Intelligent Embedded Camera for Robust Object Tracking on Mobile Platform

Imane Salhi

UNIVERSITY Gustave Eiffel

Mathematics and Information Sciences (MSTIC)

Doctoral Thesis

Computer Science, Automation and Signal Processing

Imane SALHI

Intelligent Embedded Camera for Robust Object Tracking on Mobile Platform

Thesis supervised by Valérie Gouet-Brunet

Defended on 14/02/2021

Jury :

Michèle Gouiffès	Université Paris Saclay, LIMSI	Rapporteur
Frédéric Chausse	Université Clermont Auvergne, Institut Pascal	Rapporteur
Erwan Piriou	CEA – List	Examiner
Valérie Gouet-Brunet	IGN-UGE, LASTIG	Thesis director
Martyna Poreba	IGN-UGE, LASTIG	Supervisor
Maroun Ojail	CEA – List	Supervisor

UNIVERSITY GUSTAVE EIFFEL

Abstract

Mathematics, Information and Communication Sciences and Technologies (MSTIC)

Doctor of Philosophy

Intelligent Embedded Camera for Robust Object Tracking on Mobile Platform

by Imane SALHI

The aim of this study is to analyze, compare and retain the most relevant tracking methods likely to respect the constraints of embedded systems, such as Micro Aerial Vehicles (MAVs), Unmanned Aerial Vehicles (UAVs) and intelligent glasses, in order to find a new robust embedded tracking system. A typical VINS consists of a monocular camera that provides visual data (frames), and a low-cost Inertial Measurement Unit (IMU), a Micro-Electro-Mechanical System (MEMS) that measures inertial data. This combination is very successful in the system navigation field thanks to the advantages that these sensors provide, mainly in terms of accuracy, cost and reactivity. Over the last decade, various sufficiently accurate tracking algorithms and Visual Inertial Navigation Systems (VINS) have been developed, however, they require greater computational resources. In contrast, embedded systems are characterized by their high integration constraints and limited resources. Thus,in this thesis, a solution for embedded architecture, relaying on efficient algorithms and providing less computational load, is proposed.

First, relevant tracking algorithms are studied focusing on their accuracy, robustness, and computational complexity. In parallel, numerous recent embedded tracking computation architectures are also discussed. Then, our robust visual-inertial tracking approach, called: "Context Adaptive Visual Inertial SLAM", is introduced. It alternates between visual KLT-ORB and EKF Visual-Inertial tracking, according to the navigation context, thanks to the proposed execution control module. The latter uses several parameters concerning the scene's appearance, the motion types, *etc*. Consequently, tracking continuity, robustness and accuracy are improved, even in difficult conditions. Moreover, our proposal is suited to embedded systems integration, given the low algorithms computational complexity and the implemented PoIs management leading to decrease the number of PoIs as well as the occurrences of their detection. All our experiments and tests was performed using the different EuRoC dataset sequences.

Keywords. Visual Tracking, Visual-Inertial Tracking, Robust Tracking, Simultaneous Localization And Mapping (SLAM), Odometry, Visual-Inertial Navigation, Multi-sensor Systems, Embedded Systems, Co-design, Fusion, IMU/Camera coupling, Loose Coupling, Tight Coupling.

UNIVERSITE GUSTAVE EIFFEL

Résumé

Mathématiques et Sciences et Technologies de l'Information et de la Communication (MSTIC)

Docteur en Philosophie

Caméra Intelligente Embarquée pour le Suivi Robuste d'Objet sur Plateforme Mobile

par Imane SALHI

Le suivi visuel-inertiel est une thématique d'actualité, difficile à traiter, notamment lorsqu'il s'agit de respecter les contraintes des systèmes embarqués, comme dans les drones autonomes (*Unmanned Aerial Vehicles (UAVs)*). Les questions relatives à la miniaturisation, la portabilité et la communication des systèmes électroniques s'inscrivent dans des problématiques actuelles en matière d'avancée technologique. Pour répondre de manière efficace à ces problématiques, il est nécessaire d'envisager des traitements complexes et des implémentations sur des supports contraignants en termes d'intégration et de consommation d'énergie, tels que les micro-véhicules aériens (*MAVs*), les lunettes et les caméras intelligentes.

Au cours de cette dernière décennie, différents algorithmes performants de suivi ont été développés. En revanche, ils nécessitent des ressources calculatoires conséquentes, compte tenu des différentes formes d'utilisation possibles. Or, les systèmes embarqués imposent de fortes contraintes d'intégration, ce qui réduit leurs ressources, particulièrement en termes de capacité calculatoire. Ainsi, ce type de système nécessite de recourir à des approches efficaces avec moins de charge et de complexité calculatoire. L'enjeu de cette thèse réside dans cette problématique. L'objectif est d'apporter une solution embarquée de suivi qui permettrait d'assurer un fonctionnement robuste dans différents environnements de navigation. Une analyse des algorithmes pertinents de suivi, visuel et visuel-inertiel et des environnements de navigation ainsi qu'une étude de différentes architectures embarquées de calcul sont menées, afin de proposer notre solution nommée « système de suivi inertiel-visuel adaptatif à l'environnement de navigation~». Cette dernière consiste à alterner entre deux approches de suivi : *KLT-ORB* et *EKF VI Tracking*, selon les conditions de navigation du système, grâce au module de contrôle, tout en assurant la cohérence du système global en gérant le nombre de PoIs et l'occurrence de leur détection et en respectant les contraintes des systèmes embarqués. Tous nos expérimentations et tests ont été réalisées en utilisant le jeux de données *EuRoC*.

Mots-Clés. Suivi visuel, Suivi visuel-inertiel, Suivi robuste, Localisation et Cartographie Simultanée (SLAM), Odométrie, Navigation visuelle-inertielle, Systèmes multi-capteurs, Systèmes embarqués, Co-Conception, Fusion, Couplage centrale inertielle/caméra, Couplage lâche, Couplage serré.

Acknowledgements

This PhD thesis work was carried out in the *Laboratoire L3A* of the *Institut d'Intégration des Systèmes et des Technologies* in the *Commissariat à l'énergie atomique et aux énergies alternatives (CEA)* in collaboration with the *Laboratoire LaSTIG* of the *Institut National de l'Information Géographique et Forestière (IGN)*. I would like to take this opportunity to thank these two institutions for their support and hosting.

I would like to express my sincere gratitude to Valérie Gouet-Brunet, for having accepted the direction of this thesis and having guided me throughout the whole work. I would like to acknowledge my three thesis supervisors, Martyna POREBA, Erwan PIRIOU and Maroun OJAIL, for their help not only to learn the technical aspects and to develop the integrity of my thesis, but also in developing my professional and personality skills. Working with them was a very pleasant and educational experience. I would like to address my sincerest acknowledgements to Michèle GOUIFFES and Frédéric CHAUSSE who accepted and judged my work.

A very special gratitude goes out to my friends and colleagues at L3A-CEA : Alexandre, Stéphane, Karim, Laurent, Thibault, Philippe, Youssouf, Eric, Suresh, Christophe, ... for their interesting discussion, kind and warm help and good humor. I thank also my office mates: Khan, Michal, Alix, ... for their sympathy and the good time we spent together. It was a great to share the same office with you for my PhD journey.

I would like to thank my parents as well as my sister Oumaima and my little brother Reda for supporting me every day although the distance, you are my real source of inspiration and motivation. Many thanks also to my friends who have also supported me during these years, especially Maryem who accompanied me throughout this journey and always brought me precious moral and emotional support and Mehdi who was very empathic and gave me great help and sympathy, in particular, during the last months of my thesis. Finally, I would like to thank rest of all my family and friends for their support and interest in my work. Thank you to all the people I unfortunately forgot to thank.

Imane SALHI

This thesis is dedicated to my beloved parents.

Contents

List of Figures

List of Tables

Résumé étendu

Introduction

Le sujet de cette thèse traite la problématique du suivi pour la localisation. Cette problématique représente un sujet stimulant dans des domaines d'application divers tels que la vision par ordinateur, la robotique et plus généralement l'Intelligence Artificielle (IA). Dans notre travail, nous nous intéressons au suivi dans les différents systèmes de navigation visuels et visuels-inertiels (VINS), pendant lequel les images acquises par une caméra mobile sont traitées et la position ainsi que les mouvements sont estimés afin de se localiser tout au long du déplacement du sysème. En effet, un VINS typique se compose d'une caméra monoculaire chargée de capter des images (informations visuelles) et une unité de mesure inertielle (IMU), à faible coût, capable d'intégrer les mouvements d'un mobile (accélération et vitesse angulaire) pour estimer son orientation (angles de roulis, de tangage, et de cap), sa position et sa vitesse linéaire. En effet, les deux types de capteurs produisent des données complémentaires. Malgré leur fréquence (cadence) moins rapide, les données visuelles peuvent générer des informations plus précises comparées aux données inertielles. En revanche, cette précision est très dépendante des conditions de texture, de luminosité de la scène concernée et des phénomènes d'occultations. L'intégration des données inertielles permet de remédier à ces problèmes et d'améliorer la précision. Cette combinaison est particulièrement efficace dans le domaine de la navigation et du suivi, grâce à la prise en compte des avantages conjoints de ces capteurs notamment en terme de précision, de coût et de réactivité.

