupled techniques estimate pose using individual sensor data and then merge them; tightly coupled methods use data from both sensor modalities for combined pose estimation. According to the review by Scaramuzza and Zhang [107], the latter method is more accurate. This review also describes three major paradigms. Filtering techniques sequentially process measurements and maintain only the current state estimate. Inertial data enhances accuracy but also introduces more non-linearities, resulting in more linearization errors permanently injected

into the filter state. Fixed-lag smoothing approach achieves a balance between efficiency and accuracy by optimizing over a sliding window, adjusting recent portions of the trajectory as new data arrive. Thus, prior states in the window can be re-linearized to correct a potential error. In contrast, full smoothing methods consider the entire trajectory and all measurements for estimation, providing the highest accuracy at the expense of greater computational complexity.

Factor graphs are widely used in full smoothing methods. They facilitate optimization tasks by explicitly modeling the structure of the problem, allowing efficient sparse linear solvers to be used in the optimization process. The **GTSAM** library [86] is widely used and combined with the iterative solver **iSAM2** [84]. This kind of solver greatly improves performance by identifying and updating only factors that are affected by new measurements, thus maintaining the sparsity of the factor graph.

2.2.1 Sparse methods

Feature-based methods mostly adopt a keyframe-based approach, storing selective keyframes with associated camera poses and landmarks. They are faster because they do not process the entire image data, but only the most relevant keypoints. While this allows for real-time performance and robust localization, it also results in sparse map reconstruction.

PTAM [92] introduced parallel SLAM, divided into a tracking thread that estimates relative pose at camera frame rate, and a mapping thread that optimizes the global map at keyframe rate.

Building upon this foundation, Mur-Artal presented **ORB-SLAM** [87], an algorithm that runs in real-time using a more robust version of the **ORB** feature descriptor. Loop closures are detected using the **DBoW2** library [108], which relies on vocabulary trees. Similar to a pose graph, the covisibility graph is adopted, connecting keyframes based on spatial proximity. Two nodes connect when they observe a minimum number of shared landmarks, emphasizing spatial relationships over temporal ones. This representation further optimizes data association, loop closure detection, and relocalization by limiting the search window to a relevant portion of the map.

SVO [109] is a semi-direct Visual Odometry (VO) technique commonly used in various SLAM front-ends. Motion estimation begins by minimizing the photometric error of the patches resulting from the projection of 3D landmarks across successive frames. Then, reprojected points are refined by feature-patch alignment from previous keyframes. BA is

then applied to further minimize the reprojection error from the previous step. In terms of mapping, for each identified keypoint, depth filters are initialized with the average depth of the keyframe, with a high uncertainty. As more frames are processed, depth is updated in a Bayesian fashion. When the uncertainty is low enough, the depth is incorporated into the map. This method provides a good balance between direct and feature-based approaches, resulting in a robust and fast solution that was then implemented for SLAM.

Voxel maps were introduced in [41] as an alternative map representation for keyframe-based methods. The study argues that the use of a voxel map, as opposed to a covisibility graph, improves both scalability and geometric reasoning within SLAM systems. This approach explicitly reasons about occlusions and ensures geometric consistency within the camera field of view. Furthermore, it avoids the redundancy of keyframes in identical locations, and the voxel hashing technique maintains a constant query time for a fixed frustum. The methodology has been integrated into the **SVO** framework, with empirical results showing that it achieves an optimal balance between geometric awareness and computational efficiency.

SVO+GTSAM [79] is another SLAM algorithm based on **SVO** [109] front-end, and a full smoothing back-end using factor graph. A structureless approach is introduced replacing landmarks variables in the graph with structureless factors, accelerating computations. The authors have also formally demonstrated the IMU pre-integration theory which is widely used in VIO. It constrains a single relative motion from high-frequency inertial measurements between 2 frames addressing the manifold structure of the rotation group $SO(3)$.

Basalt [2] is a state-of-the-art stereo-inertial method. It uses the double sphere camera model [71], which offers improved accuracy for cameras with a wide field of view. The system conducts VIO estimation on a dedicated thread using a fixed-lag smoothing strategy. **FAST** keypoints are detected and tracked for each frame using the **Kanade-Lucas-Tomasi (KLT)** method. Between frames, IMU data is pre-integrated. Next, a local BA is applied minimizing both the reprojection error and an IMU propagation term from [79]. In the mapping thread, each new keyframe triggers the detection of new ORB features, which are matched with other keyframes. Then, non-linear factor recovery proceeds by recovering roll-pitch, yaw, absolute position, and relative poses from the linearization resulting from front-end marginalization. Finally, Global BA optimizes reprojection errors from ORB features and an error term computed from the non-linear factors.

ORB-SLAM was followed by a visual-inertial version [96] and **ORB-SLAM 2** [99],

an RGB-D and stereo extension. Combining and improving these works led to **ORB-SLAM 3** [1]. Its structure is illustrated in Figure 2.5. It is compatible with monocular, stereo, RGB-D, and visual-inertial configurations, without restriction on the camera model. It enhances place recognition through verification of geometric consistency and implements the **Atlas** multi-map system [110]. The benchmarking results of the authors show that **ORB-SLAM 3** outperforms all the leading methods in all sensor configurations, making it the state of the art. However, even though the IMU initialization process has been accelerated, it is still suboptimal in practice. In slow motion, the initialization tends to fail, especially without pitch and roll rotations.

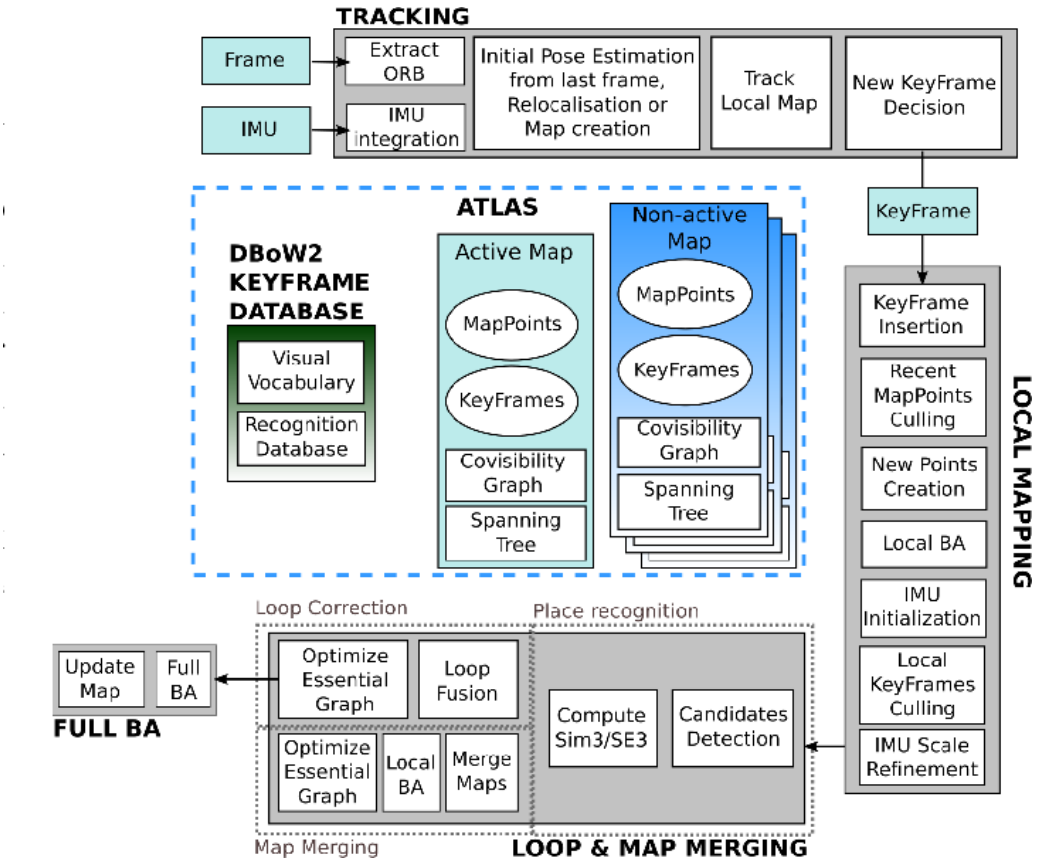


Figure 2.5 – Architecture of **ORB-SLAM 3** as illustrated in [1] © 2021 IEEE, featuring a multi-threaded design. The Tracking thread processes input data to estimate the camera pose at the camera frame rate. The Local Mapping thread refines camera poses and updates the local map at the keyframe rate. The Loop & Map Merging thread is responsible for place recognition and loop closure. The Atlas component is a multi-map system that stores maps when the algorithm loses track and merges the active map with stored ones when a previously mapped location is recognized.

Summary Conventional sparse SLAM and VIO methods have made significant progress in addressing many of the primary challenges associated with localization. Under classical conditions, characterized by static scenes without dynamic objects, constant lighting conditions, and the absence of aggressive motion, state-of-the-art approaches show remarkable accuracy and robustness in performing metric localization. The adoption of a sparse map representation allows for the application of more advanced techniques in a real-time operational context. Additionally, voxel mapping was shown to be particularly relevant for navigation tasks, providing better scalability and geometric consistency. However, these systems have mainly been evaluated using standard datasets like **KITTI** [5] for automobiles, **TUM-VI** [4] for handheld payload, and **EuRoC** [3] for drones, as described in Section 2.6. These benchmarks do not explicitly evaluate the performance of SLAM systems under more challenging conditions.

2.2.2 Dense and semi-dense methods

Dense SLAM is commonly implemented using direct methods, which process much more data as they estimate depth for each pixel in an image. This approach typically provides a 3D reconstruction as a dense point cloud or a 3D mesh. This comes at the cost of increased computational complexity, making it more challenging for real-time applications than its sparse counterparts.

DTAM [93] is a pioneer in direct methods exploiting GPU-accelerated dense computations over entire images. Textured depth maps are generated through the optimization of a projective photometric cost volume, considering factors such as photometric error, inverse depth, and a robust spatial regularization term. Camera pose is inferred by aligning the actual and a virtually reconstructed frame from the dense map. This technique not only yields a dense 3D reconstruction but also enhances robustness during fast motion. Nonetheless, the initial bootstrapping of the system and the precision of localization remain suboptimal.

LSD-SLAM [94] is another renowned direct method. It considers high-gradient regions instead of all pixels in the image to create a semi-dense map. When a new keyframe is created, the inverse depth is estimated from the previous keyframe and refined by adaptive-baseline stereo matching from subsequent frames. Motion is estimated by minimizing the variance-normalized photometric error. Keyframes and corresponding poses are inserted into a pose graph and maintained in the background by PGO. This method runs in real-time on a CPU and achieves good accuracy.

VITAMIN-E [95] builds on a feature-based approach and achieves dense reconstruction by handling a very large number of keypoints. It introduces a new form of dense feature tracking that detects local curvature extrema and uses dominant flow estimation to predict their new positions. It adopts a subspace Gauss-Newton method for BA optimization, which substantially increases computational efficiency by partially updating variables, making real-time operation feasible. The system generates a mesh by projecting the resulting dense point cloud and applying 2D Delaunay triangulation, which is then denoised and integrated within a Truncated Signed Distance Function (TSDF). The processing speed of **VITAMIN-E** on a CPU is such that it can handle every single frame, not just keyframes. While it surpasses the localization accuracy of methods like **ORB-SLAM** and **LSD-SLAM**, its measurements are not metric, and its overall performance is below that of state-of-the-art methods such as **ORB-SLAM 3**.

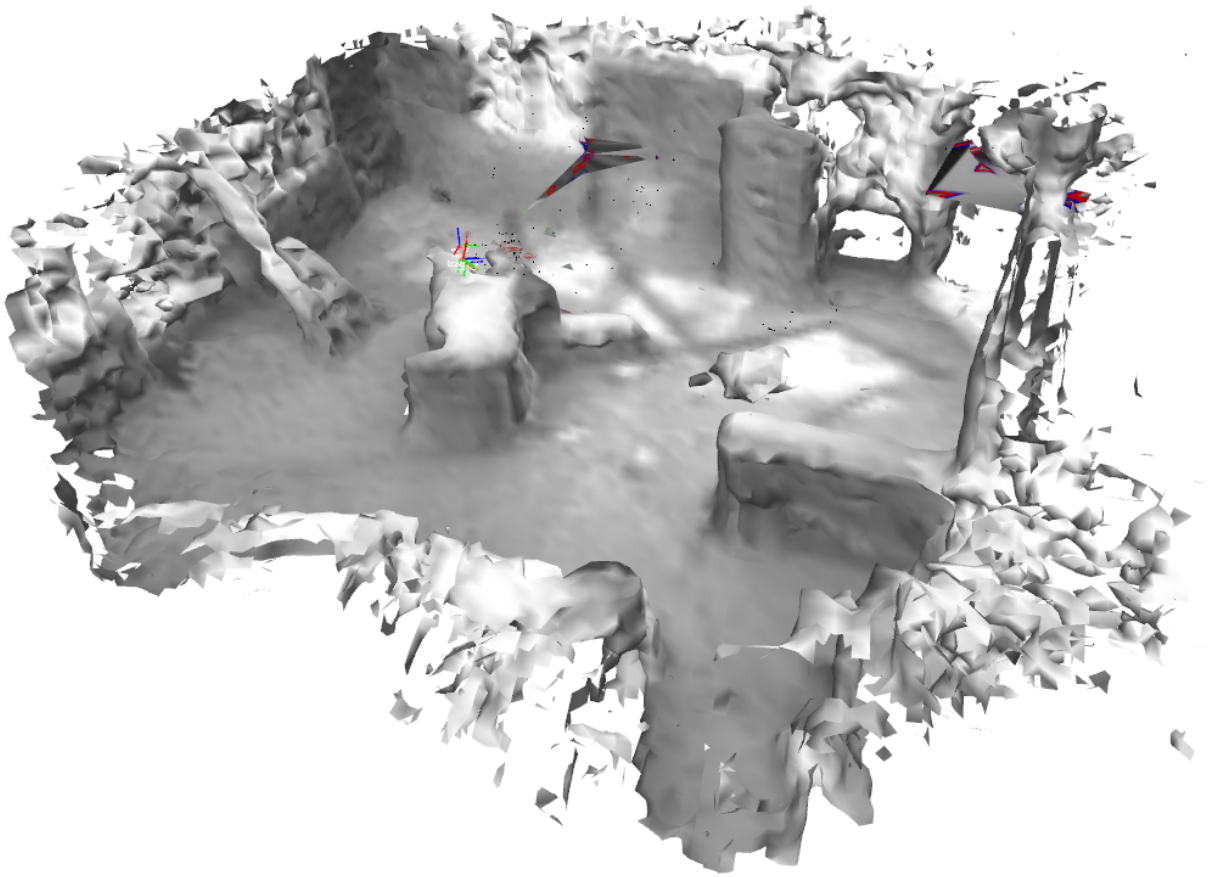


Figure 2.6 – Example of 3D reconstruction generated by Kimera [12] on the EuRoC [3] dataset.

Kimera [12] presents a dense, metric, semantic, and real-time SLAM system using a stereo-inertial configuration segmented into four primary modules: VIO, robust PGO (global sparse mapping and loop closure), fast dense 3D mapping, and global dense semantic 3D mapping. The VIO module is based on **SVO+GTSAM** [79] adapted for stereo. The robust PGO adopts the DBoW2 [108] library for loop closure detection and improves reliability through outlier rejection via Pairwise Consistency Maximization (PCM). Similar to **VITAMIN-E**, the system achieves fast dense 3D mapping by performing a 2D Delaunay triangulation of tracked features and then back-projecting this triangulation onto corresponding 3D landmarks to generate a fast per-frame and multi-frame 3D mesh. Global mapping leverages bundled raycasting, as demonstrated in **Voxblox** [39], using a 3D point cloud derived from dense stereo to create and update a TSDF from which an accurate global 3D mesh is extracted using marching cubes [111]. It also incorporates semantics by raycasting 2D segmentation.

Summary Dense SLAM, especially in its monocular form, has been a less explored area until recently because the primary focus has been on ensuring robust and accurate localization over dense mapping. Existing methods often required significant computational resources or were less efficient than sparse SLAM, which handles fewer parameters during robust PGO. The inherent depth ambiguity poses a critical limitation to real-time dense 3D map reconstruction from a monocular camera. Nevertheless, this challenge has been addressed in recent SLAM developments through the integration of advanced deep learning techniques that are presented in the following section.

2.3 Densifying Monocular SLAM

Recent advancements have integrated DNNs into monocular SLAM for end-to-end learning or specific tasks, thereby improving scene understanding and robustness [38], [112]. In particular, these methods have significantly advanced dense mapping capabilities [113]. A common approach to densify sparse or semi-dense SLAM maps is to leverage Monocular Depth Estimation (MDE), which has been shown to significantly improve 3D reconstruction quality. Yet, many of these methods predict relative depth without an absolute scale. A concise synthesis of related works is presented in Table 2.2 whose operation is briefly described in the sequel.

Method	Year	Metric	Sensors	Localization evaluation	Mapping evaluation	Computing resources	Code available
NERF-SLAM [114]	2023	No	Mono	No	Depth map	RTX 2080 Ti	Yes
Rosinol et al. [115]	2023	No	Mono	No	Point cloud	RTX 2080 Ti	No
CodeMapping [6]	2021	Yes	Mono-IMU	No	Depth map	RTX 3080	No
DROID-SLAM [116]	2021	No	Mono	Yes	N/A	2x RTX 3090	Yes
TANDEM [117]	2021	No	Mono	Yes	Depth map	RTX 2080	Yes
CodeVIO [7]	2021	Yes	Mono-IMU	Yes	Depth map	GTX 1080 Ti	No
DeepRelativeFusion [66]	2021	No	Mono	Yes	Depth map	GTX 1070	No
DeepFactors [65]	2020	No	Mono	Yes	Depth map	GTX 1080	Yes
MapSlammer [118]	2019	No	Mono	Yes	Point cloud	N/A	Yes
CodeSLAM [119]	2018	No	Mono	No	N/A	N/A	Yes
CNN-SLAM [64]	2017	No	Mono	Yes	Depth map	Quadro K5200	Yes

Table 2.2 – Comparative overview of the main monocular SLAM densification methods.

CNN-SLAM [64] was a pioneer in integrating DNNs into SLAM, providing a solution for predicting dense depth maps using Convolutional Neural Networks (CNNs) from single keyframes and subsequently merging them with depth estimates derived from **LSD-SLAM**. This technique addresses specific challenges such as absolute scale estimation, robustness during pure rotational motion, and extraction of dense depth in textureless regions.

Map Slammer [118] uses the **DeMoN** method [120] to predict depth maps from a pair of successive monocular images. A 3D point cloud is extracted from these depth maps. It is then registered and fused with the sparse points from **ORB-SLAM 2**, resulting in a dense 3D map.

DeepRelativeFusion [66] extends the capabilities of **LSD-SLAM** by incorporating **MiDaS** [121] to predict relative depth from a CNN, and then densify the semi-dense

map estimated by SLAM. The resulting dense map is then used to fine-tune keyframe poses, and the global structure is further enhanced by a two-view consistency check. This approach demonstrates superior results compared to one of the reference methods DeepFactors presented afterwards.

CodeSLAM [119] introduced Conditional Variational Auto-Encoders (CVAE) to infer depth in monocular SLAM by learning a compact depth representation. The auto-encoder[1] implicitly finds a compact representation referred to as a "code" for the depth map. The authors argue that when given the corresponding intensity image, only the depth information that cannot be recovered from the image intensities needs to be encoded. It translates into the architecture shown in Figure 2.7. The lower network represents a Variational Auto-Encoder (VAE), whose central variational component samples the code from the Gaussian distribution it has learned. The connection between the code and the reconstructed depth is enforced by avoiding non-linear activations in the decoder. The upper network extracts features from the intensity image and predicts the depth uncertainty. The encoded features are concatenated at multiple scales in the lower network to condition the VAE. The entire model is trained from RGB-D data. The codes and camera poses constitute the state variables of the system from which the Jacobian is derived. Consequently, the use of codes provides a compact representation of the 3D structure of a keyframe, thereby reducing the complexity of the optimization.

Continuing this work, **DeepFactors** [65] implements the same concept into a real-time full SLAM system relying on GPU acceleration. In particular, this approach proposes a probabilistic formulation that integrates the use of codes in a dense BA framework with a factor graph. It also introduces a third network that predicts an initial code from an image for each new keyframe, improving the efficiency of the process. **DeepFactors** represents a significant improvement in dense monocular SLAM, with promising reconstruction accuracy compared to previous methods. Nevertheless, Jacobian computation still severely penalizes its efficiency.

Addressing this issue, **CodeVIO** [7] intelligently integrates a CVAE with an IMU, presenting a real-time SLAM framework capable of dense and metric 3D mapping. The VIO module is based on **OpenVINS** [122], where the depth code is integrated into the state vector. Additionally, the sparse depth estimated by VIO is concatenated with the input intensity image of the conditioner network. On the other hand, since only the Jaco-

1. An auto-encoder is a type of neural network used to learn efficient codings of unlabeled data. The network typically consists of two parts: an encoder that maps the input to a code, and a decoder that maps the code to a reconstruction of the original input.

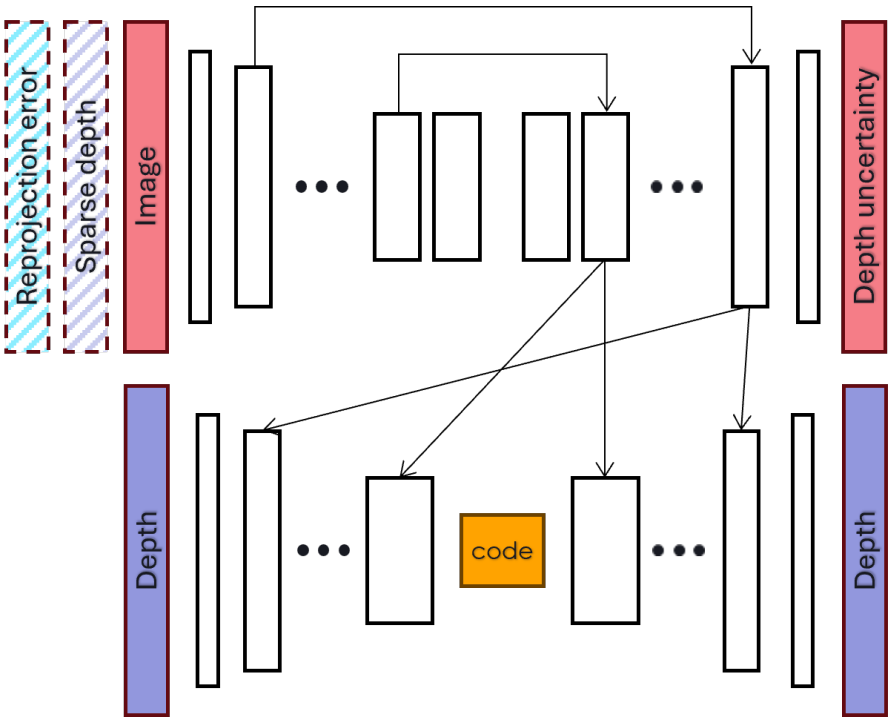


Figure 2.7 – Architecture of a CVAE as presented in **CodeSLAM**. Certain methods additionally integrate the sparse depth from SLAM as input [6], [7] and occasionally include the associated reprojection error [6].

bian of the VAE decoder is needed, they approximate it with a finite difference equation and avoid its recurrent computation using the First-Estimate Jacobians technique. This approach significantly speeds up the process and allows **CodeVIO** to achieve state-of-the-art accuracy in monocular SLAM densification.

TANDEM [117] is a dense monocular SLAM system based on photometric BA optimized over a sliding window. The method employs a multi-view stereo network, which efficiently exploits the entire active keyframe window, creating 3D cost volumes hierarchically and utilizing adaptive view aggregation. This process balances the different stereo baselines between keyframes, improving depth map prediction. The system produces a consistent global map through the fusion of depth map predictions into a TSDF voxel grid. TANDEM is not metric, but for reference, it surpasses **CodeVIO** in reconstruction accuracy after scale alignment.

CodeMapping [6] builds on the monocular inertial configuration of **ORB-SLAM 3** and uses CVAE to densify the mapping process. Building on the CVAE introduced in **CodeSLAM**, it processes a keyframe consisting of an intensity image concatenated

with the estimated sparse depth and the associated reprojection error. The authors argue that the reprojection error provides a confidence measure for the sparse depth. Similar to **DeepFactors**, the dense mapping thread processes windows of four keyframes from which depth codes are extracted and optimized in a dedicated factor graph. Only the structure is optimized since **ORB-SLAM 3** already provides accurate pose estimation. Therefore, this approach strongly relies on the robustness of the underlying SLAM method. Also, training in a supervised fashion limits the generalization capacity. However, it is currently one of the state-of-the-art methods for dense and metric monocular SLAM.

DROID-SLAM [116] marks a significant advancement in SLAM research, introducing an end-to-end differentiable architecture that integrates deep learning with traditional geometric methods. The problem is structured as a covisibility graph and adapts techniques introduced in the **RAFT** [123] optical flow estimator. Each edge (image pair) is processed through a feature network to construct correlation volumes, complemented by another network that gathers contextual features. A learned recurrent update operator is the core of this method. It iteratively updates the hidden state, pose, and depth estimates. The operator incorporates a convolutional Gated Recurrent Unit (GRU) that interprets correlation, flow, and context features to predict dense optical flow adjustments. These predictions are integrated into a dense BA layer, which refines the pose and depth within the graph. Although **DROID-SLAM** is computationally intensive and follows a supervised learning paradigm, it demonstrates impressive localization performance and robustness after scale adjustment.

Following this work, a few approaches have been proposed to specifically improve 3D reconstruction by enhancing depth map fusion. The paper [115] introduced a probabilistic volumetric fusion technique based on **DROID-SLAM** aiming to improve the quality of its noisy reconstructed maps. Depth uncertainty is derived from the dense BA optimization procedure, specifically from the marginal covariance of the depth maps. The volumetric fusion is applied to the depth maps weighted by their computed uncertainty to build a TSDF. A mesh is then generated using marching cubes, taking into account the uncertainty bounds.

Continuing this approach, **NeRF-SLAM** [114] incorporates Neural Radiance Fields (NeRF) into SLAM to replace the previous uncertainty-based volumetric construction. The original NeRF model [124] has demonstrated remarkable capabilities in rendering 3D scenes from different viewpoints, although it requires extensive training on multiple posed images. However, following subsequent research on instant [125] and pose-free [126],

[127] techniques, **NeRF-SLAM** proposes to learn and apply a model from dense SLAM outputs and computed uncertainty in real-time. This novel method significantly enhances performance compared to most prior techniques, although the authors acknowledge that the computational demands in real-time operations are still too high for drone applications.

Summary Thus, recent studies aim to enhance monocular SLAM by densifying sparse and semi-dense maps to increase localization robustness or to improve 3D reconstruction. The majority of these approaches employ deep learning techniques, which have resulted in remarkable advances in depth estimation accuracy. However, as Table 2.2 illustrates, several of these methods fail to achieve metric precision due to their reliance on strictly monocular configurations. Furthermore, these methods have primarily been tested on high-powered desktop GPUs, raising concerns about their suitability for low-power embedded computing environments. Among these, **CodeVIO** and **CodeMapping** stand out as leading methods for dense and metric monocular SLAM. Unfortunately, they do not make their source codes publicly accessible, restricting broader experimentation and adoption.

2.4 Monocular Depth Estimation

In the previous section, we illustrated the benefit of predicting dense depth maps from single images for dense SLAM. In this section, we will provide a concise overview of the various deep learning methods and approaches for Monocular Depth Estimation (MDE).

The review by Ming et al. [128] presents a thorough review of research on MDE. Initial methods relied primarily on visual depth cues such as vanishing points, shadows, and focus variations. These were not very effective and were limited to scenes with shallow depth fields. Subsequently, with the rise of machine learning paradigms, techniques based on feature detectors and probabilistic graph models were introduced. Although they attained better performance, they were still sparse, complex, slow, and inefficient in textureless regions. Finally, the emergence of Deep Learning has led to considerable progress in the field. Typically, encoder-decoder architectures are used to first extract deep features from a single image and then use deconvolution layers to regress the corresponding depth map. Methods can be divided into three categories depending on the employed learning method: **supervised**, **semi-supervised**, and **self-supervised** (or unsupervised).

Supervised methods, presented in subsection 2.4.1, rely on extensively labeled datasets where ground truth depth information is used to train models to understand and predict depth from single images. These methods achieve high precision but require substantial data collection with accurate labeling, which is a tedious task.

Semi-supervised methods mitigate this by using a combination of labeled and unlabeled data, improving the model's ability to generalize from limited ground truth information while still leveraging the large amount of unlabeled data. This approach balances the robustness of supervised methods with the scalability of unsupervised ones.

On the other hand, self-supervised methods, reviewed in subsection 2.4.2, do not require labeled data and instead rely on innovative training schemes derived from multi-view geometry, often using stereo pairs or monocular sequences. While this approach provides greater scalability, it is typically less accurate than its supervised counterparts. It is also more prone to scale ambiguity and affected by occlusions.

The influence of absolute scale estimation on the depth estimation accuracy was established in [129]. Supervised methods tend to achieve higher accuracy when the predicted scale is close to that of the training ground truth [128]. This approach leads to limited generalization and increased uncertainty [129]. The alternative approach is to learn relative depth by predicting depth maps up to an unknown scale factor, typically implementing

scale-invariant loss [129], which has also been shown to improve generalization [121]. As a result, most of the algorithms are evaluated either using scale-invariant metrics [129] or after performing a scale alignment using ground truth data. Certain self-supervised methods [8], [130] also address this issue by adopting a scale-aware[1] loss function, allowing the prediction of depth maps that are both metric and scale-consistent.

Ming et al.'s review provides a comprehensive comparison of these methods, highlighting quantitative results measured on the **KITTI** benchmark [5]. The results, particularly in terms of the squared relative error metric, indicate that supervised methods outperform others, with self-supervised techniques next in line. Based on these findings, this section will briefly cover a few renowned techniques from both categories in chronological order.

2.4.1 Supervised methods

Eigen et al. [129] pioneered the use of DNNs in depth estimation, adopting a multi-scale strategy with CNNs. Their method initially generates a coarse depth map by interpreting global structure information, which is subsequently refined for specific local regions by a fine-scale network. Nonetheless, this approach has some limitations, such as the use of successive spatial pooling operations which degrades the quality of the deep feature extraction.

Fu et al. proposed the **DORN** method [131], which redefines depth estimation as an ordinal regression task instead of traditional direct regression. By discretizing continuous depth values into distinct ordinal bins, it overcomes some of the inherent challenges of direct regression, thereby achieving increased stability and accuracy in its predictions. Additionally, training the model to predict relative scale enhances its ability to stay robust despite scale changes since the relative depth remains consistent, even if the absolute scale varies.

BTS [132] proposes to replace traditional skip connections with local planar guidance, establishing direct and explicit links between encoder and decoder layers at different resolutions. This approach relies on the assumption that small image regions can often be approximated as planar surfaces, providing a geometric context that helps maintain consistency across scales and resolutions.

MiDaS [121] presents a novel training strategy designed for robust depth estimation.

1. In Monocular Depth Estimation, "scale-aware" refers to the ability of a model to predict depth maps where the absolute scale of the depth values corresponds to the real-world measurements, as opposed to predicting relative or arbitrary scales.

The approach relies on mixing multiple datasets, as the combination is likely to contain different depth ranges, varying depth representations (direct or inverse depth), and potential disparity and scale ambiguity (relative or absolute). To align these diverse data sources, **MiDaS** introduces a novel loss function that is both scale and shift invariant to account for the disparate nature of the integrated datasets. This strategic fusion enables the model to learn depth estimation with enhanced generalizability across different scenarios and conditions.

The authors of **MiDaS** then introduced the Dense Prediction Transformer (**DPT**) [133], which leverages Vision Transformers (ViTs) to enhance the encoder's capabilities. Notably, the ViT backbone in DPT maintains a constant and relatively high resolution throughout its depth. This consistency allows the model to make more detailed predictions compared to Fully Convolutional Neural Networks (FCNNs), which lose resolution as they go deeper when using pooling operations. The depth map prediction is generated by a convolutional decoder, which reassembles the set of tokens processed by the encoder. Since ViTs are not the primary focus of our work, we refer readers to the comprehensive survey [134] for a detailed introduction to the topic.

AdaBins [135] builds on a similar approach to **DORN**, but introduces a novel approach to adaptively divide the depth range into bins. The method uses an encoder-decoder architecture, followed by a unique module that uses a simplified version of ViT to adaptively determine the bin boundaries for each image. The ultimate depth map is computed as a linear combination of these adaptive bin centers.

Recently introduced, **ZoeDepth** [136] addresses the challenges of generalization and absolute scale in depth estimation. Building on the **MiDaS** strategy, it adopts an encoder-decoder architecture derived from **DPT** and incorporates a modern ViT backbone for the encoder. While the **MiDaS** model predicts relative depth, **ZoeDepth** attaches a metric bins module to the decoder. Inspired by the adaptive binning technique introduced in **AdaBins**, **ZoeDepth** effectively infers a metric depth map.

ZeroDepth [9] introduces a novel approach to predict metric scale depth in a zero-shot cross-domain setting. The Transformer-based model is illustrated in Figure 2.8. By integrating camera intrinsic parameters, the model gains insight into object size and shape, which aids in scale estimation. Using variational inference, it captures uncertainty in depth predictions, producing a distribution over potential depth maps rather than a singular estimate. The encoder samples a latent depth representation from image and camera embeddings, which is then decoded multiple times into a depth map, with the final depth

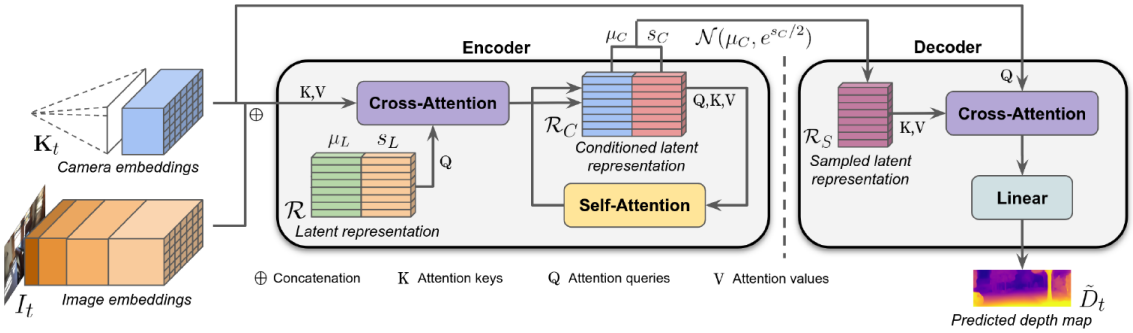


Figure 2.8 – Architecture of **ZeroDepth** as published in [9] © 2023 IEEE.

being the mean of these samples and the standard deviation being the uncertainty. Currently, **ZeroDepth** is among the leading methods for monocular depth estimation.

2.4.2 Self-supervised methods

Self-supervised methods, such as the one introduced by Garg et al. [137], eliminate the need for ground truth depth data by leveraging epipolar geometry constraints. They reformulate monocular depth estimation as an image reconstruction problem. Their pioneering work introduces a form of unsupervised learning that relies on multi-view geometry, specifically applied within an auto-encoder framework. The training process uses stereo images, with the encoder first predicting the inverse depth from the left image. This predicted depth, which is inherently related to disparity, is then integrated with the right image to create an input for the decoder. The role of the decoder is to warp the right image to reconstruct the original left view. The network then computes a reconstruction error between the warped image and the original, and uses this as feedback for learning. During inference, only the encoder is operational, predicting depth maps from single images. This innovative approach bypasses the need for actual depth data, relying solely on stereo images and known camera calibration parameters.

Monodepth [138] enhances this approach by implementing a FCNN and a novel loss function that enforces consistency between the depth maps inferred from the left and right images. Rather than reconstructing one image from another, **Monodepth** ensures that the inferred depth maps maintain coherence as viewpoints are interchanged, facilitating the management of occlusions and textureless regions. Unlike Garg et al., who linearize their loss through a Taylor expansion, **Monodepth** employs a bilinear sampler for image synthesis, resulting in a training loss that is fully differentiable and thereby more straight-

forward to optimize. Moreover, the depth smoothness is enforced by a regularization term.

SfMLearner [139] extends the self-supervised learning paradigm by simultaneously learning depth and camera pose from monocular video sequences. It uses a pose estimation network that takes a target view and its adjacent source views as input and predicts the relative poses between the target and each source view. It results in a set of multiple views of the same scene with estimated relative poses, allowing the previous methodology to be applied. The training dataset comprises sequences of successive frames, each with an average optical flow magnitude of at least one pixel to ensure sufficient scene dynamics. These sequences are fixed to 3 frames, with the target view positioned as the central frame.

Monodepth2 [140] addresses the inherent occlusion challenge of training on monocular sequences. For this purpose, it introduces an appearance matching loss that exploits the per-pixel minimum reprojection error. An automasking procedure is also used to exclude pixels without relative motions, as these pixels do not provide sufficient information for depth computation. Additionally, the authors implement multi-scale supervision by calculating the photometric reconstruction error over different scales using intermediate depth maps upsampled to full resolution.

PackNet-SfM [8] presents a novel neural network structure designed for scale-aware depth estimation from monocular video sequences. The architecture of the self-supervised model for depth and camera pose prediction is depicted in Figure 2.9. This method introduces symmetrical packing and unpacking blocks with 3D convolutions, allowing detailed spatial information to be preserved for precise high-resolution depth predictions. **PackNet-SfM** also aims to infer depth with absolute scale by integrating a velocity supervision loss when ground truth velocity data is available. This loss function constrains the estimated translation and guides the model in scale prediction. **PackNet-Sfm** outperforms many existing methods without scale adjustment, including supervised learning approaches. While reported experiments demonstrate promising generalization capabilities, the primary evaluation context remains outdoor car navigation scenarios.

Recently, several advanced methods have been proposed, achieving remarkable performance improvements. **DIFFNet** [141] employs a semantic segmentation network and incorporates multi-scale feature fusion and spatial attention modules into its decoder to improve skip connections processing. **MonoFormer** [142] evaluates the generalization capabilities of state-of-the-art methods. Based on the observation that CNNs are texture-biased and Transformers are shape-biased, they exploit the strengths of both and intro-

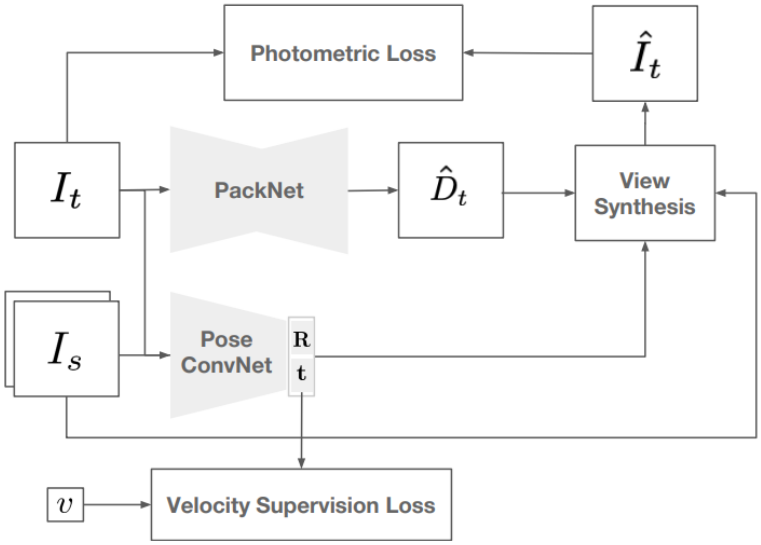


Figure 2.9 – Architecture of **PackNet-Sfm** as published in [8] © 2020 IEEE. The model is designed for joint depth and camera ego-motion estimation.

duce **Monoformer**, a hybrid architecture that offers improved performance in various scenarios. In comparison, **MonoViT** [143] also combines CNNs and ViT, but stands out for its higher accuracy and significantly reduced network complexity, with 2 to 16 times fewer parameters than **MonoFormer**.