Récemment, les systèmes de navigation ont évolué et sont devenus plus intégrés, mobiles et autonomes grâce à l'utilisation d'architectures de calcul de plus en plus embarquées. Plusieurs implémentations embarquées de différents algorithmes de suivi ont été réalisées dans ce sens. Elles permettent d'assurer un suivi suffisamment précis mais peuvent impliquer une complexité calculatoire importante. Par ailleurs, la forte densité d'intégration des systèmes électronique impose des limitations en termes d'énergie, d'encombrement et de capacité calculatoire. La problématique majeure des systèmes embarqués de suivi réside dans l'adéquation des différents algorithmes efficaces aux architectures de calculs embarqués, en respectant les contraintes imposées, en particulier en termes de capacité calculatoire. Le sujet de cette thèse, intitulée "Caméra intelligente embarquée pour le suivi robuste d'objets sur plateforme mobile", s'intéresse à cette problématique et vise à proposer un système de suivi visuel-inertiel adaptatif à l'environnement de navigation respectant les contraintes des systèmes embarqués.

Afin de mener à bien ce travail de recherche, une étude bibliographique des divers algorithmes et techniques de suivi a été mene. Ensuite, les différentes architectures et contraintes d'un système embarqué de navigation visuelle-inertielle, ainsi que les méthodes d'adéquation algorithme-architecture, connues sous le nom de « co-design », sont décrites et discutées. Dans le même temps,

une étude sur les conditions et les environnements de navigation a été réalisée en se fondant principalement sur le jeu de données public EuRoC. Le choix de ce jeu de données repose sur la disponibilité des informations inertielles et visuelles qui illustrent des mouvements de drones dans des environnements variés. Ces environnements incluent des scènes avec différentes textures et luminosités et avec des vitesses linéaires et angulaires et des types des déplacements et des rotations divers. Les algorithmes et techniques exploités dans la proposition qui émane de cette thèse prennent en considération les différentes catégories d'environnement de navigation identifiées suite à ces analyses et études de ce jeu de données. En effet, la solution proposée se base sur un module de contrôle d'exécution qui permet d'analyser le contexte de navigation et d'alterner par la suite, entre deux différentes techniques de suivi. La première, purement visuelle, elle est fondée sur l'algorithme Kanade-Lucas-Thomasi (KLT) et destinée à la navigation dans des environnements faciles, caractérisés par des mouvements lents et stables et des scènes texturées et illuminées. La deuxième, visuelle-inertielle, elle est basée sur l'algorithme Extended Kalman Filter (EKF). Elle est adaptée à la navigation dans des environnements complexes, qui se caractérisent principalement par des mouvements rapides et des problèmes de texture et de luminosité. En s'adaptant à l'environnement de navigation tout en réduisant le nombre de points à suivre et en basculant entre ces deux méthodes de suivi, le système proposé peut produire des résultats efficaces, robustes et adaptés à des implémentations embarquées pour des systèmes comme les Micro-Aerial Vehicles (MAVs).

○ Contributions

Pour traiter la problématique posée dans cette thèse, présentée et discutée auparavant, nous avons adopté une orientation de recherche qui se focalise, tout d'abord, sur l'analyse, la comparaison et l'utilisation des algorithmes les plus pertinents pour le suivi, tout en respectant les contraintes des systèmes embarqués, et qui implique, ensuite, de mettre en place :

- Une étude des différents environnements de navigation , ainsi qu'une identification et une description des caractéristiques des environnements faciles et difficiles de navigation,
- Un système de suivi adaptatif alternant deux méthodes de suivi grâce à un module de contrôle d'exécution et une méthodologie de sélection de PoIs. Il est constitué de trois parties :
    1. la méthode de suivi visuel basée sur le KLT-ORB "*KLT-ORB Tracking*", dédiée aux environnements faciles de navigation (sans vibration, sans mouvement rapide, sans flou, suffisamment de texture.),
    2. la méthode de suivi visuel-inertiel basée sur EKF "*EKF VI Tracking*", destinée aux environnements difficiles de navigation (mouvements rapides, zones noires, scènes non texturées.),
    3. le module de contrôle d'exécution, basé sur de nombreuses métriques afin de décrire l'état de l'environnement de navigation et la qualité du suivi actuel.

Système embarqué pour un suivi robuste

Cette thèse s'organise en cinq chapitres. Les deux premiers chapitres s'intéressent aux systèmes de navigation visuels et visuels-inertiels, respectivement. Ensuite, le troisième chapitre traite les différents systèmes et architectures embarquées utilisés spécifiquement dans des systèmes de suivi.

Enfin, les chapitres quatre et cinq se focalisent sur la présentation et la discussion de notre proposition et de ses résultats, respectivement. Dans ce qui suit, nous résumons les différents points traités dans chaque chapitre de cette thèse afin de fournir un aperçu sur son contenu global.

○ Etat de l'art

De nombreux travaux de recherches traitant la thématique de la navigation, particulièrement le suivi, ont été proposés. Les méthodes de navigation peuvent être classées selon différents critères. La catégorisation la plus répandue dans la littérature est celle qui divise ces méthodes en "Odométrie" et "Localisation et Cartographie Simultanées (Simultaneous Localization And Mapping (SLAM))". Tout d'abord, l'odométrie est le processus d'estimation du mouvement d'un robot (translation et rotation) en observant une séquence d'images de son environnement. Ce processus de traitement peut impliquer des accumulations d'erreur odométriques au fil de temps. L'odométrie visuelle (Visual Odometry (VO)), plus précisément, représente un cas particulier d'une technique nommée "Structure From Motion (SFM)" [70]. Cette dernière traite la problématique de la reconstruction 3D de la structure de l'environnement en utilisant un ensemble d'images séquentiellement ordonnées ou non. Or, le SLAM [56][11] représente un moyen pour qu'un robot arrive à se localiser dans un environnement inconnu. En plus, contrairement à l'odométrie, le SLAM autorise la construction progressive d'une carte de son environnement et assure un retour qui permet de réduire l'erreur accumulée au fil de temps. Le SLAM a été largement étudié au cours des deux dernières décennies [47], ce qui a donné lieu à de nombreuses solutions [108][153][175][211] basées sur différents capteurs, notamment des caméras et des IMUs.

Systèmes visuels La majorité des systèmes de suivi se décompose en deux étapes de calcul. La première est destinée au prétraitement des images. Pendant cette étape, il est possible d'utiliser des techniques basées sur le flux optique ou sur l'extraction des caractéristiques de l'image, telles que les Points d'Intérêt (PoIs). En effet, les PoIs représentent des caractéristiques d'image connus pour leur précision et leur aspect générique. Notre étude de différentes approches de la littérature a conduit à identifier les algorithmes de détection/description des PoIs comme le plus approprié pour un système de suivi à ressources limitées. En effet, cet algorithme permet de détecter un nombre suffisant de PoIs tout en étant invariant à la majorité des transformations requises, notamment la rotation. Quant aux méthodes basées sur le flux optique, l'algorithme KLT présente également un intérêt particulier. C'est une méthode différentielle qui prend comme hypothèse que le flux est constant dans un voisinage local du pixel considéré, et résout l'équation du flux optique pour tous les pixels dans ce voisinage par la méthode des moindres carrés. En effet, l'algorithme KLT offre un suivi visuel précis particulièrement en cas des déplacements courts entre deux images successives dans une séquence d'images. Il est caractérisé également par sa légère complexité calculatoire par rapport aux algorithmes basés sur la détection/description des PoIs.

Différents systèmes visuels de navigation et de suivi (VO et SLAM) sont proposés dans la littérature et fournissent des résultats précis et fiables. Cependant, ils sont influencés par différentes contraintes liées à l'environnement de navigation, telles que les environnements sombres, peu texturés ou des mouvements qui brouillent l'image, ce qui réduit considérablement leur robustesse et peut provoquer l'échec du suivi. De plus, il faut garder à l'esprit qu'un système visuel monoculaire souffre de la dérive d'échelle dans le temps due à l'utilisation d'une seule caméra. Par conséquent, afin d'améliorer les performances de suivi, il est nécessaire soit de multiplier le nombre de caméras (stéréovision), ce qui imposera un calcul supplémentaire, soit d'utiliser d'autres types de capteurs

afin d'assurer des données complémentaires à celles de la caméra, comme l'intégration des mesures inertielles issues de l'IMU.

Systèmes visuel-inertiels Les systèmes multi-capteurs de suivi revêtent un grand intérêt dans la recherche académique et industrielle. La combinaison caméra/IMU est parmi les couplages les plus répandus. Ces deux capteurs se caractérisent par leur complémentarité. En effet, l'IMU fournit des mesures d'accélération et de vitesse angulaire à une fréquence importante tandis que la caméra produit des informations visuelles précises mais à une fréquence réduite. Les systèmes de navigation multi-capteurs, visuels-inertiels, peuvent être divisés, en fonction du type de couplage, en deux catégories : couplage serré ou couplage lâche. Cette catégorisation est orthogonale au découpage des méthodes de navigation en se basant sur la distingtion VO et SLAM. De même que les systèmes visuels purs, les systèmes de suivi visuel-inertiel ont besoin de passer par le calcul de la pose. Bien évidemment, cette étape est différente de celle utilisée pour les systèmes visuels purs à cause de l'introduction des mesures inertielles. Ainsi, pour ce type de système, un algorithme de fusion de données est recommandé, notamment l'algorithme EKF vu son adéquation avec nos besoins en terme de précision et du respect de la capacité calculatoire des systèmes à ressources limitées.

Architecture embarquées pour le suivi Actuellement, il existe différentes architectures de calculs dédiées et d'accélération. Afin de choisir l'architecture la plus adaptée pour l'implémentation d'une application donnée, il est nécessaire de trouver un compromis entre la qualité des résultats et les contraintes imposées par un système embarqué. La limitation de la capacité calculatoire est considérée parmi les principales contraintes à prendre en compte lors de la conception d'un systèmes embarqué. Or, l'utilisation de plusieurs capteurs engendre un nombre important de données à traiter et à fusionner, ce qui a un effet sur la complexité calculatoire du système. Par conséquent, concevoir un système embarqué de suivi visuel-inertiel nécessite des architectures de calcul embarqué plus spécialisées et plus raffinées selon les objectifs du système.

○ Suivi visuel-inertiel adaptatif à l'environnement de navigation

Dans cette thèse, nous proposons un système basé sur les données visuelles-inertielles qui peut assurer le suivi dans différentes conditions de navigation, grâce à un module de contrôle qui alterne entre deux approches de suivi, visuelle et visuelle-inertielle. Pour cela, tout d'abord, une analyse de l'environnement de navigation est réalisée pour permettre, d'une part, d'identifier les environnements de navigation problématiques qui nécessitent un plus grand effort en termes de temps et de capacité de calcul, d'autre part, d'identifier les cas où la navigation peut être plus facilement assurée et les contraintes sont assouplies. Ainsi, un suivi robuste basé sur la combinaison de différentes méthodes algorithmiques est proposé et est assuré par un module de contrôle déié. Ce dernier est utilisé pour analyser les données inertielles, la qualité de l'image, le nombre de PoIs et le retour d'information sur la qualité du suivi, entre les images actuelles et précédentes, afin d'identifier le type de mouvement (rotation, translation) et la qualité des données visuelles. Il vise également à choisir la méthode de suivi la plus adaptée à l'environnement de navigation actuel. Dans ce travail, un choix motivé est fait entre l'approche visuelle *KLT-ORB Tracking*, basée sur le flux optique via l'algorithme KLT, et l'approche visuelle-inertielle *EKF VI Tracking*, basée sur la fusion de données en utilisant l'algorithme EKF.

Le suivi visuel *KLT-ORB Tracking* permet un suivi plus rapide, en particulier dans un contexte de navigation facile (scènes à faible mouvement et texturées), ce qui est avantageux pour les systèmes

embarqués mobiles. En effet, dans la plupart des travaux, un suivi visuel dans un système SLAM est effectué en détectant les PoIs et en les décrivant pour chaque image. Cette approche nécessite un temps d'exécution et un coût de calcul importants (*e.g.* au moins 10*ms* pour ORB SLAM). Cela explique les avantages de l'utilisation de la méthode KLT-ORB Tracking basée sur le flux optique. Elle permet de calculer rapidement la pose relative entre l'image actuelle et l'image précédente, en diminuant la fréquence de détection/description des PoIs. En particulier, dans le cas de mouvements de translation et de rotation relativement faibles. Aussi si la cartographie et la création de KeyFrame ("image clé") ne sont pas nécessaires, la redétection d'un nombre élevé de PoIs est dispensable ce qui réduit le temps de traitement. Ainsi, le suivi visuel *KLT-ORB Tracking* limite la complexité du calcul des PoIs et de la localisation des poses pour le SLAM. Néanmoins, lorsque l'environnement de navigation est difficile, le suivi est effectué en fusionnant les données inertielles et visuelles à l'aide de l'algorithme EKF. C'est une technique de suivi précise et efficace malgré les coûts de calcul élevés par rapport à la méthode visuelle pure. Il est possible de réduire ce temps et d'en tirer parti pour une mise en œuvre solide du SLAM sur les systèmes mobiles embarqués. Ceci est réalisé en alternant entre les deux méthodes, grâce au module de contrôle d'exécution qui vise à analyser l'environnement de navigation et, en fonction du type de mouvement et des conditions de la scène, l'approche de suivi appropriée est sélectionnée et exécutée.

○ Expérimentations

Nous avons réalisé des différentes expérimentations afin de pouvoir évaluer la qualité globale du système proposé. Dès nos premières expérimentations et tests des approches de la littérature, tel que ORB SLAM, nous avons pu soulever trois aspects problématiques, liés principalement à l'évaluation des trajectoires estimées, qui sont le non-déterminisme, l'estimation de l'échelle et l'alignement des trajectoires. Afin d'évaluer correctement notre proposition et de la comparer par rapport aux autres travaux de l'état de l'art nous avons adopté la méthodologie décrite ci-après. Tout d'abord, pour remédier au non-déterminisme, nous avons utilisé dans nos calculs la trajectoire moyenne calculée suite à dix exécutions de chaque approche testée. Ensuite, nous avons analysé plusieurs techniques d'estimation d'échelle parmi lesquelles nous avons retenu la méthode qui consiste à calculer le ratio entre la pose inertielle et la pose fusionnée (visuelle-inertielle), cette estimation est inspirée de la proposition présentée dans [198] et dédiée aux systèmes de navigation visuelle-inertielle. Par ailleurs, la trajectoire estimée et la vérité terrain ne sont pas représentées dans le même référentiel. Par conséquent, le calcul des erreurs entre les deux trajectoires n'est pas concluant. Pour cette raison, un pré-traitement est effectué afin d'appliquer les transformations nécessaires pour pouvoir comparer l'estimation et la vérité terrain et pouvoir calculer les erreurs et quantifier la précision du système étudié, ce pré-traitement est nommée : "alignement de trajectoire". Dans cette thèse, nous avons adopté la méthodologie proposée dans [230], basée sur la technique *Umeyama* [212].

Par la suite, notre système est évalué en tenant compte des différents aspects discutés auparavant. Nous avons commencé tout d'abord par évaluer le fil de traitement de suivi d'ORB SLAM, ensuite chaque composante du système proposé est testée indépendamment : *KLT-ORB Tracking* et *EKF VI Tracking*. Finalement la qualité de l'ensemble de la solution est quantifiée. Pour cela, nous avons choisi le jeu de données EuRoC comme base pour nos tests. Dans ces expérimentations nous nous sommes principalement basés sur l'exactitude et la robustesse du suivi dans différents types d'environnements, ainsi que sur le temps d'exécution nécessaire pour les différents calculs. Ces évaluations commencent par décrire la partie suivi de la méthode ORB SLAM puisqu'il représente un point de départ pour notre proposition. ORB SLAM fournit un suivi robuste et précis sur des

séquences faciles de la navigation. Or, dans les environnements difficiles, il rencontre des difficultés pour pouvoir assurer le suivi continu. En effet, la qualité du suivi peut se voir détériorée voire le système se perdre et par conséquent le suivi échoue sur le reste de la séquence. De plus, le suivi ORB SLAM impose une charge de calcul importante en détectant 1000 PoIs à chaque image et sur toutes les images de la séquence. En évaluant notre implémentation du suivi *KLT-ORB Tracking* nous remarquons une réduction importante en terme de temps de calcul. Cela revient à la réduction de la fréquence de détection des PoIs et de leur nombre. En effet, ce temps de calcul est estimé en moyenne à $32fps$. Comme attendu, le KLT-ORB n'est fonctionnel que sur des séquences faciles et des distances courtes. Pour y remédier le suivi *EKF VI Tracking* est utilisé pour pouvoir se compléter avec le suivi *KLT-ORB Tracking*. Lors de l'évaluation du le suivi *EKF VI Tracking*, nous avons pu confirmer sa robustesse à des environnements difficiles (grandes rotations, vibrations causant le flou dans l'image,*etc*). Sa précision dans les séquances dites "faciles" est intéressante et est estimée en moyenne à 0.15 mètres alors que dans les cas difficiles cette précision se dégrade et l'erreur peut augmenter jusqu'à 0.69 mètres. Puisque le suivi *EKF VI Tracking* se base sur la fusion des données visuelles et inertielles, la charge de calcul et le temps d'execution sont affectés. Ainsi, la cadence de traitement est de $35fps$ en moyenne pour les environnements faciles et $37fps$ pour ceux difficiles. Afin de conserver des temps de traitement correctes nous avons proposé de réduire drastiquement le nombre de PoIs à savoir jusqu'à 5 nouveaux PoIs au maximum par image.

Finalement, le système de suivi adaptatif a été évalué dans sa globalité. Nous avons ainsi pu vérifier l'intérêt de l'utilisation du module de contrôle en étudiant la qualité du suivi, le temps de calcul mais aussi l'impact de la phase de transition entre les deux méthodes. La transition entre les deux modes impose une erreur qui peut aller jusqu'à 0.30 mètres et un temps de calcul entre 0.25 et 0.27 secondes, en fonction de la méthode qui prendra le relais. En effet, chacune des deux méthodes de suivi a une propre phase d'initialisation dont la différence majeure réside dans le nombre d'images exploitées. La qualité globale du suivi adaptatif représente une erreur ATE de 0.08 mètres sur les séquences faciles et de 0.19 mètres sur les séquences difficiles, ainsi qu'un temps de calcul estimé par $32fps$ pour les séquences difficile et par $29fps$ pour les séquences faciles. La solution proposée fournit des résultats de précision comparable avec la plupart des résultats des travaux connus de la littérature. Elle permet aussi de garantir la robustesse du suivi dans différents environnements de navigation, même les plus contraignants. Enfin, cette méthode permet d'avoir des résultats en terme de temps de calcul acceptables et rend envisageable une implémentation sur une architecture embarquée.