Summary This section has provided a selective review of monocular depth estimation techniques. Overall, supervised methods are more precise, although they have been severely hampered by the scarcity of large labeled datasets. Their generalization capabilities remained limited until the recent emergence of Transformers, with the example of **ZeroDepth**, which has an extensive learning capacity but requires massive amounts of training data. During our study, self-supervised strategies initially emerged as a more viable option, providing decent generalization without reliance on labeled data. Notably, **PackNet-Sfm** stood out by approaching the accuracy of its supervised counterparts. However, a common limitation of these diverse approaches is their predominant training and evaluation in the context of autonomous driving, with insufficient exploration and adaptation to indoor environments. In the context of this work, we intend to leverage the generalization capabilities of these methods in indoor scenarios specific to drone navigation. To achieve this, in the following sections, we will review the evaluation metrics and datasets that are most relevant to our study.

2.5 Evaluation metrics

This section presents the classic metrics used to evaluate SLAM systems [144] and depth estimation [63] algorithms. These metrics will be used in our experiments for the measurement of the accuracy of our work and the comparison with related works.

2.5.1 Localization evaluation

SLAM and VIO systems need relevant metrics to benchmark their accuracy. The most used metrics are the Absolute Trajectory Error (ATE) and the Relative Pose Error (RPE) introduced in [145]. The trajectory of a drone can be represented as a sequence of poses or as a set of rigid body transformations. In the following, the estimated trajectory is defined as $P = \{\mathbf{P}_i \in SE(3)\}_{0<i\leq N}$ and the corresponding ground truth trajectory as $Q = \{\mathbf{Q}_i \in SE(3)\}_{0<i\leq N}$, where N denotes the number of poses.

Absolute Trajectory Error

The most appropriate metric for evaluating SLAM localization is the ATE, as it assesses global trajectory consistency. Before comparison, the estimated and ground truth trajectories need to be aligned since they might be expressed in different coordinate frames. Horn's alignment method [146] is commonly used. It solves an optimization problem to determine the rigid body transformation $\mathbf{S} \in SE(3)$ that aligns the estimated poses $\mathbf{P}_i$ with the ground truth poses $\mathbf{Q}_i$. Other methods have been proposed, such as the one discussed by Salas et al. [147], which presents a least-squares optimization approach on the manifold. Consequently, the error at time step i can be expressed as:

$$\mathbf{E}_i = \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{P}_i \tag{2.17}$$

This approach is relevant when the evaluated algorithm can determine the absolute scale. However, in settings such as pure monocular SLAM, where this is not possible, scale adjustment becomes necessary. Consequently, the transformation $\mathbf{S}$, initially in $SE(3)$, transitions to a similarity transformation in the $Sim(3)$ group. In this case, the Umeyama method [148] is commonly used to align trajectories, since it additionally corrects the scale. Similarity transformations can be defined by the following equation where $s \in \mathbb{R}$ is a scale factor:

$$\forall \mathbf{T} \in Sim(3), \mathbf{T} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.18}$$

The ATE assumes that rotational errors that accumulate will eventually manifest as global translation errors at the end of the trajectory. Therefore, the ATE is defined as the Root Mean Square Error (RMSE) of the translational error. Using the RMSE provides a significant statistical measure to quantify the average magnitude of the error. Therefore, the formula for the error is as follows::

$$ATE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \left\|\text{trans}(\mathbf{E}_i)\right\|^2} \tag{2.19}$$

with trans($\mathbf{T}$) being the translation part of $\mathbf{T} \in SE(3)$.

Relative Pose Error

The RPE is another metric particularly suited for VIO, as it evaluates the trajectory over segments of length Δ, providing a more granular evaluation of camera pose tracking accuracy. The error is typically divided into translation and rotation components. The relative pose error at time i is expressed as follows:

$$\mathbf{F}_i = (\mathbf{Q}_i^{-1}\mathbf{Q}_{i+\Delta})^{-1}(\mathbf{P}_i^{-1}\mathbf{P}_{i+\Delta}) \tag{2.20}$$

The translation error RPE_{trans} is calculated following the same principle as the ATE, but over the segments, where $m = N - \Delta$ is the number of error matrices considered:

$$RPE_{trans} = \sqrt{\frac{1}{m}\sum_{i=1}^{m} \left\|\text{trans}(\mathbf{F}_i)\right\|^2} \tag{2.21}$$

On the other hand, the rotational error RPE_{rot} is defined as follows:

$$RPE_{rot} = \sqrt{\frac{1}{m}\sum_{i=1}^{m} \left\|\text{angle}(\text{rot}(\mathbf{F}_i))\right\|^2} \tag{2.22}$$

with the operator rot($\mathbf{T}$) extracting the rotation matrix from a transformation $\mathbf{T} \in SE(3)$, and angle($\mathbf{R}$) = $\arccos\left(\frac{\text{tr}(\mathbf{R})-1}{2}\right)$ computing the rotation angle of a rotation $\mathbf{R} \in SO(3)$ around a rotation axis [149].

2.5.2 Depth estimation evaluation

Monocular depth estimation models are evaluated using various metrics introduced in [129] to quantitatively analyze the accuracy and precision of the predicted depth maps. Some of these metrics were then also applied to benchmark the dense reconstruction of SLAM. Although other methods have been proposed to evaluate point clouds, to our knowledge there are no standardized metrics used in dense SLAM to evaluate voxel maps.

The key metrics used to evaluate monocular depth estimation and dense SLAM are outlined below. We have excluded scale-invariant metrics as we aim to evaluate metric reconstruction and therefore penalize scale errors. In the sequel, Ω is the set of pixels p considered, N is the number of 3D points in Ω, $\hat{D}_p$ is the predicted depth, and Z_p the ground truth depth.

Depth map evaluation

— Absolute Relative Difference

This metric calculates the mean absolute relative error, emphasizing the absolute difference between the predicted and ground truth values relative to the true depth. By normalizing errors against the ground truth, it compensates for the different impacts between farther and closer points. Basically, an identical error magnitude has a greater impact on closer points than on farther points.

$$abs\_rel = \frac{1}{N} \sum_{p \in \Omega} \frac{|\hat{D}_p - Z_p|}{Z_p} \tag{2.23}$$

— Squared Relative Difference

Compared to the previous metric, the squared relative difference amplifies larger errors by squaring the discrepancies. This procedure amplifies the magnitude of larger errors, making it a sensitive metric that further penalizes significant deviations between predicted and actual values.

$$sq\_rel = \frac{1}{N} \sum_{p \in \Omega} \frac{\left\| \hat{D}_p - Z_p \right\|^2}{Z_p} \tag{2.24}$$

— Root Mean Squared Error

The RMSE calculates the standard deviation of the depth errors and emphasizes larger differences due to its quadratic nature. Nonetheless, it does not differentiate

between errors from near points and those from far points, treating all errors with the same weight regardless of depth.

$$rmse = \sqrt{\frac{1}{N}\sum_{p\in\Omega}\left\|\hat{D}_p - Z_p\right\|^2} \tag{2.25}$$

— RMSE in log scale
This variant of the RMSE operates in the logarithmic domain, reducing the influence of extreme outliers and focusing on relative errors.

$$rmse\_log = \sqrt{\frac{1}{N}\sum_{p\in\Omega}\left\|\log\hat{D}_p - \log Z_p\right\|^2} \tag{2.26}$$

— Accuracy rate
The accuracy rate (or threshold accuracy) evaluates the proportion of predicted values that fall within a certain factor threshold of the actual depth, providing a gradual evaluation of prediction accuracy. It is expressed by the following formulation, where t is typically in $\{1, 2, 3\}$ and $\mathbb{I}(\cdot)$ represents the indicator function that returns 1 if the specified condition is met and 0 otherwise:

$$\delta_t = \frac{1}{N}\sum_{p\in\Omega}\mathbb{I}\left(\max\left(\frac{\hat{D}_p}{Z_p}, \frac{Z_p}{\hat{D}_p}\right) < 1.25^t\right) \tag{2.27}$$

Polygon mesh and point cloud evaluation

In [12], [115], [150], a distinctive approach is proposed for the evaluation of 3D meshes and point clouds. This method defines both accuracy and completeness scores. For 3D meshes, point clouds are generated by sampling uniformly over the mesh surface. Subsequently, the estimated and ground truth point clouds are registered using the Iterative Closest Point (ICP) algorithm using the CloudCompare library [151]. The accuracy of the estimation is then evaluated by computing the average distance from each point in the ground truth point cloud to its nearest neighbor in the estimated point cloud. Similarly, completeness is determined by calculating the distance between each estimated point and the closest point in the ground truth cloud. This approach provides a new and interesting way to compare reconstructed maps, taking into account both the accuracy of the points and the completeness of the reconstruction.

2.6 Datasets

Name	Year	Sensors	IMU	Pose GT	Map GT	Environment	Task	Platform
NYU Depth v2 [152]	2012	Mono			DDM	Indoor	MDE	Car navigation
KITTI [5]	2012	Stereo	10 Hz	✓	S-DDM	Outdoor	SLAM, MDE	Car navigation
TUM-RGB-D [145]	2012	Mono	500 Hz	✓	DDM	Indoor	SLAM, MDE	Handheld
Cityscapes [153]	2016	Stereo		✓		Outdoor	SLAM, MDE	Car navigation
EuRoC [3]	2016	Stereo	200 Hz	✓	3D scan	Indoor	SLAM, MDE	Drone indoor flights
MVSEC [154]	2018	Stereo	200 Hz	✓	S-DDM	Indoor, Outdoor	SLAM, MDE	Multi vehicle
ETH3D [155]	2018	Stereo	460 Hz	✓	S-DDM	Indoor	SLAM, MDE	Handheld
TUM-VI [4]	2018	Stereo	200 Hz	partial		Indoor	SLAM	Handheld
Blackbird [156]	2018	Stereo	100 Hz	✓		Indoor	SLAM	FPV drone simulation
UZH-FPV Drone Racing [10]	2019	Stereo	1 kHz	✓		Indoor, Outdoor	SLAM	FPV drone flight
nuScenes [157]	2019	Stereo	1 kHz	✓		Outdoor	SLAM, MDE	Car navigation
Waymo [158]	2019	Stereo			S-DDM	Outdoor	MDE	Car navigation
KITTI-360 [159]	2020	Stereo	✓	✓	S-DDM	Outdoor	SLAM, MDE	Car navigation
TartanAir [160]	2020	Stereo	✓	✓	DDM	Indoor, Outdoor	SLAM, MDE	Drone simulation
Omnidata [161]	2021	Mono			DDM	Indoor, Outdoor	MDE	Simulation
Hilti SLAM Challenge 2021 [11]	2021	Stereo	800 Hz	sparse		Indoor, Outdoor	SLAM	Handheld
Hilti-Oxford [162]	2022	Stereo	400 Hz	✓	3D scan	Indoor, Outdoor	SLAM	Handheld

Table 2.3 – Overview of various SLAM and MDE datasets detailing camera sensor types (Mono/Stereo), IMU frequency, availability of pose and map ground truth (GT), environmental settings, and data collection platforms. Map ground truth formats: Dense Depth Map (DDM), Semi-Dense Depth Map (S-DDM), or 3D scan. ✓indicates that the data is available. Blue rows identify indoor datasets with monocular-inertial data and pose ground truth, while green rows identify those that additionally provide map ground truth.

To test and evaluate SLAM and VIO methods, relevant datasets have to be identified. These datasets should include synchronized input data (camera images, IMU measurements) and associated ground truth (trajectory and point cloud or mesh model). The range of cameras includes visible (RGB), Infrared (IR), event-based (Events), and depth (RGB-D) sensors. Different IMU sensors can be used, their frequency usually ranges from 10 Hz to 1 kHz, and the synchronization with cameras can be software or hardware-based, the latter being the most precise. The ground truth trajectory can be a set of 3D positions or 6D poses for the whole track, or only for the start and the end of the sequence. On the other hand, the map ground truth is rarely available. If available, it is usually provided as depth maps captured from an RGB-D camera or a 3D point cloud collected by a high-precision LiDAR.

We present an overview of various SLAM and MDE datasets in Table 2.3. The datasets highlighted in blue are well adapted for the evaluation of monocular-inertial SLAM algorithms in indoor environments, as they provide pose ground truth. Those highlighted in green provide additional structure ground truth, allowing for the evaluation of dense

SLAM or MDE. The remaining datasets either focus exclusively on outdoor environments or serve to train and evaluate DNNs for MDE. Below, we discuss some of these classic benchmarks, focusing on those relevant to our study of dense monocular-inertial SLAM.

KITTI [5]

The **KITTI** dataset [5] is renowned in the computer vision community for providing a rich collection of data from urban driving scenarios for many tasks such as 3D object detection, depth estimation, and SLAM. It includes global shutter stereo camera images, IMU, GPS, and LIDAR data, along with accurate ground truth annotations for 3D point clouds and objects. This benchmark has been widely used to evaluate SLAM algorithms and is a reference in depth estimation for training and evaluating DNNs.

EuRoC [3]

The **EuRoC** dataset [3] serves as a benchmark, providing data captured by Micro-Aerial Vehicles (MAVs) over 11 sequences, categorized into three levels of difficulty, and conducted in two different indoor environments. This resource provides stereo inertial data complemented by precise ground truth for pose estimation. One of the environments also includes structure ground truth (3D scan of the room), facilitating the evaluation of 3D reconstruction and depth estimation capabilities. While the EuRoC dataset is a common benchmark for evaluating SLAM and VIO methods, it is limited by its confined indoor environments, lack of significant lighting variations, and absence of scenarios involving rapid, aggressive motion.

TUM datasets [4], [145]

TUM-RGB-D [145] represents a reference benchmark in the history of visual SLAM and has greatly inspired the trajectory evaluation procedure. The dataset consists of monocular RGB-D images captured with a Microsoft Kinect camera, especially in small indoor spaces. Meanwhile, **TUM-VI** [4] is a stereo-inertial dataset recorded indoors with a handheld setup, featuring a high dynamic range. It includes both short and long sequences, including loop closures, and is an essential benchmark for evaluating VIO and visual-inertial SLAM systems. However, this benchmark only provides partial pose ground truth, specifically at the beginning and end of the sequences.

UZH FPV Drone Racing dataset [10]

This dataset, later referred to as **UZH-FPV**, is known for its comprehensive coverage of FPV drone flight dynamics, challenging high-speed, aggressive flight maneuvers in both indoor and outdoor environments. Notably, this benchmark includes event camera data, facilitating comparative analysis between standard and event-based SLAM methods. With high-precision ground truth for drone pose tracking, UZH-FPV is specifically designed to evaluate VIO-based drone navigation systems.

HILTI SLAM Challenge datasets [11], [162]

The **HILTI** SLAM Challenge is a competition designed to push the boundaries of SLAM, highlighting the need for millimeter-level precision 3D mapping in complex environments. Datasets were collected from a handheld device integrating multiple visible cameras, IMUs, and LiDARs. They capture long indoor trajectories with high illumination changes in offices, labs, galleries, basements, and construction sites. However, they do not provide dense trajectory ground truth for most scenes, only a few highly accurate reference poses. Although precise LiDAR-generated point clouds are accessible, the absence of complete structure ground truth, primarily due to unknown corresponding poses and imperfect synchronization with the camera frame rate, is a limitation. However, the **Hilti-Oxford 2022** dataset recently included a detailed 3D scan for one of its scenes, which could be considered for evaluating dense reconstruction.

TartanAir [160]

The **TartanAir** dataset is a dataset simulated using the **AirSim** simulator [163], which renders realistic 3D environments using Unreal Engine. Scenarios are characterized by dynamic obstacles, varying lighting, and weather conditions that challenge conventional SLAM algorithms. The dataset includes multimodal sensor inputs, including stereo RGB, depth images, optical flow, and object semantics. The ground truth odometry is provided, but the structure ground truth has not been released.

Summary The field of visual SLAM and MDE has seen the adoption of various datasets, but not all are ideally suited for evaluating dense monocular-inertial SLAM in the context of indoor drone navigation. Such evaluation requires datasets with synchronized visual-inertial data, complemented by pose and map ground truths. We have

Figure 2.10 – Sample images from various datasets: EuRoC (top-left), UZH-FPV (middle), HILTI (top-right), and KITTI (bottom).

selected a few datasets among those presented above to validate localization robustness under challenging conditions. Specifically, we chose the **HILTI** dataset for its large-scale environments and variable lighting, and the **UZH-FPV** dataset for its high-speed motion sequences. The **EuRoC** dataset was chosen for the evaluation of dense mapping capabilities due to its common use in monocular SLAM densification research.

2.7 Conclusion

In this chapter, we have conducted a comprehensive presentation of the landscape of visual SLAM in the context of our study, which aims to produce dense and metric 3D maps in real-time for drone navigation.

The first section provided a thorough overview of the mathematical foundations of visual SLAM, including sensor modeling and the SLAM problem representation. This provided the basis for a better understanding of the monocular depth ambiguity and the complexity of the dense 3D mapping problem when applying classical approaches.

Therefore, we presented a comprehensive review of conventional visual SLAM methods, comparing sensor modalities and highlighting sparse and dense methods. We synthesized a broad view of this comparison in Table 2.1. We have seen that sparse visual-inertial methods have reached a level of maturity where they can achieve high localization accuracy and robustness in real-time. However, these methods have not been extensively evaluated for challenging indoor drone navigation scenarios characterized by larger exploration areas, rapid motion, and high illumination changes. We believe that conducting such a broader evaluation of state-of-the-art methods would provide crucial insights for our study, as such conditions are most likely to occur when operating a drone inside a building.

On the other hand, we have also noted how the early advancements of dense monocular SLAM were hindered by their complexity. However, as sparse SLAM became quite performant, the interest in dense mapping approaches was renewed. Therefore, in the following section, we have focused on the recent work toward densifying monocular SLAM, typically using deep learning-based MDE. The features of some of the main methods in this field are also summarized in Table 2.2. While substantial improvements have been made, a majority of the existing work has focused on achieving superior 3D reconstruction, often relying on large DNNs while ignoring some aspects such as metric scale or embedded computing. For drone navigation, we previously identified the voxel representation as appropriate. Given the aforementioned suitability of voxel representation for drone navigation, our approach seeks to densify monocular SLAM using MDE to build coarse but metric voxel maps tailored for real-time applications.

Naturally, we then presented a synthetic overview of Monocular Depth Estimation. that encompasses the most notable techniques of supervised and self-supervised approaches. This review highlights significant advancements in both categories, with leading methods claiming to predict the absolute scale and to provide good generalization capa-

bilities, although mostly demonstrated in outdoor environments. By integrating MDE with monocular-inertial SLAM, we expect that the robustly triangulated landmarks from SLAM algorithms can further improve the performance of MDE. Specifically, since we intend to apply indoor datasets that may be unfamiliar to the chosen DNNs.

Finally, we have defined the evaluation metrics for SLAM and depth estimation in the last sections, but we have also shed light on the key datasets that will allow us to benchmark our work. In the forthcoming chapters, we will present our main contributions towards the densification of monocular SLAM. Our objective is to facilitate dense and metric 3D mapping, paving the way for its future application in real-time drone navigation.

Chapter 3

From sparse to dense SLAM for drone indoor navigation

SLAM has become the preferred method for performing both localization and mapping tasks essential for autonomous drone navigation. However, when relying on a monocular-inertial setup, the scope is limited to less complex sparse methods to ensure real-time performance. This is mainly because many dense solutions are computationally intensive and require significant memory. To densify monocular SLAM, a common strategy is to use DNNs to predict the detailed depth information. Although much of the research in this area focuses on the accuracy of 3D reconstructions, our focus shifts to building a less detailed but metrically accurate representation. Our objective is to optimize its efficiency to make it viable for coarse 3D mapping, real-time navigation, and obstacle avoidance. The work described in this chapter was presented at the Geospatial Informatics XIII session of the SPIE Defense + Commercial Sensing 2023 conference in Orlando [67].

In this chapter, we present a SLAM pipeline tailored for autonomous drone navigation in indoor environments. This pipeline is based on a state-of-the-art SLAM algorithm, which estimates the camera pose and builds a sparse metric map in real-time. We incorporate a deep learning method to infer a dense depth map from a single image, which is then combined with the SLAM outputs to obtain a dense and metric depth map. These resulting depth data are then converted into a voxel map. The design of the system architecture is specifically tailored towards future real-time applications on embedded systems.

Then, we carry out a quantitative performance comparison of visual SLAM approaches in order to select a state-of-the-art method performing robust localization and sparse mapping. To determine its suitability for our specific indoor drone navigation context, we evaluate it on previously unexplored datasets. These datasets capture indoor environments with long trajectories, rapid motions, low lighting, and high illumination changes. Our comparison contrasts the capabilities of the selected method with other leading conventional visual SLAM solutions, shedding light on its strengths and limitations.

3.1 Navigation system architecture

We aim to create a navigation system that can be embedded into a small drone for real-time use. Therefore, the architecture design should take advantage of all the optimization capabilities available in the embedded systems under consideration. Since dense 3D mapping involves multiple processing and extensive computation, each module in the data pipeline should be optimized. Nowadays, embedded systems typically include hardware accelerators dedicated to camera acquisition to minimize CPU usage. Furthermore, since most SLAM methods are CPU-based, this leaves room in the GPU to run deep learning-based MDE solutions. The NVIDIA Jetson series is a perfect example of such embedded systems, featuring a compact form factor and specialized hardware accelerators for video and deep learning. Specifications for various NVIDIA Jetson modules are compiled in Table 3.1 from the resource published by the manufacturer [164]. This table, sorted by computing power, shows the continuous growth of the capabilities of these systems over the years, especially for deep learning applications. This trend is promising and supports the approach of using DNNs for depth prediction.

	Nano	**TX2**	**Xavier NX**	**AGX Xavier**	**Orin NX**	**AGX Orin**
Year	2019	2017	2020	2018	2023	2023
AI	0.472 TFLOPS	1.33 TFLOPS	21 TOPS	32 TOPS	100 TOPS	275 TOPS
CPU	4 core ARM A57	4 core ARM A57 2 core Denver	6 core Carmel ARM v8	8 core Carmel ARM v8	8 core A78 ARM v8	12 core A78 ARM v8
Memory	4 GB	8 GB	8/16 GB	32/64 GB	16 GB	64 GB
GPU	128 core Maxwell	256 core Pascal	384 core Volta	512 core Volta 1x NVDLA	1024 core Ampere	2048 core Ampere 2x NVDLA v2
Power	5-10 W	7.5-15 W	10-30 W	10-30 W	10-25 W	15-60 W
Weight	61 g	85 g	76 g	280 g	28 g	-

Table 3.1 – Comparison of different NVIDIA Jetson modules. The weights reported are for the modules alone, excluding the carrier board. The row labeled "AI" indicates the evaluated performance of the module for AI computations, measured in TFLOPS (Tera Floating Point Operations Per Second) or TOPS (Tera Operations Per Second).

3.1.1 Structuring software around the SLAM baseline

If we intend to rely on the aforementioned type of embedded systems, we should first outline the current architecture and computational load of typical SLAM solutions. In this study, we develop our system based on a state-of-the-art sparse monocular SLAM algorithm, prioritizing a solution that offers both accuracy and real-time performance.

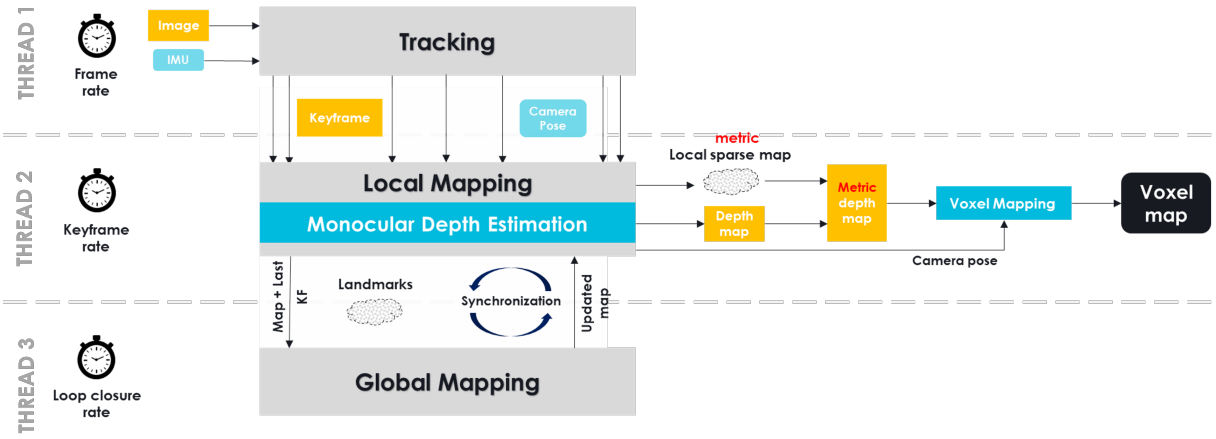


Figure 3.1 – Scheme of our proposed pipeline based on multi-threaded SLAM. The Local Mapping thread is extended with dense depth map prediction. The predicted dense depth and metric sparse depth are then fused together. The keyframe pose along with the metric dense depth map are processed through voxel mapping to build and update the voxel map.

Currently, most methods implement a multi-threaded architecture that distributes the computation across the CPU. As shown in Figure 3.1, a Tracking thread estimates the camera pose from monocular images combined with pre-integrated inertial measurements at the camera frame rate. At this point, a sparse set of keypoints is extracted and tracked over a sequence of frames for pose estimation. Selected frames that meet certain criteria, such as an overlap threshold, are designated as keyframes. Afterward, the Local Mapping thread processes each keyframe to triangulate keypoints and update the local map at the keyframe rate. Some algorithms implement an additional Global Mapping thread to perform place recognition and loop closure at a reduced frequency. Given the potential complexity of the mapping tasks, a sparse map representation is preferred for this operation, which is typically implemented as an iterative optimization of a factor graph.

3.1.2 Structural evolution: incorporating densification into SLAM

When exploring indoor environments at a reasonable flight speed, the experiments described in Section 3.2 revealed that the average keyframe rate remains moderate. It ranges from 2 to 4 Hz in the **EuRoC** [3] and **UZH-FPV** [10] datasets, and fluctuates around 6 Hz in the **HILTI** dataset [11]. Consequently, considering a keyframe rate ranging from 2 to 6 Hz reduces the constraints on deep learning inference to predict depth, which

in typical computer vision tasks is expected to run at camera frame rates of 20 to 30 Hz. Naturally, the keyframe rates may vary depending on specific factors such as the SLAM method complexity, the number of keypoints set, and the speed of the camera.

On a related note, research in [46], which benchmarks SLAM algorithms on various NVIDIA Jetson platforms, shows that while sparse SLAM requires a significant amount of CPU, it leaves the GPU largely free. Consider the results reported for the NVIDIA Xavier NX, whose specifications are reported in Table 3.1. The benchmark conducted on **Kimera** [12] shows an average load of 250% CPU (fluctuating between 100% and 360%), 30% memory, and 3% GPU. By comparison, **ORB-SLAM 2** [99] (stereo) exhibits slightly higher efficiency, with CPU consumption of 210% (ranging from 140% to 315%), memory consumption of 8%, and GPU consumption of 5%. Another study [45] demonstrated the real-time performance of **ORB-SLAM 2** by porting the algorithm on the NVIDIA Jetson TX2, a predecessor with fewer capabilities than the Xavier NX. These studies confirm that embedded systems can effectively run SLAM algorithms while primarily using the CPU without truly involving the GPU. This observation supports our belief that such systems can simultaneously use a DNN on the GPU to predict a dense depth map for each keyframe. Adopting a decoupled framework in which the tracking and mapping threads are independent of the densification extension will minimize the impact on the SLAM baseline speed.

Additionally, since MDE cannot reliably ensure metric scale, we propose to combine the densely predicted depth map with the sparse metric depth computed by SLAM within the Local Mapping thread. Inspired by Kimera [12], we opt for voxel mapping [39] to construct a voxel map using the scaled depth map and the estimated camera pose. This technique constructs a TSDF through bundled raycasting, where pixels converging on the same voxel are raycasted together. This method is highly efficient and works exclusively on the CPU, but can also be ported to the GPU using nvblox [165]. It is worth mentioning again that the choice of voxel representation proves to be advantageous for navigation tasks [40]. The voxel mapping approach [41] highlighted the enhanced scalability and geometric reasoning of this kind of structure. Additionally, the use of TSDF, and by extension ESDF, is optimal for path planning tasks as it inherently builds an occupancy grid and provides a metric distance to any obstacle for each voxel. Finally, this provides an opportunity for further optimization, such as voxel hashing, which can further improve map usage and storage [41].

In this section, we have explored the densification of monocular SLAM tailored to

drone navigation and have proposed a pipeline that is aimed toward future applications on embedded systems. However, such a framework strongly relies on the chosen sparse SLAM baseline, the accuracy of which is therefore critical. Consequently, the following section focuses on the evaluation of state-of-the-art visual SLAM algorithms in order to select an appropriate monocular-inertial method as a baseline to integrate into our work.

3.2 Sparse SLAM benchmark

This section presents a thorough evaluation of selected visual SLAM solutions in challenging drone navigation scenarios. The purpose of this benchmark is restated by presenting the algorithms and datasets under consideration and then detailing the experimental procedure. Quantitative results are reported and discussed, notably to justify the selection of the sparse SLAM baseline for our study, but more generally to highlight the strengths and limitations of the algorithms studied.

3.2.1 Robust pose estimation in challenging conditions

In dense 3D mapping, accurately estimating trajectory is critical because landmarks are typically projected from the estimated camera poses. However, tracking the camera ego-motion in challenging scenes can be tricky. As we noted in Section 2.2, **ORB-SLAM 3** [1] is nowadays considered the leading solution across most configurations, and especially in the monocular-inertial category as demonstrated by the authors. Therefore, we have decided to study this method to integrate it into our work. In this section, we confirm this choice by further evaluating its performance and robustness in monocular-inertial and stereo-inertial settings in various indoor drone navigation scenarios, including fast motion, low illumination, and high illumination changes. For comparison, we also evaluate the competitive stereo-inertial methods, **Basalt** [2] and **Kimera** [12].

All methods considered have already been benchmarked on the **EuRoC** dataset [3], which was collected from a small drone in small rooms. This dataset does not include large environments and presents few challenging sensing conditions. Still, to verify the consistency of our results with those reported by the original authors, we retested these three algorithms on the **EuRoC** benchmark. Subsequently, we used the **UZH-FPV** dataset [10] to specifically evaluate the algorithms under aggressive motion and high speeds in both indoor and outdoor environments. The dataset was collected with two camera pitch configurations: one facing downwards and the other facing forward. Given the low flight altitudes, sequences recorded with the camera pointing downward exhibit particularly high optical flow. Additionally, the camera's focus on the ground results in many captured images displaying uniform textures. Lastly, we integrated the **HILTI** dataset [11] to address the challenges associated with large indoor scenes, particularly those with low lighting and significant illumination variations. However, the absence of calibration sequences for this dataset has prevented the use of the Double Sphere camera model preferred for **Basalt**.

3.2.2 Experimental procedure

The experiments were carried out on a laptop with an Intel i7-8750H CPU, 16GB memory, and an NVIDIA RTX 2080 Mobile GPU. For our experimental setup, we developed a script that ran each method 10 times on each scene and then saved the trajectory results. We use the evo library [166] to align the estimated trajectories with the ground truth on $SE(3)$. We calculate ATE and RPE_{trans} in meters, RPE_{rot} in degrees, and generate statistics for these metrics, such as *median*, *mean*, and *std*. To evaluate the different methods, we report the median and the standard deviation (std) of the results from the 10 executions, providing valuable insight into robustness [1]. Furthermore, the system's ability to consistently track the pose throughout the entire trajectory was measured by the coverage percentage from the 10 executions, thus providing an additional measure of the robustness of the algorithms. Additionally, we present plots juxtaposing the estimated trajectories with the ground truth to visualize the shape of the estimated trajectory and the potential scale errors. For this purpose, we use the median trajectory (based on the ATE) among the 10 executions, as indicated by the index following the method name in the plots legend. To conduct a thorough analysis, we also present color plots of the RPE errors, which effectively highlight local performance nuances, such as during rotations.

— **ORB-SLAM 3**: The algorithm uses a unique format for calibration parameters and includes fine-tuned configuration files for the **EuRoC** dataset. For other datasets, we developed a script to convert calibration parameters from the **Kalibr** format [167]. Furthermore, we adapted the original ORB-SLAM 3 implementation to the data formats of **UZH-FPV** and **HILTI**. This adaption also allows to save trajectories, sparse map keypoints, and the associated reprojection errors.

— **Kimera**: This method uses a custom format for camera calibration parameters and provides configuration files specific to **EuRoC**. A conversion script for calibration files from the **Kalibr** format is readily available. We use ROS (Robot Operating System), a flexible framework for writing robot software, to run Kimera and capture the estimated camera poses.

— **Basalt**: This system leverages the double sphere camera model which requires specific camera calibration. The authors provide the full configuration files for **EuRoC** and **UZH-FPV** datasets. However, we could not run Basalt on the **HILTI** dataset since the calibration sequences were not published. Basalt implementation is quite advanced and provides options to save the estimated trajectory and other statistics.

3.2.3 Results

In the following, we present the quantitative results for each dataset and analyze the performance of each method with respect to different metrics.

EuRoC

We first assess the algorithms on the EuRoC dataset [3] to ensure that each algorithm is correctly set up and to attempt to replicate the reported results. The outcomes are presented in Table 3.2. Generally, they align with the results reported for **Basalt** [2] and **ORB-SLAM 3** [1], exhibiting an average absolute difference of less than 20 mm. While the authors did not publish results on the difficult scene *V203*, we were able to successfully execute the algorithm using their updated code without any issues. On the other hand, it is notable that we could not achieve the same outcomes for **Kimera** [12], where we noticed a more significant average absolute difference of 352 mm, particularly in challenging scenes *MH04* and *MH05*. Detailed results are presented in Table 3.2 and discussed subsequently.

EuRoC scenes		**MH01**	**MH02**	**MH03**	**MH04**	**MH05**	**V101**	**V102**	**V103**	**V201**	**V202**	**V203**	**Mean**
Basalt	reported	0,080	0,060	0,050	0,100	0,080	0,040	0,020	0,030	**0,030**	0,020		0,051
	measured	0,082	0,041	0,050	0,095	0,133	0,043	0,049	0,060	0,039	0,053	0,239	0,080
	difference	0,002	0,019	0,000	0,005	0,053	0,003	0,029	0,030	0,009	0,033		0,018
Kimera	reported	0,080	0,090	0,110	0,150	0,240	0,050	0,110	0,120	0,070	0,100	0,190	0,119
	measured	0,112	0,090	0,79	3,533	0,173	0,060	0,059	0,175	0,051	0,081	0,394	0,437
	difference	0,032	0,000	0,031	3,383	0,067	0,010	0,051	0,055	0,019	0,019	0,204	0,352
OS3	reported	**0,036**	**0,033**	**0,035**	**0,051**	0,082	**0,038**	**0,014**	**0,024**	0,032	**0,014**	**0,024**	**0,035**
	measured	0,037	**0,028**	**0,027**	**0,052**	0,083	**0,036**	**0,015**	0,025	**0,026**	**0,014**	0,026	**0,034**
	difference	0,001	0,005	0,008	0,001	0,001	0,002	0,001	0,001	0,006	0,000	0,002	0,003
OS3 (MI)	reported	0,062	0,037	0,046	0,075	**0,057**	0,049	0,015	0,037	0,042	0,021	0,027	0,043
	measured	**0,031**	0,053	0,034	0,129	**0,073**	0,044	**0,015**	**0,022**	0,039	0,019	**0,022**	0,044
	difference	0,031	0,016	0,012	0,054	0,016	0,005	0,000	0,015	0,003	0,002	0,005	0,014

Table 3.2 – *ATE* measured on the EuRoC dataset using Basalt, Kimera, and ORB-SLAM 3 (OS3) in stereo-inertial and monocular-inertial (MI) settings. The table contrasts results published by the authors with those that we measured and presents the absolute differences. The best results from each set for each scene are shown in bold.