Conclusions et perspectives

Cette thèse présente notre proposition d'une méthode de suivi visuelle-inertielle robuste adaptée à la mise en œuvre des systèmes embarqués mobiles tels que les MAVs. D'une part, les algorithmes de suivi sont contraints par les conditions du contexte de navigation, en particulier, ceux basés sur la vision. Ces derniers sont sensibles aux scènes noires à faible texture et aux mouvements rapides de la caméra qui peuvent provoquer des flous dans l'image. Par conséquent, les mesures inertielles sont utilisées dans l'objectif d'améliorer ces systèmes visuels, leur intégration impose une complexité calculatoire plus importante à cause des pré-traitements des mesures brutes de l'IMU et de la fusion avec les données visuelles. D'autre part, les systèmes embarqués sont contraints en termes de ressources. Ils sont limités par la puissance de calcul, la mémoire embarquée et la charge utile des capteurs. Ainsi, ils ne peuvent pas gérer facilement et directement la mise en œuvre d'algorithmes

de suivi à haute complexité de calcul. Cette thèse décrit et évalue notre proposition dédié aux systèmes embarqués tout en assurant l'adéquation algorithme-architecture. Elle consiste à alterner entre deux méthodes de suivi : le suivi visuel *KLT-ORB Tracking* et le suivi *EKF VI Tracking*, grâce au module de contrôle d'exécution qui analyse le contexte de navigation (mouvement et scène) et permet de passer à la méthode de suivi appropriée. L'évaluation du suivi visuel-inertiel adaptatif proposé assure un fonctionnement robuste dans les différents environnements de navigation, faciles et difficiles, fournissant une précision, exprimée en termes de RMSE d'ATE de 0.08 mètres dans les environnements faciles et de 0.19 mètres dans les environnements difficiles. Ces résultats sont obtenus grâce à l'efficacité des algorithmes de suivi choisis et au module de contrôle d'exécution.

Les recherches présentées dans cette thèse offrent des prolongements et perspectives potentiels pour la suite des travaux. Ces derniers visent à étendre les contributions fournies et à améliorer la praticabilité de la solution proposée dans les applications réelles. Pour les futurs travaux, nous avons comme prévisions les sujets suivants :

- **Amélioration le processus d'estimation de l'échelle :** l'échelle est un problème de recherche difficile dans le domaine du suivi visuel et visuel-inertiel. Des recherches futures peuvent être entreprises pour développer de meilleures stratégies d'estimation à l'échelle en utilisant des données visuelles et inertielles indépendamment du traitement de la pose fusionnée, même avec des capteurs étroitement couplés. Cela devrait permettre d'éviter les grandes variations d'échelle, ainsi que le bruit et les erreurs supplémentaires.

- **Amélioration les mesures de cohérence des trajectoires :** des études futures peuvent également élargir les mesures utilisées pour le module de contrôle d'exécution. Les résultats expérimentaux ont démontré l'importance du module de contrôle d'exécution et de ses métriques de contrôle utilisées pour alterner entre les deux approches de suivi proposées. Suite à ces résultats, des études futures peuvent également améliorer les métriques existantes. Ainsi, la précision du suivi pourrait donc être améliorée.

- **Intégration du suivi inertiel et visuel adaptatif en fonction du contexte, sous contrainte matérielle :** le suivi inertiel visuel adaptatif en fonction du contexte de navigation peut être utilisé dans différents systèmes de navigation. En particulier, les MAVs, qui ont des limites en termes de complexité de calcul, de mémoire, d'espace et de ressources énergétiques. La méthode proposée a été développée pour être implémentée sur de tels systèmes embarqués tout en respectant leurs exigences et leurs contraintes. Par conséquent, les travaux futurs pourront se concentrer sur le portage de la solution sur une architecture SoC multiprocesseur (MPSoC) embarquée de la famille Xilinx, et/ou de type GPGPU de chez NVIDIA par exemple Jetson NX ou nano.

Introduction

Problem Background

Motion tracking is a challenging field of research in computer vision (CV), robotics and more generally artificial intelligence (AI). This topic is presented, especially in our context, as a measure of the positioning of an object moving in a predefined space. Thus, body position and/or orientation are estimated. Actually, it is possible to measure only position (x, y, z) so this is called the 3 Degrees of Freedom (3 DoF or 3D) tracking, or to measure position (3 coordinates) and orientation (3 independent angular coordinates), simultaneously, then this is known as the 6 DoF or 6D tracking.

Nowadays, motion tracking is becoming an increasingly important task in a wide range of application and intelligent devices, for example in military, entertainment, sports, medical applications and also in validation of CV and robotics. The following paragraphs describe some of the relevant applications examples of motion tracking:

Health Care and Person Aid for example, in [45][63], an application based on motion tracking and designed specifically for visually impaired people (figure 1) is proposed. It is a real-time virtual blind cane system, that combine three sensors in a loosely coupled way : a camera, an IMU and a laser. The distance of the obstacles, regarding the camera, is calculated using the triangulation based on the laser stripe center position and the camera parameters. Inertial data are integrated at this step, applying Kalman Filter algorithm, in order to perform the motion tracking and compute the system pose. The latter is used to determine the distance to obstacles with respect to the user's body.

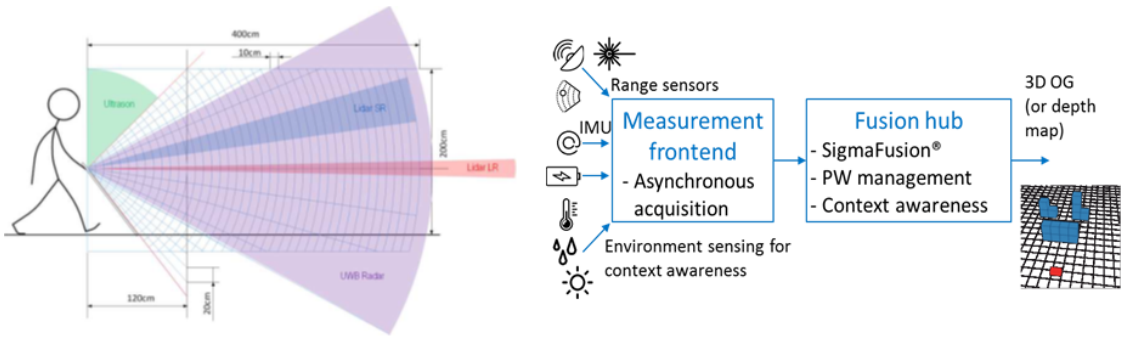

FIGURE 1: Overview of the real-time virtual blind cane system proposed in [173] .

Motion Tracking and Environmental Understanding Google ARCore [80] is an example of works on Android devices, and focuses on three functionalities which are motion tracking, environmental understanding and light estimation. Motion tracking uses the phone's camera to track Points of Interest (PoIs) (figure 2), in the room coupled to IMU sensor data, in order to determine the 6D pose of the phone as it moves. Environmental understanding detects horizontal surfaces using the same PoIs computed for motion tracking. Light estimation leverages the ambient light sensors in order to

observe the ambient light in the environment and light the virtual objects in ways that match their surroundings, making their appearance more realistic.

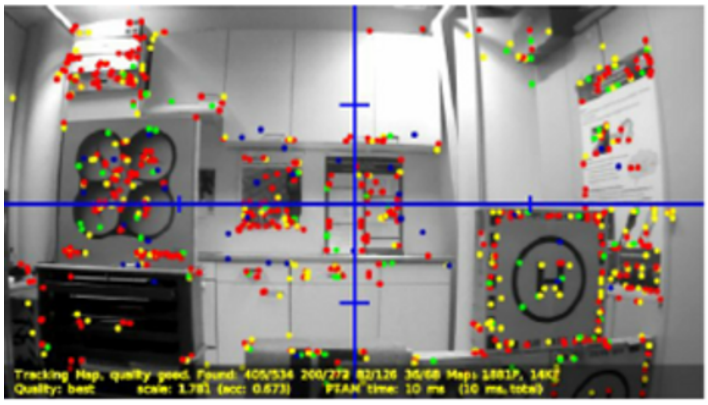

FIGURE 2: PoIs matching with the real world and used for motion tracking (figure from [143])

Autonomous Navigation for instance, Aircraft Inspection application [54] based on Unmanned Aerial Vehicles (UAVs), as shown on figure 3. UAVs, commonly known as a drones, are an aircraft without a human pilot aboard. For such aircraft inspection systems, the precise localization and obstacle detection are crucial to maintain safe distance to the parked plane and guarantee the anti-collision of the whole inspection system, while allowing close inspections if necessary. Therefore, the motion tracking may be provided by captured images, which are immediately geo-referenced to the aircraft structure, then processed to localize and measure aircraft surface target damage. By using drone, it is possible to perform more accurate and low-cost inspections. In fact, the operator can in real-time map visualize and get damage reporting for inspected plane, and then flag visible defects for closer examination.

FIGURE 3: Donecle aircraft inspection drone system [54].