In addition, we report our measurements in more detail in Table 3.3, including the std and the percentage of trajectory coverage. The following will discuss the results for the *V102* scene, which are representative of the general performance in most scenes. This sequence spans 83.50 seconds, during which the ground truth encompasses 17,702 poses,

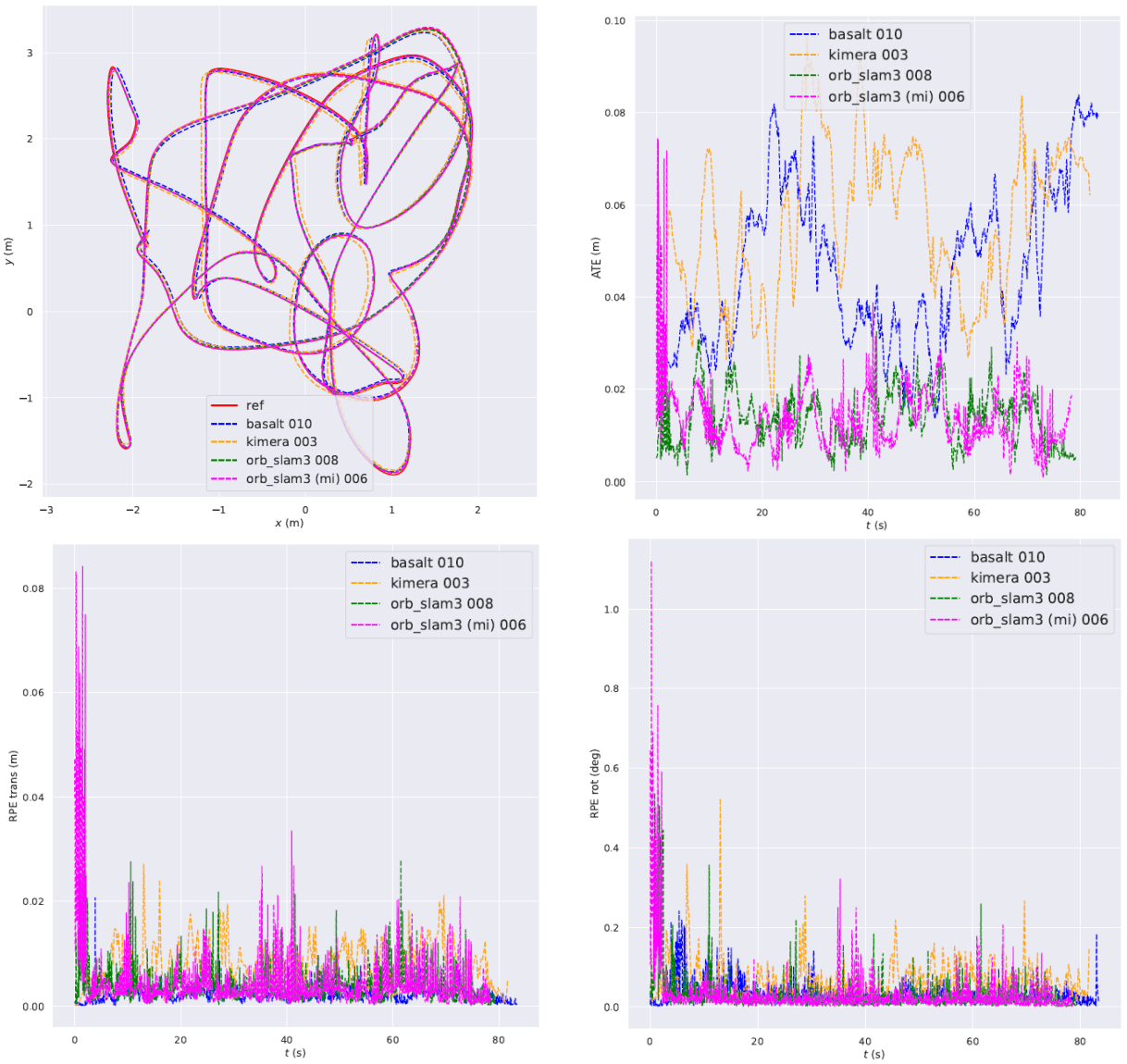


Figure 3.2 – Evaluation of Basalt, Kimera, ORB-SLAM 3, and ORB-SLAM 3 (MI) on the *V102* sequence from the EuRoC dataset. The plots presented are: aligned trajectories (top-left), ATE (top-right), RPE_{trans} (bottom-left), and RPE_{rot} (bottom-right).

resulting in a trajectory of 75.89 m at an average speed of 2.19 m/s. Figure 3.2 shows the estimated trajectories for each method alongside the ground truth. The error metrics measured are also plotted against time. Generally, we can observe peaks in the ATE curves associated with scaling errors or accumulated drift. Therefore, the RPE provides a more appropriate representation of local performance because it allows local estimation comparisons independent of previous conditions.

Scene	Basalt		Kimera		ORB-SLAM3		ORB-SLAM3 (MI)	
	ATE (m)	cover	ATE (m)	cover	ATE (m)	cover	ATE (m)	cover
MH01	0.082 (0.082 ± 0)	100%	0.112 (0.118 ± 0.019)	99%	<u>0.037</u> (0.036 ± 0.004)	100%	**0.031** (0.029 ± 0.008)	99%
MH02	<u>0.041</u> (0.041 ± 0)	100%	0.090 (0.090 ± 0.010)	99%	**0.028** (0.028 ± 0.006)	100%	0.053 (0.056 ± 0.015)	80%
MH03	0.050 (0.050 ± 0)	100%	0.079 (0.093 ± 0.091)	50%	**0.027** (0.028 ± 0.002)	87%	<u>0.034</u> (0.085 ± 0.152)	87%
MH04	<u>0.095</u> (0.095 ± 0)	100%	3.533 (3.002 ± 1.231)	85%	**0.052** (0.052 ± 0.002)	82%	0.129 (0.127 ± 0.028)	90%
MH05	0.133 (0.133 ± 0)	100%	0.173 (0.172 ± 0.018)	99%	<u>0.083</u> (0.080 ± 0.012)	84%	**0.073** (0.075 ± 0.018)	81%
V101	<u>0.043</u> (0.043 ± 0)	100%	0.060 (0.061 ± 0.004)	99%	**0.036** (0.037 ± 0.000)	97%	0.044 (0.043 ± 0.004)	97%
V102	0.049 (0.049 ± 0)	100%	0.059 (0.061 ± 0.005)	98%	**0.015** (0.014 ± 0.002)	95%	<u>0.015</u> (0.016 ± 0.002)	94%
V103	0.060 (0.060 ± 0)	100%	0.175 (0.175 ± 0.013)	99%	<u>0.025</u> (0.024 ± 0.002)	95%	**0.022** (0.025 ± 0.008)	93%
V201	<u>0.039</u> (0.039 ± 0)	100%	0.051 (0.052 ± 0.010)	99%	**0.026** (0.026 ± 0.006)	97%	0.039 (0.041 ± 0.004)	97%
V202	0.053 (0.053 ± 0)	100%	0.081 (0.083 ± 0.010)	99%	**0.014** (0.014 ± 0.002)	98%	<u>0.019</u> (0.018 ± 0.003)	98%
V203	0.239 (0.239 ± 0)	100%	0.394 (0.397 ± 0.022)	99%	<u>0.026</u> (0.031 ± 0.010)	96%	**0.022** (0.025 ± 0.007)	96%
Mean	0.080	100.0%	0.437	93.2%	**0.034**	93.7%	<u>0.044</u>	92.0%
std	0.060	0.0	1.032	14.9	0.019	6.4	0.033	6.7

Table 3.3 – Evaluation of Basalt, Kimera, and ORB-SLAM 3 on the EuRoC dataset. The ATE values are presented in the format: $median$ ($mean \pm std$). The last two rows report the mean and std of the median and cover values. For each sequence, the best result is highlighted in boldface, and the second is underlined.

In this experiment, running **Basalt** was straightforward and did not require any significant modifications. As a result, we observe excellent robustness, as the algorithm provides constant performance and full trajectory coverage, as shown in Table 3.2. Thus, the average error across all scenes is 80 mm. Looking more closely, the RPE plots show that the system initializes quickly and keeps relative errors below 0.25 degrees for rotations and 21 mm for translations.

ORB-SLAM 3 achieves outstanding accuracy on this benchmark by reaching an average error of 34 mm across all scenes, and 44 mm in monocular-inertial setup. However, its robustness appears to be weaker than **Basalt**, as the results are less consistent and the coverage is not always complete. This is primarily due to the initialization process, which requires parallax and sufficient motion with translation and rotation. The initialization is even more sensitive in monocular-inertial settings, as we can see in the RPE plots, where high error peaks are present at the beginning of tracking.

Regarding **Kimera**, even though we could not exactly reproduce the published results, we obtained an average error of 448mm instead of 119 mm. While it performs correctly on the easiest scenes keeping the error below 100 mm, it is severely impacted on difficult scenes. In particular, the algorithm failed to track the camera multiple times on the *MH04* scene.

Supplementary results for this difficult scene are reported in Appendix B.1 to maintain focus on the main discussion and avoid overloading it with extensive figures. The RPE plots for the *MH04* scene suggest that **Kimera** encounters difficulties in precisely

estimating some rotations and the scale of some translations. However, it is important to note that the original authors of **Kimera** achieved greater accuracy on this sequence compared to us, potentially due to the use of different parameterizations. This particular sequence contains some very dark scenes. Still, **ORB-SLAM 3** demonstrates remarkable resilience under such conditions, using **ORB** features that are much more robust than the **Shi-Tomasi** corners used by **Kimera**. On the other hand, **Basalt** relies on the **FAST** feature detector and tracks corresponding patches with **KLT**. It also employs a pyramidal method to handle large displacements. Corresponding plots of these two previous systems demonstrate excellent trajectory estimation.

UZH-FPV

In the following analysis, we benchmark these algorithms on the UZH-FPV dataset [10] to confront them with high dynamic motion. The overall results are shown in the Table 3.4. The scenes captured with the camera facing forward are more in line with the building exploration scenario under the scope of this thesis. However, we will also examine the results of the sequence with the camera facing down at an angle of 45 degrees.

Scene	Basalt		Kimera		ORB-SLAM3		ORB-SLAM3 (MI)	
	ATE (m)	cover	ATE (m)	cover	ATE (m)	cover	ATE (m)	cover
indoor_45_2	**1.083**(1.083 ± 0.000)	100%	$16.027(15.228 \pm 5.114)$	100%	3.952(11.964 ± 17.188)	70%	-	-
indoor_45_4	$0.740(0.740 \pm 0.000)$	100%	$14.716(20.064 \pm 15.584)$	99%	0.568(10.289 ± 17.520)	83%	**0.440**(0.440 ± 0.052)	54%
indoor_45_12	$1.392(1.392 \pm 0.000)$	100%	$6.437(17.157 \pm 19.916)$	99%	1.342(1.343 ± 0.032)	97%	**0.645**(0.668 ± 0.096)	61%
indoor_45_13	**1.649**(1.649 ± 0.000)	100%	28.349(29.880 ± 5.967)	99%	-	-	-	-
indoor_45_14	**2.965**(2.964 ± 0.003)	100%	51.599(173.591 ± 360.774)	99%	-	-	-	-
outdoor_45_1	**1.722**(1.784 ± 0.188)	100%	12.010(10.123 ± 3.492)	99%	-	-	-	-
indoor_forward_3	$0.731(0.731 \pm 0.000)$	100%	$3.365(3.982 \pm 1.660)$	100%	**0.493**(0.500 ± 0.023)	100%	0.508(0.516 ± 0.064)	95%
indoor_forward_5	1.130(1.130 ± 0.002)	100%	$2.998(3.125 \pm 1.173)$	99%	$1.368(1.286 \pm 0.331)$	100%	**1.023**(1.149 ± 0.684)	98%
indoor_forward_6	$1.404(1.404 \pm 0.000)$	100%	$6.017(5.861 \pm 0.954)$	99%	0.496(0.503 ± 0.033)	100%	**0.432**(0.467 ± 0.088)	100%
indoor_forward_7	$1.268(1.268 \pm 0.000)$	100%	$6.251(5.631 \pm 1.188)$	100%	0.341(0.858 ± 1.065)	100%	**0.167**(0.176 ± 0.034)	100%
indoor_forward_9	$1.583(1.583 \pm 0.001)$	100%	$3.533(3.558 \pm 0.126)$	99%	0.984(0.974 ± 0.195)	100%	**0.575**(0.703 ± 0.193)	99%
indoor_forward_10	$1.520(1.520 \pm 0.002)$	100%	$3.502(3.477 \pm 0.101)$	99%	0.877(1.142 ± 0.811)	100%	**0.607**(0.669 ± 0.128)	100%
outdoor_forward_1	$1.618(1.618 \pm 0.000)$	100%	-	-	1.383(1.405 ± 0.096)	100%	**1.200**(1.224 ± 0.086)	100%
outdoor_forward_3	$2.126(2.126 \pm 0.001)$	100%	$14.367(20.157 \pm 11.222)$	100%	1.553(1.600 ± 0.330)	100%	**1.051**(1.169 ± 0.342)	100%
outdoor_forward_5	**3.031**(3.031 ± 0.000)	100%	11.269(11.714 ± 1.353)	100%	-	-	-	-
Mean	1.432	100.0	12.900	99.3	1.158	94.4	**0.550**	88.4
std	0.582	0.0	14.297	0.5	1.112	10.7	0.242	19.2

Table 3.4 – Evaluation of Basalt, Kimera, and ORB-SLAM 3 on the UZH-FPV dataset. The ATE values are presented in the format: $median$ $(mean \pm std)$.

Camera facing forward:

We first discuss the results for the *forward* sequences. For illustration, we present results from the *indoor_forward_9* sequence in Figure 3.3, which is representative of the overall performance. This scene recorded a trajectory of 136.34 m over 29.31 seconds, with 14,658 ground truth poses. The average speed is 4.65 m/s and the peak speed is 8.80 m/s. As a result, significant optical flow is observed in the images captured as detailed in [10].

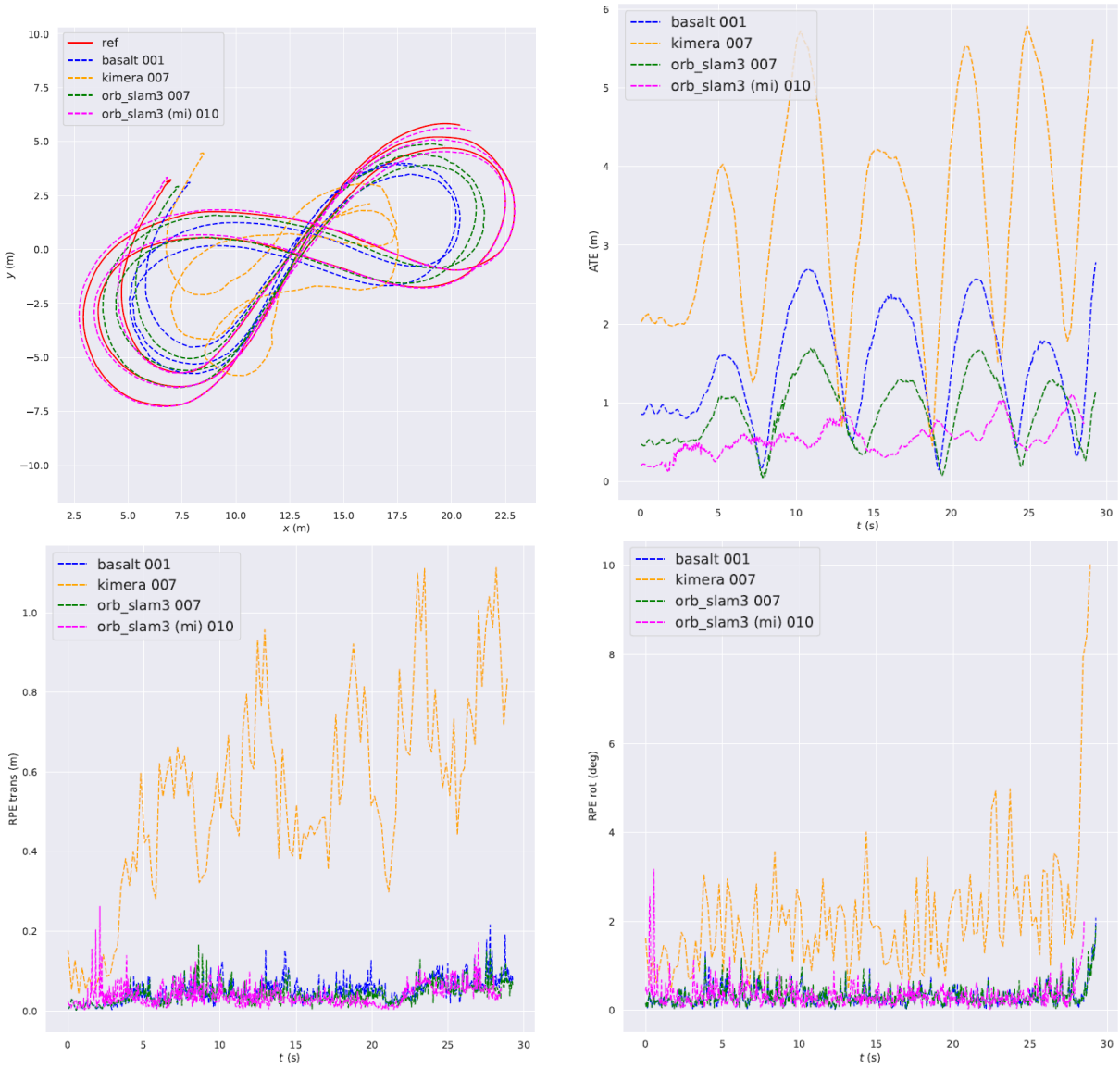


Figure 3.3 – Evaluation of Basalt, Kimera, ORB-SLAM 3, and ORB-SLAM 3 (MI) on the *indoor_forward_9* sequence from the UZH-FPV dataset. The plots presented are: aligned trajectories (top-left), ATE (top-right), RPE_{trans} (bottom-left), and RPE_{rot} (bottom-right).

Basalt [2], previously configured for this dataset by its authors, demonstrates a high level of robustness with a negligible standard deviation in all 10 runs. As depicted in Figure 3.3, the shape of the trajectory is accurate, however, the scale estimation is incorrect, leading to reduced accuracy of the measurements. Thus, errors tend to amplify on curves, making the peaks on the ATE plot more pronounced.

Results in Table 3.4 demonstrate that **ORB-SLAM 3** performs best in monocular-inertial configuration, surpassing the performance of **Basalt**. Although we did not quantify the scale error, the trajectory plot shows that the scale estimation is excellent. Nonetheless, the stability of the algorithm is lower, with an average std of 0.373 m and up to 2.05 m in the most difficult scene. The top speed in indoor scenes is 12.8 m/s, and the ATE error ranges from 0.167 m to 1.041 m, which is fairly good considering the high speed and the important optical flow. We observed that the initialization tends to fail at the start of the track when the motion is slow. This translates to a reduced coverage percentage, as seen in the initial seconds on the RPE plots in Figure 3.3.

Unexpectedly, the stereo-inertial version of **ORB-SLAM 3** is slightly less accurate. We explain this primarily by the sensitivity of the algorithm to calibration accuracy. In practice, stereo-inertial calibration is more complex than its monocular-inertial counterpart. Nevertheless, the algorithm initialization is more efficient and produces a greater coverage and a lower standard deviation.

As for **Kimera** [12], it seems to have great difficulty in keeping good track of the drone's trajectory, especially when estimating rotations in curves. Furthermore, scale estimation is poor, as shown by the plots in Figure 3.3. It is possible that adjusting further the algorithm's parameters could enhance the results, given that our outcomes were already below those of the authors on EuRoC. As before, we report more detailed plots for each method on this sequence in Appendix B.1.

Among the *forward* sequences, *indoor_forward_7* is particularly difficult since it is longer (313.90 m) and features significant challenges with complex rotations and aggressive motions. Some consequences are high optical flow and motion blur, as shown in Figure 3.4, which make feature detection and matching very difficult. For detailed results, we refer to Appendix B.1. To sum up, we observed again that **Kimera** struggled to accurately track camera motion, particularly during rotations, as indicated by high RPE_{rot} values. Unexpectedly, in this specific sequence, despite providing a rough trajectory estimate, **Basalt** exhibited large shifts due to accumulated drift, with occasional extreme RPE_{rot} peaks (up to 10 degrees) affecting trajectory accuracy. On the other hand, in line with previous observations, **ORB-SLAM 3** showed relatively good performance considering the length and difficulty of this sequence, especially in its monocular-inertial configuration.

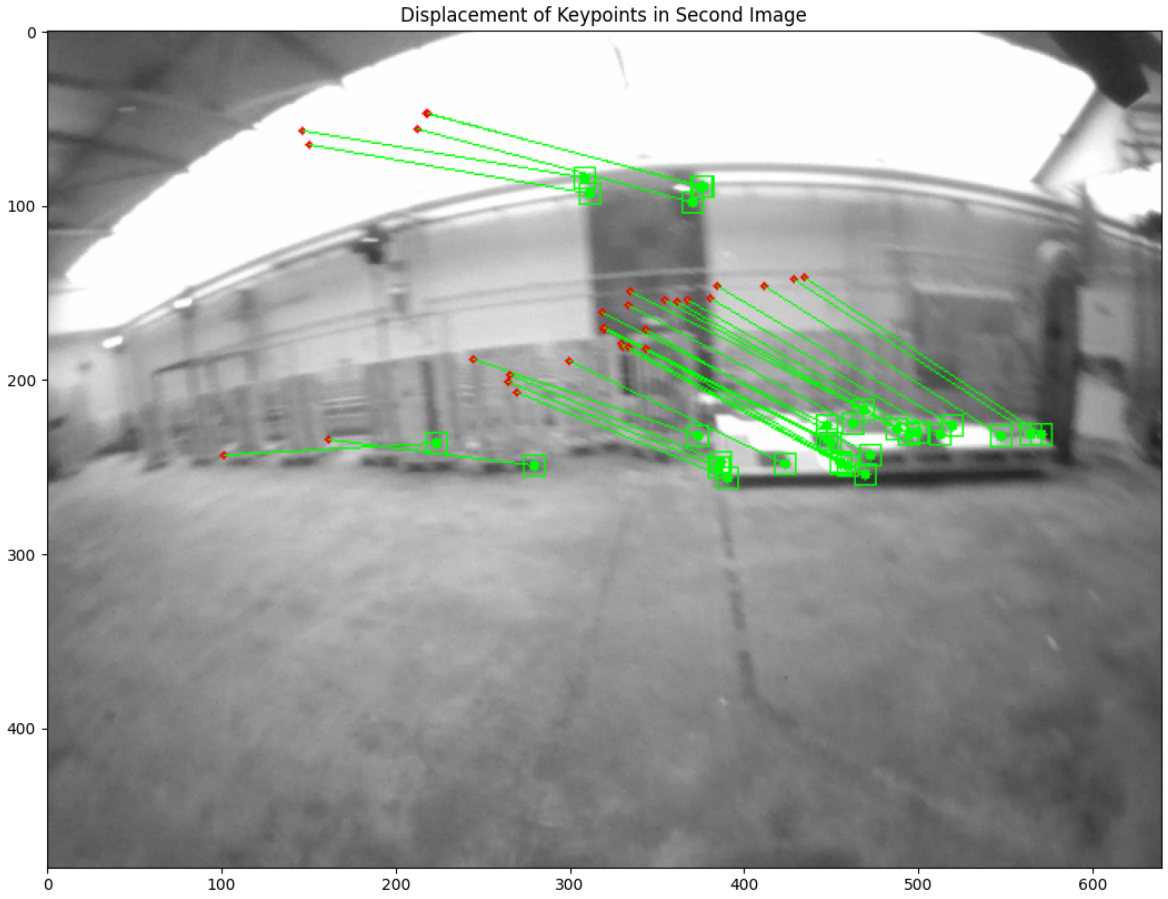


Figure 3.4 – ORB feature detection and matching on the *indoor_forward_7* sequence of the HILTI dataset. The figure shows the displacement of keypoints between successive frames, with start points in red and end points in green.

Camera facing downward:

For information, we also comment on the results of the sequences captured with the camera pointing downwards. In Figure 3.5, we report the different plots for the sequence *indoor_45_12*. This sequence covers a distance of 111.12 m during 40.82 seconds, with an average speed of 2.72 m/s and a maximum speed of 4.33 m/s. The low flight altitude and camera angle engender significant optical flow in the *indoor_45* and *outdoor_45* sequences. The flight altitude also leads to predominantly textureless scenes, as can be seen in Figure 3.6. Consequently, detecting and tracking keypoints in such frames is particularly challenging.

Overall, the observed behavior aligns with the patterns seen under previous conditions. **Basalt** demonstrates excellent robustness but continues to struggle with metric scale es-

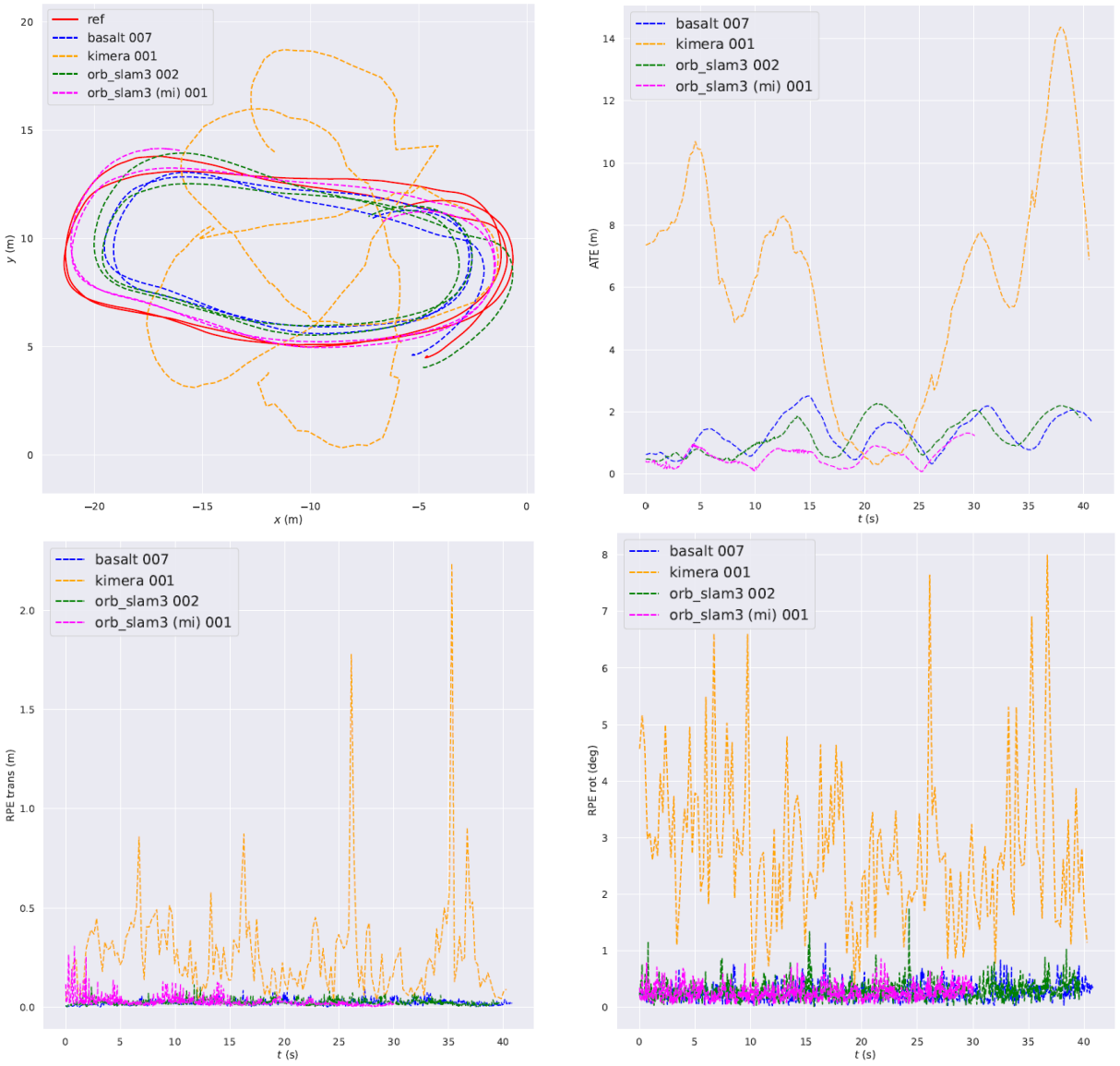


Figure 3.5 – Evaluation of Basalt, Kimera, ORB-SLAM 3, and ORB-SLAM 3 (MI) on the *indoor_45_12* sequence from the UZH-FPV dataset. The plots presented are: aligned trajectories (top-left), ATE (top-right), RPE_{trans} (bottom-left), and RPE_{rot} (bottom-right).

timation. **Kimera**, under these conditions, fails to track the camera, leading to highly inconsistent results. The estimated trajectory significantly loses its way during curves, leading to notable rotational errors. **ORB-SLAM 3** encounters difficulties during initialization and often loses track in more difficult scenes. However, in monocular-inertial setups, once the system manages to initialize successfully, it delivers exceptional accu-

racy. We have observed a predominant failure of the system during the initialization of the IMU parameters. Moreover, there were several instances in which the algorithm struggled to track features with large displacements in the image, or to detect new features, and was not able to make effective use of the inertial data during these periods. Individual measurements for each method are also reported in the Appendix B.1.

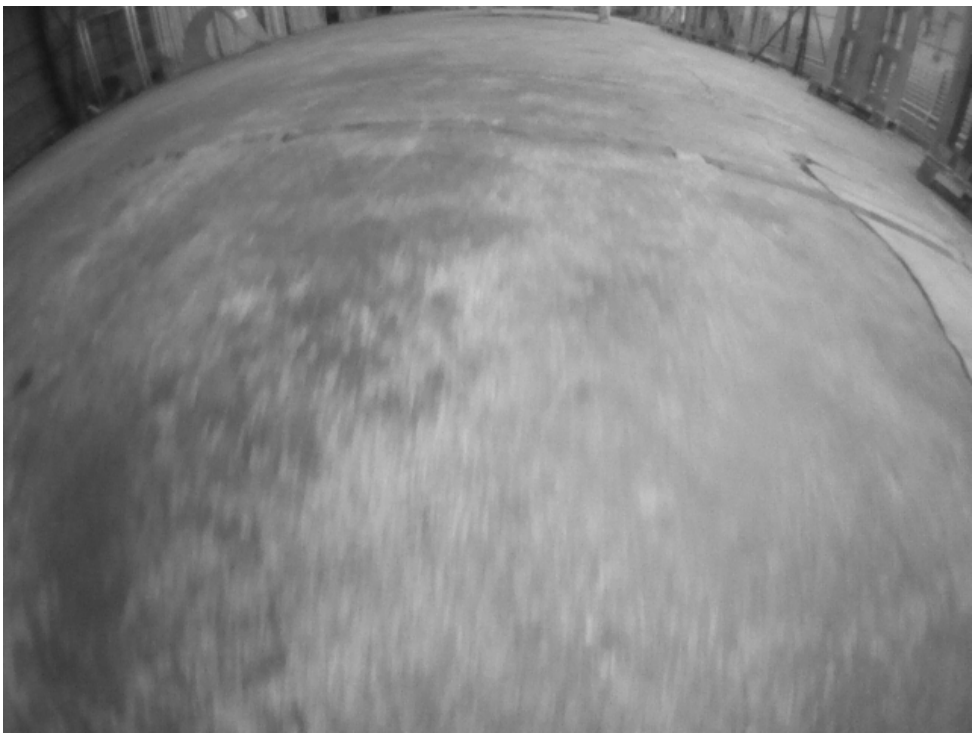


Figure 3.6 – Image sample from *indoor_45_12* sequence of UZH-FPV dataset.

HILTI

We present in Figure 3.7 the results of **ORB-SLAM 3** running in a monocular-inertial configuration on the HILTI dataset. The estimated trajectory is plotted against the highly sparse reference. Unfortunately, we could not successfully run the other algorithms on this dataset, either due to problems with the calibration data or their inability to track the camera effectively. **ORB-SLAM 3** provides accurate trajectory estimations across various scenes, with an ATE ranging from 20 mm to 300 mm. This is a respectable performance, especially considering the length and challenges of some of the scenes.

The sequence *Basement_1* is particularly interesting because it was captured in underground corridors where lights are activated as the operator approaches, resulting in

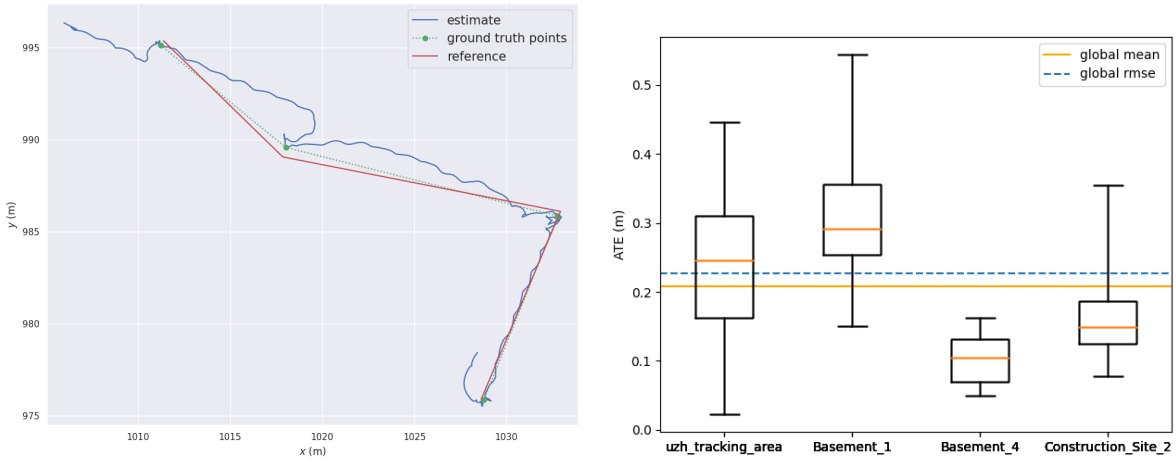


Figure 3.7 – Evaluation of ORB-SLAM 3 in monocular-inertial setup on the HILTI dataset. Left image: Trajectory estimated on scene *Basement_1*. Right image: Boxplots of the ATE on different scenes.

significant lighting changes. An illustration of these illumination changes can be seen in Figure 3.8. Despite these conditions, **ORB-SLAM 3** manages to maintain tracking and provides accurate metric scale pose estimation. The corresponding trajectory plot is depicted in Figure 3.7. We have observed comparable findings in the other scene, as shown in Appendix B.1.

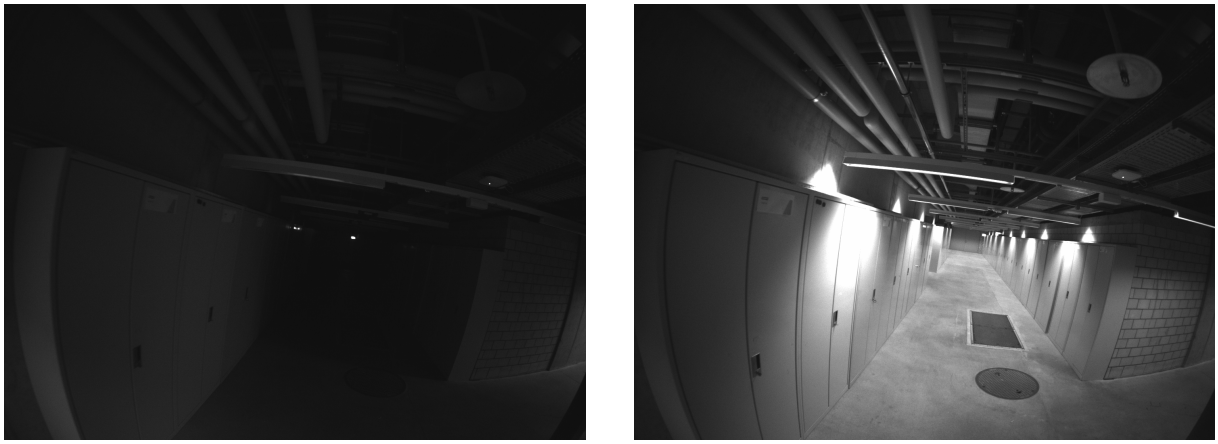


Figure 3.8 – Consecutive frames from the *Basement_1* sequence of the HILTI dataset, showcasing significant illumination changes.

3.2.4 Discussion

These initial experiments suggest that **ORB-SLAM 3**, when configured in a monocular-inertial setup, provides a promising SLAM baseline for our drone navigation system. Across all three datasets selected, it consistently attained competitive results, successfully handling high-speed motion and changes in illumination. In some sequences, it even outperformed other stereo-inertial algorithms. Nonetheless, some limitations were also observed, such as with the predominantly textureless images. Moreover, the algorithm requires a thorough tuning of parameters and precise calibration. In other experiments conducted with our data, while we were able to obtain fair results in a purely monocular setup, the monocular-inertial configuration did not yield satisfactory results. We primarily attribute this to the inferior quality of the built-in IMU in our setup when compared to the IMUs used in the previous benchmarks. Furthermore, we noted an initialization delay at the beginning of some sequences, especially when there was insufficient motion for the algorithm to initialize the IMU.

While our experiments might not fully represent the potential of **Kimera** due to possible enhancements through fine-tuning by the authors, the results do reveal some weaknesses in the accurate estimation of challenging trajectories, especially regarding rotations. Still, it is important to remember that the conditions in the **UZH-FPV** dataset are particularly challenging for feature detection and matching. Although **Basalt** demonstrated excellent robustness across all sequences, it struggled with accurate scale estimation. Additionally, in the most challenging scenes with aggressive motion, **Basalt** occasionally failed to correctly estimate rotations, resulting in significant cumulative drift in its trajectory estimates.

3.3 Conclusion

This chapter has presented a preliminary study for prototyping a real-time dense and metric 3D mapping system suitable for drone navigation. Indeed, existing work on densifying monocular SLAM often lacks metric scale and is not specifically designed for drone navigation. Our goal was to explore the densification of monocular SLAM for future implementation in embedded systems. Additionally, we aimed to expand the scope of visual SLAM benchmarking by focusing on the challenging conditions found in drone applications in order to identify an appropriate SLAM baseline for our project.

We have introduced a pipeline to densify monocular SLAM by integrating DNNs for dense depth map prediction. The architecture design is motivated by reference benchmark analysis of SLAM performance on embedded systems. Considering that conventional SLAM architectures predominantly use the CPU, we extend the mapping thread to the GPU for densification. In particular, this involves inferring a dense depth map for each keyframe using a DNN while minimizing disruption to the original SLAM process. The feasibility of this approach is further justified by both the low keyframe rate observed in our experiments and the growing computational power of modern embedded systems. Nevertheless, the pipeline is designed under the assumption of a moderate navigation speed to maintain a reasonable keyframe rate. Finally, we incorporate an incremental voxel mapping strategy, which will be discussed in the following chapter.

Subsequently, we selected a sparse SLAM baseline to integrate into the presented densification pipeline. This analysis is supported by a thorough evaluation on several benchmarks, some of which have not previously been used for visual SLAM assessment. In fact, the **UZH-FPV** dataset has been primarily used to evaluate VIO algorithms, while the **HILTI** dataset has been predominantly applied to LiDAR-based SLAM research. Our experiments have shown that **ORB-SLAM 3** offers accurate camera tracking in a variety of difficult scenarios, although there are some resilience issues, especially regarding IMU initialization. Given its satisfactory overall performance, we have confirmed the selection of **ORB-SLAM 3** in its monocular-inertial configuration for our subsequent work.

Building on the presented pipeline and the selected SLAM baseline, the next chapter introduces a scale recovery process that combines the outputs of the sparse SLAM with the dense depth map predicted by a DNN. This work is evaluated and compared against existing methods. Furthermore, an initial presentation of the resulting voxel map is provided which illustrates the full implementation of the proposed framework.

Chapter 4

Monocular SLAM densification

In the previous chapter, we proposed a dense and metric 3D mapping pipeline designed for real-time operation on embedded systems, where we established ORB-SLAM 3 as our sparse SLAM baseline. In this chapter, we implement the proposed framework and focus on the challenge of achieving dense and metric mapping through monocular SLAM densification.

In this context, we incorporate a state-of-the-art DNN, which infers a dense depth map from a single image, into our loosely coupled framework, allowing the restoration of metric scale from the sparse depth data estimated by SLAM. We conduct a quantitative comparison with related works, as well as a qualitative analysis of the results.

We then proceed to construct a voxel map using the rescaled dense depth maps and the estimated camera poses. This map is created through a raycasting process that allows for an iterative volumetric fusion of depth maps from multiple viewpoints. Finally, our experimental results will be presented through a qualitative analysis.

The loosely coupled strategy for producing dense and metric depth maps, in particular the scale recovery procedure, was presented at the 9th International Conference on Automation, Robotics and Applications (ICARA) 2023 in Abu Dhabi [68]. Additional results and the work on volumetric fusion were also presented at the SPIE Defense + Commercial Sensing 2023 conference in Orlando, for the Geospatial Informatics XIII session [67].