Motion Tracking in Autonomous Navigation

In this thesis we focus on the motion tracking in autonomous navigation application. Before discussing further on this, it is worth noting that navigation methods (including tracking) can be subdivided into: Odometry, which is an incremental motion tracking with respect to the starting image;

and SLAM (Simultaneous Localization and Mapping), which concerns a combination of the motion tracking used for the localization, and the map building task that is done simultaneously with camera motion tracking.

Generally, tele-operated and autonomous navigation systems (odometry- or SLAM-based) are able to capture, process, and actively make sense of the information provided by their sensors. The acquired information depends mainly on the system purpose and its integrated sensors. For example, the landmarks, which are detected on the images captured by the system-mounted cameras, and used in visual tracking and localization tasks, are sufficient to ensure a navigation environment mapping, obstacle avoidance and path planning, however, an additional dense occupancy map can be recommended. Actually, motion tracking is tied to localization and environment mapping. An effective motion tracking process provides accurate navigation environment features, thus ensures accurate localization and environmental mapping.

Meanwhile, the miniaturization of electronic systems enabled small-scale, mobile and wearable smart communicating objects (*e.g.* Internet of Things (IoT)) to become increasingly widespread. This miniaturization aims to perform various complex systems in different fields, such as mobile multi-sensor embedded systems for navigation and tracking, intelligent glasses, smart cameras, UAVs and Micro Air Vehicles (MAVs). The latest examples are among the highly constrained embedded devices. Therefore, the design of such systems imposes serious challenges in terms of computing capacity, system integration (form factor) and resource constraints, like battery life and energy consumption. The main object of this thesis is to find an adequacy between the algorithms quality and the embedded architecture constraints in order to provide a robust and accurate embedded tracking system. Thus, different tracking approaches are discussed, including visual and visual-inertial ones. In the following, an overview of these discussions is introduced.

○ Visual Motion Tracking

There are numerous methods to process visual data for different tracking tasks. Recently, deep learning techniques have been largely applied to CV and robotics technology. These techniques are used in various applications, such as facial recognition, object recognition, camera pose improvement and motion tracking. However, they require a high computing capacity and an interconnection between the system and external data resources. This involves an additional cost, in terms of energy consumption and computational complexity, and makes the system dependent instead of autonomous. Besides deep learning techniques, there are various other handcrafted visual data processing techniques that are employed in different applications. For example, in PoIs detection, description and matching tasks. These image processing techniques do not require any external network connectivity for data exchange, and save energy with regard to the deep learning techniques. Compared to other systems, only-vision based ones provide an accurate results. This is not only thanks to the algorithms and processing methods used, but also to the precision of the visual data coming from the camera. However, this sensor provides data at a slower frequency, compared to other sensors (cadence). It can also suffer from occultation and less textured environments that cause loss of information, thus loss of system accuracy and robustness. To overcome these problems, it is required to combine several homogeneous/heterogeneous sensors like IMU and camera coupling.

○ Visual-Inertial Motion Tracking

Many literature papers are written about visual-inertial data fusion [175][93][175], obtained from a camera/IMU combination. Cameras and IMUs are particularly attractive for providing navigation

methods as they are characterized by their low power consumption, light weight and wide availability in the market at attractive prices. An IMU may contain one, two (gyroscope and accelerometer (6 axis)), or three inertial sensors (gyroscope, accelerometer and magnetometer, *etc*.), that provide raw inertial measurements. Integrating inertial data provides a pose information that enables to overcome the problems of only vision-based navigation methods.

The camera/IMU combination comes in different configuration categories, which can depend on different fusion topologies: centralized, decentralized or hierarchical fusion, as well as on different sensors coupling: tightly coupled or loosely coupled. The fusion topology and/or coupling type choice relies on the system's resources and application field. A loosely coupled system requires less computational complexity than a tightly coupled one, but it does not provide the same accuracy as the latter. Furthermore, the visual-inertial fusion requires the use of a data fusion algorithm. There are a wide variety of these algorithms, the most commonly ones in case of tracking systems are: Kalman Filter (KF), Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) and Particle Filter (PF). Selecting the appropriate algorithm for a given navigation system computation is supported by its nature. For example, Kalman Filter algorithm cannot deal with the non-linearity problem, however, the other three algorithms are designed for this fusion case. The decision depends also on the algorithm computational complexity, and its result accuracy. Moreover, camera/IMU combination requires a hardware integration thought. It is necessary to take into account the processing cost and the management of all this data when making processing algorithmic choices. Computational complexity and overall performance are among the main criteria to be considered when choosing the most appropriate algorithm.

Research Objectives & Contributions

Based on all the previous discussions, our aim in this thesis is to design and develop a new tracking system that ensures a robust and accurate motion tracking and respect the embedded systems constraints, simultaneously. To do so, first, our work's positioning within the relevant literature and its main issues are presented below. Afterwards, the main researches objectives are listed. This is before ending with a summary of the main contributions of this work, the list of publications and the thesis structure.

○ Positioning

Further to the previous analysis, this work proposes an embedded multi-sensor tracking system based on the camera/IMU combination. It aims at tracking the system's location continuously at any moment. To achieve this, the proposed solution is a visual-inertial tracking based on PoIs. These latter are image features that provide a generic and accurate representation of the environment. Inertial data derived from the IMU, often fused with visual data, is also used for tracking processes in a specific computation framework. Using visual or visual-inertial data for tracking involves two different computation approaches which are presented and discussed in details in this thesis. In addition, this work focuses on various embedded computing architectures and their main constraints regarding the processing architecture implementation. Computational architectures can be programmed with a programming language such as C, C++, *etc*., for instance GPP, GPU, DSP and ASIP, or described by a descriptive language like VHDL, for example ASIC and FPGA. Choosing the right target architecture becomes more obvious when a compromise is found between the computational complexity of the used algorithms and the different embedded system constraints, such as its performance, space

exploration and energy consumption. In addition, this decision may also be enhanced by adapting existing algorithms to be implemented on an embedded architecture. Therefore, the integration on different embedded architectures is improved and a large number of algorithms and methods are used despite their computational complexity.

○ Research Objectives

In this thesis, a new, robust, controlled tracking approach based on tightly camera/IMU coupling is proposed. This tracking approach is studied in the framework of a SLAM system by taking inspiration from different well-known literature methods: the ORB SLAM [153], the R-VIO [93] and the accurate Monocular VI EKF SLAM [177], *etc.* A novel dedicated execution control module, ensuring the continuity of the tracking process is proposed (figure 4).

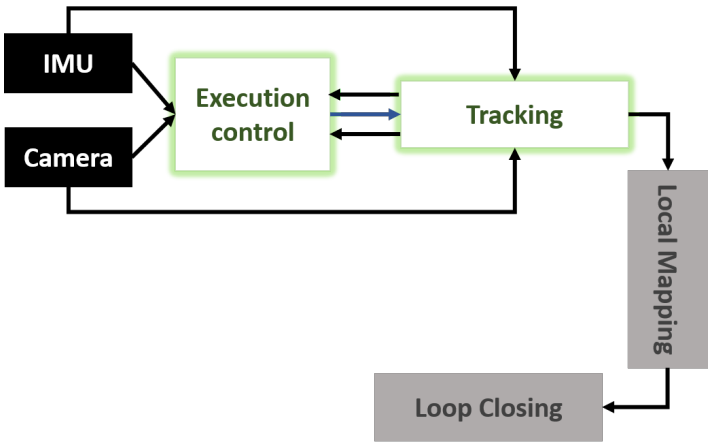

FIGURE 4: Proposed System Overview

One of the major challenges of an embedded tracking system is its robustness to:

- fast and complex motions,
- changes of scene lighting,
- difficult or recurring textured scenes.

Therefore, the main objective of this thesis is to improve the robustness of the tracking and its efficiency, while keeping the accuracy or even enhancing it, in various navigation contexts.

○ Main contributions

To resume, the main contributions provided by this thesis are:

- Effect evaluation of different types of navigation environments on tracking quality, and the identification of the more complex and problematic navigation conditions;
- Development and proof of interest in using the execution control module allowing a switch between two tracking methods and the ability to select the suitable method to be performed, according to the system's navigation context;
- Development and combination of two different tracking algorithm in the same system to ensure tracking robustness and continuity: Visual KLT-ORB and Visual-Inertial EKF tracking, as well as their adaptation to the remaining ORB SLAM processing parts;

- PoIs management methodology performed between different processing functions, that provides the system consistency and saves memory and computational time by limiting the number of the PoIs at 500 or 5 PoIs.

○ List of Publications

The papers published during this thesis are provided herein:

Imane Salhi, Martyna Poreba, Erwan Piriou, Valérie Gouet-Brunet, and Maroun Ojail (Jan. 2019). "Multi-Modal Localization for Embedded Systems." Book chapter in *Multi-Modal Scene Understanding, M. Ying Yang (eds.), Elsevier (80 pages: 199-278)*. [1]

Imane Salhi, Erwan Piriou, Martyna Poreba, Maroun Ojail, and Valérie Gouet-Brunet (June 2018). "Suivi d'objet par capteurs visuels et inertiels sur systèmes embarqués." Conference paper in *Conférence Française de Photogrammétrie et de Télédétection (CFPT) (8 pages)* [2]

Thesis Structure

The thesis is organised in five main chapters.