4.1 Loosely coupled dense and metric depth estimation: scale recovery

We propose to densify SLAM sparse maps for UAV navigation in two stages. The overall framework is illustrated in Figure 4.1 which operates at keyframe rate. In addition to **ORB-SLAM 3** [1], we have initially integrated **PackNet-Sfm** [8] for depth estimation. This self-supervised method was qualified as scale-aware and shows promising generalization capabilities, as described in Section 2.4. The scale-awareness refers to the ability to estimate the absolute scale. Nevertheless, since the transition from outdoor to indoor scenes represents a significant domain shift, we expect the predicted depth maps to not be exactly metric, but still consistent. Following the recent introduction of **ZeroDepth** [9], we have finally opted to incorporate this model into our approach. **ZeroDepth** is supposed to be scale-aware as well, benefiting not only from training on a huge dataset, but also from the advanced learning capabilities of Transformers.

Thus, as depicted in Figure 4.1, the first step is to predict a dense depth map using a DNN and scale it using SLAM triangulated points, as described in Section 4.1.1. In our experiments, we opted for **ORB-SLAM 3** as our monocular-inertial SLAM baseline. Given the high precision in localization of **ORB-SLAM 3**, we can assume a sufficient accuracy of the resulting sparse map, which we will validate in subsequent experiments. This section outlines a procedure for global scale factor recovery that can operate independently of ground truth data during navigation. Following the scale-correction process,

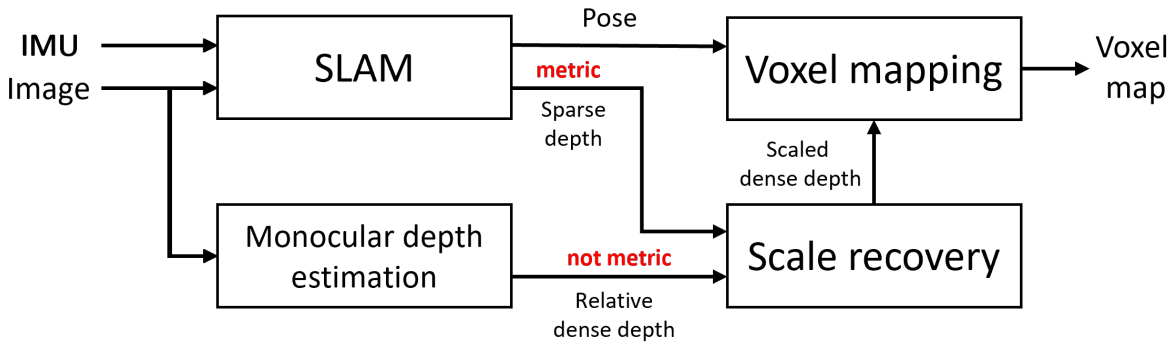


Figure 4.1 – Architecture of our proposed loosely coupled pipeline which recovers the scale of the predicted dense depth from the sparse depth estimated by SLAM. The final voxel map is built and maintained by multi-view fusion from the estimated camera pose and the scaled dense depth map.

we evaluate the scaled depth maps to assess the impact of our methodology. In the second stage, we aim to further improve the process through multi-view refinement using volumetric fusion, which is detailed in section 4.2.

4.1.1 Scale recovery

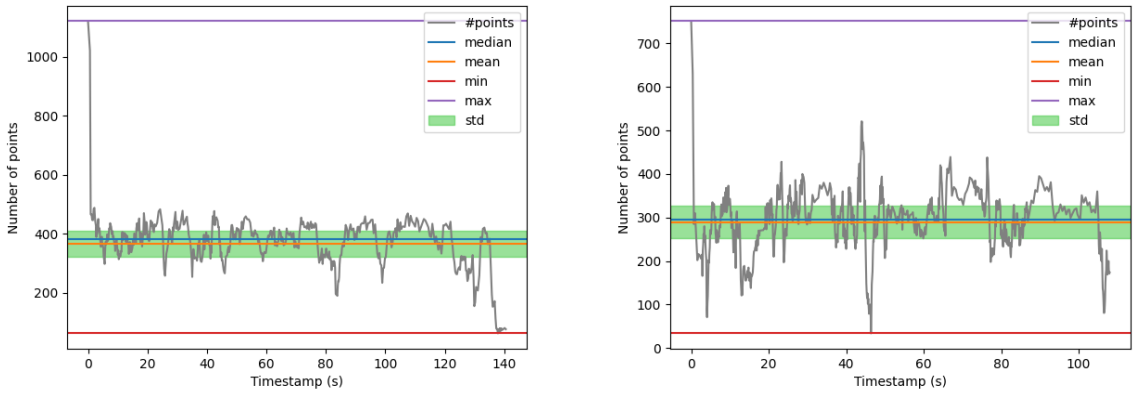
For a keyframe I_k we define $\Omega_k \subset I_k$ as the subset of pixels for which **ORB-SLAM 3** estimated the depth such that for $p \in \Omega_k$ the estimated depth is $D_p^k \in \mathbb{R}_+^*$. Likewise, the chosen Deep Learning model infers a dense depth map such that each pixel $p \in I_k$ has a predicted depth $\hat{D}_p^k$. Assuming that the predicted depth map is consistent, we obtain:

$$\exists \alpha_k \in \mathbb{R}, \forall p \in I_k \rightarrow Z_p^k = \alpha_k \hat{D}_p^k \tag{4.1}$$

where Z_p^k is the ground truth and α_k the scale factor for this keyframe. When evaluating depth prediction, this consistency assumption is widely assumed, and the typical approach is to use the ratio of the median predicted depth to the median of the ground truth in order to scale the estimated depth map [138], [139]. We will refer to this global scale factor as the **GT-scale** which is defined by the following expression:

$$\alpha_k = \frac{med(\{Z_p^k, p \in I_k\})}{med(\{\hat{D}_p^k, p \in I_k\})} \tag{4.2}$$

Using medians to estimate scale is less prone to outliers compared to using means. Nevertheless, this approach may become less reliable for smaller datasets, where variance can be high and outliers more impactful. In the context of depth estimation, ground truth data derived from sparse LiDAR point clouds typically contain a fair number of points, representing about 5% of the image density [168]. On the other hand, the number of points tracked by a system like **ORB-SLAM 3** is much lower, about 0.1% of the image density on **EuRoC**/*V101* and 0.02% on **HILTI**/*Basement_1*. The algorithm detects up to 1,200 points per frame, but the actual number of robustly mapped points per keyframe is much lower, as shown in Figure 4.2. On average, it represents only 366 mapped points per keyframe on the *V101* scene, and 289 on *Basement_1*. The higher number of points observed at the beginning of the sequences can be attributed to the initialization procedure of the system, which is more permissive and accepts more points during this phase [1]. The lower numbers at the end of the **EuRoC** sequences are associated with frames with uniform textures, which typically occur when the drone is landing.



(a) EuRoC: *V101* (b) HILTI: *Basement_1*

Figure 4.2 – Evolution of the landmarks density per keyframe running ORB-SLAM 3.

Therefore, to account for the fewer points and their potential variability, we suggest estimating the global scale factor by minimizing the square relative error, as outlined by Equation (2.24). To the best of our knowledge, this particular approach has not been previously explored. As mentioned in Section 2.5.2, this metric aims to counterbalance the disproportionate influence of distant points with large absolute errors, while still penalizing significant deviations robustly. Hence, when working with sparse depths from SLAM, we define the global scale factor as follows:

$$\hat{\alpha}_k = \min_{\alpha} \frac{1}{N} \sum_{p \in \Omega_k} \frac{\left\| \alpha \hat{D}_p^k - D_p^k \right\|^2}{D_p^k} \tag{4.3}$$

here, N represents the total number of points within the set Ω_k. For simplification in the minimization process, we omit the factor of $\frac{1}{N}$ as it does not affect the value of α that minimizes the function. Then, expanding the sum yields a quadratic polynomial, which is trivial to minimize:

$$\hat{\alpha}_k = \min_{\alpha} \alpha^2 \sum_{p \in \Omega_k} \frac{\hat{D}_p^{k2}}{D_p^k} - 2\alpha \sum_{p \in \Omega_k} \hat{D}_p^k + \sum_{p \in \Omega_k} D_p^k \tag{4.4}$$

In the following, we will refer to the resulting scale factor as the **SR-scale**. Although the solution to this equation is straightforward, we emphasize that the proposed procedure still relies on a strong assumption regarding the consistency of the predicted depth map.

4.1.2 Experimental procedure

All experiments were conducted on the same system described in Section 3.2.2, featuring an NVIDIA RTX 2080 mobile GPU with 8 GB of dedicated memory. In addition, we adapted **ORB-SLAM 3** to not only store the trajectory, but also to export each keyframe's associated landmarks, reprojection errors, and timestamps. For each keyframe, dense depth maps were predicted using the PyTorch implementation of **PackNet-Sfm** and **ZeroDepth** using the weights published by their authors. **PackNet-Sfm** was trained using velocity supervision on two autonomous driving datasets: **Cityscapes** [153] and **KITTI** [5]. On the other hand, **ZeroDepth** training involved a much larger set of data, including both outdoor environments such as **Tartan Air** [160], **Waymo** [158], **Parallel Domain** [9], and the **Large-Scale Driving** dataset [9], and indoor environments represented by the immense **OmniData** dataset [161]. Furthermore, we developed a script to construct ground-truth depth maps for each keyframe by projecting the scene LiDAR point cloud onto the image plane, using the ground-truth camera poses and the camera's intrinsic parameters.

For quantitative analysis, we exclusively focus on the *V101* scene from the **EuRoC** dataset. This choice is motivated by the limited availability of evaluation datasets that provide both trajectory and 3D structure ground truth. The chosen scene is comprehensive and has also been used in several related works to evaluate their performance. Additionally, we provide a qualitative evaluation of **PackNet-Sfm** and **ZeroDepth** networks on the *Basement_1* scene from the **HILTI** dataset. As mentioned earlier, this scene presents particular challenges due to lighting variations and the presence of long corridors which result in a greater depth of field.

Regarding the evaluation process, we use three types of depth maps: dense (DNN-predicted), semi-dense (ground truth), and sparse (SLAM-estimated). To ensure consistency despite density variations, a mask is applied to align the DNN and SLAM depth maps with the semi-dense ground truth, and values outside the specified range are filtered out. Values are capped at 30 m since the room size is approximately 8 m x 8.4 m [3] and the valid DNN predictions largely fall within this range. In this study, we evaluate the original SLAM sparse depth, the original DNN predictions, the predictions scaled to ground truth (GT-scale), and the predictions aligned with our proposed method (SR-scale). The evaluation metrics outlined in Section 2.5.2 are measured on every keyframe of the scene, and the mean values for each metric are reported. We contrast our results with other approaches that densify monocular SLAM with reported quantitative evaluations on the

V101 scene, specifically referencing **DeepFactors** [65], **TANDEM** [117], **CodeVIO** [7] and **CodeMapping** [6]. Unfortunately, although the authors reported standard metrics, they did not all use the same ones. Furthermore, we present a 3D visualization of the results via the camviz Python library [169]. This library allows us to display the evaluated depth maps as 3D point clouds for each keyframe by projecting them using camera intrinsic parameters.

4.1.3 Quantitative results

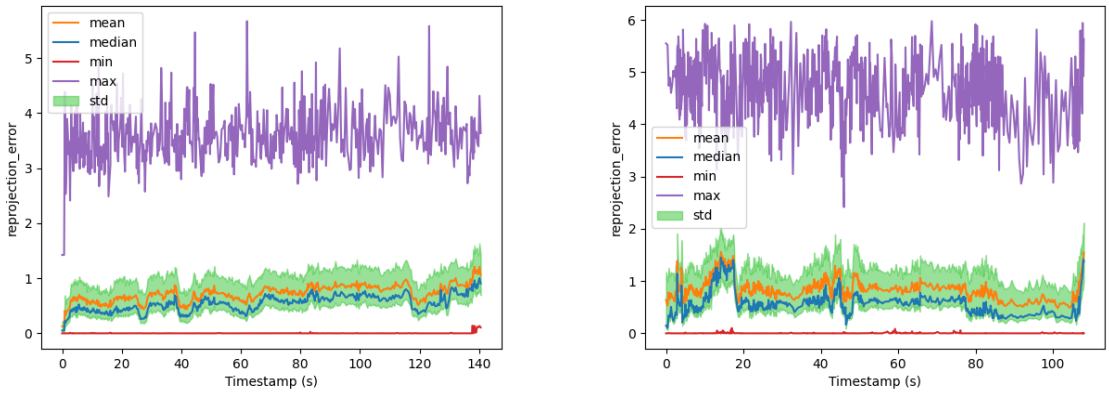
Our experimental evaluation results are summarized in Table 4.1, which presents the metrics we measured alongside those from related works. The table is organized in three sections. The first section evaluates the sparse map produced by **ORB-SLAM 3**. The second section shows the results of other methods, and the third one presents the results of the DNN-generated depth maps without scaling, with median-scaling (GT-scale), and rescaled with our proposed method (SR-scale).

	abs_diff	abs_rel	sq_rel	rmse	rmse_log	δ_1	δ_2	δ_3
ORB-SLAM 3 [1] (sparse)	0.284	0.156	0.266	0.572	0.222	89.8%	94.6%	96.5%
DeepFactors [65] (GT-scale)	0.842			1.050				
TANDEM [117] (GT-scale)						**94.25%**		
CodeVIO [7]				0.468		87.0%	**95.2%**	**97.9%**
CodeMapping [6]	**0.192**			**0.381**				
PackNet-Sfm	6.309	2.720	22.945	7.267	1.258	1.3%	4.4%	12.0%
PackNet-Sfm (GT-scale)	0.807	0.331	0.530	1.145	0.396	48.2%	76.0%	89.6%
PackNet-Sfm (SR-scale)	0.792	0.318	0.443	1.063	0.418	43.2%	72.7%	87.9%
ZeroDepth	0.791	0.285	0.326	0.953	0.392	38.8%	70.6%	88.8%
ZeroDepth (GT-scale)	0.386	0.167	0.152	0.545	**0.217**	81.3%	92.5%	96.6%
ZeroDepth (SR-scale)	0.412	**0.167**	**0.145**	0.569	0.222	77.0%	92.5%	96.6%

Table 4.1 – Evaluation of depth estimation on the EuRoC *V101* scene, with units in meters except for the δ_i metrics. The best value for each metric is highlighted in bold for dense evaluations, excluding the first row.

Our analysis begins with the sparse depth estimation provided by **ORB-SLAM 3**. Due to the inherent sparsity of the SLAM-generated maps, the evaluation is based on significantly fewer points compared to dense depth maps, making the measurements more susceptible to the influence of errors or outliers. However, as shown in Figure 4.3, the mean reprojection error remains below 1 pixel despite a few peaks up to about 5 pixels. This demonstrates a great level of precision in estimating depth, as evidenced by the high percentage of valid points ($\delta_1 = 89.8\%$) and the low errors reported in Table 4.1.

These observations affirm our belief that the sparse depth estimated by **ORB-SLAM 3** provides a reliable source for scale recovery.



(a) EuRoC: *V101* (b) HILTI: *Basement_1*

Figure 4.3 – Evolution of the reprojection error per keyframe running ORB-SLAM 3.

The second group of Table 4.1 presents results related to different metrics, such as abs_diff, $rmse$, δ_1, δ_2, and δ_3. According to the values published in [6], [117], even with multi-view optimization and scaling to ground truth, **DeepFactors** shows limitations on new data. In contrast, **TANDEM**, which was also trained on indoor datasets, shows outstanding performance after scale alignment. Considering metric methods that do not depend on ground truth scaling, **CodeVIO** and **CodeMapping** emerge as top performers in terms of $rmse$. Both techniques are based on CVAE (presented in Section 2.3) which allows to integrate dense depth into the multi-view optimization framework by exploiting the compact code representation. Furthermore, **CodeMapping** improves the performance even further by exploiting the reprojection error.

In the final group, we present the evaluation of **PackNet-Sfm** and **ZeroDepth**, both unscaled and scaled with ground truth as well as with our proposed method. We note that there is no multi-view processing, and each depth map is predicted solely from a single image. Our findings reported in Table 4.1 indicate that **PackNet-Sfm** faces difficulty in absolute scale estimation, with a significant drop in $rmse$ from 7.27 m to 1.15 m following scale adjustment, but the result remains suboptimal. Yet, it is worth noting that our approach successfully recovers the scale factor using the sparse depth data from **ORB-SLAM 3**. The results reported in the table sometimes surpass those using GT-scale for the first metrics. We attribute this to our method considering fewer points and offering

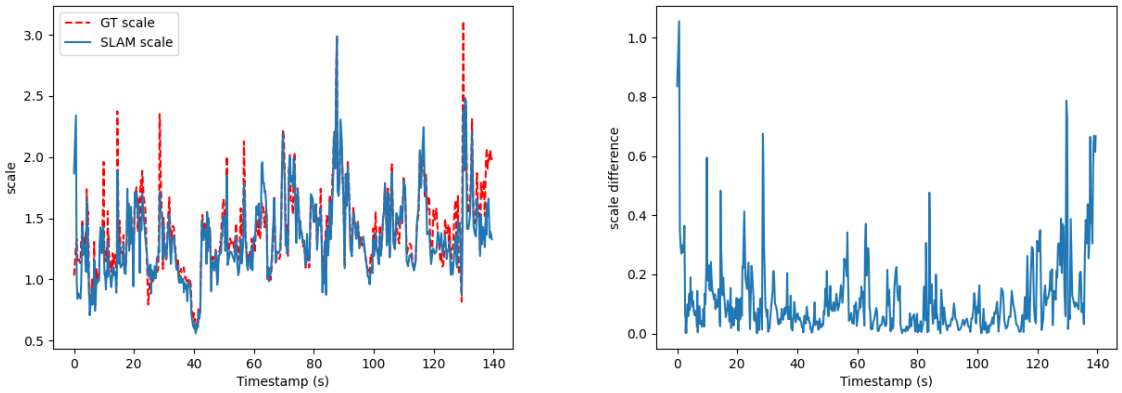


Figure 4.4 – Left: A plot of the global scale factors estimated for ZeroDepth [9] predictions on the *V101* scene of the EuRoC dataset, comparing the use of ground truth (GT) scaling with our SLAM-based approach. Right: A plot of the absolute difference between the two scale estimates over the sequence.

a finer approach than the median technique, as described in section 4.1.1. As a result, in some cases, our method is less affected by predicted depth outliers.

In contrast, **ZeroDepth** performs very well thanks to its training dataset, which includes a significant number of indoor samples. Throughout the entire sequence, it achieves an average *rmse* of less than 1 meter. After rescaling, the error decreases to 0.545 m with the GT-scale and 0.569 m with the SR-scale. Figure 4.4 presents the GT-scale and SR-scale plots for **ZeroDepth**, along with their absolute differences. A mean absolute difference of 0.118 demonstrates that our method yields a reasonably accurate estimation of the global scale factor, even if it relies on much fewer reference points. Another notable observation is the inconsistency of the predicted scale over time. The scales estimated show significant variability, even between consecutive frames, as shown in the figure. During our experiments, this phenomenon occurs more frequently on frames with mainly uniform textures. Similar observations have been made using **PackNet-Sfm**, and the corresponding results are reported in Appendix B.2.

As introduced in Section 2.4, **ZeroDepth** uses a learned variational distribution to sample depth maps from a latent space constructed from image and geometric embeddings. In practice, 10 samples are generated, from which the final depth map is derived using the mean, and the uncertainty is obtained from the standard deviation. The experiments conducted by the authors on the **KITTI** dataset indicate that this sampling number is reasonable to efficiently exploit the resulting uncertainty and filter out low-

	PackNet-Sfm	ZD01	ZD02	ZD03	ZD04	ZD05	ZD06	ZD07	ZD08	ZD09	ZD10
Inference (ms)	9.2	13.9	19.7	22.5	24.8	29.1	144.6	456.0	804.9	1152.9	1501.1

Table 4.2 – Average inference time in milliseconds for PackNet-Sfm and ZeroDepth (ZD) on the *V101* scene of the EuRoC dataset, measured using 1 to 10 samples.

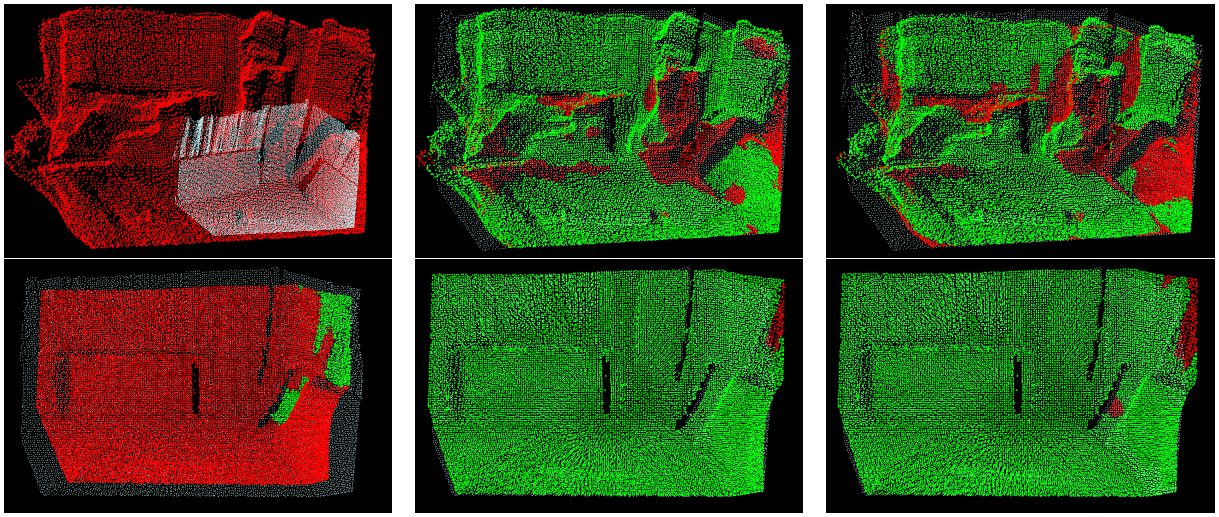
confidence depth values. Nonetheless, the repetitive sampling process significantly affects speed performance, as shown in Table 4.2. We conducted our own experiments on **EuRoC**, testing the model with 1 to 10 samples over 10 iterations each. This comparison revealed only negligible differences in both mean uncertainty and accuracy over the entire sequence. Therefore, a smaller number of samples can be considered, especially for use on embedding systems. This observation is specific to the *V101* sequence and more exhaustive experiments would be required on other sequences or datasets to be conclusive.

4.1.4 Qualitative analysis

We will now present and analyze our qualitative results. Figure 4.5 illustrates the evaluation of our scale recovery procedure in 3D. This visualization effectively highlights the initial incorrect prediction of scale by the deep learning models and demonstrates the effectiveness of our scaling approach with respect to the δ_1 metric. Through the use of the δ_1 accuracy rate metric (cf. Equation 2.27), valid points are shown in green, incorrect points in red, and the reference point cloud is shown in white to provide the context. For detailed views of the corresponding color images and depth maps, refer to Figure 4.6 for **ZeroDepth** and Figure B.27 for **PackNet-Sfm**.

In addition to evaluating the scale recovery, we analyze the overall depth estimation quality. We note that **PackNet-Sfm** fails to accurately infer the depth of some structures and the predictions happen to be very noisy on some edges. However, as we can see from the Figures reported in the Appendix B.2, the model manages to correctly distinguish most of the surfaces in the scene. We also note that the network fails to predict some planar surfaces or predicts them as if they were outdoors, with larger depths in the upper part of the image. This indicates that the predicted scale does not appear to be consistent within the image. In contrast, **ZeroDepth** predicts much higher quality depth maps, as shown in Figure 4.6. The predictions are notably smoother and exhibit less noise at surface discontinuities.

Applying these models to the *Basement_1* scene of the **HILTI** dataset (as seen in Figure 4.7) presents challenges due to the scene's dim lighting and the extended depth of field



original GT-scale SR-scale

Figure 4.5 – 3D visualization of the evaluation of PackNet-Sfm and ZeroDepth predictions using the δ_1 metric on the *V101* scene of EuRoC. The depth maps, both with and without scaling, are projected into a 3D point cloud. The ground truth points are shown in white for reference, the correct predictions in green, and the incorrect ones in red.

in its long corridors. Surprisingly, **PackNet-Sfm** manages to yield reasonable predictions in these long corridors, even under darker conditions. However, as highlighted in the side view shown in the figure, the predictions are significantly poorer in the upper part of the image. This limitation is likely because **PackNet-Sfm** was trained on datasets focused on autonomous driving, where such image regions typically depict the sky, thus leading to its unfamiliarity with indoor scene characteristics. The predictions of **ZeroDepth**, although less planar than those of **PackNet-Sfm**, maintain a notable level of smoothness across surfaces. Nonetheless, in the darkest images, the network often predicts overly flat depth maps, failing to accurately capture the extended depth characteristic of the corridor. We include supplementary qualitative results in the Appendix B.2.

Since both models were trained on color images, we tested them using a sequence captured in the corridors of our laboratory. **PackNet-Sfm** exhibited poor performance in these tests, while **ZeroDepth** demonstrated significantly better results, as shown in Figure 4.7. With the use of color images, the planarity of the wall and floor surfaces was correctly represented in the **ZeroDepth** predictions. However, it is worth noting that the scene captured in our laboratory was significantly better illuminated than the *Basement_1* scene. Additionally, when surfaces at the end of the corridor are too distant, the network tends to incorrectly predict them as being closer than they actually are.

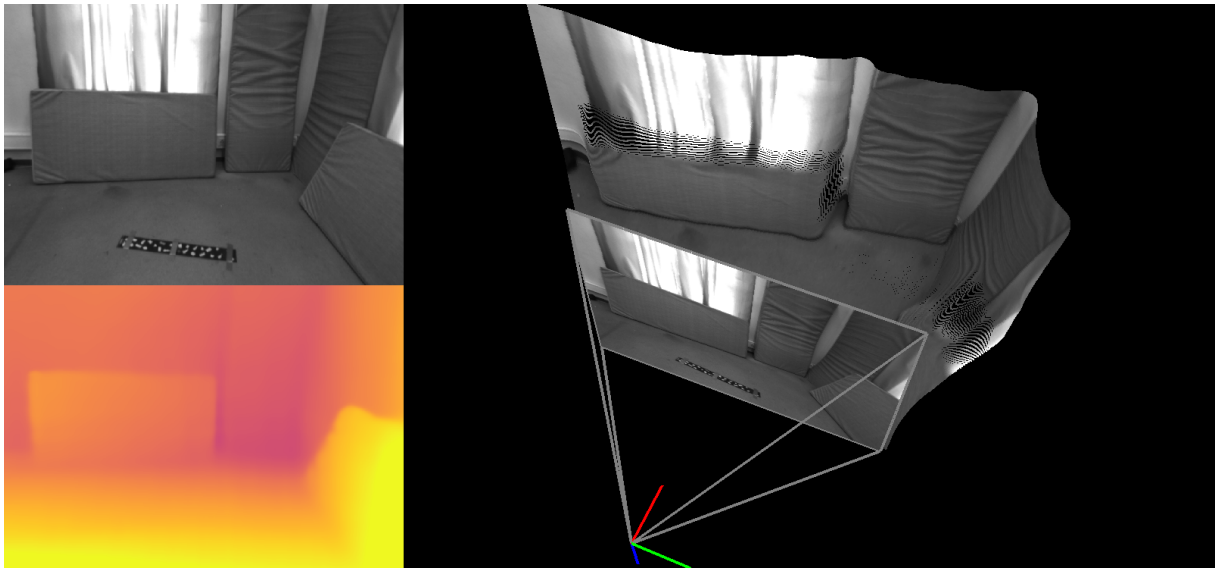
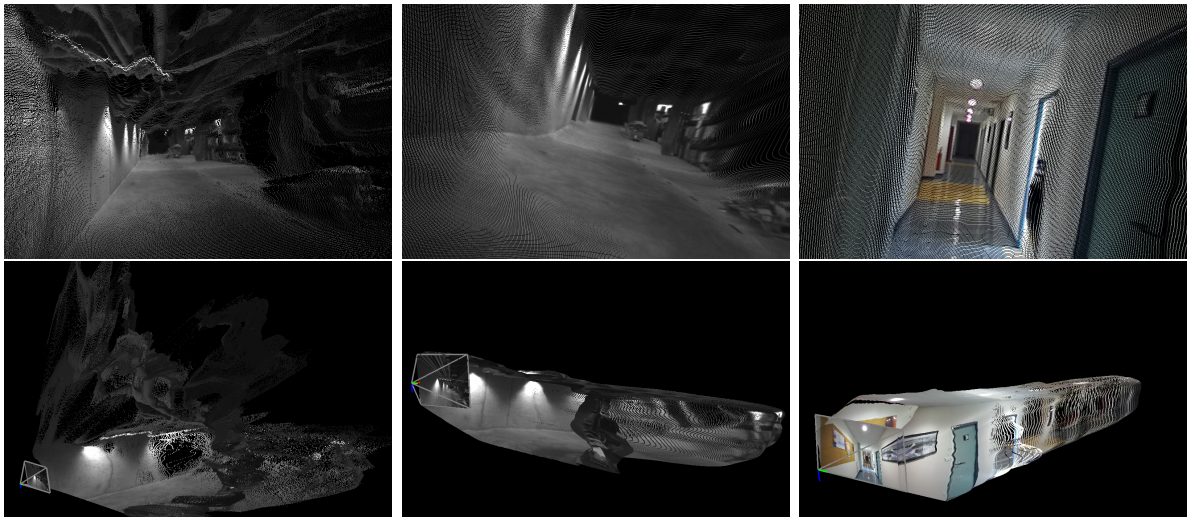


Figure 4.6 – Visualization of ZeroDepth predictions on the scene *V101* of EuRoC dataset.



(a) PackNet-Sfm (b) ZeroDepth (c) ZeroDepth

Figure 4.7 – Visualization of depth prediction from *Basement_1* scene of HILTI dataset and a sequence that we collected in IMT Atlantique school (last column). The first line shows a frontal view, and the second one is a side view.

4.1.5 Discussion

In this section, we have presented our scale recovery procedure, which leverages the sparse depth estimated by **ORB-SLAM 3** to adjust the scale of dense depth maps predicted by deep learning models, specifically **PackNet-Sfm** and **ZeroDepth**. Our method has demonstrated its effectiveness in helping to provide metrically accurate dense depth maps through evaluation on the established **EuRoC** dataset. The experiments further supported that these DNNs have promising generalization capabilities.

However, our approach heavily depends on the accuracy of the chosen model's depth predictions. Although the qualitative evaluation in **EuRoC** yielded satisfactory results, the dataset mainly comprises confined room settings. To assess the models under different conditions, we additionally applied them in long corridors with different lighting scenarios. These experiments uncovered certain constraints for depth prediction, especially in such specific environments.

Given that **PackNet-Sfm** was trained only on outdoor color images, we first considered retraining or fine-tuning the network. The usual process in depth estimation for self-supervised learning using monocular sequences involves electing successive frames with sufficient optical flow to ensure parallax [129], [139]. Despite applying this method with multiple datasets, our attempts to train the network were unsuccessful. Still, the original authors reported successful training on EuRoC, but neither the weights nor the scores for that specific training were made available.

ZeroDepth performed reasonably well on the grayscale images of **EuRoC**, but its effectiveness seems to decrease in darker scenes without color. Furthermore, it may not have encountered comparable scenarios with deep hallways during training. Unfortunately, we have not attempted to fine-tune **ZeroDepth** for these specific scenarios due to the extensive computational resources required.

Even though these results are specific to single-view depth estimation, they lay the groundwork for the next stage of our research. In the following section, we build on this procedure for generating dense and metric depth maps and introduce voxel mapping as a means to achieve efficient multi-view volumetric fusion.

4.2 Voxel map construction

This section delves into the voxel mapping procedure, an integral part of the pipeline previously presented in Figure 4.1. Notably, our research focuses on creating a metric occupancy grid for navigation purposes, rather than a high-precision 3D model. Here, we will show how voxel mapping effectively serves this goal, providing an optimal approach for both building and maintaining the 3D map, and facilitating multi-view fusion refinement.

4.2.1 Multi-view volumetric refinement

The use of multi-view techniques is essential in Visual SLAM, as previously explained in Section 2.1.3 with Bundle Adjustment. This process jointly optimizes camera poses and 3D landmarks across multiple frames observing the same scene. Unlike MDE, which relies solely on a single input image, the multi-view approaches enforce both temporal and geometric consistency.

Consequently, other works aimed at densifying SLAM have also leveraged multi-view schemes to improve depth estimation. In particular, this has been achieved by integrating depth code representations into a factor graph [6], [7], [65], employing multi-view stereo networks [117], or using recurrent neural networks [114], [116]. Although some of these methods have shown high accuracy, as mentioned in the previous section, they often require significant computational resources because they involve the joint optimization of a problem with a greater number of variables.

Applying multi-view to the mapping task by fusing depth maps is another common strategy in 3D reconstruction. More recently, the integration of NeRF into SLAM has been explored [114], [170], [171], but this approach is computationally intensive as it requires learning a NeRF model from SLAM outputs in real-time. On the other hand, the conventional solution involves volumetric fusion of depth maps obtained from either RGB-D sensors or stereo matching [12], [39], [56], [172], [173]. Here, fusion requires the weighing of depth maps based on the corresponding depth sensor model.

In our work, we use a fast and iterative volumetric method derived from **Voxblox** [39], following the same approach as **Kimera** [12]. Instead of relying on stereo for dense depth estimation, we follow our previously presented pipeline for predicting metric and dense depth maps. Moreover, unlike **Kimera** which proceeds to construct a 3D model via marching cubes, we stop our process at an earlier stage and focus solely on the voxel level. We elaborate on this approach in the remainder of this section.

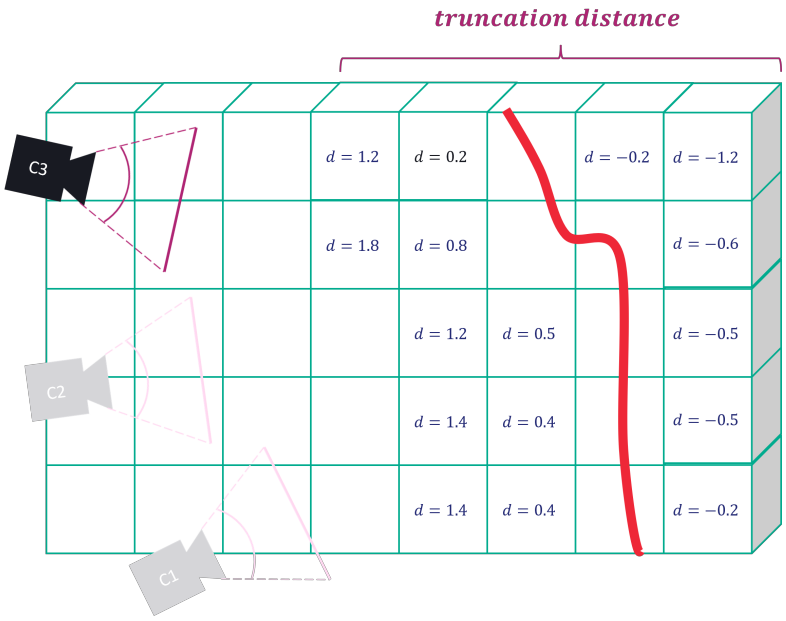


Figure 4.8 – Schema illustrating the construction of a voxel map from multiple viewpoints using a TSDF to update voxels' distance from the nearest surface.

Voxblox [39] uses dense depth maps to construct and maintain a voxel map through raycasting. At its core, **Voxblox** relies on the TSDF to represent 3D space. In this representation, each voxel stores a weight and a distance to the nearest surface, as shown in Figure 4.8. During the process of raycasting, a ray is projected for each pixel from the optical center of the camera to the corresponding point projected in 3D space, updating each voxel it intersects along the way. This update is based on the Signed Distance Function, which calculates the distance from the voxel's center to the target point, being positive outside the object and negative inside. The application of the TSDF improves efficiency by limiting updates to voxels within a predefined truncation distance from the surface.

The voxel updating process involves a weighing function that balances new ray data with existing voxel information. **Voxblox** adopts a weighing strategy that is inspired by RGB-D sensor models, assigning a fixed weight of 1 in front of the surface and decreasing quadratically with distance behind the surface. Additionally, to optimize the fusion process, **Voxblox** introduces bundled raycasting. This technique aggregates multiple pixels that terminate in the same voxel into a single ray, which significantly enhances performance with minimal loss in accuracy. This approach enables fast volumetric fusion by iteratively building and updating the voxel map from multiple viewpoints.

In our proposed pipeline, **Voxblox** receives at each keyframe the rescaled dense depth map computed via our approach and the corresponding camera pose estimated by **ORB-SLAM 3**. This integration enables the iterative construction and refinement of the voxel map by bundled raycasting from successive viewpoints.

4.2.2 Experimental results

To test this method, we have integrated **Voxblox** into our current framework. **Voxblox** was incorporated through its implementation [174] in ROS. As of now, while a few methods have been proposed for evaluating voxel maps [41], [175], to the best of our knowledge, while few methods were proposed to evaluate voxel maps [41], [175], there is no standardized quantitative metric for comparing voxel maps that has been applied to dense monocular SLAM or MDE. Nevertheless, we provide experimental results of voxel mapping applied to **EuRoC** and **HILTI** datasets. For visualization, we use RViz, a ROS 3D visualization tool, to display both the generated voxel map and the 3D mesh rendered via marching cubes. To facilitate qualitative comparison, we add a reference point cloud, displayed in white or red. It is generated using **Voxblox** by raycasting the ground truth depth maps and displaying the centers of the occupied voxels, i.e. those that lie on a surface. Additionally, for **EuRoC**, we run the example provided in **Voxblox**, which uses the ground truth trajectory and stereo inputs to estimate dense depth via stereo matching [176]. We will refer to it as the stereo-based reference.

Fine-grained voxel map reconstruction: Figure 4.9 presents the results of voxel mapping for scene *V101* of the EuRoC dataset, using a voxel size of 100 mm.

The map was constructed using the estimated trajectory from **ORB-SLAM 3** and the dense depth maps predicted by **ZeroDepth**, scaled using our method. The ground truth point cloud in red was aligned with the estimated trajectory. Assuming perfect alignment and reconstruction, all ground truth points would fall within surface voxels. Nonetheless, given the ATE of 49 mm measured for this sequence, a slight misalignment may exist. The map generated by **Voxblox** using stereo input is displayed in the second row. It is worth noting that the stereo-based approach processes all frames of the sequence, while our method only processes the keyframes selected by the SLAM algorithm.

Overall, our approach produces a decent reconstruction of the room, capturing most of the major structural elements that are important for drone navigation and obstacle avoidance. The majority of voxels are well aligned with the reference point cloud, indicating a good fit. However, we observe some shortcomings in the mapping of thinner objects.

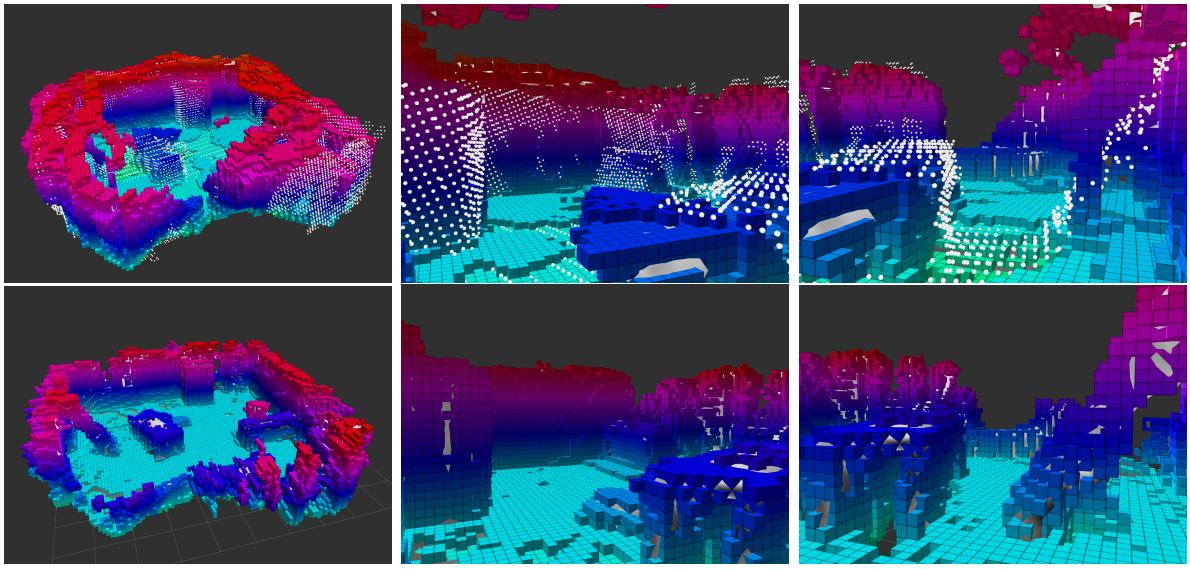


Figure 4.9 – Visualization of voxel mapping results in EuRoC, using Voxblox with a voxel size of 100 mm. The first row presents our results, while the second row shows those of the stereo-based reference.

As observed in the top-right image in Figure 4.9, a ladder against the wall is inaccurately mapped, with only part of it being reconstructed. A more detailed view of the reconstructed mesh and depth prediction for this area is illustrated in Figure 4.10. Upon closer examination of the corresponding depth predictions, it is apparent that there is considerable noise surrounding thin objects. Additionally, this region of the scene received limited coverage in the sequence, restricting the potential for correction from other viewpoints.

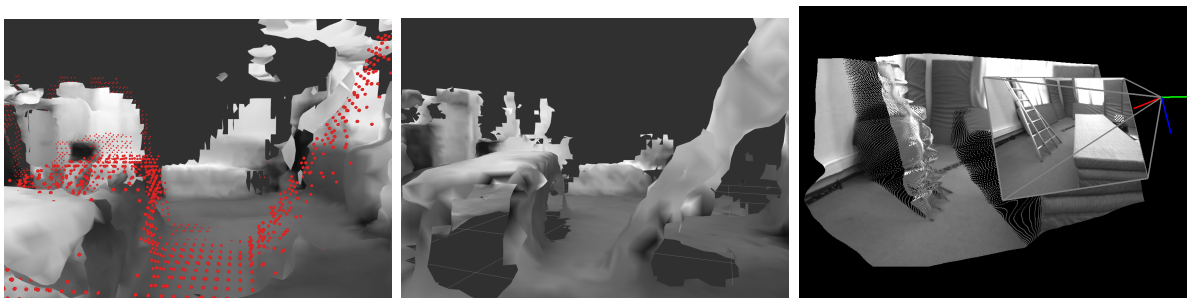


Figure 4.10 – 3D meshes generated via marching cubes from the voxel maps: our approach (left) and stereo-based method (middle). The right image displays the original depth prediction by ZeroDepth in this area. Notably, the point cloud of the mattress at the bottom left of our map, covered by only a single image, was not mapped.

Coarse-grained voxel map reconstruction: Our experiment using a 200 mm voxel size produced the map featured in Figure 4.11. When we increased the voxel size, the resulting map became coarser but smoother, mitigating some of the noise we noticed in the input depth maps. Moreover, increasing the voxel size yields also results in a more restrictive navigable space that maintains a greater distance between the drone and the actual surfaces.

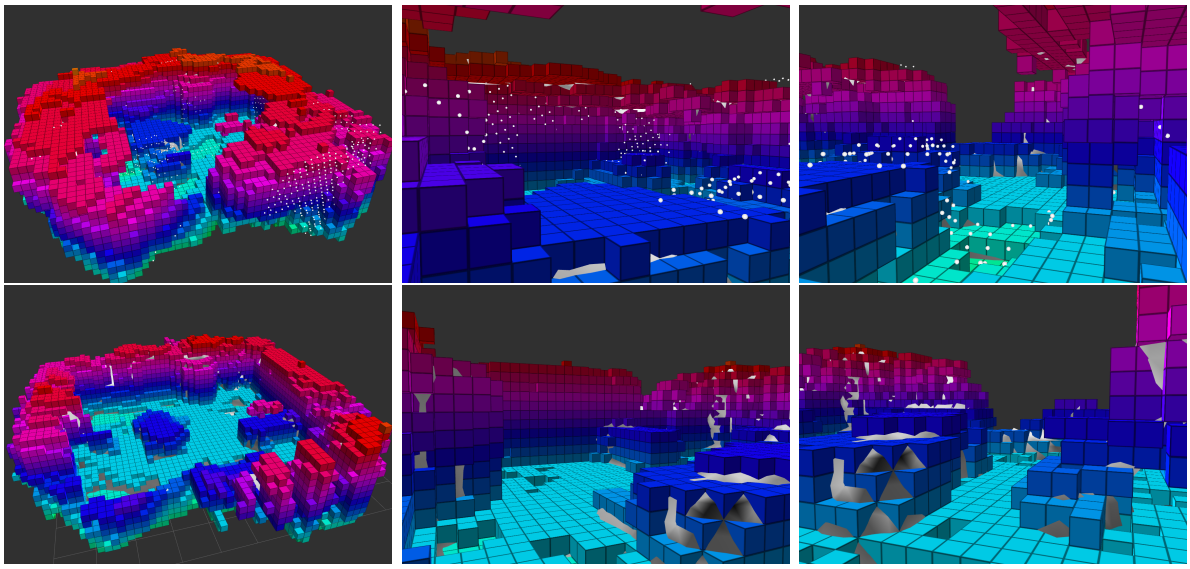
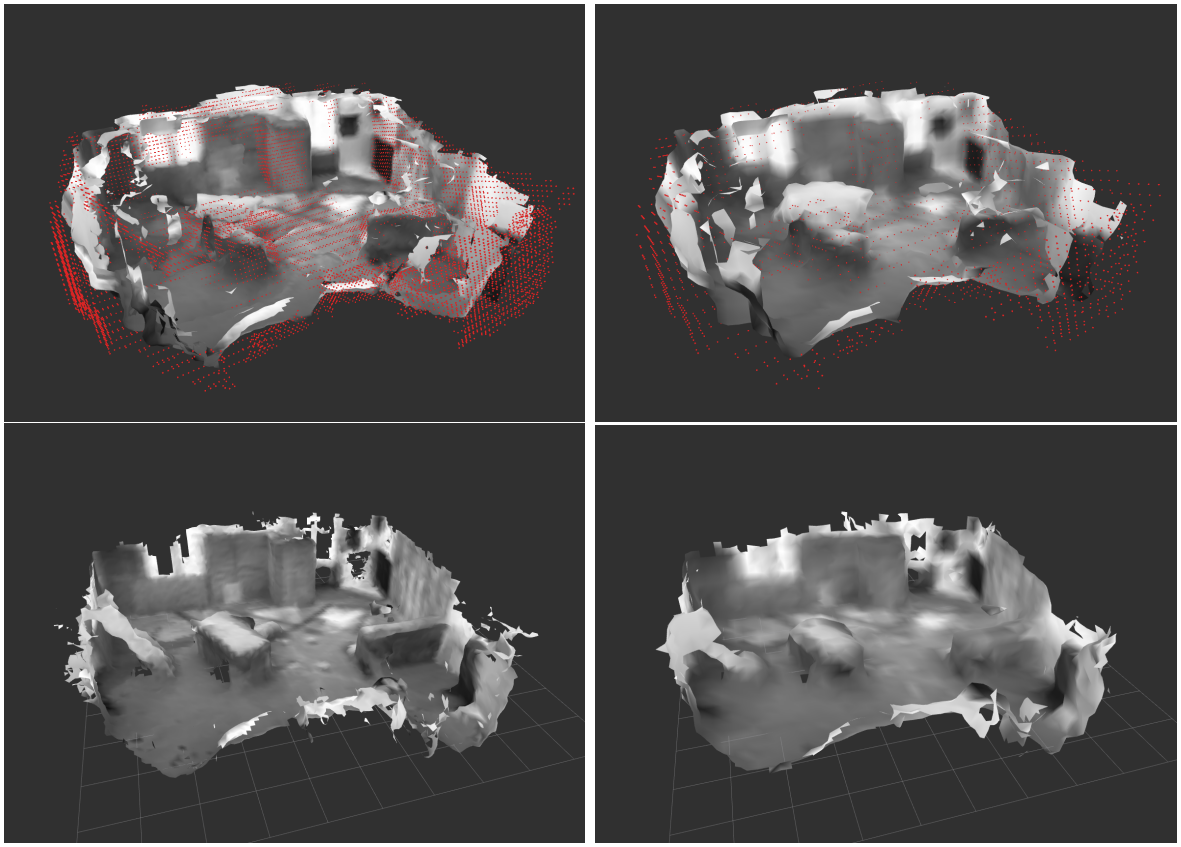


Figure 4.11 – Visualization of voxel mapping results using Voxblox in EuRoC, with a voxel size of 200 mm. The first row presents our results, while the second row shows those of the stereo-based example.

For completeness, we further present the corresponding textured 3D meshes in Figure 4.12 for a more detailed visualization of the captured structures. In general, the stereo-based method produces a more detailed reconstruction. Nevertheless, our approach yields reasonable performance considering that it processes fewer images (529 keyframes against 1,151 frames). Interestingly, our approach provides more detail in the upper parts of the images, especially around the windows, where stereo matching struggles due to the lack of texture.

Afterward, we tested our pipeline on the *Basement_1* scene from the **HILTI** dataset, choosing **PackNet-Sfm** for depth estimation since **ZeroDepth** tended to predict flat depth maps, which significantly affected voxel mapping. Further details on the suboptimal results of **ZeroDepth** are provided in Appendix B.2. The results of the experiment using **PackNet-Sfm** are thus shown in Figure 4.13, depicting voxel maps and corresponding 3D

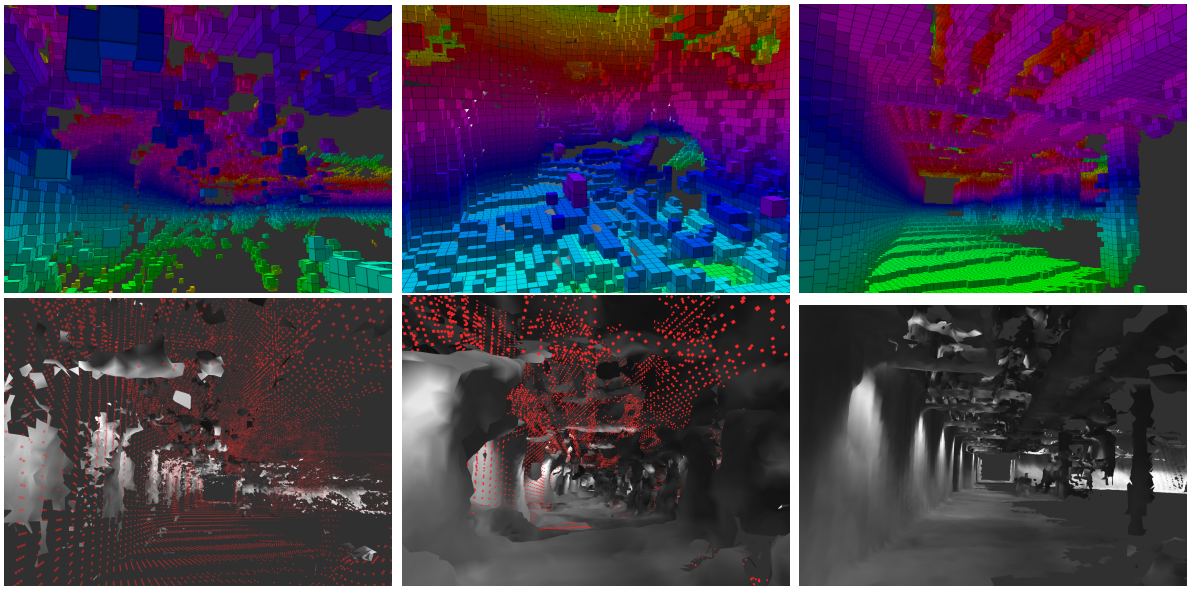


(a) voxel size = 100 mm (b) voxel size = 200 mm

Figure 4.12 – 3D mesh reconstructions from voxel maps: the first row displays results using our approach, while the second row features the stereo-based method.

meshes. As a reference, we projected LiDAR measurements using the trajectory estimated by **ORB-SLAM 3**, since dense ground truth trajectory data was not available.

The analysis of the results indicates that the sparse depth information from **ORB-SLAM 3** effectively outlines the geometric structure of the scene, but really lacks density. Furthermore, within empty spaces in the voxel map, we observed outliers, which likely arose from the strong illumination changes typical of this scene. As for the densified map enhanced by multi-view fusion, it does not sufficiently correct for structural inaccuracies. Although the extended hallway is somewhat noticeable, the resulting map is too coarse and irregular, making it unsuitable for navigation purposes.



(a) ORB-SLAM 3 (b) our (c) LiDAR

Figure 4.13 – Multi-view volumetric fusion results using Voxblox in HILTI dataset: the first row presents voxel maps, and the second row depicts 3D meshes reconstructed via marching cubes. For reference, the LiDAR-measured point cloud is shown in red.

In conclusion, our experiments with voxel mapping using **Voxblox** for volumetric fusion have provided valuable insights. Our approach demonstrated the potential for creating coherent and geometrically consistent voxel maps for drone navigation, in particular within the *V101* scene of the **EuRoC** dataset. While the sparse depth information from **ORB-SLAM 3** successfully captured the primary geometric structures, its density limitation became apparent. Furthermore, although our densification method improved the completeness of the map, the multi-view fusion scheme could not compensate for the structural inaccuracies in the initial DNN depth predictions, resulting in maps unsuitable for precise navigation. These findings highlight the importance of the initial depth map quality in voxel-based mapping, and point to the inherent trade-offs when choosing a lightweight volumetric fusion approach.

4.3 Conclusion

In this chapter, we have successfully implemented and demonstrated our proposed framework which effectively densifies monocular-inertial SLAM for drone navigation. The framework relies on the pose estimation and sparse depth from SLAM, and requires reliable initial monocular depth predictions, but operates without direct depth measurements.

We introduced a loosely coupled approach that efficiently retrieves the metric scale of DNN dense depth predictions using sparse SLAM depths. This method differs from most related works in dense monocular SLAM that generally depend on ground truth measurements for scaling or use tightly coupled approaches with inertial measurements to determine the metric scale. Our approach, which relies on a single image input, has demonstrated fair results in comparison to state-of-the-art performances, while most of these methods incorporate more complex multi-view schemes. The experiments we have conducted validate the use of SLAM sparse depth as a reliable source for our method and confirm successful scale recovery across different deep learning models.

Expanding on this foundation, we have integrated a voxel mapping solution into our framework. This system constructs and updates a voxel map using SLAM-estimated camera poses and corresponding scaled dense depth maps. In contrast to the typical approach of using multi-view schemes for depth map estimation, we instead apply multi-view fusion for mapping which merges depth maps from different viewpoints. This decision aligns with our objective of developing a lightweight solution optimized for navigation applications, as opposed to more complex but highly accurate alternatives. The selected approach involves the volumetric fusion of dense depth maps through bundled raycasting, which iteratively refines the occupancy grid through a TSDF. The presented qualitative analysis illustrates the ability of our system to map essential structural elements of an environment, allowing for obstacle avoidance, provided that the initial depth predictions are sufficiently accurate.

Chapter 5

Conclusion

5.1 Synthesis of research contributions

The work presented in this thesis falls within the scope of research aimed at densifying monocular SLAM in the context of 3D mapping for autonomous drone navigation. We have outlined the significant potential of autonomous drones for first responders in scenarios like USAR to carry out reconnaissance missions and 3D mapping. This thesis addresses key scientific challenges associated with the use of small drones in indoor environments, equipped with a payload limited to passive sensors: a monocular visible camera and an IMU. In this context, real-time dense and metric 3D mapping is required for drone navigation, with SLAM emerging as the most viable solution.

How can we perform monocular SLAM densification with metric scale in real-time on an embedded system ?
As discussed earlier, achieving real-time dense 3D mapping via monocular SLAM poses significant challenges due to depth ambiguity and the inherent complexity of conventional SLAM approaches. Within the domain of monocular SLAM densification, a few approaches have recently emerged that leverage deep learning for depth estimation and achieve detailed 3D reconstruction. We proposed a loosely coupled framework combining sparse SLAM and MDE designed for dense and metric voxel mapping for drone navigation. The architectural decisions of the system are strategically targeted for future real-time applications on embedded systems at reasonable navigation speeds. Given that the majority of conventional SLAM algorithms run primarily on the CPU, our decoupled approach to densification exploits the GPU to minimize the impact on the original frame rate. Additionally, while other studies typically adopt a multi-view scheme for depth estimation, which tends to increase complexity, we instead apply the multi-view strategy in the voxel mapping process. The chosen lightweight mapping solution iteratively builds a voxel map via raycasting by performing a volumetric fusion of the provided depth maps.

To the best of our knowledge, no similar framework exists for comparison, which leads us to implement the entire pipeline and evaluate its individual components against related works.

To what extent can state-of-the-art visual SLAM be applied to drone navigation in challenging conditions?
The proposed framework relies on a real-time monocular-inertial SLAM baseline that estimates camera pose and sparse depth, a domain that is relatively mature and extensively evaluated. We conducted additional benchmarks in more challenging drone-specific scenarios, typically reserved for VIO or LiDAR-based SLAM, to evaluate several algorithms in various indoor environments featuring larger scales, high-speed motion, or varying lighting conditions. These experiments led us to adopt **ORB-SLAM 3** as our baseline, confirming its robustness in challenging environments while also revealing limitations in its IMU initialization process.

How can we combine SLAM sparse and metric depth to recover the absolute scale of a predicted dense depth map?
A key feature of our framework is a decoupled scale recovery procedure that uses the metric sparse depth from SLAM to correct the scale of DNN-predicted dense depth maps. This differs from common monocular dense SLAM methods that either rely on ground truth depth for scaling or incorporate dense depth into their SLAM optimization. In contrast, our proposed method restores the absolute scale of a depth map that is predicted from a single input image using the corresponding SLAM estimated sparse depth. We applied this method using state-of-the-art MDE models, namely **PackNet-Sfm** and **ZeroDepth**, and successfully demonstrated effective scale recovery, yielding comparable results with related works. Complementing this work with the voxel mapping stage, we also demonstrated through a qualitative evaluation that the proposed system is able to provide a coarse map of the environment adapted for drone navigation given good initial dense depth predictions.

5.2 Future directions and research perspectives

This section discusses potential avenues for advancing the research initiated in this study. We propose investigating several essential areas, such as extending the benchmarking of algorithms, studying a tightly coordinated approach for MDE, improving the voxel mapping process, and strengthening the robustness of SLAM localization through the use of densified maps. This section will review these potential research directions, highlighting their expected impact and the methods that could be used to achieve their benefits.

Training and validation datasets

In this study, our experiments to evaluate our proposed workflow were constrained by the limited availability of suitable datasets. To conduct a comprehensive monocular-inertial SLAM evaluation, we require datasets that provide synchronized visual-inertial data, calibration parameters, precise trajectory ground truth, and more particularly structural ground truth. As mentioned in section 2.6, published indoor datasets rarely provide both pose and map ground truths.

To overcome this limitation, we can consider several approaches. The most direct method is to collect our own data, but it requires significant resources and a very rigorous setup. Alternatively, we can use existing datasets that combine visual-inertial data with precise depth measurements [11], [154], [162] to generate a pseudo ground truth of the environment. Given the 6D pose trajectory and accurate LiDAR measurements, it would be possible to reconstruct a global point cloud of the map. In the absence of a ground truth trajectory, it may be worthwhile to investigate the use of state-of-the-art LiDAR-based SLAM solutions [53]–[55] for this task. Another interesting approach is the use of realistic 3D simulations, as demonstrated by the TartanAir dataset [160], which was created using the AirSim simulator [163] together with Gazebo for physics simulation and Unreal Engine for graphics rendering. We have initiated similar simulations controlling a drone in a virtual environment as shown in Figure 5.1. However, with the discontinuation of AirSim development, certain aspects such as accurate data synchronization and realistic sensor modeling (camera distortion and IMU noise) need further improvements.

During this research, we observed a lack of standardized metrics for evaluating the 3D mapping of dense monocular SLAM. As discussed in Section 2.5.2, a common way to evaluate the generated maps is to measure the error of the corresponding depth maps. Nevertheless, according to a recent study [177], these 2D-based distances do not capture

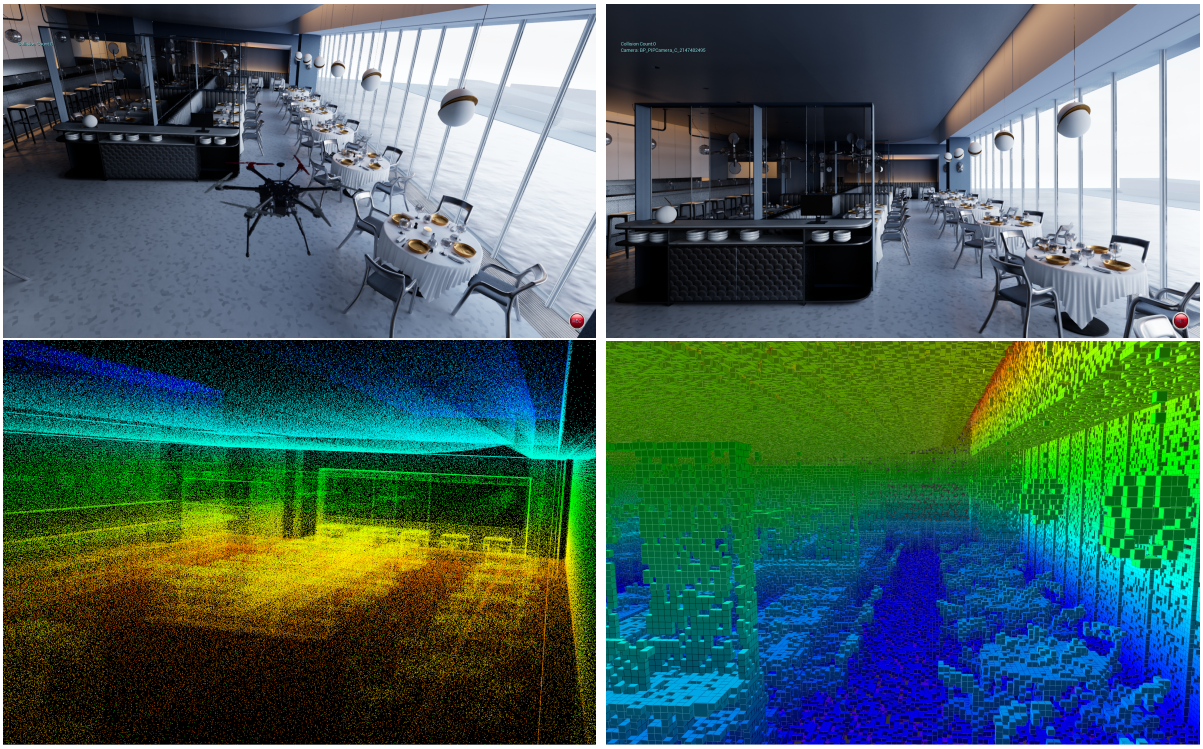


Figure 5.1 – Simulated data from a drone flying in an indoor environment, created using AirSim and Unreal Engine. The first row displays a capture of the drone and a rendered image, and the second row shows the corresponding point cloud and voxel map generated for the scene.

the 3D geometry well. Instead, the study proposes the application of various 3D metrics to MDE. Specifically, common measures such as precision, F-score, completeness, and Intersection Over Union are adapted for 3D. As mentioned previously, Rosinol et al. applied similar measures for precision and completeness in dense monocular SLAM [12], [114], [115], [150], but to the best of our knowledge they are the only ones. In Voxblox [39], voxel mapping is evaluated by comparing the estimated TSDF to a reference one reconstructed from the ground truth point cloud. A more recent study [175] also proposed to evaluate occupancy grids by comparing probability distributions computed from the 3D reconstruction and the ground truth point cloud.

We believe that the use of standard depth estimation metrics remains a reasonable approach for comparing our scale recovery procedure with similar methods. However, for a more meaningful analysis and to allow for future comparisons, it would seem wise to include additional 3D metrics, as suggested by recent research [177]. Still, these metrics may not be quite appropriate for evaluating voxel mapping, especially if the objective

is to construct a coarse map as we do. A more relevant evaluation for voxel mapping might be to evaluate the reconstructed occupancy grid based on its TSDF or probability distribution. However, this form of evaluation would require applying this approach to the existing works, but only a few have made their code publicly available [114], [116], [117].

Finally, since we have targeted this research for real-time application on drones, a comprehensive study of the framework running on embedded systems is crucial. A thorough analysis similar to [46] would be of great value, providing key insights into sparse SLAM performance, MDE inference times, as well as CPU and GPU loads, and memory usage. Implementing different acceleration methods can further optimize performance, including using half-precision computation for DNN inference, reducing camera resolution, tuning keyframe selection criteria, and reducing the number of keypoints. Each factor has a significant impact on the overall accuracy and the achievable navigation speed. Therefore, conducting such benchmarking would provide a deeper understanding of the algorithms' performance under real conditions and help identify the gaps that need to be bridged for practical deployment.

Tightly coupled approach

In this thesis, we proposed a decoupled pipeline to minimize any adverse effects on the baseline performance of sparse SLAM. However, as reported previously, predicting depth from a single image resulted in scale inconsistencies. An interesting avenue for future research is to include sparse SLAM depth points as a DNN input. These points, which reflect multiple observations, carry not only metric scale but also temporal information. In our current loosely coupled approach, we rely on a custom model to recover the scale. Moving towards a tightly coupled framework could take advantage of the strong capabilities of DNNs to model the relationship between the sparse SLAM depth information and the input image. As shown in Figure 5.2, such an approach could allow the prediction to be metric. Additionally, incorporating the reprojection error could also contribute to the prediction of the uncertainty. Overall, this integration has the potential to enhance the consistency and accuracy of depth predictions.

For example, we had described **CodeVIO** [7] and **CodeMapping** [6] implementing such an approach with a CVAE. Additionally, the field of depth completion, which initially focused on completing the dense depth prediction from semi-dense LiDAR measurements only, provides valuable insights. Currently, RGB-guided depth completion methods, which employ a fusion of point clouds and corresponding color images, are the most advanced

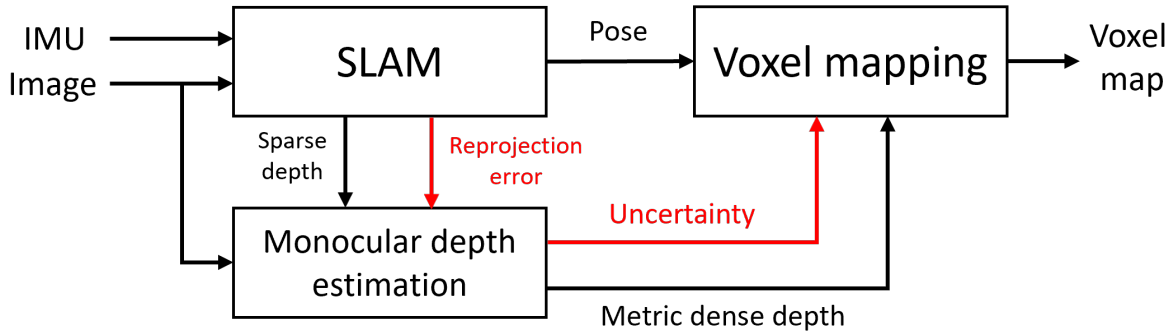


Figure 5.2 – Architecture of a tightly coupled pipeline that would predict metric dense depth from a monocular image and the sparse depth estimated by SLAM. The integration of SLAM reprojection error could also contribute to the prediction of depth uncertainty.

techniques [168]. Despite the considerably lower density of points in SLAM compared to LiDAR data, these techniques can be highly advantageous. Notably, several operators have been developed to process unstructured data, such as point clouds, and to integrate them with image features. For example, Du et al. [178] employed a graph neural network and introduced edge convolution and Dynamic Graph Representation to extract embeddings from point clouds. Such a technique could potentially be adapted to augment models like **ZeroDepth** by incorporating sparse SLAM depth, especially since the model already merges image and geometric embeddings derived from camera intrinsics. Furthermore, the adjustment of the loss function to merge image and point cloud data, as well as to predict uncertainty, could also be inspired by depth completion research [168].

Therefore, the pursuit of a tightly coupled approach to MDE emerges as a compelling research direction. This approach aims to enhance the accuracy of depth predictions by more closely integrating MDE with SLAM processes, potentially resulting in mapping solutions that are more precise and reliable.

Voxel mapping enhancement

The voxel mapping module of our system integrates Voxblox, which was developed for drone navigation [39]. Voxblox constructs a TSDF and introduces an optimized method to derive an Euclidean Signed Distance Function (ESDF). The ESDF is particularly useful for navigation purposes, as it provides the Euclidean distance to the nearest obstacle for each voxel. In terms of performance, Voxblox is efficient, capable of real-time operation on CPU only, and its speed can be further enhanced by grouped raycasting. However,

our implementation has the potential for further optimization. The recent development of nvblox [165], which parallelizes the algorithm and enables GPU execution, has shown a significant boost in computational speed and has been demonstrated on an NVIDIA Jetson embedded platform. This advancement demonstrates the possibility of substantial computational acceleration, which we aim to explore for future enhancements.

Furthermore, the refinement of the volumetric fusion of depth maps can be achieved through the adaptation of the raycasting weighing function. As previously stated, prior research has established weight functions based on sensor models [12], [39], [56], [172], [173], while a few others have used neural networks for depth map fusion [179], [180]. A notable dense SLAM method [115] has successfully implemented probabilistic volumetric fusion by weighing depth maps by their respective uncertainties. We also believe that incorporating depth uncertainty into our fusion process could significantly reduce the effects of noise and outliers. In the continuation of our work, we are considering using the uncertainty prediction provided by **ZeroDepth** for this purpose. SLAM algorithms usually detect keypoints in areas with significant gradients, such as object edges. However, the depth predicted in these regions often exhibits deformations and increased noise, especially around occlusion boundaries. This issue is discussed in [177] and is illustrated in the **ZeroDepth** prediction example shown in Figure 4.10 on the ladder. Therefore, we believe that the SLAM sparse depth could help alleviate this problem.

Overall, enhancements in voxel mapping should result in more detailed and navigable environmental maps. Accelerating the process will increase performance on embedded systems, and adjusting the weighting function should prevent the map from being corrupted by erroneous predictions.

Robustifying localization

A promising avenue for future research is to use the densified map to enhance the robustness of SLAM localization. This approach is particularly relevant in scenarios where **ORB-SLAM 3**, as shown in Figure 4.2, exhibits a significant reduction in the number of points, which is often observed in areas with uniform textures or during some pure rotational motions. While high-quality IMUs can compensate for these constraints, their lower-cost counterparts may not offer the same level of reliability. In typical SLAM systems, keypoints are added to the map after triangulation from multiple viewpoints and a map consistency check. We assume that the information from a densified map could be valuable to the initialization of additional keypoints. By retrieving the keypoint depth

from the densified map, we could potentially loosen the criteria for adding new keypoints to the map, potentially overcoming some of the current limitations of SLAM systems in complex environments. This method could lead to improved navigation accuracy and reliability.

Appendix A

Publications

This thesis has been the subject of the following publications:

1. Y. Habib, P. Papadakis, C. Buche, C. Le Barz, and A. Fagette, *Dense, metric and real-time 3d reconstruction for autonomous drone navigation*, Journées des Jeunes Chercheurs en Robotique (JJCR 2021), Poster, Oct. 2021. [Online]. Available: https://hal.science/hal-04184995
2. Y. Habib, P. Papadakis, C. Le Barz, A. Fagette, T. Gonçalves, and C. Buche, « Densifying slam for uav navigation by fusion of monocular depth prediction », *in 2023 9th International Conference on Automation, Robotics and Applications (ICARA)*, 2023, pp. 225–229. DOI: 10.1109/ICARA56516.2023.10125712
3. Y. Habib, P. Papadakis, A. Fagette, C. L. Barz, T. Gonçalves, and C. Buche, « From sparse SLAM to dense mapping for UAV autonomous navigation », *in Geospatial Informatics XIII*, K. Palaniappan, G. Seetharaman, and J. D. Harguess, Eds., International Society for Optics and Photonics, vol. 12525, SPIE, 2023, p. 125250C. DOI: 10.1117/12.2663706

Appendix B

Additional results

B.1 Sparse SLAM evaluation

This section presents additional results from our evaluation of several sparse SLAM algorithms, including **Kimera** [12], **Basalt** [2], and **ORB-SLAM 3** [1] in both stereo-inertial and monocular-inertial configurations.

EuRoC: *MH04* sequence

Specifically, we report the performance of these algorithms on the *MH04* scene from the EuRoC dataset, which is identified as one of the more challenging scenarios in the dataset. Despite the relatively slow speed of the drone and the rich texture of the scene, one of the areas explored by the drone is particularly dark with limited visibility. Under these challenging conditions, all of the algorithms we tested demonstrated robust tracking capabilities, with the exception of **Kimera**. We present the estimated trajectories aligned with the ground truth through $SE(3)$ alignment. The corresponding colormaps represent ATE, RPE_{rot}, and RPE_{trans}.

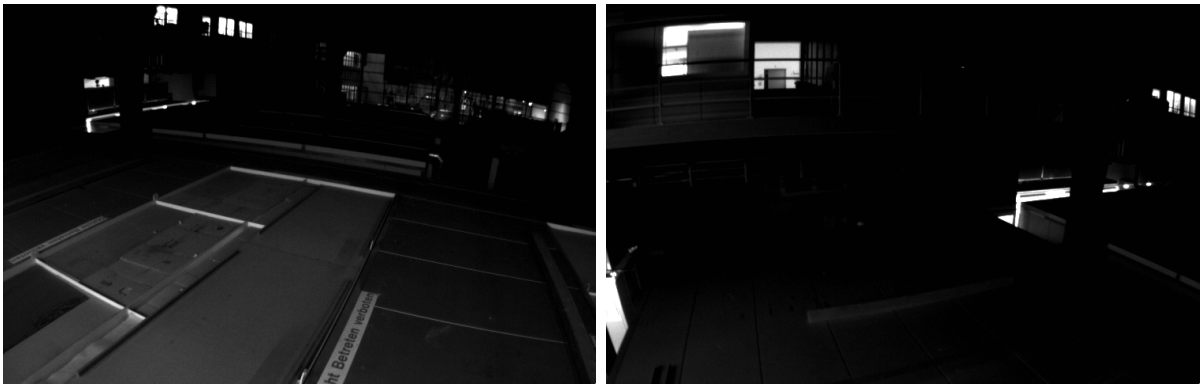


Figure B.1 – Dark images from the *MH04* scene of the EuRoC dataset. Despite the low light, distinct areas are visible, and the drone navigates at a slow speed

Method	ATE (m)	RPE_{rot} (deg)	RPE_{trans} (m)	cover
Basalt	0.0951	0.0345	0.0040	100.0%
Kimera	3.5326	0.0943	0.1127	99.0%
ORB_SLAM3	0.0515	0.0196	0.0064	81.5%
ORB_SLAM3 (MI)	0.1288	0.1599	0.0253	80.1%

Table B.1 – Median evaluation results on the *MH04* scene of EuRoC.

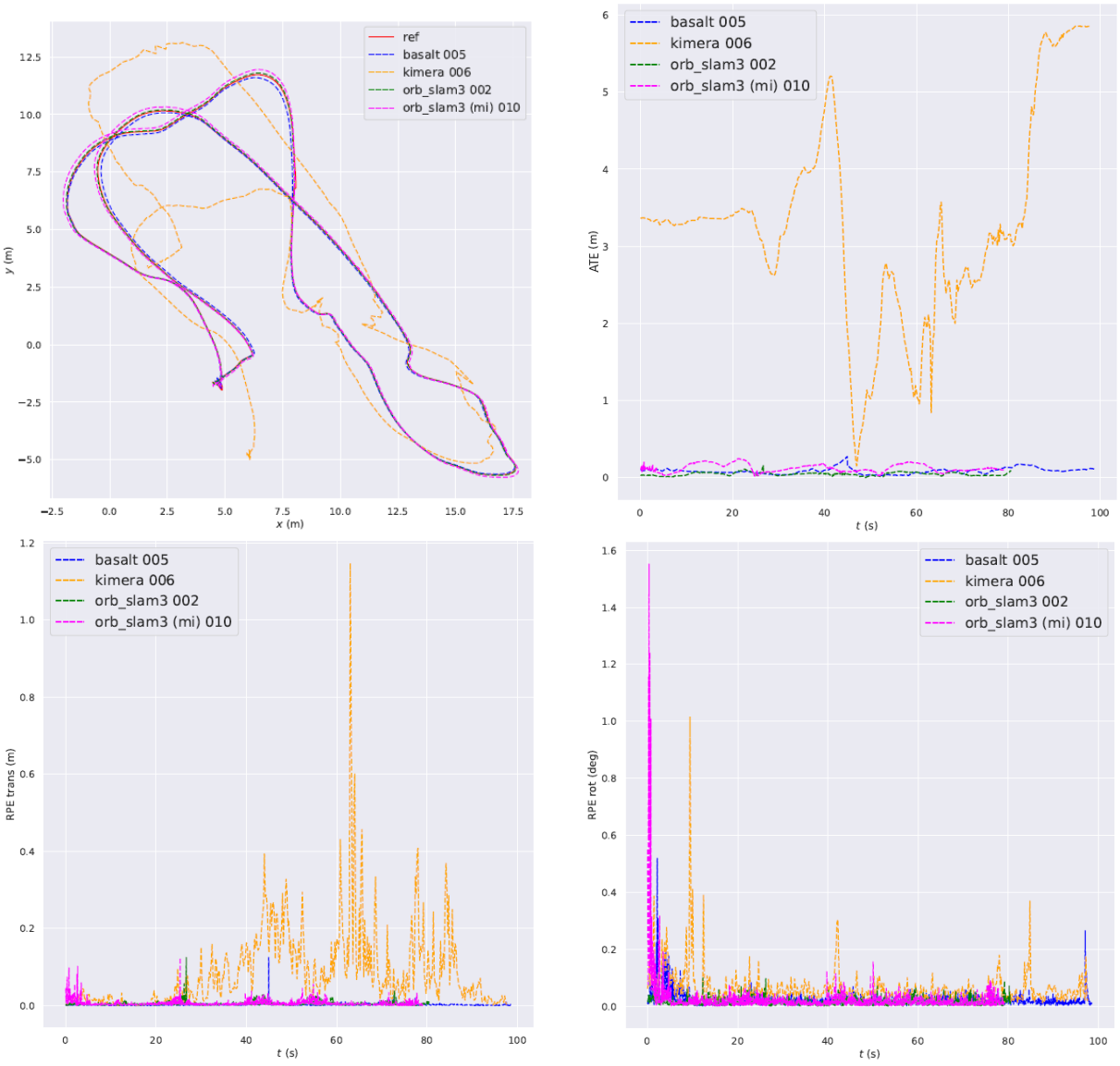


Figure B.2 – Evaluation of Basalt, Kimera, ORB-SLAM 3, and ORB-SLAM 3 (MI) on the *MH04* sequence from the EuRoC dataset. The plots presented are: aligned trajectories (top-left), ATE (top-right), RPE_{trans} (bottom-left), and RPE_{rot} (bottom-right).