◇ Chapter 1, provides an overview of state-of-the-art on visual navigation systems. Starting by introducing the main visual systems classification. Then, the main components of visual tracking techniques are discussed. Herein we focus on detailing the different PoIs detection and matching methods, as well as on the explanation of various visual pose estimation. We also introduce some of relevant vision-based tracking and navigation systems.

◇ Chapter 2, deals with visual-inertial navigation systems. Different Visual Inertial Navigation System (VINS) categories and methods are introduced. In addition, a thorough overview of some well-known examples in literature is provided in order to illustrate techniques explained before in this chapter.

◇ Chapter 3 presents a discussion of multi-sensor embedded navigation systems. In fact, in this chapter, we present the embedded systems classification and constraints. Subsequently, we focus on the different computational architectures, which are used mainly in embedded tracking systems. Lastly, an overview of the current tracking and navigation embedded systems, based on visual-inertial combination, is discussed.

◇ After all previous discussions and state-of-the-art overviews, the proposed approach is described in chapter 4. The latter provides the navigation environments analysis, performed in order to identify the problematic environments and the main metric impacted by these latter. Next, the overall workflow of our Context Adaptive Visual-Inertial SLAM is presented. Further to this, the Context Adaptive Visual-Inertial Tracking is explained in depth details. First the proposed execution control module metrics and workflow are detailed, then visual KLT-ORB and EKF Visual-Inertial tracking are addressed.

◇ Chapter 5 deals the previously mentioned Context Adaptive Visual-Inertial Tracking experimental assessment. First, the experimental and assessment environment is described and the

[1] https://www.sciencedirect.com/science/article/pii/B9780128173589000147
[2] https://hal.archives-ouvertes.fr/hal-02338377

main trajectory evaluation issues, including the non-determinism, the scale estimation and the trajectory alignment, are highlighted and addressed. Then, the trajectory evaluation is discussed based on exhaustive analysis and experiments. Afterwards, results of the different proposition components assessments are reported and explained, and the results of the overall solution appraisal is presented and compared to other relevant tracking methods.

- ◇ Finally, Conclusion summarizes the thesis by presenting an overview of the proposed approach and discussing their experimental results.In addition, research perspectives and final thoughts are also discussed.

Chapter 1

Visual Navigation System

Navigation systems can be roughly seen as a process that has the ability to determine an appropriate and safe path between a starting point and a destination. A lot of navigation methods have been proposed but each often respond to different constraints. Therefore, the required navigation method is highly depending on the field of application. In this chapter, we are particularly interested in visual navigation and its various classes and components.

In the early 2000s, vision-based navigation has proven to be a promising and primary research topic for navigation systems. This is, especially, due to the rapid development of computer vision domain field, which is extending their application area. These navigation systems include obstacle detection, autonomous navigation, localization, *etc*. Firstly, the visual sensors used for vision-based navigation systems provide complete information about the navigation environment. Also, they are very suitable for dynamic environment perception thanks to their anti-interference capability. Moreover, such sensors allow to acquire more precise data.

Generally, the main steps in the navigation system process are: image acquisition, features detection and matching, pose estimation, tracking and map building/updating if it is required (depending on the navigation system category).

Feature extraction involves identifying robust features in the image, as edges, points of interest (PoIs), etc. It is an essential step as it is relied upon during the rest of all computations. Therefore, in this chapter this topic is addressed with a focus on PoIs as being the image features most suitable for this thesis project. This is because they are a generic and precise image features that facilitate many navigation and tracking process in different environment conditions. Also, the main algorithms for detecting and matching these PoIs are discussed. In addition, the localization and estimation of the pose (position, orientation) impact significantly a navigation system's quality and performance. Therefore, pose estimation is considered one of the most important challenges in a navigation system. For this reason, in our review we are particularly interested in different pose estimation methods applied and developed for visual navigation and tracking systems.

In this chapter, firstly, the visual navigation system classification, as well as its two main methods which are Visual Odometry (VO) and Simultaneous Localization And Mapping (SLAM) are discussed in section 1.1. Then, in section 1.2, the upstream image processing techniques that are usually involved in a visual navigation system are addressed. The focus is on PoIs detection, description and matching algorithms, because most of the relevant literature systems rely on these image features as they are more generic and accurate for vision-based localization and tracking processing. Afterwards, the relevant methods and processes used in the literature for estimating visual pose are presented in section 1.3. Finally, section 1.4 gives a detailed description of widely used works of the state-of-the-art (EKF MonoSLAM [108], PTAM [117], and ORB SLAM [153]) to illustrate the discussions of the previous sections.

1.1 Systems Classification

Several vision-based navigation classifications are proposed in the literature [52] [22] [18]. First of all, vision-based navigation was categorized into two main types: *indoor navigation* and *outdoor navigation* [52]. Additionally, each of these two categories is further classified into different subcategories. Outdoor navigation can be subdivided in structured and unstructured environments. For the first type, the system navigates on the basis of the road following technique, where the system (robot) detects the road lines and followed them in order to navigate consistently. While for the second one, the system has no properties to follow for navigation, and other alternative solutions are used

as presented in [52] and [22]. Whereas, indoor navigation is subdivided in three categories (figure 1.1): firstly, *map-based approach*, this concerns navigation systems that require user-created geometric models or topological environment maps. In this category, there may be global (or absolute) systems in which the initial position is unknown, or local (or relative) systems which assume a known initial system position. Secondly, *map-building-based approach*, it is about the sensor-based systems that construct their own geometric or topological environment models for subsequent use in navigation. This systems type has been widely used in autonomous and semi-autonomous navigation areas, employing different methods as Simultaneous Localization And Mapping (SLAM). As discussed in [133], these map-based systems evolved considerably and derived three approaches: indirect, direct, and hybrid approaches, which depend on how the visual sensor images of the system are processed. Finally, in *mapless navigation*, the system does not use any explicit representation of the navigation space. They are more based on object recognition that exist in the navigation environment and/or tracking them using their motion visually observed. Odometry is one of the most common mapless navigation technique. It is based on optical flow and features tracking-based methods.

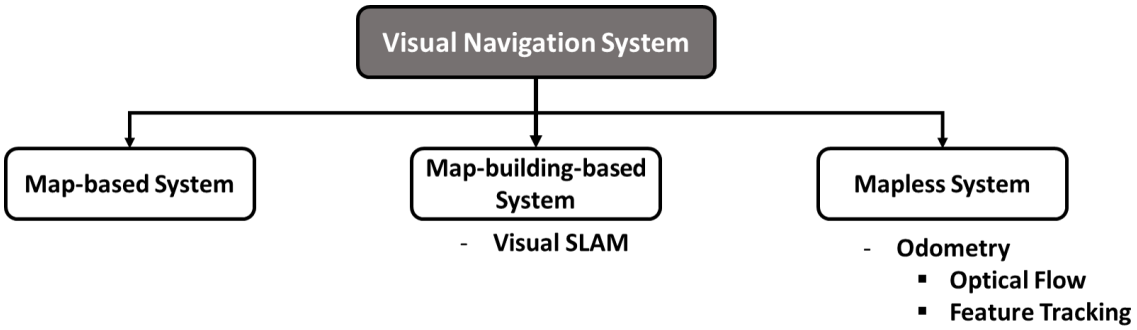

FIGURE 1.1: Example of a common visual navigation systems classification

In addition, according to recent literature reviews as [18] and [146], vision-based navigation systems can be classified according to other criteria. For instance, it is possible to distinguish between filter-based navigation and optimization-based navigation (figure 1.2). This classification is among the relevant ones discussed recently. It relies on the processing techniques used to solve the navigation problems.

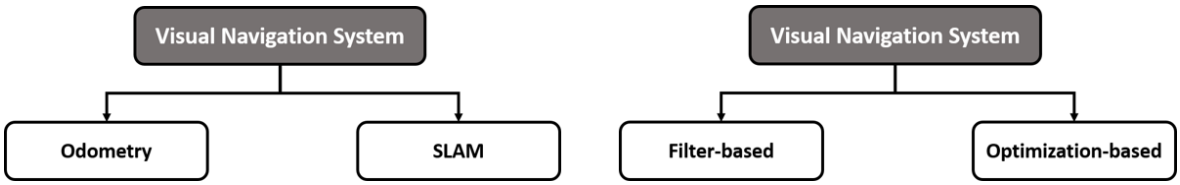

FIGURE 1.2: Visual navigation systems classification discussed in this section

Beside the previously presented classifications, vision-based navigation systems can be classified according to other factors [18], *e.g.* according to the navigation vehicle type, sensor configuration, etc. However, it should be noted that currently a system may belong to one, two or more different categories at the same time. For example, there are various filter-based odometries as well as other optimization-based odometries, same for SLAM, etc. (figure 1.2). This diversity comes back to the significant improvements that navigation systems have achieved in recent years.

In this section, the most common and recent classes of the visual navigation systems are discussed, as well as the main pose estimation techniques are addressed. Firstly, visual odometry and

visual SLAM are introduced, then the differences between them are presented. Subsequently, the optimization-based and the filter-based techniques are discussed.

1.1.1 Visual Odometry and SLAM

Many studies have recently been carried out on the Odometry and Simultaneous Localization and Mapping (SLAM) approaches, as example [18][163][73]. These techniques are mainly applied to compute vehicle position and orientation from data obtained using sensors mounted on it.