Basalt

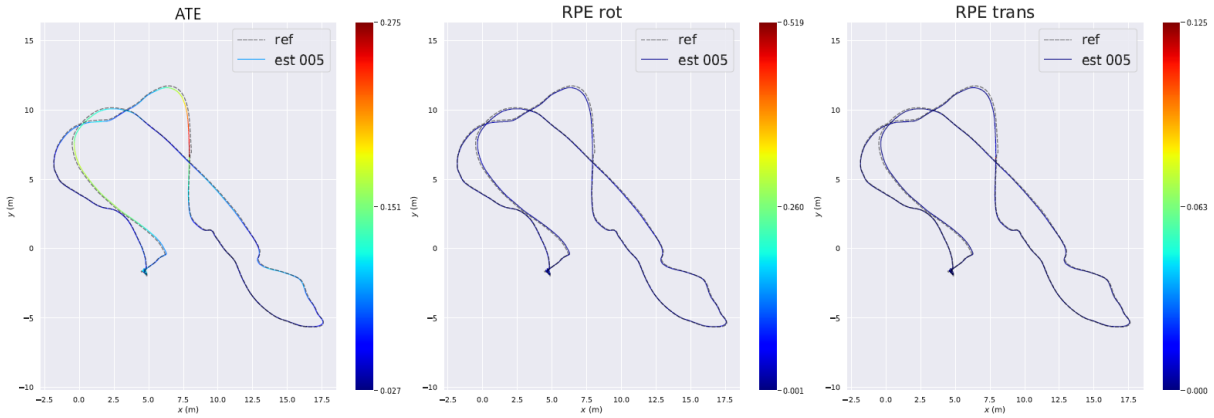


Figure B.3 – Trajectory evaluations of Basalt results on scene *MH04*.

Kimera

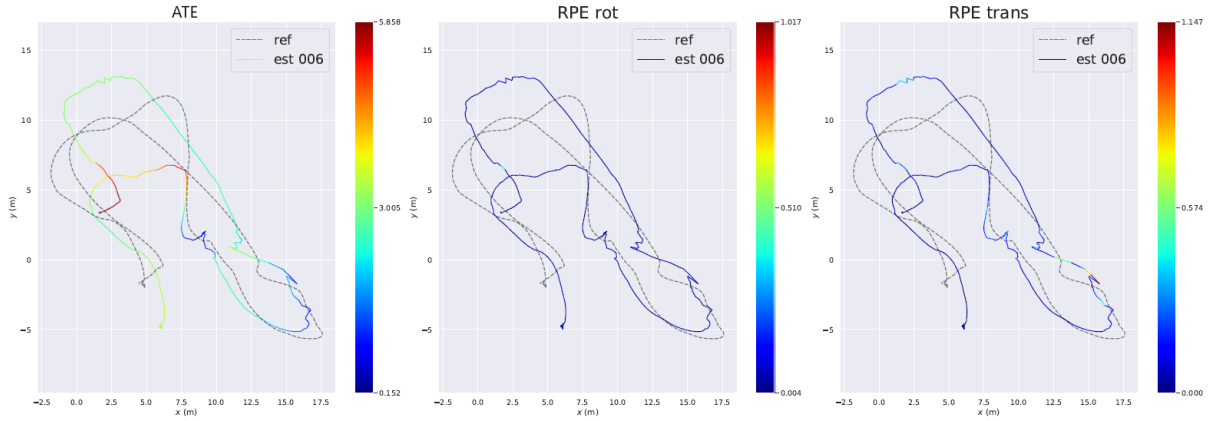


Figure B.4 – Trajectory evaluations of Kimera results on scene *MH04*.

ORB-SLAM 3

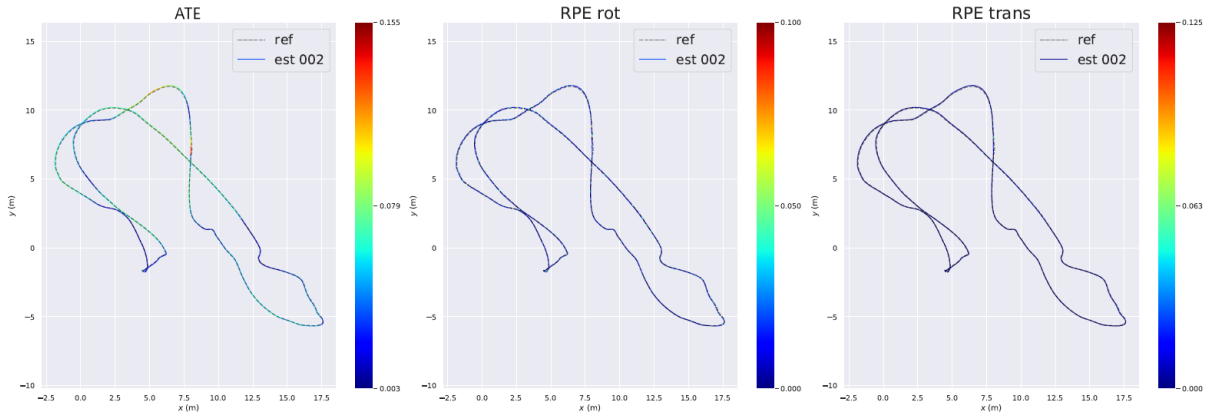


Figure B.5 – Trajectory evaluations of ORB-SLAM 3 results on scene *MH04*.

ORB-SLAM 3 (MI)

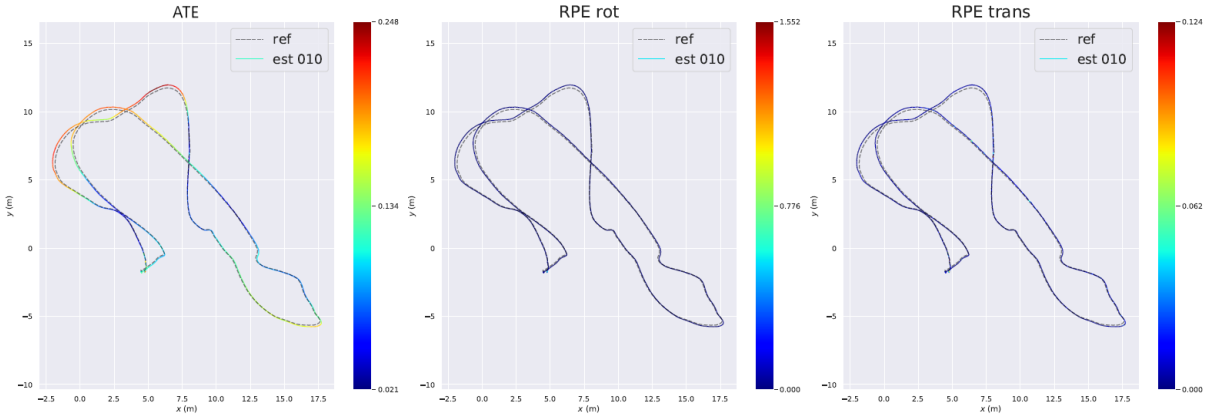


Figure B.6 – Trajectory evaluations of ORB-SLAM 3 (MI) results on scene *MH04*.

UZH-FPV: *indoor_forward_9* sequence

Below, we report the the results of the SLAM algorithms on the *indoor_forward_9* sequence which is not the most difficult but is representative of most *forward* sequences. Similarly, we report the trajectories under the same conditions as before. Comparative plots of all methods were discussed in Section 3.2. This section presents detailed measurements for each method. While **Basalt** does not accurately estimate scale, the RPE plots demonstrate that the relative errors remain acceptable. The results also indicate that **Kimera** struggles with estimating rotation, as evidenced by the REP_{rot} plots. Overall, **ORB-SLAM 3** performs well, even though in stereo-inertial settings, the scale is slightly inaccurate.

Method	ATE (m)	RPE_{rot} (deg)	RPE_{trans} (m)	cover
Basalt	1.583	0.366	0.055	99.9%
Kimera	3.533	2.780	0.625	99.4%
ORB_SLAM3	0.953	0.376	0.048	99.9%
ORB_SLAM3 (MI)	0.575	0.376	0.043	97.3%

Table B.2 – Median evaluation results on the *indoor_forward_9* scene of UZH-FPV.

Basalt

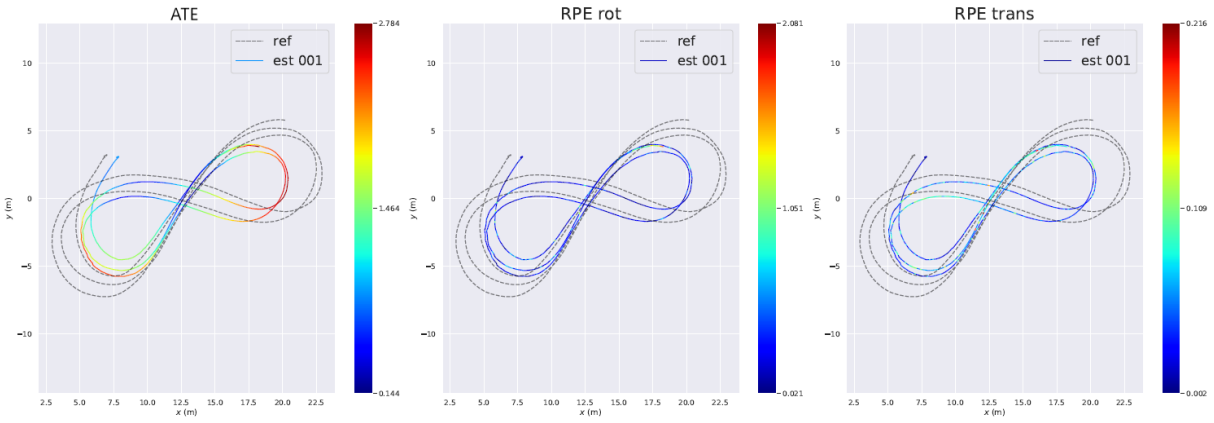


Figure B.7 – Evaluation of Basalt on scene *indoor_forward_9* sequence.

Kimera

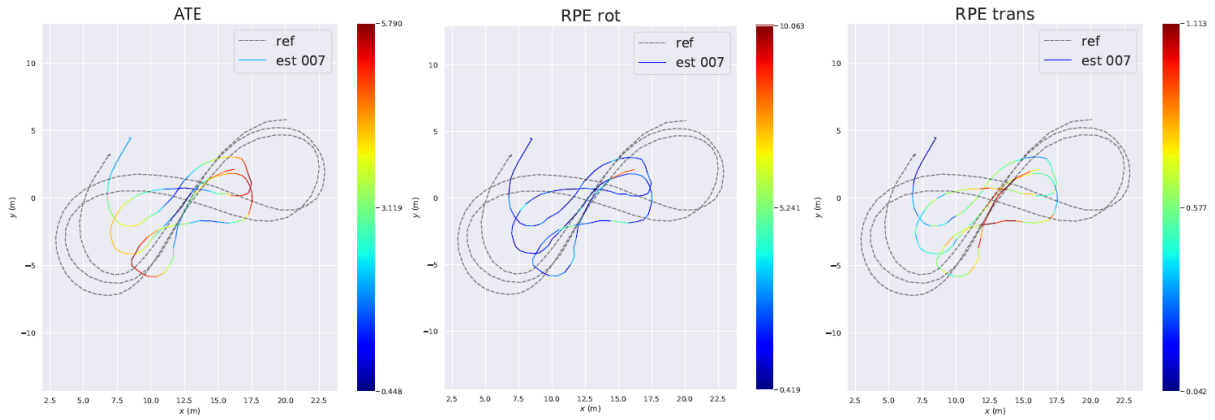


Figure B.8 – Evaluation of Kimera on scene *indoor_forward_9* sequence.

ORB-SLAM 3

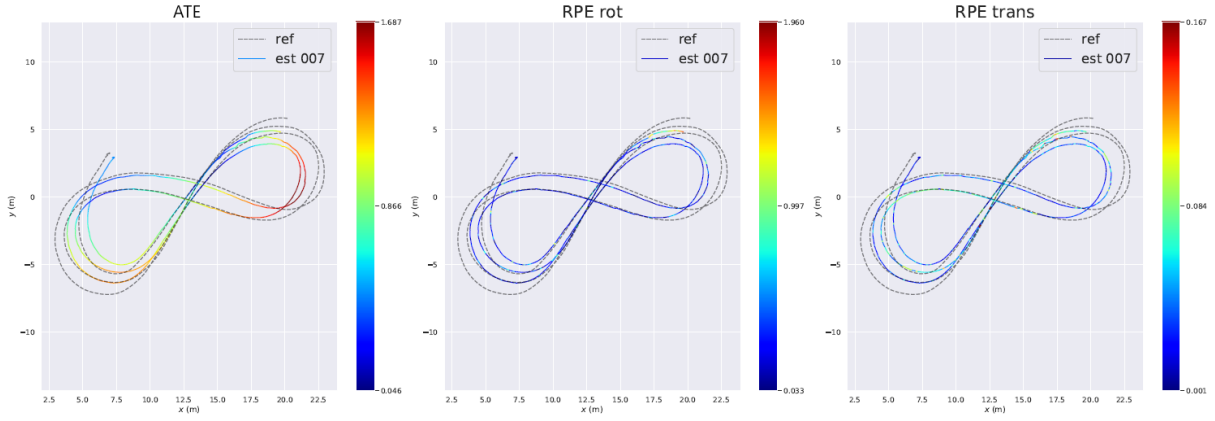


Figure B.9 – Evaluation of ORB-SLAM 3 on scene *indoor_forward_9* sequence.

ORB-SLAM 3 (MI)

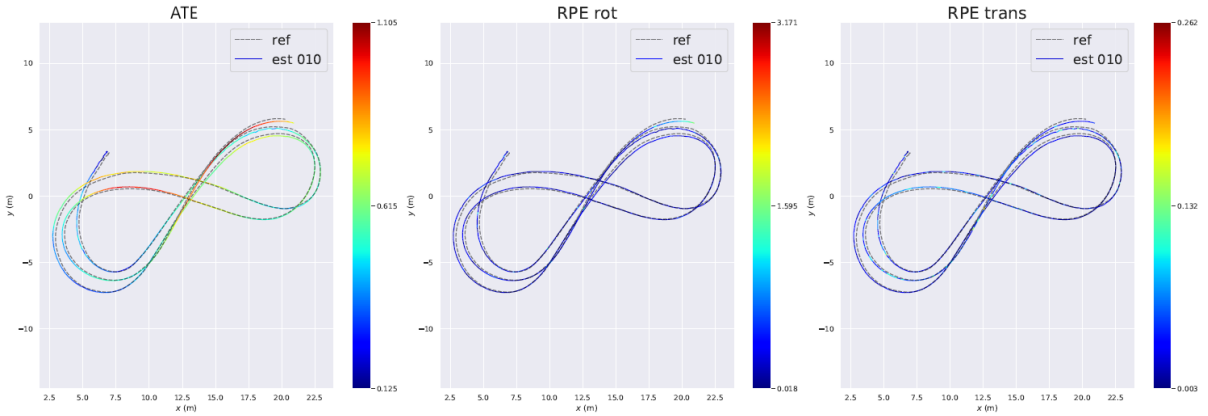


Figure B.10 – Evaluation of ORB-SLAM 3 (MI) on scene *indoor_forward_9* sequence.

UZH-FPV: *indoor_forward_7* sequence

The *indoor_forward_7* sequence represents a notable challenge among the *forward* sequences due to its length and several aggressive motions and rotations. It also includes complex maneuvers such as circling objects while maintaining camera focus on a central position. Detailed trajectory and measurement data are provided below. For this sequence, **Kimera** struggles with accurate camera pose tracking, as evidenced by high translational and rotational errors. The intense optical flow poses a challenge to the feature detection and matching capabilities of these algorithms. Thus, while **Basalt** is generally consistent, it experiences sporadic but severe rotational errors, resulting in cumulative trajectory drift. Conversely, **ORB-SLAM 3** demonstrates robust performance in both sensor configurations, benefiting from the resilience of ORB features, as consistent with its performance in other scenes.

Method	ATE (m)	RPE_{rot} (deg)	RPE_{trans} (m)	cover
Basalt	1.268	0.894	0.054	100.0%
Kimera	6.251	5.593	0.716	99.7%
ORB_SLAM3	0.341	0.705	0.042	100.0%
ORB_SLAM3 (MI)	0.167	0.675	0.034	100.0%

Table B.3 – Median evaluation results on the *indoor_forward_7* scene of UZH-FPV.

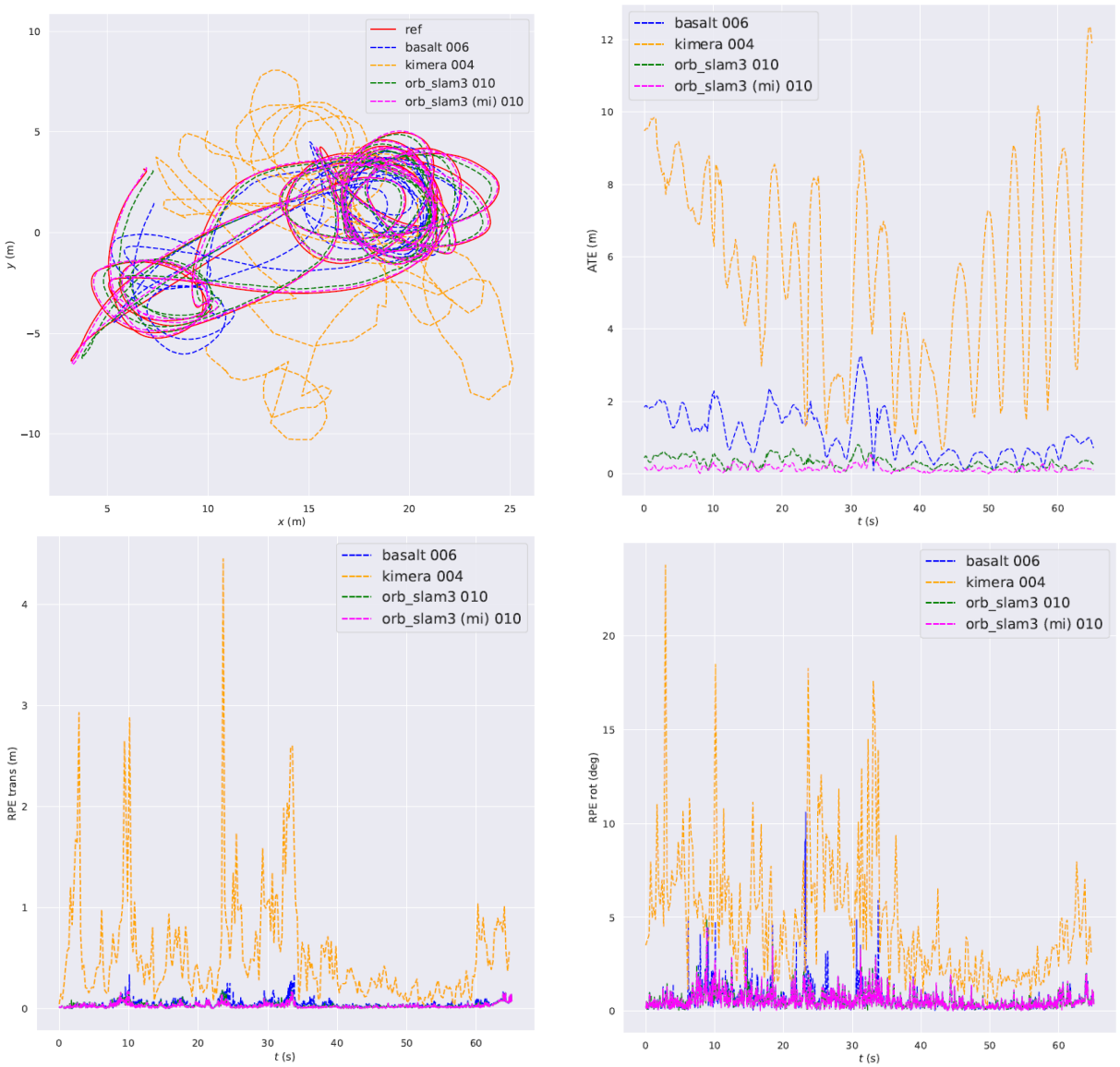


Figure B.11 – Evaluation of Basalt, Kimera, ORB-SLAM 3, and ORB-SLAM 3 (MI) on the *indoor_forward_7* sequence from the EuRoC dataset. The plots presented are: aligned trajectories (top-left), ATE (top-right), RPE_{trans} (bottom-left), and RPE_{rot} (bottom-right).

Basalt

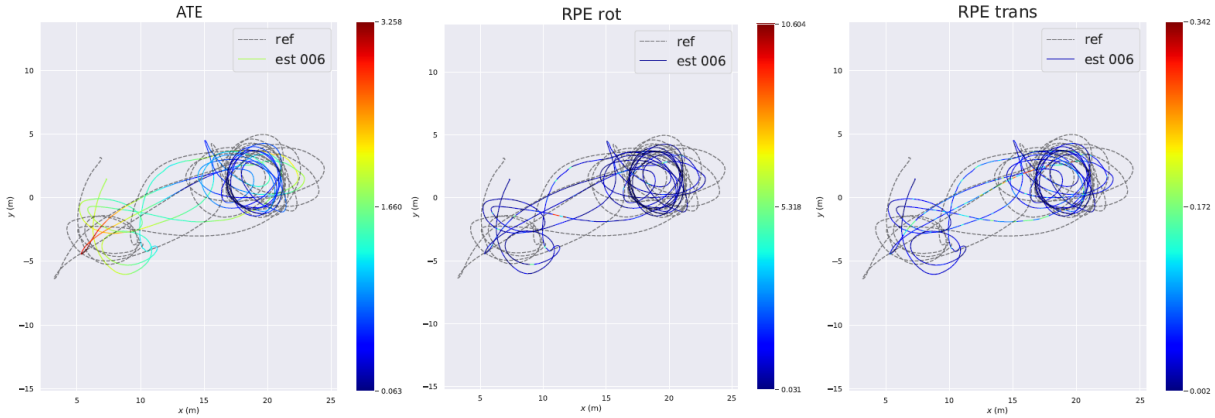


Figure B.12 – Evaluation of Basalt on scene *indoor_forward_7* sequence.

Kimera

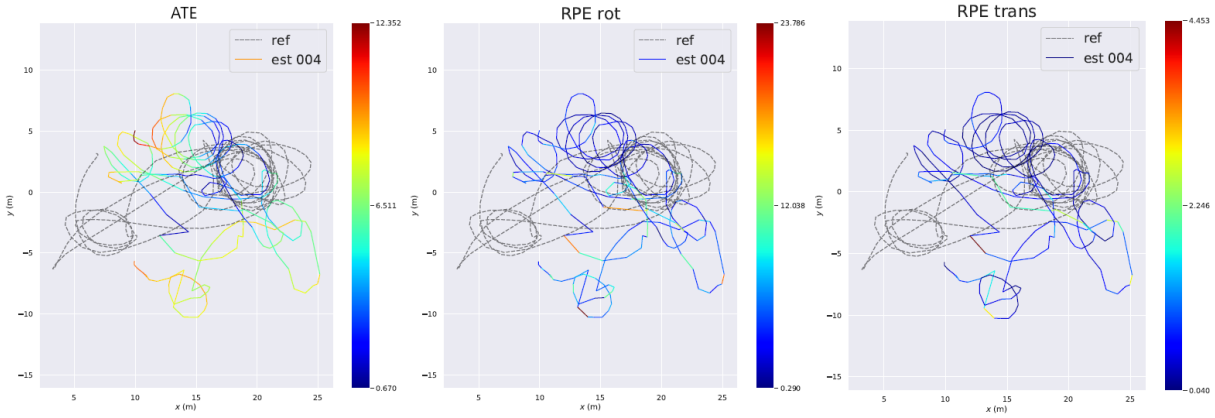


Figure B.13 – Evaluation of Kimera on scene *indoor_forward_7* sequence.

ORB-SLAM 3

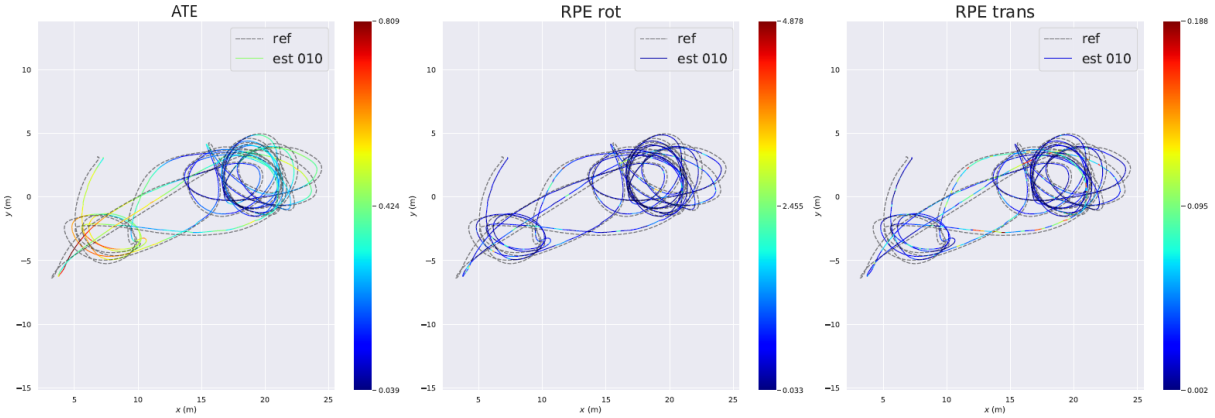


Figure B.14 – Evaluation of ORB-SLAM 3 on scene *indoor_forward_7* sequence.

ORB-SLAM 3 (MI)

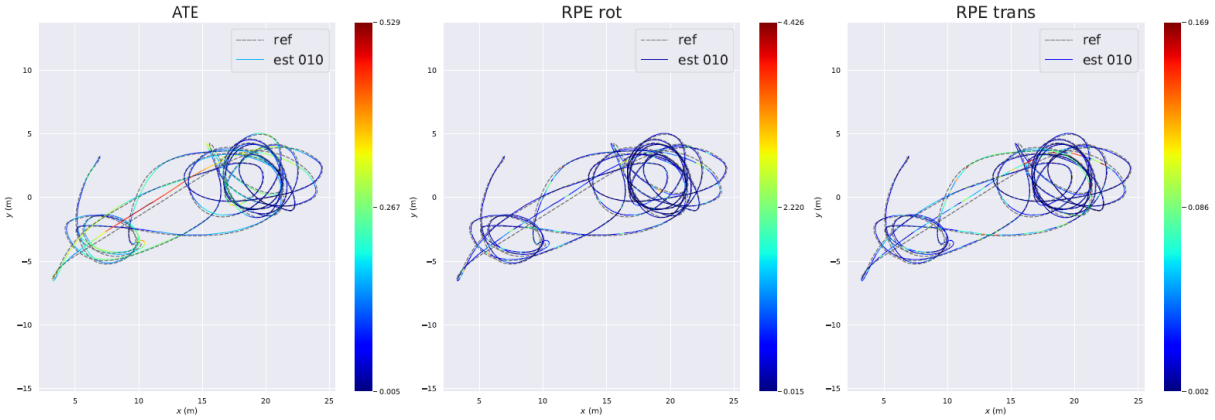


Figure B.15 – Evaluation of ORB-SLAM 3 (MI) on scene *indoor_forward_7* sequence.

UZH-FPV: *indoor_45_12* sequence

The individual performance of each SLAM method on the *indoor_45_12* sequence is reported below. This sequence was captured by the camera pointing down, mostly capturing textureless ground images. **Basalt** exhibits good trajectory estimation except for scale precision. **Kimera** struggles with translation estimation and especially with rotation, as indicated by the RPE plots. **ORB-SLAM 3** performs similarly to **Basalt** in its stereo-inertial configuration. On the other hand, its monocular-inertial configuration, while more accurate, especially in scale estimation, shows less consistency. Notably, it failed to complete the sequence in the 10 trials, achieving an average trajectory coverage of only 73.6%.

Method	ATE (m)	RPE_{rot} (deg)	RPE_{trans} (m)	cover
Basalt	1.392	0.276	0.026	100.0%
Kimera	6.437	2.713	0.367	99.2%
ORB_SLAM3	1.342	0.319	0.029	97.2%
ORB_SLAM3 (MI)	0.645	0.296	0.045	73.6%

Table B.4 – Median evaluation results on the *indoor_45_12* scene of UZH-FPV.

Basalt

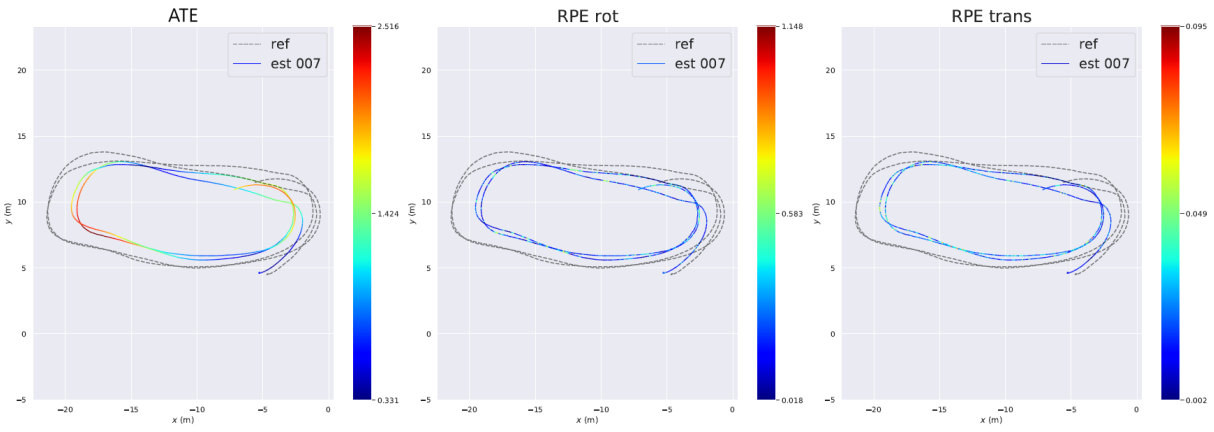


Figure B.16 – Evaluation of Basalt on scene *indoor_45_12* sequence.

Kimera

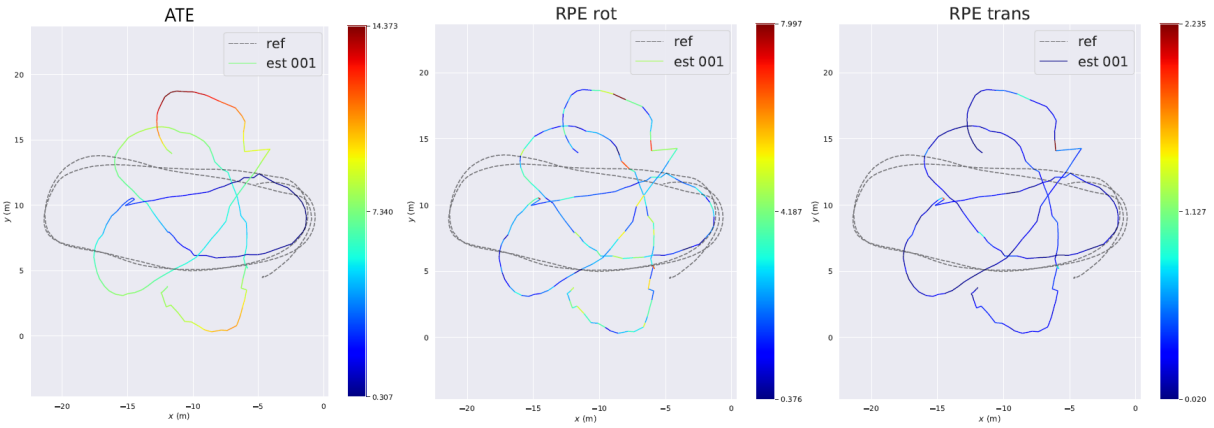


Figure B.17 – Evaluation of Kimera on scene *indoor_45_12* sequence.

ORB-SLAM 3

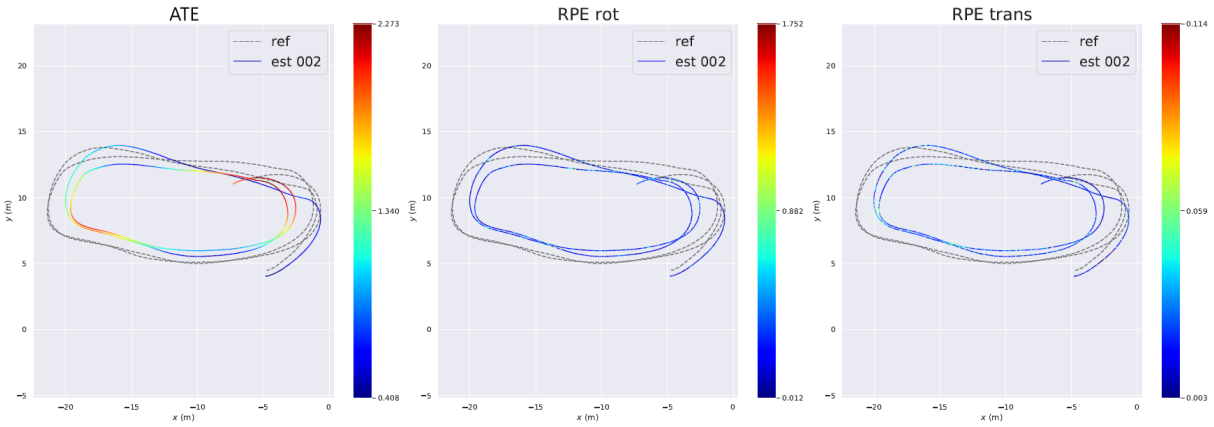


Figure B.18 – Evaluation of ORB-SLAM 3 on scene *indoor_45_12* sequence.

ORB-SLAM 3 (MI)

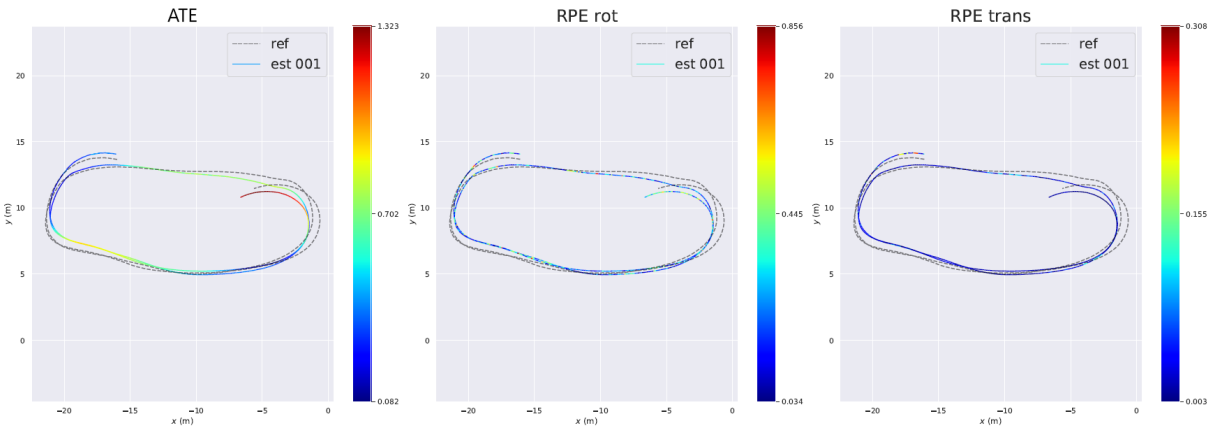


Figure B.19 – Evaluation of ORB-SLAM 3 (MI) on scene *indoor_45_12* sequence.

HILTI

Below, we present the evaluation of ORB-SLAM 3 on a monocular-inertial setup using selected sequences from the HILTI dataset with released ground truth. The trajectory ground truth are extremely sparse except for the last sequence. We plot the estimated trajectories aligned with the ground truth and the evolution of the ATE over the sequence. The *Basement_1* and *Basement_4* scenes were recorded in a long underground corridor with many objects in the second sequence. The other scenes were taken at a construction site and in a large empty hall.

Basement_1

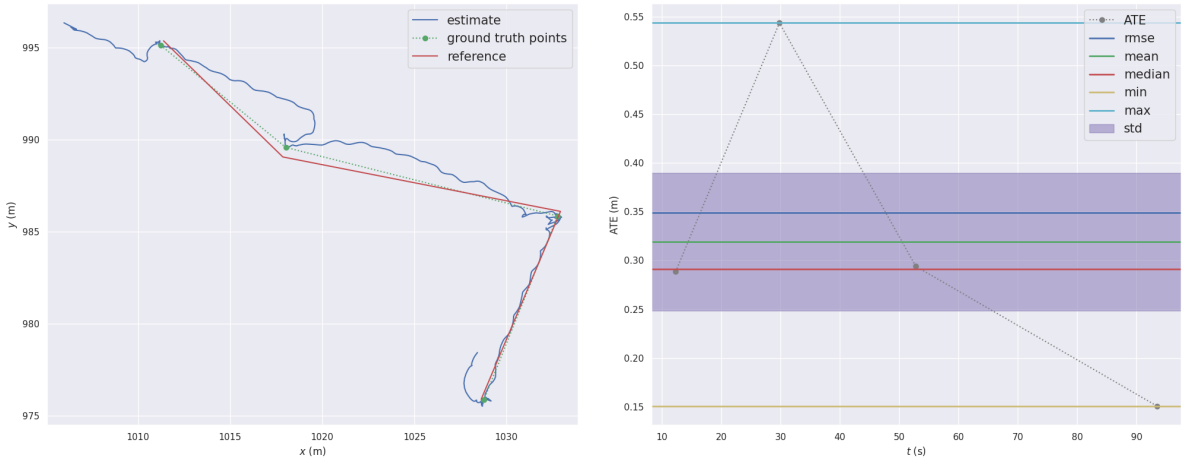


Figure B.20 – Results for the *Basement_1* scene from the HILTI dataset. Left: The estimated trajectory aligned with the ground truth. Right: Evolution of the ATE throughout the sequence.

Basement_4

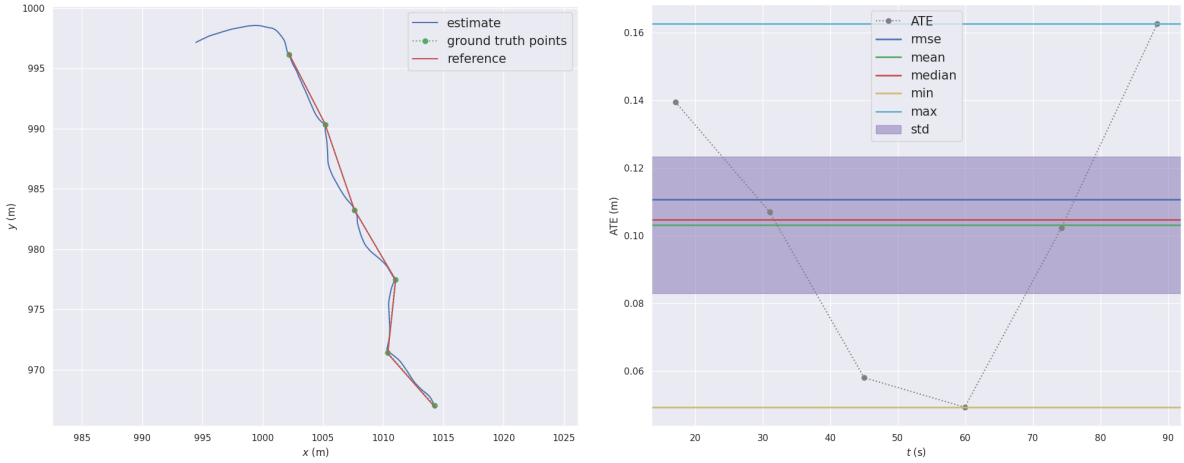


Figure B.21 – Results for the *Basement_4* scene from the HILTI dataset. Left: The estimated trajectory aligned with the ground truth. Right: Evolution of the ATE throughout the sequence.

Construction_Site_2

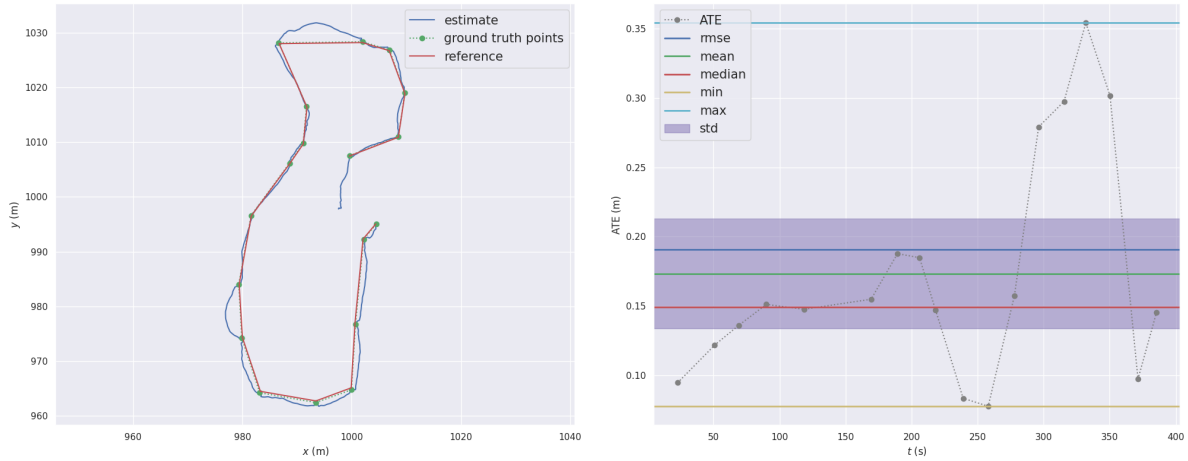


Figure B.22 – Results for the *Construction_Site_2* scene from the HILTI dataset. Left: The estimated trajectory aligned with the ground truth. Right: Evolution of the ATE throughout the sequence.

uzh_tracking_area_run2

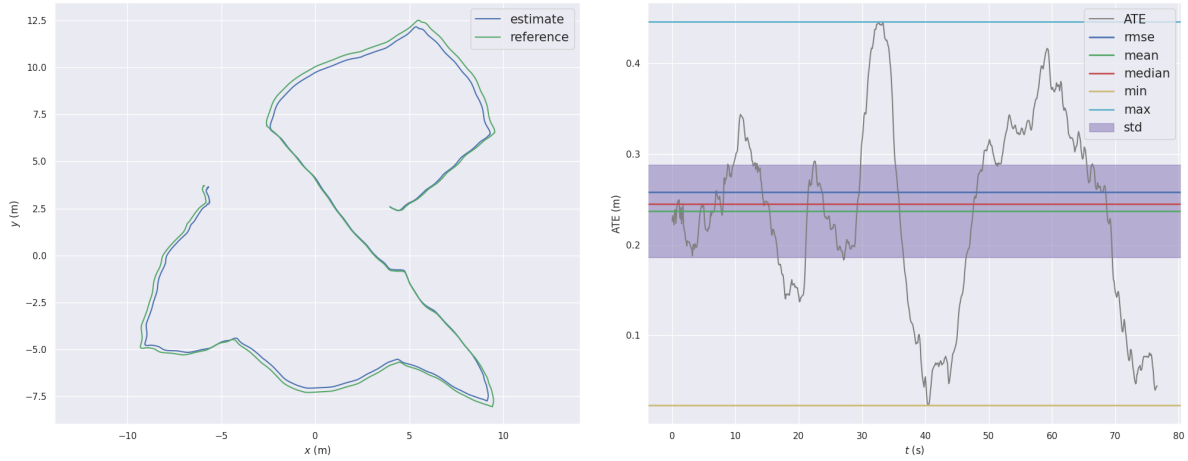


Figure B.23 – Results for the *uzh_tracking_area_run2* scene from the HILTI dataset. Left: The estimated trajectory aligned with the ground truth. Right: Evolution of the ATE throughout the sequence.

B.2 Scale recovery

In this section, we provide additional depth prediction results from **PackNet-Sfm** [8] and **ZeroDepth** [9] on the same input images for comparison. The **camviz** tool [169] was employed to visualize the results, showcasing the source image (top-left), the predicted depth map (bottom-left), and the corresponding 3D projection (right).

EuRoC: *V101* sequence

In the following, we report some results on the *V101* sequence of the EuRoC [3] dataset. Additionally, we present the estimated scales for **PackNet-Sfm** in Figure B.24 using both ground truth and our approach. A plot of the corresponding absolute differences is also included. The same results were discussed for **ZeroDepth** in Section 4.1.1.

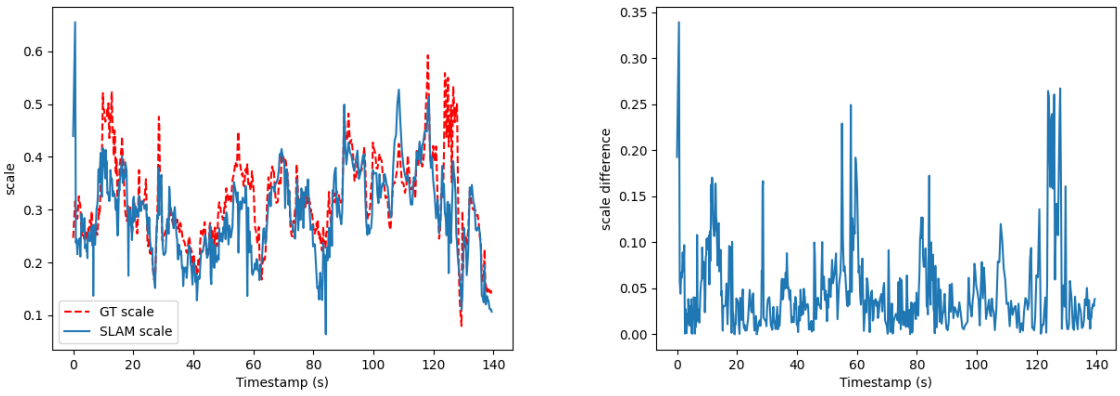


Figure B.24 – Left: A plot of the global scale factors estimated for PackNet-Sfm [9] predictions on the *V101* scene of the EuRoC dataset, comparing the use of ground truth (GT) scaling with our SLAM-based approach. Right: A plot of the absolute difference between the two scale estimates over the sequence.

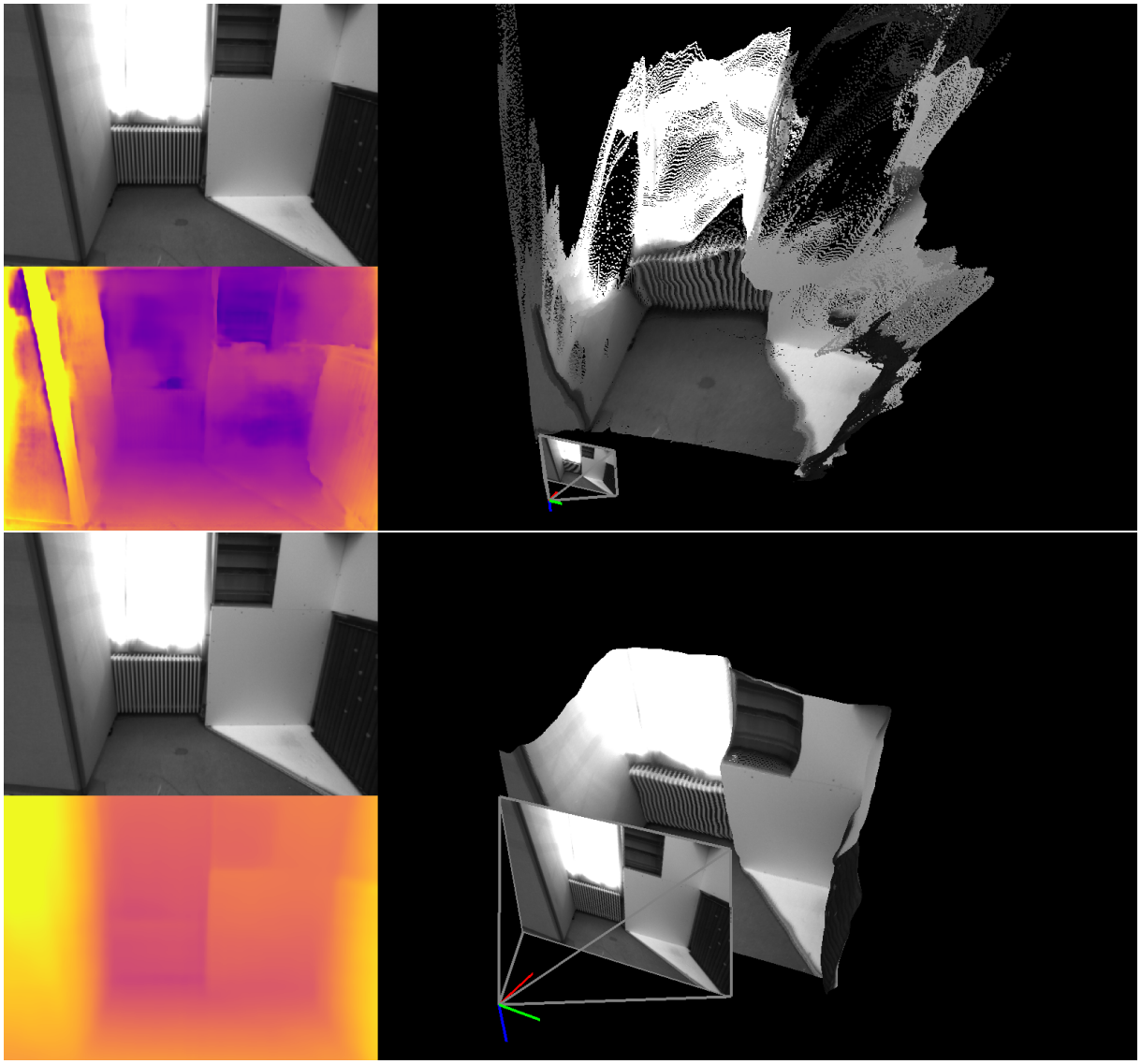


Figure B.25 – Keyframe 182: PackNet-Sfm (top) and ZeroDepth (bottom) depth predictions on the *V101* scene of EuRoC.

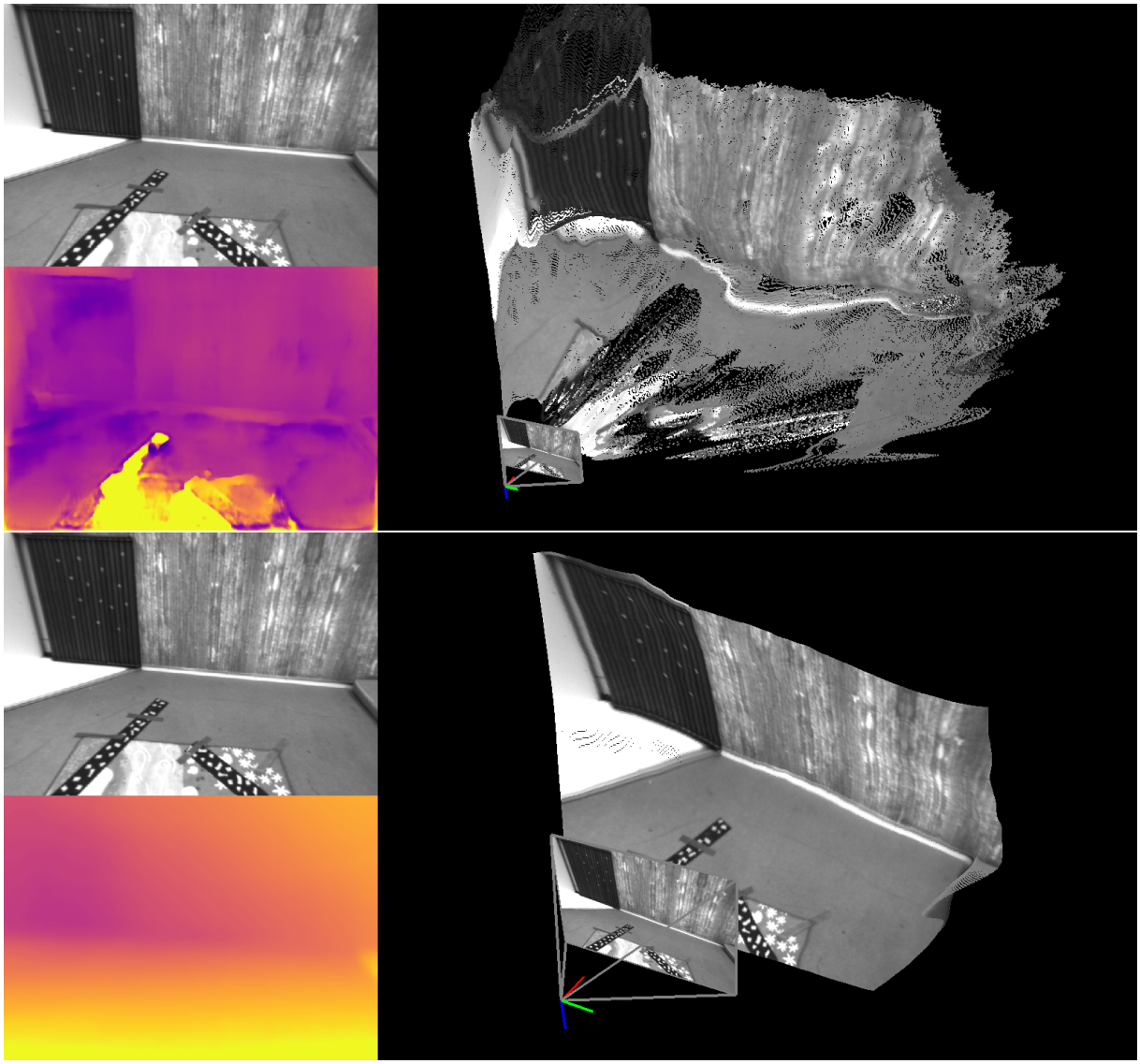


Figure B.26 – Keyframe 200: PackNet-Sfm (top) and ZeroDepth (bottom) depth predictions on the *V101* scene of EuRoC.

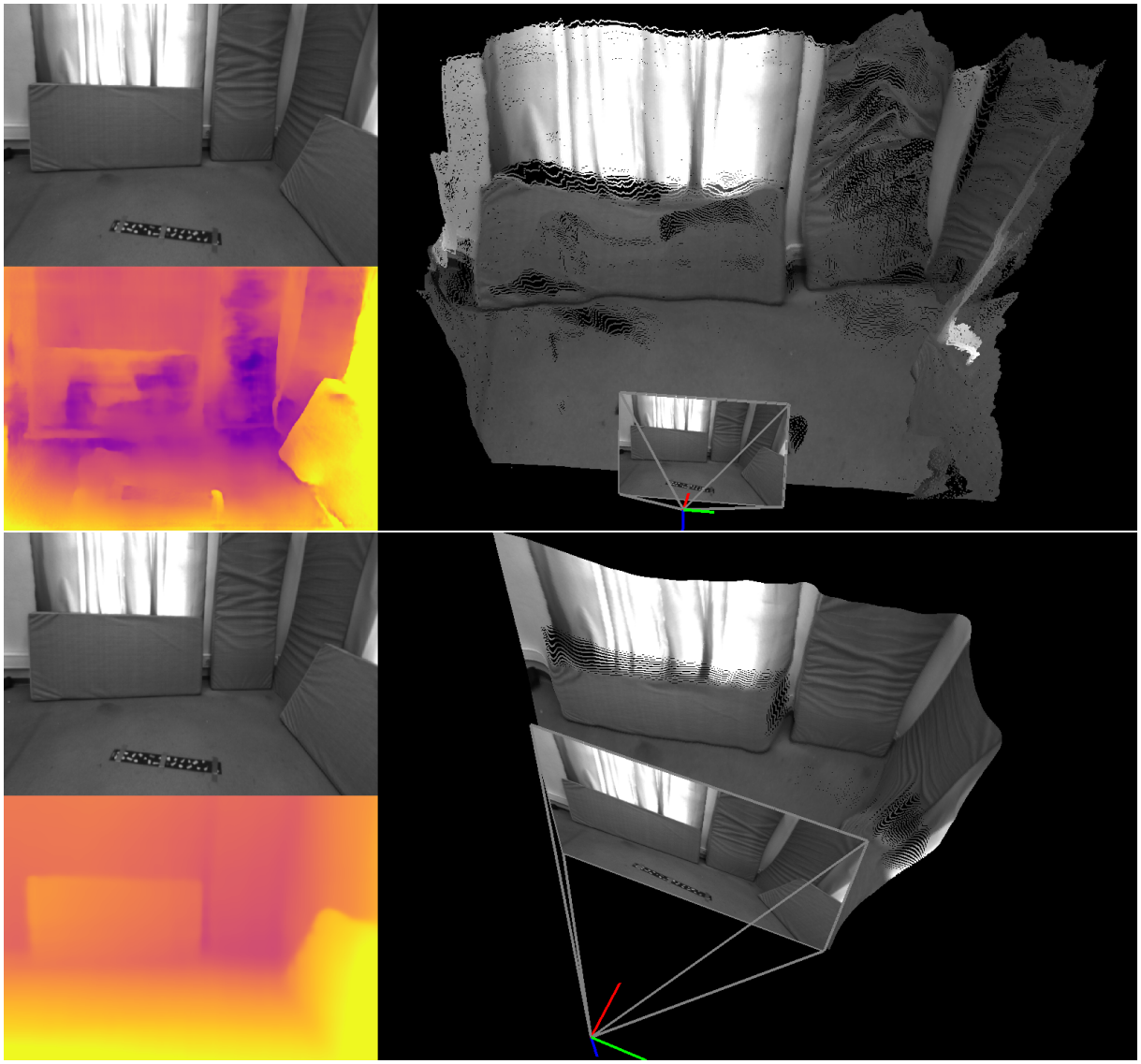


Figure B.27 – Keyframe 353: PackNet-Sfm (top) and ZeroDepth (bottom) depth predictions on the *V101* scene of EuRoC.

HILTI: *Basement_1*

Overall performance on the *Basement_1* sequence of the HILTI [11] dataset was discussed in Section 4.1.1. Our focus here is to highlight specific situations where **ZeroDepth** failed to correctly estimate depth. We noticed inconsistent results in this scene with several flat depth maps as illustrated in Figure B.28. **PackNet-Sfm** managed to provide rough predictions of the geometric structure of the corridor, despite a very noisy prediction in the upper part.

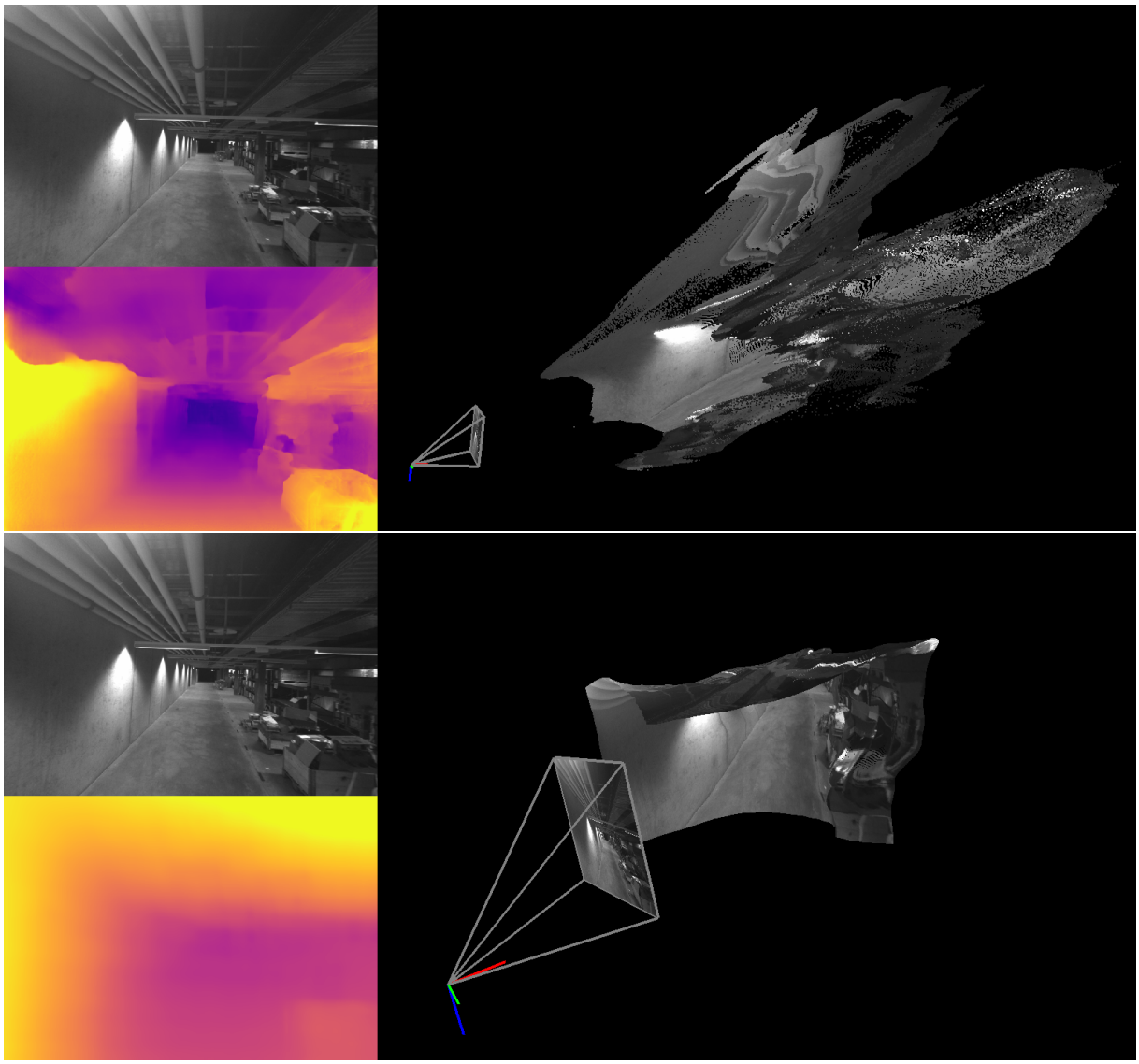


Figure B.28 – Keyframe 024: PackNet-Sfm (top) and ZeroDepth (bottom) depth predictions on the *Basement_1* scene of HILTI.

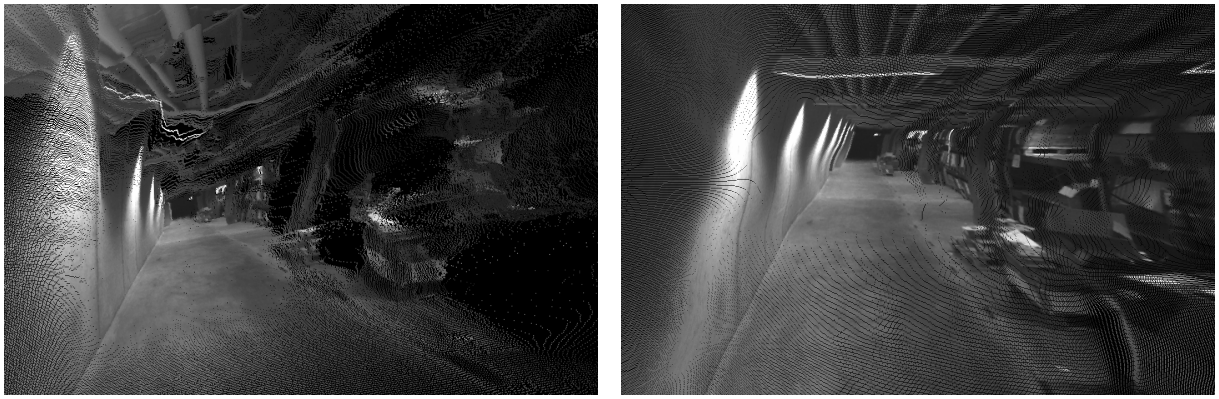


Figure B.29 – Keyframe 024: Front view of PackNet-Sfm (top) and ZeroDepth (bottom) depth predictions on the *Basement_1* scene of HILTI.

Collected data

We provide additional results for monocular depth estimation using **ZeroDepth** on color images collected in the corridors of the IMT Atlantique school. The purpose of this study is to evaluate the indoor performance of the model, particularly in lengthy hallways, as opposed to the scenarios of the HILTI dataset. We observed that **ZeroDepth** struggles to estimate depth accurately in poor lighting conditions. Figures B.30 and B.31 illustrate this with depth predictions in frames 138 and 157 of our sequence with front and side views . The brighter image seems to have an accurate depth prediction along the length of the corridor, while the darker image has a very shallow depth of field. The model estimates the depth for the first few well-lit meters properly, but the more distant, darker areas appear unnaturally flat.

These scenarios present a significant challenge as the limited lighting provides little visual information. Nevertheless, in situations with dark illumination but limited depth of field, the model shows improved performance, as shown in Figures B.32 and B.33.

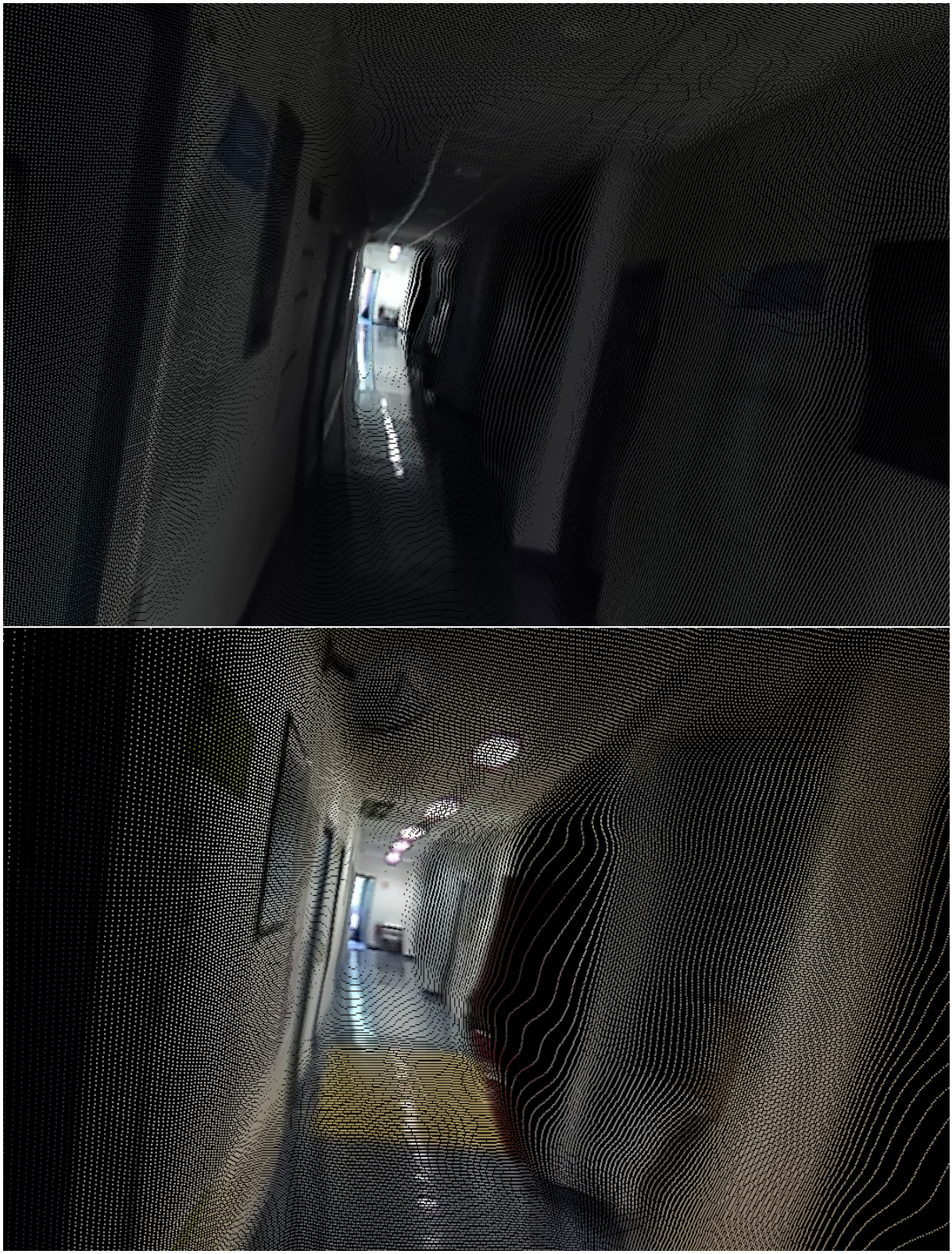


Figure B.30 – Front view of ZeroDepth results on our data: frame 138 (left) and 157 (right).

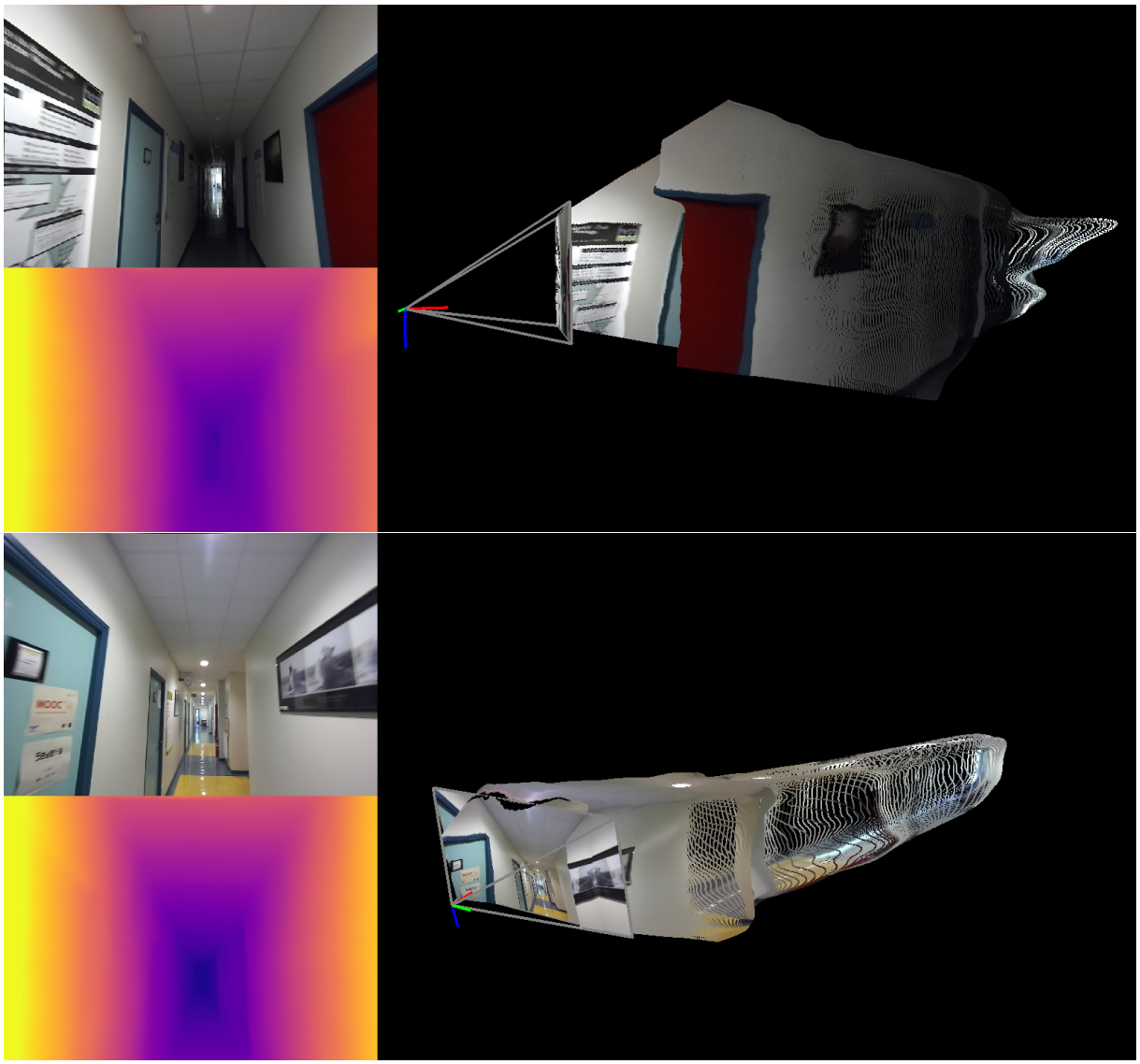


Figure B.31 – ZeroDepth prediction on our data: frame 138 (top) and 157 (bottom).

Figure B.32 – Frame 581: ZeroDepth prediction on our data.

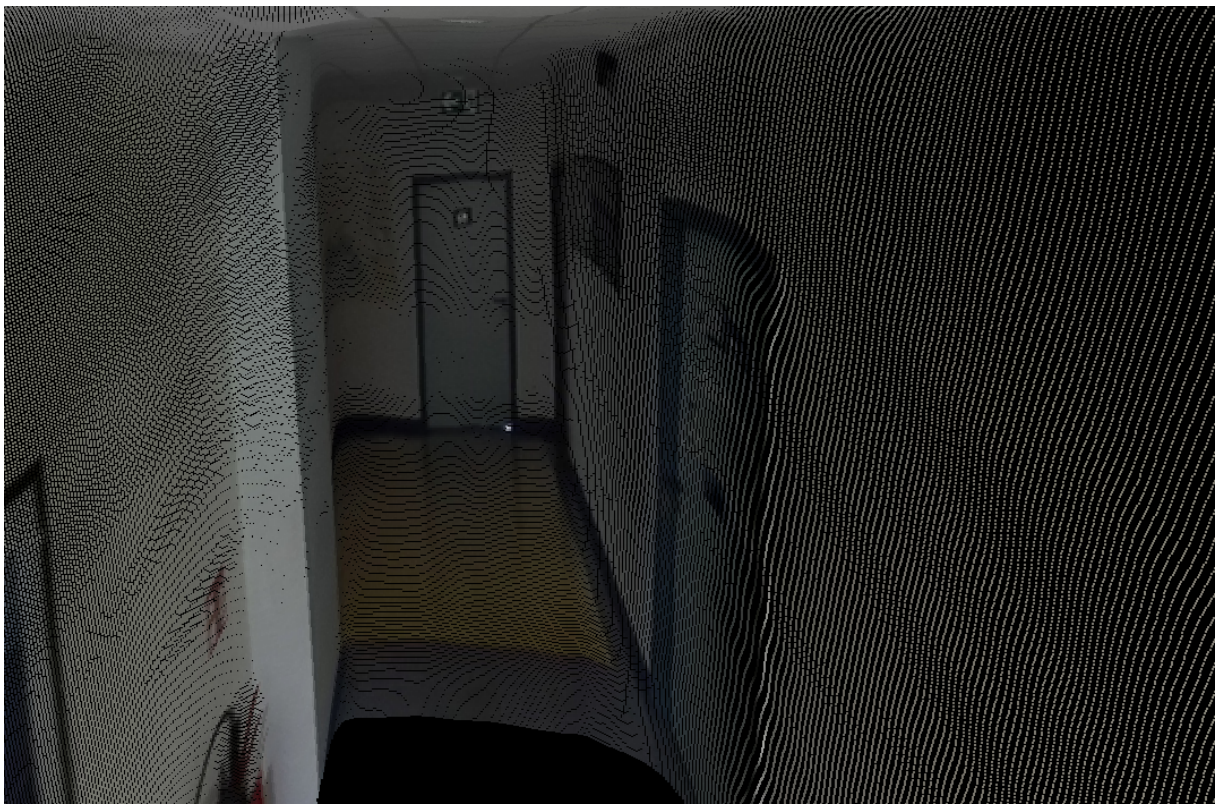


Figure B.33 – Frame 581: Front view of ZeroDepth prediction on our data.

Appendix C

Bibliography

References

[1] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, « Orb-slam3: an accurate open-source library for visual, visual-inertial, and multimap slam », *IEEE Transactions on Robotics*, vol. 37, *6*, pp. 1874–1890, 2021.

[2] V. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers, « Visual-inertial mapping with non-linear factor recovery », *IEEE Robotics and Automation Letters*, vol. 5, *2*, pp. 422–429, 2020.

[3] M. Burri, J. Nikolic, P. Gohl, *et al.*, « The euroc micro aerial vehicle datasets », *The International Journal of Robotics Research (IJRR)*, vol. 35, *10*, pp. 1157–1163, 2016.

[4] 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)*, 2018, pp. 1680–1687. DOI: 10.1109/IROS.2018.8593419.

[5] A. Geiger, P. Lenz, and R. Urtasun, « Are we ready for autonomous driving? the kitti vision benchmark suite », *in Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[6] H. Matsuki, R. Scona, J. Czarnowski, and A. J. Davison, « Codemapping: real-time dense mapping for sparse slam using compact scene representations », *IEEE Robotics and Automation Letters*, vol. 6, *4*, pp. 7105–7112, 2021.