Odometry approach comes from robotics domain. It uses the motion sensor data or visual sensor data (camera frames), to estimate the pose changes over time. As the focus of this chapter is on vision-based systems, the discussion below will only focus on Visual Odometry (VO). In the computer vision community, this name refers to a family of low-level techniques used to calculate the current pose. For this purpose, changes of relative position and orientation are added to the previous pose, and are estimated by comparing two successive frames taken at two different locations by the processed system's embedded cameras during its motion. In fact, the main aim of VO is to ensure the local coherence of the estimated trajectory. As discussed before, VO can be categorized in different ways [146]. In particular, it is possible to distinguish between Optical Flow (OF)- and Feature Tracking (FT)-based methods.

On the one hand, optical flow-based approach uses raw visual measurements (pixels) for estimating a vehicle's location. This approach can also be described as direct method. Basically, an optical flow algorithm takes as input an image sequence received from one or more cameras, and computes the 2D motion vector from one image to the next. Using the pixel intensity calculation, the algorithm allows the points motion between two successive images to be shown. Kanade-Lucas-Tomasi (KLT) is one of the most widely used optical flow. For this reason, its processing details are presented on section 1.3.3 to illustrate this type of approach.

On the other hand, FT-based methods, also known in the literature as indirect methods. They are based on the PoIs detection in each captured image using feature detectors, such as corner or edge detectors. There are different algorithms proposed for this purpose, most of them are based on corner detection (section 1.2). The operating concept of this approach is illustrated in figure 1.3. Firstly, once each image is acquired, the feature detection and matching is done. Subsequently, the epipolar geometry and error optimization are performed in order to finally have an optimized pose estimation. Actually, the main advantage of this kind of method is its robustness against geometric distortions and brightness inconsistencies. However, despite the advantages of PoIs, using only these features causes a significant loss in image information. This is because they represent visual information of an image, which can be subject to several problems such as brightness, texture and blur. In addition, it is more complex in terms of calculation and requires a significant computational effort and energy compared to the optical flow.

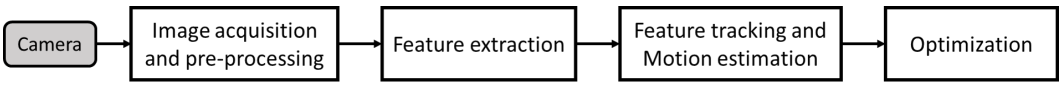

FIGURE 1.3: Visual Odometry steps overview.

Actually, odometry is implemented in SLAM methods as the first step of the process. It aims to obtain a moving system pose, followed by next SLAM steps such as local mapping and loop closing.

SLAM refers to a specific process used by a mobile system to build the global environment map and to maintain its consistency. There are various methods to perform SLAM [47]. Firstly, many

visual SLAM (VSLAM) applications consist of detecting and tracking PoIs in the image, these are called FT-based SLAM applications. As presented in [47][200], it is possible to distinguish between two types of methods: filter-based methods and Bundle Adjustment (BA)-based methods. Indeed, MonoSLAM [108], PTAM [117] and ORB SLAM [153] are among the most widespread methods in literature, whereas ORB SLAM remains the most used method. In fact, SLAM [56][11], particularly ORB SLAM [153], is made up of several parts: feature detection, feature matching, 3D pose estimation, global pose update and pose correction, and update or integration of new features on the map, figure 1.4 illustrates a global SLAM overview.

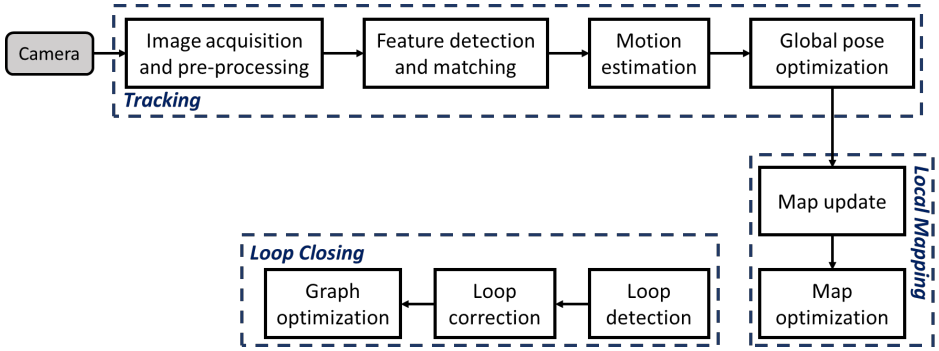

FIGURE 1.4: Visual SLAM steps overview

Implementing such a system is a tedious task, because it requires a map for localization and a good pose estimation for mapping. The SLAM problem in robotics is essentially to determine the features map and the system's trajectory from its controls and relative observations.

In ORB SLAM, map errors and pose estimates are correlated. Therefore, they must be corrected to obtain a global consistency of the trajectory and the map over time. This global consistency can generally be achieved using different optimization techniques such as BA [210] or pose-graph optimization [81]. Loop constraints are very valuable constraints for such optimization, they can be obtained by evaluating the visual similarity between current and past images. This visual similarity can be achieved by matching features. Besides, this already known information can be used to refine the map and the pose. This recognition process of an already mapped area or a previously visited place is known as the loop closing. Actually, ORB SLAM includes an interesting selection of various techniques used to improve the SLAM performances. Also, it has been improved by using different algorithms to manage different sensor configurations: monocular, stereo, RGB-D camera and even IMU/camera coupling. The disadvantages of these algorithms are the complexity and large number of input parameters as well as the complexity of the algorithms used.

In addition, among the well common direct SLAM works there are DTAM [156], LSD-SLAM [59], SVO [68], and DSO [58][215]. This SLAM category is known for using the whole image in its process without the need to extract features. Thus, it relies on photometric consistency to register two successive images. Recently, due to the development of deep learning, other works, belonging to this category, have appeared as [207] and [21]. But these are complex and time consuming, so they often require GPU implementations. Then there is RGB-D SLAM, which uses RGB-D cameras [78][233]. These cameras based on structured light can provide real-time 3D information but are only used in indoor navigation. This is to avoid problems related to range and sunlight. Furthermore, with the emergence and evolution of event cameras, event camera-based visual SLAM has appeared, as in [115][218]. Indeed, these cameras are bio-inspired image sensors that can provide an "infinite" frame

rate by detecting visual variations in the "visual event" images. However, as they are insufficiently mature, their performance for SLAM applications cannot yet be concluded.

In general, SLAM and VO are two important navigation approaches, however there are interesting differences between them. Table 1.1 summarizes a comparison between the main features of each approach. The visual SLAM provides a consistent global system trajectory estimation based on the

	VO	**Visual SLAM**
Definition	A method of estimating the ego-motion (camera's motion relative to a rigid scene) of a system using mainly input of camera(s) attached (frames)	V-SLAM is a process that allows system building a map of an environment and using it to know its pose from camera(s)
Goal	Aims to recover the trajectory incrementally, pose after pose	Aims to obtain a global, consistent estimate of the robot trajectory and keeping a track of environment map
Optimization	Optimizing only over the last poses	Uncertainty will be optimized, when loop closure is detected
Consistency	Only local consistency of the trajectory and the local map is used to obtain estimate of the local trajectory	Whereas SLAM is concerned with the global map consistency

TABLE 1.1: Comparison between VO versus visual SLAM [190]

environmental map record. In fact, the latter is used to close the loop when the robot returns to an area already visited, thus allowing to reduce the drift of the map and the camera trajectory, respectively. While the VO allows, first of all, to obtain the trajectory in a progressive way, pose after pose. As well, it enables optimization by using only the last trajectory positions (this is also called window BA) without using loop closure. In fact, the main objective of the VO is to ensure consistency of the local trajectory which gives a more accurate estimation of the local trajectory (e.g. using the BA). Actually, visual SLAM is in general more accurate than VO. Thanks to to its ability to solve multiconstraints problem for trajectory estimation, although this does not necessarily mean it is more robust than VO. Commonly, choosing between VO and visual SLAM relies on the application's need and the possible trade-off between performance, consistency and implementation simplicity. Visual SLAM may sometimes seem more appropriate, especially, when overall consistency of the camera trajectory is required. However, VO gives real-time performance without the need to keep a complete record of the camera's track history, which may be a constraint of the targeted system.

1.1.2 Back-end Optimization Types

Visual navigation systems can also be divided into optimization-based or filter-based systems. In this section each of these two categories is presented. Although in this chapter the interest is in purely visual methods, so this categorization will be largely confined to the visual SLAM approach. In the next chapter, dealing with inertial visual navigation systems, this categorization will be more general and odometry will be even more concerned by this categorization.