[7] X. Zuo, N. Merrill, W. Li, Y. Liu, M. Pollefeys, and G. Huang, « Codevio: visual-inertial odometry with learned optimizable dense depth », *in 2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 14 382–14 388. DOI: 10.1109/ICRA48506.2021.9560792.

[8] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon, « 3d packing for self-supervised monocular depth estimation », *in IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2485–2494.

[9] V. Guizilini, I. Vasiljevic, D. Chen, R. Ambruș, and A. Gaidon, « Towards zero-shot scale-aware monocular depth estimation », *in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2023, pp. 9233–9243.

[10] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, « Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset », *in 2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6713–6719. DOI: 10.1109/ICRA.2019.8793887.

[11] M. Helmberger, K. Morin, B. Berner, N. Kumar, G. Cioffi, and D. Scaramuzza, « The hilti slam challenge dataset », *IEEE Robotics and Automation Letters*, vol. 7, *3*, pp. 7518–7525, 2022. DOI: 10.1109/LRA.2022.3183759.

[12] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, « Kimera: an open-source library for real-time metric-semantic localization and mapping », *in IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1689–1696.

[13] I. Kostavelis and A. Gasteratos, « Robots in crisis management: a survey », *in Information Systems for Crisis Response and Management in Mediterranean Countries*, I. M. Dokas, N. Bellamine-Ben Saoud, J. Dugdale, and P. Díaz, Eds., Cham: Springer International Publishing, 2017, pp. 43–56, ISBN: 978-3-319-67633-3.

[14] G.-J. M. Kruijff, F. Pirri, M. Gianni, *et al.*, « Rescue robots at earthquake-hit mirandola, italy: a field report », *in IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2012.

[15] A. Restas *et al.*, « Drone applications for supporting disaster management », *World Journal of Engineering and Technology*, vol. 3, *03*, p. 316, 2015.

[16] N. Ruangpayoongsak, H. Roth, and J. Chudoba, « Mobile robots for search and rescue », *in IEEE International Safety, Security and Rescue Rototics, Workshop, 2005.*, 2005, pp. 212–217. DOI: 10.1109/SSRR.2005.1501265.

[17] H. Surmann, T. Kaiser, A. Leinweber, G. Senkowski, D. Slomma, and M. Thurow, « Small commercial uavs for indoor search and rescue missions », *in 2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, 2021, pp. 106–113. DOI: 10.1109/ICARA51699.2021.9376551.

[18] M. Kopuletý and T. Palasiewicz, « Advanced military robots supporting engineer reconnaissance in military operations », *in Modelling and Simulation for Autonomous Systems*, J. Mazal, Ed., Cham: Springer International Publishing, 2018, pp. 285–302, ISBN: 978-3-319-76072-8.

[19] C. Paucar, L. Morales, K. Pinto, *et al.*, « Use of drones for surveillance and reconnaissance of military areas », *in Developments and Advances in Defense and Security*, Á. Rocha and T. Guarda, Eds., Cham: Springer International Publishing, 2018, pp. 119–132, ISBN: 978-3-319-78605-6.

[20] Thales, *Uas100 long range surveillance drone*, `https://www.thalesgroup.com/en/markets/aerospace/drone-solutions/uas100-long-range-surveillance-drone`, 2021.

[21] Thales, *A guide for safety rules for drone operators*, `https://www.thalesgroup.com/en/guide-safety-rules-drone-operators`.

[22] Thales, *Thales scaleflyt*, `https://www.scaleflyt.com`.

[23] A. Fagette, J. S. Lopez Yepez, S. Allogba, *et al.*, « A novel swarm defense end-to-end system for autonomous drone detection, tracking, and neutralization », *in Proceedings of the NATO Symposium SET-315 Research Symposium on Detection, Tracking, ID and Defeat of Small UAVs in Complex Environments*, 2023.

[24] COOPOL, *Capacité d'appui aux opérations de secours et police*, `http://coopol.eurecom.fr/en`, 2016.

[25] TeamAware, *Teamaware*, `https://teamaware.eu`, 2021.

[26] AURORA, *Safe urban air mobility for european citizens*, `https://aurora-uam.eu`, 2020.

[27] ResponDrone, *Situational awareness system for first responders*, `https://respondroneprojec` `com`, 2019.

[28] Thales, *Ground-breaking new role for autonomous drone technology*, `https://www.thalesgroup.com/en/united-kingdom/news/ground-breaking-new-role-autonomous-drone-technology`, Accessed: 2021-01-22, 2021.

[29] L. Brown, R. Clarke, A. Akbari, *et al.*, « The design of prometheus: a reconfigurable uav for subterranean mine inspection », *robotics*, vol. 9, *4*, p. 95, 2020.

[30] G. De Croon and C. De Wagter, « Challenges of autonomous flight in indoor environments », *in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1003–1009. DOI: 10.1109/IROS.2018.8593704.

[31] L. CAMILLI, « Emerging technologies, applications, regulations, and market challenges in the consumer aerial drone industry », *in Conference: San Francisco State College of Business, At San Francisco*, 2015.

[32] S. M. Asaad and H. S. Maghdid, « A comprehensive review of indoor/outdoor localization solutions in iot era: research challenges and future perspectives », *Computer Networks*, vol. 212, p. 109 041, 2022, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2022.109041. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622001918.

[33] N. Shiroma, N. Sato, Y. Chiu, and F. Matsuno, « Study on effective camera images for mobile robot teleoperation », *in RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No.04TH8759)*, 2004, pp. 107–112. DOI: 10.1109/ROMAN.2004.1374738.

[34] K. Sedlmajer, D. Bambu? ek, and V. z. Beran, « Effective remote drone control using augmented virtuality », *in Proceedings of the 3rd International Conference on Computer-Human Interaction Research and Applications*, 2019, pp. 177–182.

[35] J. Thomason, P. Ratsamee, K. Kiyokawa, *et al.*, « Adaptive view management for drone teleoperation in complex 3d structures », *in Proceedings of the 22nd International Conference on Intelligent User Interfaces*, 2017, pp. 419–426.

[36] L. O. Rojas-Perez and J. Martinez-Carranza, « Metric monocular slam and colour segmentation for multiple obstacle avoidance in autonomous flight », *in 2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, 2017, pp. 234–239. DOI: 10.1109/RED-UAS.2017.8101672.

[37] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, « Unmanned aerial vehicles (uavs): collision avoidance systems and approaches », *IEEE Access*, vol. 8, pp. 105 139–105 155, 2020. DOI: 10.1109/ACCESS.2020.3000064.

[38] C. Cadena, L. Carlone, H. Carrillo, *et al.*, « Past, present, and future of simultaneous localization and mapping: toward the robust-perception age », *IEEE Trans-*

actions on Robotics, vol. 32, *6*, pp. 1309–1332, 2016. DOI: 10.1109/TRO.2016.2624754.

[39] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. 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, pp. 1366–1373.

[40] Y. Choe, I. Shim, and M. J. Chung, « Geometric-featured voxel maps for 3d mapping in urban environments », *in 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, pp. 110–115. DOI: 10.1109/SSRR.2011.6106760.

[41] M. Muglikar, Z. Zhang, and D. Scaramuzza, « Voxel map for visual slam », *in 2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4181–4187. DOI: 10.1109/ICRA40945.2020.9197357.

[42] J. Rennie, *Drone types: multi-rotor vs fixed-wing vs single rotor vs hybrid vtol*, https://www.auav.com.au/articles/drone-types, Nov. 2016.

[43] A. Bachrach, R. He, and N. Roy, « Autonomous flight in unknown indoor environments », *International Journal of Micro Air Vehicles*, vol. 1, *4*, pp. 217–228, 2009.

[44] 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)*, 2018, pp. 2502–2509. DOI: 10.1109/ICRA.2018.8460664.

[45] S. Aldegheri, N. Bombieri, D. D. Bloisi, and A. 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)*, 2019, pp. 5370–5375. DOI: 10.1109/IROS40897.2019.8967814.

[46] J. Jeon, S. Jung, E. Lee, D. Choi, and H. Myung, « Run your visual-inertial odometry on nvidia jetson: benchmark tests on a micro aerial vehicle », *IEEE Robotics and Automation Letters*, vol. 6, *3*, pp. 5332–5339, 2021. DOI: 10.1109/LRA.2021.3075141.

[47] M. Zaffar, S. Ehsan, R. Stolkin, and K. M. Maier, « Sensors, slam and long-term autonomy: a review », *in 2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2018, pp. 285–290. DOI: 10.1109/AHS.2018.8541483.

[48] J. Cheng, L. Zhang, Q. Chen, X. Hu, and J. Cai, « A review of visual slam methods for autonomous driving vehicles », *Engineering Applications of Artificial Intelligence*, vol. 114, p. 104 992, 2022, ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2022.104992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622001853.

[49] I. Abaspur Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, « A survey of state-of-the-art on visual slam », *Expert Systems with Applications*, vol. 205, p. 117 734, 2022, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.117734. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422010156.

[50] D. Scaramuzza and F. Fraundorfer, « Visual odometry [tutorial] », *IEEE Robotics & Automation Magazine*, vol. 18, *4*, pp. 80–92, 2011. DOI: 10.1109/MRA.2011.943233.

[51] E. P. Herrera-Granda, J. C. Torres-Cantero, and D. H. Peluffo-Ordoñez, « Monocular visual slam, visual odometry, and structure from motion methods applied to 3d reconstruction: a comprehensive survey », *Visual Odometry, and Structure from Motion Methods Applied to 3D Reconstruction: A Comprehensive Survey*,

[52] S. Jiang, C. Jiang, and W. Jiang, « Efficient structure from motion for large-scale uav images: a review and a comparison of sfm tools », *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 167, pp. 230–251, 2020, ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2020.04.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271620301131.

[53] H. Lim, D. Kim, B. Kim, and H. Myung, « Adalio: robust adaptive lidar-inertial odometry in degenerate indoor environments », *in 2023 20th International Conference on Ubiquitous Robots (UR)*, 2023, pp. 48–53. DOI: 10.1109/UR57808.2023.10202252.

[54] T. Shan, B. Englot, C. Ratti, and D. Rus, « Lvi-sam: tightly-coupled lidar-visual-inertial odometry via smoothing and mapping », *in 2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5692–5698. DOI: 10.1109/ICRA48506.2021.9561996.

[55] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, « Fast-lio2: fast direct lidar-inertial odometry », *IEEE Transactions on Robotics*, vol. 38, *4*, pp. 2053–2073, 2022. DOI: 10.1109/TRO.2022.3141876.

[56] R. A. Newcombe, S. Izadi, O. Hilliges, *et al.*, « Kinectfusion: real-time dense surface mapping and tracking », *in 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.

[57] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison, « Elasticfusion: dense slam without a pose graph », *Robotics: Science and Systems XI*, 2015.

[58] K. Huang, S. Zhang, J. Zhang, and D. Tao, « Event-based simultaneous localization and mapping: a comprehensive survey », *arXiv preprint arXiv:2304.09793*, 2023.

[59] J. Hidalgo-Carrió, D. Gehrig, and D. Scaramuzza, « Learning monocular dense depth from events », *in 2020 International Conference on 3D Vision (3DV)*, 2020, pp. 534–542. DOI: 10.1109/3DV50981.2020.00063.

[60] Y. Zhou, G. Gallego, H. Rebecq, L. Kneip, H. Li, and D. Scaramuzza, « Semi-dense 3d reconstruction with a stereo event camera », *in Proceedings of the European Conference on Computer Vision (ECCV)*, Sep. 2018.

[61] M. Bujanca, X. Shi, M. Spear, P. Zhao, B. Lennox, and M. Luján, « Robust slam systems: are we there yet? », *in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5320–5327. DOI: 10.1109/IROS51168.2021.9636814.

[62] D. Prokhorov, D. Zhukov, O. Barinova, K. Anton, and A. Vorontsova, « Measuring robustness of visual slam », *in 2019 16th International Conference on Machine Vision Applications (MVA)*, 2019, pp. 1–6. DOI: 10.23919/MVA.2019.8758020.

[63] X. Dong, M. A. Garratt, S. G. Anavatti, and H. A. Abbass, « Towards real-time monocular depth estimation for robotics: a survey », *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, *10*, pp. 16 940–16 961, 2022. DOI: 10.1109/TITS.2022.3160741.

[64] K. Tateno, F. Tombari, I. Laina, and N. Navab, « CNN-SLAM: real-time dense monocular SLAM with learned depth prediction », *CoRR*, vol. abs/1704.03489, 2017. arXiv: 1704.03489. [Online]. Available: http://arxiv.org/abs/1704.03489.

[65] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, « Deepfactors: real-time probabilistic dense monocular slam », *IEEE Robotics and Automation Letters*, vol. 5, *2*, pp. 721–728, 2020.

[66] S. Y. Loo, S. Mashohor, S. H. Tang, and H. Zhang, « Deeprelativefusion: dense monocular slam using single-image relative depth prediction », *in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 6641–6648.

[67] Y. Habib, P. Papadakis, A. Fagette, C. L. Barz, T. Gonçalves, and C. Buche, « From sparse SLAM to dense mapping for UAV autonomous navigation », *in Geospatial Informatics XIII*, K. Palaniappan, G. Seetharaman, and J. D. Harguess, Eds., International Society for Optics and Photonics, vol. 12525, SPIE, 2023, p. 125250C. DOI: 10.1117/12.2663706.

[68] Y. Habib, P. Papadakis, C. Le Barz, A. Fagette, T. Gonçalves, and C. Buche, « Densifying slam for uav navigation by fusion of monocular depth prediction », *in 2023 9th International Conference on Automation, Robotics and Applications (ICARA)*, 2023, pp. 225–229. DOI: 10.1109/ICARA56516.2023.10125712.

[69] Y. Ma, S. Soatto, and J. Košecká, *An invitation to 3-d vision: from images to geometric models*. Springer, vol. 26.

[70] J. Kannala and S. Brandt, « A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, *8*, pp. 1335–1340, 2006. DOI: 10.1109/TPAMI.2006.153.

[71] V. Usenko, N. Demmel, and D. Cremers, « The double sphere camera model », *in 2018 International Conference on 3D Vision (3DV)*, 2018, pp. 552–560. DOI: 10.1109/3DV.2018.00069.

[72] D. G. Lowe, « Distinctive image features from scale-invariant keypoints », *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[73] E. Rosten and T. Drummond, « Machine learning for high-speed corner detection », *in Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, Springer, 2006, pp. 430–443.

[74] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, « Orb: an efficient alternative to sift or surf », *in 2011 International Conference on Computer Vision*, 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[75] H. C. Longuet-Higgins, « A computer algorithm for reconstructing a scene from two projections », *Nature*, vol. 293, *5828*, pp. 133–135, 1981.

[76] D. Nister, « An efficient solution to the five-point relative pose problem », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, *6*, pp. 756–770, 2004. DOI: 10.1109/TPAMI.2004.17.

[77] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, « Extending kalibr: calibrating the extrinsics of multiple imus and of individual axes », *in 2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4304–4311. DOI: 10.1109/ICRA.2016.7487628.

[78] P. Furgale, J. Rehder, and R. Siegwart, « Unified temporal and spatial calibration for multi-sensor systems », *in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1280–1286. DOI: 10.1109/IROS.2013.6696514.

[79] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, « On-manifold preintegration for real-time visual–inertial odometry », *IEEE Transactions on Robotics*, vol. 33, *1*, pp. 1–21, 2017. DOI: 10.1109/TRO.2016.2597321.

[80] A. Jurić, F. Kendeš, I. Marković, and I. Petrović, « A comparison of graph optimization approaches for pose estimation in slam », *in 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, 2021, pp. 1113–1118. DOI: 10.23919/MIPRO52101.2021.9596721.

[81] H. Strasdat, J. M. Montiel, and A. J. Davison, « Visual slam: why filter? », *Image and Vision Computing*, vol. 30, *2*, pp. 65–77, 2012.

[82] H. Durrant-Whyte and T. Bailey, « Simultaneous localization and mapping: part i », *IEEE Robotics & Automation Magazine*, vol. 13, *2*, pp. 99–110, 2006. DOI: 10.1109/MRA.2006.1638022.

[83] F. Fraundorfer and D. Scaramuzza, « Visual odometry : part ii: matching, robustness, optimization, and applications », *IEEE Robotics & Automation Magazine*, vol. 19, *2*, pp. 78–90, 2012. DOI: 10.1109/MRA.2012.2182810.

[84] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, « Isam2: incremental smoothing and mapping using the bayes tree », *The International Journal of Robotics Research*, vol. 31, *2*, pp. 216–235, 2012. DOI: 10.1177/0278364911430419. eprint: https://doi.org/10.1177/0278364911430419.

[85] F. Dellaert, « Factor graphs and gtsam: a hands-on introduction », GT RIM, Tech. Rep. GT-RIM-CP&R-2012-002, Sep. 2012. [Online]. Available: https://research.cc.gatech.edu/borg/sites/edu.borg/files/downloads/gtsam.pdf.

[86] F. Dellaert and G. Contributors, *Borglab/gtsam*, version 4.2a8, May 2022. DOI: 10.5281/zenodo.5794541. [Online]. Available: https://github.com/borglab/gtsam.

[87] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, « Orb-slam: a versatile and accurate monocular slam system », *IEEE Transactions on Robotics*, vol. 31, *5*, pp. 1147–1163, 2015. DOI: 10.1109/TRO.2015.2463671.

[88] G. Xu, Q. Zhang, and N. Li, « Study on the method of slam initialization for monocular vision », *IOP Conference Series: Materials Science and Engineering*, vol. 569, *5*, p. 052 085, Jul. 2019. DOI: 10.1088/1757-899X/569/5/052085. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/569/5/052085.

[89] G. Huang, « Visual-inertial navigation: a concise review », *in 2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9572–9582. DOI: 10.1109/ICRA.2019.8793604.

[90] Meta AI, *Powered by ai: oculus insight*, https://ai.meta.com/blog/powered-by-ai-oculus-insight/, 2019. (visited on 11/08/2022).

[91] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, « An overview to visual odometry and visual slam: applications to mobile robotics », *Intelligent Industrial Systems*, vol. 1, *4*, pp. 289–311, 2015.

[92] G. Klein and D. Murray, « Parallel tracking and mapping for small ar workspaces », *in 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852.

[93] 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*, 2011, pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513.

[94] J. Engel, T. Schöps, and D. Cremers, « Lsd-slam: large-scale direct monocular slam », *in European conference on computer vision*, Springer, 2014, pp. 834–849.

[95] M. Yokozuka, S. Oishi, S. Thompson, and A. Banno, « Vitamin-e: visual tracking and mapping with extremely dense feature points », *in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9633–9642. DOI: 10.1109/CVPR.2019.00987.

[96] R. Mur-Artal and J. D. Tardós, « Visual-inertial monocular slam with map reuse », *IEEE Robotics and Automation Letters*, vol. 2, *2*, pp. 796–803, 2017. DOI: 10.1109/LRA.2017.2653359.

[97] T. Qin, P. Li, and S. Shen, « Vins-mono: A robust and versatile monocular visual-inertial state estimator », *CoRR*, vol. abs/1708.03852, 2017. arXiv: 1708.03852. [Online]. Available: http://arxiv.org/abs/1708.03852.

[98] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, « Keyframe-based visual–inertial odometry using nonlinear optimization », *The International Journal of Robotics Research*, vol. 34, *3*, pp. 314–334, 2015. DOI: 10.1177/0278364914554813. eprint: https://doi.org/10.1177/0278364914554813. [Online]. Available: https://doi.org/10.1177/0278364914554813.

[99] 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*, vol. 33, *5*, pp. 1255–1262, 2017. DOI: 10.1109/TRO.2017.2705103.

[100] T. Qin, S. Cao, J. Pan, P. Li, and S. Shen, *Vins-fusion: an optimization-based multi-sensor state estimator*, https://github.com/HKUST-Aerial-Robotics/VINS-Fusion, 2019.

[101] M. Ferrera, A. Eudes, J. Moras, M. Sanfourche, and G. Le Besnerais, « Ov^{2}slam: a fully online and versatile visual slam for real-time applications », *IEEE Robotics and Automation Letters*, vol. 6, *2*, pp. 1399–1406, 2021. DOI: 10.1109/LRA.2021.3058069.

[102] I. Cvišić, I. Marković, and I. Petrović, « Soft2: stereo visual odometry for road vehicles based on a point-to-epipolar-line metric », *IEEE Transactions on Robotics*, vol. 39, *1*, pp. 273–288, 2023. DOI: 10.1109/TRO.2022.3188121.

[103] J. Leonard and H. 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*, 1991, 1442–1447 vol.3. DOI: 10.1109/IROS.1991.174711.

[104] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, « Fastslam: a factored solution to the simultaneous localization and mapping problem », *Aaai/iaai*, vol. 593598, 2002.

[105] T. Bailey and H. Durrant-Whyte, « Simultaneous localization and mapping (slam): part ii », *IEEE Robotics & Automation Magazine*, vol. 13, *3*, pp. 108–117, 2006. DOI: 10.1109/MRA.2006.1678144.

[106] J. Shi and Tomasi, « Good features to track », *in 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[107] D. Scaramuzza and Z. Zhang, « Visual-inertial odometry of aerial robots », *CoRR*, vol. abs/1906.03289, 2019. arXiv: 1906.03289. [Online]. Available: http://arxiv.org/abs/1906.03289.

[108] D. Galvez-López and J. D. Tardos, « Bags of binary words for fast place recognition in image sequences », *IEEE Transactions on Robotics*, vol. 28, *5*, pp. 1188–1197, 2012. DOI: 10.1109/TRO.2012.2197158.

[109] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, « Svo: semidirect visual odometry for monocular and multicamera systems », *IEEE Transactions on Robotics*, vol. 33, *2*, pp. 249–265, 2017. DOI: 10.1109/TRO.2016.2623335.

[110] R. Elvira, J. D. Tardós, and J. Montiel, « Orbslam-atlas: a robust and accurate multi-map system », *in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6253–6259.

[111] W. E. Lorensen and H. E. Cline, « Marching cubes: a high resolution 3d surface construction algorithm », *ACM siggraph computer graphics*, vol. 21, *4*, pp. 163–169, 1987.

[112] S. Li, D. Zhang, Y. Xian, B. Li, T. Zhang, and C. Zhong, « Overview of deep learning application on visual slam », *Displays*, vol. 74, p. 102 298, 2022, ISSN: 0141-9382. DOI: https://doi.org/10.1016/j.displa.2022.102298. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141938222001160.

[113] R. Li, S. Wang, and D. Gu, « Ongoing evolution of visual slam from geometry to deep learning: challenges and opportunities », *Cognitive Computation*, vol. 10, pp. 875–889, 2018.

[114] A. Rosinol, J. J. Leonard, and L. Carlone, *Nerf-slam: real-time dense monocular slam with neural radiance fields*, 2022. arXiv: 2210.13641 [cs.CV].

[115] A. Rosinol, J. J. Leonard, and L. Carlone, « Probabilistic volumetric fusion for dense monocular slam », *arXiv preprint arXiv:2210.01276*, 2022.

[116] Z. Teed and J. Deng, « DROID-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras », *CoRR*, vol. abs/2108.10869, 2021. arXiv: 2108.10869. [Online]. Available: https://arxiv.org/abs/2108.10869.

[117] L. Koestler, N. Yang, N. Zeller, and D. Cremers, « Tandem: tracking and dense mapping in real-time using deep multi-view stereo », *in Conference on Robot Learning*, PMLR, 2022, pp. 34–45.

[118] J. M. Torres Cámara, *Map slammer. densifying scattered kslam 3d maps with estimated depth*, 2019.

[119] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, « Codeslam-learning a compact, optimisable representation for dense visual slam », *in IEEE conference on computer vision and pattern recognition*, 2018, pp. 2560–2568.

[120] B. Ummenhofer, H. Zhou, J. Uhrig, *et al.*, « Demon: depth and motion network for learning monocular stereo », *in IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 5038–5047.

[121] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, « Towards robust monocular depth estimation: mixing datasets for zero-shot cross-dataset transfer », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, *3*, pp. 1623–1637, 2022. DOI: 10.1109/TPAMI.2020.3019967.

[122] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, « Openvins: a research platform for visual-inertial estimation », *in 2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4666–4672. DOI: 10.1109/ICRA40945.2020.9196524.

[123] Z. Teed and J. Deng, « Raft: recurrent all-pairs field transforms for optical flow », *in Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 402–419, ISBN: 978-3-030-58536-5.

[124] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, « Nerf: representing scenes as neural radiance fields for view synthesis », *in Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 405–421.

[125] T. Müller, A. Evans, C. Schied, and A. Keller, « Instant neural graphics primitives with a multiresolution hash encoding », *ACM Trans. Graph.*, vol. 41, *4*, Jul. 2022, ISSN: 0730-0301. DOI: 10.1145/3528223.3530127. [Online]. Available: https://doi.org/10.1145/3528223.3530127.

[126] S.-F. Chng, S. Ramasinghe, J. Sherrah, and S. Lucey, *Garf: gaussian activated radiance fields for high fidelity reconstruction and pose estimation*, 2022. arXiv: 2204.05735 [cs.CV].

[127] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, « Barf: bundle-adjusting neural radiance fields », *in 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5721–5731. DOI: 10.1109/ICCV48922.2021.00569.

[128] Y. Ming, X. Meng, C. Fan, and H. Yu, « Deep learning for monocular depth estimation: a review », *Neurocomputing*, vol. 438, pp. 14–33, 2021.

[129] D. Eigen, C. Puhrsch, and R. Fergus, « Depth map prediction from a single image using a multi-scale deep network », *Advances in neural information processing systems*, vol. 27, pp. 2366–2374, 2014.

[130] L. Wang, Y. Wang, L. Wang, Y. Zhan, Y. Wang, and H. Lu, « Can scale-consistent monocular depth be learned in a self-supervised scale-invariant manner? », *in 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 12 707–12 716. DOI: 10.1109/ICCV48922.2021.01249.

[131] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, « Deep ordinal regression network for monocular depth estimation », *in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2002–2011. DOI: 10.1109/CVPR.2018.00214.

[132] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, « From big to small: multi-scale local planar guidance for monocular depth estimation », *arXiv preprint arXiv:1907.10326*, 2019.

[133] R. Ranftl, A. Bochkovskiy, and V. Koltun, « Vision transformers for dense prediction », *in 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 12 159–12 168. DOI: 10.1109/ICCV48922.2021.01196.

[134] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, « Transformers in vision: a survey », *ACM Comput. Surv.*, vol. 54, *10s*, Sep. 2022, ISSN: 0360-0300. DOI: 10.1145/3505244. [Online]. Available: https://doi.org/10.1145/3505244.

[135] S. Farooq Bhat, I. Alhashim, and P. Wonka, « Adabins: depth estimation using adaptive bins », *in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 4008–4017. DOI: 10.1109/CVPR46437.2021.00400.

[136] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, *Zoedepth: zero-shot transfer by combining relative and metric depth*, 2023. DOI: 10.48550/ARXIV.2302.12288. [Online]. Available: https://arxiv.org/abs/2302.12288.

[137] R. Garg, V. K. Bg, G. Carneiro, and I. Reid, « Unsupervised cnn for single view depth estimation: geometry to the rescue », *in European conference on computer vision*, Springer, 2016, pp. 740–756.

[138] C. Godard, O. Mac Aodha, and G. J. Brostow, « Unsupervised monocular depth estimation with left-right consistency », *in IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 270–279.

[139] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, « Unsupervised learning of depth and ego-motion from video », *in IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 1851–1858.

[140] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, « Digging into self-supervised monocular depth estimation », *in 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3827–3837. DOI: 10.1109/ICCV.2019.00393.

[141] H. Zhou, D. Greenwood, and S. Taylor, « Self-supervised monocular depth estimation with internal feature fusion », *in British Machine Vision Conference (BMVC)*, 2021.

[142] J. Bae, S. Moon, and S. Im, « Deep digging into the generalization of self-supervised monocular depth estimation », *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, *1*, pp. 187–196, Jun. 2023. DOI: 10.1609/aaai.v37i1.25090. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/25090.

[143] C. Zhao, Y. Zhang, M. Poggi, *et al.*, « Monovit: self-supervised monocular depth estimation with a vision transformer », *in 2022 International Conference on 3D Vision (3DV)*, 2022, pp. 668–678. DOI: 10.1109/3DV57658.2022.00077.

[144] 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)*, 2018, pp. 7244–7251. DOI: 10.1109/IROS.2018.8593941.

[145] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, « A benchmark for the evaluation of rgb-d slam systems », *in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.

[146] B. K. P. Horn, « Closed-form solution of absolute orientation using unit quaternions », *J. Opt. Soc. Am. A*, vol. 4, *4*, pp. 629–642, Apr. 1987. DOI: 10.1364/JOSAA.4.000629. [Online]. Available: https://opg.optica.org/josaa/abstract.cfm?URI=josaa-4-4-629.

[147] M. Salas, Y. Latif, and I. D. Reid, « Trajectory alignment and evaluation in slam: horns method vs alignment on the manifold ».

[148] S. Umeyama, « Least-squares estimation of transformation parameters between two point patterns », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, *4*, pp. 376–380, 1991. DOI: 10.1109/34.88573.

[149] D. Q. Huynh, « Metrics for 3d rotations: comparison and analysis », *Journal of Mathematical Imaging and Vision*, vol. 35, pp. 155–164, 2009.

[150] A. Rosinol, « Densifying sparse vio: a mesh-based approach using structural regularities », en, M.S. thesis, ETH Zurich, Zurich; Cambridge, MA, 2018. DOI: 10.3929/ethz-b-000297645.

[151] Cloudcompare.org, *Cloudcompare - open source project*, https://www.cloudcompare.org, 2019.

[152] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, « Indoor segmentation and support inference from rgbd images », *in Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, Springer, 2012, pp. 746–760.

[153] M. Cordts, M. Omran, S. Ramos, *et al.*, « The cityscapes dataset for semantic urban scene understanding », *in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223. DOI: 10.1109/CVPR.2016.350.

[154] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, « The multivehicle stereo event camera dataset: an event camera dataset for 3d perception », *IEEE Robotics and Automation Letters*, vol. 3, *3*, pp. 2032–2039, 2018. DOI: 10.1109/LRA.2018.2800793.

[155] T. Schöps, T. Sattler, and M. Pollefeys, « Bad slam: bundle adjusted direct rgb-d slam », *in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 134–144. DOI: 10.1109/CVPR.2019.00022.

[156] A. Antonini, W. Guerra, V. Murali, T. Sayre-McCord, and S. Karaman, « The blackbird dataset: a large-scale dataset for uav perception in aggressive flight », *in 2018 International Symposium on Experimental Robotics (ISER)*, 2018. DOI: 10.1007/978-3-030-33950-0_12. [Online]. Available: https://doi.org/10.1007/978-3-030-33950-0_12.

[157] H. Caesar, V. Bankiti, A. H. Lang, *et al.*, « Nuscenes: a multimodal dataset for autonomous driving », *in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 618–11 628. DOI: 10.1109/CVPR42600.2020.01164.

[158] P. Sun, H. Kretzschmar, X. Dotiwalla, *et al.*, « Scalability in perception for autonomous driving: waymo open dataset », *in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2443–2451. DOI: 10.1109/CVPR42600.2020.00252.

[159] Y. Liao, J. Xie, and A. Geiger, « KITTI-360: a novel dataset and benchmarks for urban scene understanding in 2d and 3d », *Pattern Analysis and Machine Intelligence (PAMI)*, 2022.

[160] W. Wang, D. Zhu, X. Wang, *et al.*, « Tartanair: a dataset to push the limits of visual slam », *in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4909–4916. DOI: 10.1109/IROS45743.2020.9341801.

[161] A. Eftekhar, A. Sax, J. Malik, and A. Zamir, « Omnidata: a scalable pipeline for making multi-task mid-level vision datasets from 3d scans », *in 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 10 766–10 776. DOI: 10.1109/ICCV48922.2021.01061.

[162] L. Zhang, M. Helmberger, L. F. T. Fu, *et al.*, « Hilti-oxford dataset: a millimeter-accurate benchmark for simultaneous localization and mapping », *IEEE Robotics and Automation Letters*, vol. 8, *1*, pp. 408–415, 2023. DOI: 10.1109/LRA.2022.3226077.

[163] S. Shah, D. Dey, C. Lovett, and A. Kapoor, « Airsim: high-fidelity visual and physical simulation for autonomous vehicles », *in Field and Service Robotics*, M. Hutter and R. Siegwart, Eds., Cham: Springer International Publishing, 2018, pp. 621–635, ISBN: 978-3-319-67361-5.

[164] NVIDIA, *Jetson modules*, https://developer.nvidia.com/embedded/jetson-modules, Accessed: 2023-11-30.

[165] A. Millane, H. Oleynikova, E. Wirbel, *et al.*, *Nvblox: gpu-accelerated incremental signed distance field mapping*, 2023. arXiv: 2311.00626 [cs.RO].

[166] M. Grupp, *Evo: python package for the evaluation of odometry and slam.* https://github.com/MichaelGrupp/evo, 2017.

[167] ETH Zurich - Autonomous Systems Lab, *Kalibr Wiki*, https://github.com/ethz-asl/kalibr/wiki, Online, 2014.

[168] J. Hu, C. Bao, M. Ozay, *et al.*, « Deep depth completion from extremely sparse data: a survey », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, *7*, pp. 8244–8264, 2023. DOI: 10.1109/TPAMI.2022.3229090.

[169] T. R. Institute, *Camviz*, https://github.com/TRI-ML/camviz, 2021.

[170] Z. Zhu, S. Peng, V. Larsson, *et al.*, « Nice-slam: neural implicit scalable encoding for slam », *in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 776–12 786. DOI: 10.1109/CVPR52688.2022.01245.

[171] C.-M. Chung, Y.-C. Tseng, Y.-C. Hsu, *et al.*, « Orbeez-slam: a real-time monocular visual slam with orb features and nerf-realized mapping », *in 2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9400–9406. DOI: 10.1109/ICRA48891.2023.10160950.

[172] C. V. Nguyen, S. Izadi, and D. Lovell, « Modeling kinect sensor noise for improved 3d reconstruction and tracking », *in 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 2012, pp. 524–530. DOI: 10.1109/3DIMPVT.2012.84.

[173] B. Curless and M. Levoy, « A volumetric method for building complex models from range images », *in Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[174] ETH Zurich - Autonomous Systems Lab, *Voxblox*, Online, 2016. [Online]. Available: https://github.com/ethz-asl/voxblox.

[175] S. Aravecchia, M. Clausel, and C. Pradalier, « Comparing metrics for evaluating 3d reconstruction quality in large-scale environment mapping », 2023.

[176] ETH Zurich - Autonomous Systems Lab, *image_undistort: dense_stereo_node*, https://github.com/ethz-asl/image_undistort, Online, 2016.

[177] E. P. Örnek, S. Mudgal, J. Wald, Y. Wang, N. Navab, and F. Tombari, « From 2d to 3d: re-thinking benchmarking of monocular depth prediction », *arXiv preprint arXiv:2203.08122*, 2022.

[178] W. du, H. Chen, H. Yang, and Y. Zhang, « Depth completion using geometry-aware embedding », *in 2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8680–8686. DOI: 10.1109/ICRA46639.2022.9811556.

[179] S. Weder, J. Schönberger, M. Pollefeys, and M. R. Oswald, « Routedfusion: learning real-time depth map fusion », *in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4886–4896. DOI: 10.1109/CVPR42600.2020.00494.

[180] S. Weder, J. L. Schönberger, M. Pollefeys, and M. R. Oswald, « Neuralfusion: online depth fusion in latent space », *in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3161–3171. DOI: 10.1109/CVPR46437.2021.00318.

[181] Y. Habib, P. Papadakis, C. Buche, C. Le Barz, and A. Fagette, *Dense, metric and real-time 3d reconstruction for autonomous drone navigation*, Journées des Jeunes Chercheurs en Robotique (JJCR 2021), Poster, Oct. 2021. [Online]. Available: `https://hal.science/hal-04184995`.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Densification du SLAM monoculaire pour la cartographie 3D et la navigation autonome de drone

Mot clés : prédiction de profondeur, cartographie métrique, apprentissage profond, IA embarqué

Résumé : Les drones aériens sont essentiels dans les missions de recherche et de sauvetage car ils permettent une reconnaissance rapide de la zone de la mission, tel qu'un bâtiment effondré. La cartographie 3D dense et métrique en temps réel est cruciale pour capturer la structure de l'environnement et permettre une navigation autonome.

L'approche privilégiée pour cette tâche consiste à utiliser du SLAM (Simultaneous Localization and Mapping) à partir d'une caméra monoculaire synchronisée avec une centrale inertielle (IMU). Les algorithmes à l'état de l'art maximisent l'efficacité en triangulant un nombre minimum de points, construisant ainsi un nuage de points 3D épars. Quelques travaux traitent de la densification du SLAM monoculaire, généralement en utilisant des réseaux neuronaux profonds pour prédire une carte de profondeur dense à partir d'une seule image. La plupart ne sont pas métriques ou sont trop complexes pour être utilisés en embarqué.

Dans cette thèse, nous identifions une méthode de SLAM monoculaire à l'état de l'art et l'évaluons dans des conditions difficiles pour les drones. Nous présentons une architecture fonctionnelle pour densifier le SLAM monoculaire en appliquant la prédiction de profondeur monoculaire pour construire une carte dense et métrique en voxels 3D.L'utilisation de voxels permet une construction et une maintenance efficaces de la carte par projection de rayons, et permet la fusion volumétrique multi-vues. Enfin, nous proposons une procédure de récupération d'échelle qui utilise les estimations de profondeur éparses et métriques du SLAM pour affiner les cartes de profondeur denses prédites. Notre approche a été évaluée sur des benchmarks conventionnels et montre des résultats prometteurs pour des applications pratiques.

Title: Monocular SLAM densification for 3D mapping and autonomous drone navigation

Keywords: depth prediction, metric mapping, deep learning, embedded AI

Abstract: Aerial drones are essential in search and rescue missions as they provide fast reconnaissance of the mission area, such as a collapsed building. Creating a dense and metric 3D map in real-time is crucial to capture the structure of the environment and enable autonomous navigation.

The recommended approach for this task is to use Simultaneous Localization and Mapping (SLAM) from a monocular camera synchronized with an Inertial Measurement Unit (IMU). Current state-of-the-art algorithms maximize efficiency by triangulating a minimum number of points, resulting in a sparse 3D point cloud. Few works address monocular SLAM densification, typically by using deep neural networks to predict a dense depth map from a single image. Most are not metric or are too complex for use in embedded applications.

In this thesis, we identify and evaluate a state-of-the-art monocular SLAM baseline under challenging drone conditions. We present a practical pipeline for densifying monocular SLAM by applying monocular depth prediction to construct a dense and metric 3D voxel map. Using voxels allows the efficient construction and maintenance of the map through raycasting, and allows for volumetric multi-view fusion. Finally, we propose a scale recovery procedure that uses the sparse and metric depth estimates of SLAM to refine the predicted dense depth maps. Our approach has been evaluated on conventional benchmarks and shows promising results for practical applications.