Optimization-based methods are implemented by solving the graph from scratch each time it evolves, but sparsifying it by removing everything except a small subset of the past poses T_i, as illustrated in figure 1.5. In some cases, the poses used are stored in a sliding window of the camera's most recent poses. Usually, it is a set of heuristically selected keyframes. While the other poses T_i and all their measurements X_i are rejected because they do not contribute to the estimates (gray elements on figure 1.5). Therefore, in visual ORB SLAM [153], this approach is also known as keyframe-based SLAM. It aims to calculate the camera pose according to the pose of the points of the 3D map already reconstructed. The goal is to reconstruct new 3D points if necessary, and to refine jointly the reconstructed 3D points and the camera pose for some selected keyframes in the sequence. Keyframe-based or optimization-based visual SLAM maps the features detected in the current frame

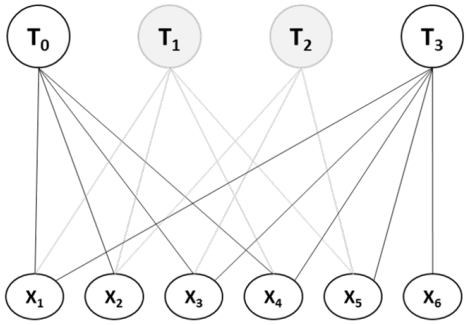

FIGURE 1.5: Optimization/Keyframe-based approach (T: camera pose, X: feature/landmark)

to the features already reconstructed (3D/2D mapping [190]), instead of mapping the features detected in the successive frames called (2D-2D mapping). This process is done using BA [210] which is a non-linear least-square refinement. This latter aims to minimize the reprojection error defined as the sum of the square residues between the detected points in the frames and the reprojections obtained from the computed model. Thus, the positioning accuracy is improved. Finally, the motion estimation is deduced from the resolution of the Perspective-n-Point (PnP) problem [223].

On the other hand, the filter-based approach, shown in figure 1.6. It marginalizes all poses that are different from the current pose (gray "T_i") and can keep the features that could be measured again in the future X_i. This approach is based on the construction of a probabilistic 3D map. this

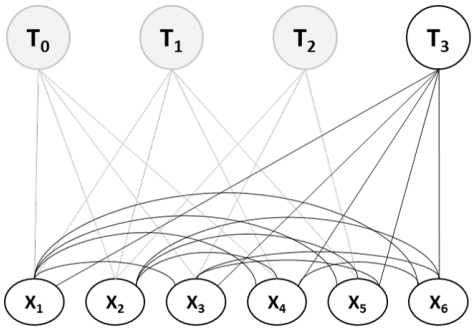

FIGURE 1.6: Filtering-based approach (T: camera pose, X: feature/landmark)

latter describes the current estimated camera pose and the reconstructed 3D map features in a global framework. The uncertainties related to the estimations are also provided. In fact, in the filter-based approaches, the map is initialized at the beginning of the process. So, as the system moves, the map is updated by different Bayesian filters such as Kalman Filter (KF), Extended Kalman Filter (EKF) or the Particulate Filter (PF), etc. It is mathematically represented by a state vector and a covariance matrix. In this case, the state vector consists of two parts: the estimated camera pose and the 3D features. Generally, the probability density function of the map parameters is approximated by a multi-variable Gaussian distribution.

1.2 Points of Interest Detection and Matching

As explained before, PoIs extraction, PoIs description and image matching are the cornerstone of image processing involved in features-based navigation systems (section 1.1). This work especially

deals with these steps. These processes are described below. So, to ensure the consistency of this section discussion, the introduction of concepts and terms that will be used is necessary:

Detector is an algorithm that uses an image as input and outputs a set of regions, in this work it is a set of PoIs.

Point of interest (PoI), according to Moravec, it is classically a dual discontinuity in the intensity function, gray-scale, depth, texture, geometry, and others of the image characteristics, also known as keypoint. Figure 1.7 shows several examples of the different types of points of interest.

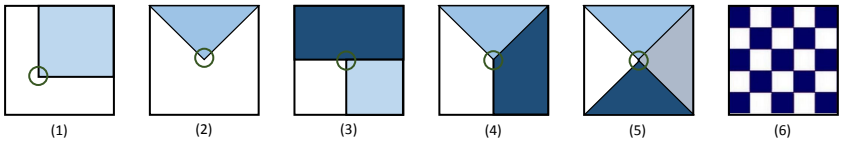

FIGURE 1.7: Several classical types of classical points of interest. (1) L junction, (2) V junction, (3) T junction, (4) Y junction, (5) X junction and (6) point of strong texture variations (checkerboard junction)

Descriptor is a representation of an intensity function (color,etc.) on an image region defined by a detector.

Matching aims to look, in two subsequent images, for the pairs of PoIs with the best similarities.

Quality evaluation metrics a detection/description algorithm can be characterized from others by different properties including repeatability, invariance, robustness and computational complexity. However, to evaluate PoIs detection/description algorithms and compare them together, a sufficient number of detected PoIs is a necessary prerequisite. Therefore, this condition can be included among the evaluation criteria of a detection/description algorithm:

- **Repeatability** of a detector: it is the ratio between the number of point-to-point matches and the number of points detected in a given pair of images [27]. It is related to an algorithm's robustness facing photommetric and/or geometric transformations that can exist between two images to match. Indeed, a repeatable detector is a detector that is robust to the changes and transformations that can be subjected to an image.
- **Robustness** of a description: it is defined by its **invariance** level against the different changes and transformations that the image can undergo. Among the most influential transformations: rotation, scale change (zoom), viewpoint change, affine transformations, brightness (light change), compression, etc. Figure 1.8 illustrates the different changes and transformations mentioned above that allow evaluating the robustness of a detector/descriptor.
- **Computational complexity**: it is the metric that identifies the computing elementary steps. Generally, it depends on the size of the input data. According to some studies, the computational complexity can be expressed in runtime (calculation time), which refers to the amount of steps required to solve a given problem.

Image processing steps listed above are considered as preliminary ones used in a wide range of computer vision applications, like 3D recognition, scene analysis, interpretation, tracking, localization, and object detection. All detection-description algorithms presented in this section are the most

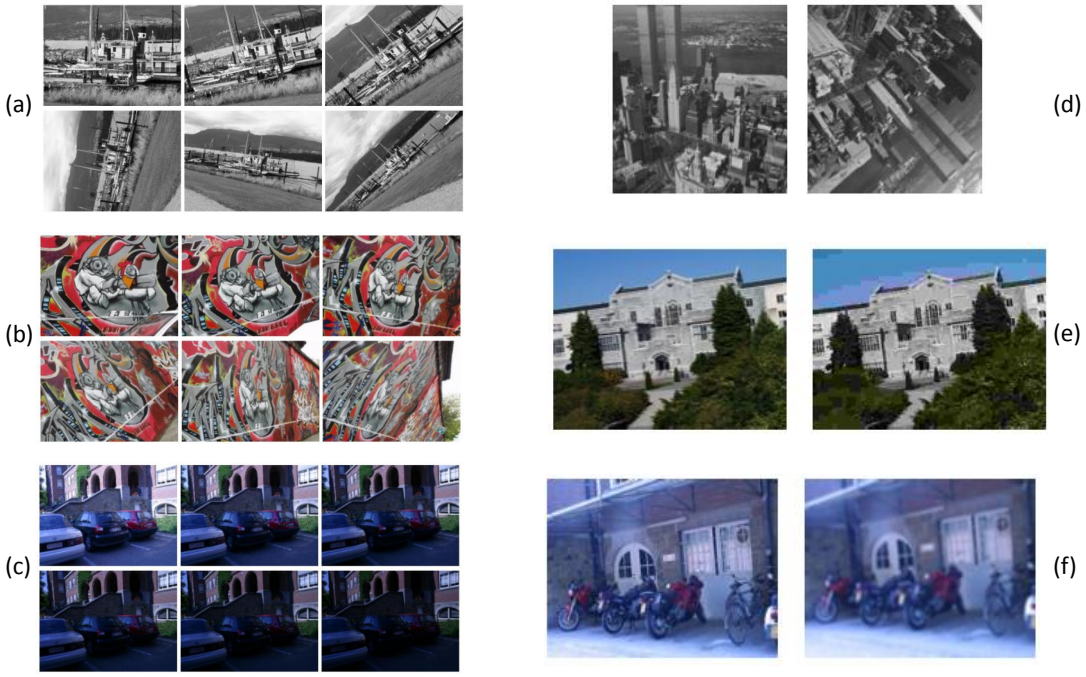

FIGURE 1.8: Examples of sequences subject to different image transformations : (a) scale and rotation change, (b) viewpoint change, (c) light change , (d) pure rotation, (e) JPEG compression and (f) blur

common used ones. However as presented in figure 1.9, deep learning techniques are now increasingly employed. These techniques are very efficient on many learning and pattern recognition tasks, providing precise and accurate results [160][225][36]. Nevertheless, these methods remain inappropriate for real-time application since they cannot achieve real-time results [43][12][134], which is a critical challenge for an autonomous embedded mobile navigation systems. In fact, as discussed in

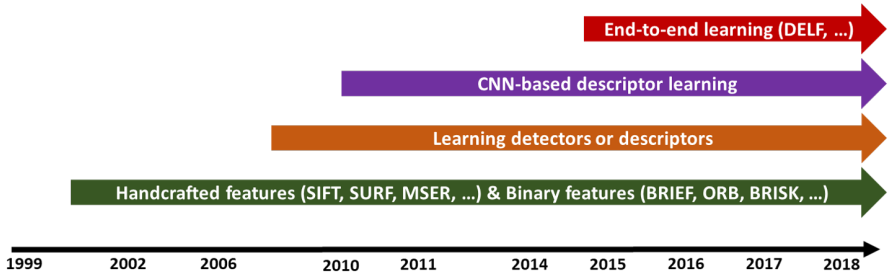

FIGURE 1.9: Methodologies and example methods chronological overview (inspired from [43])

[214], handcrafted algorithms are proven consistent, and optimized in terms of performance and energy efficiency. Whereas learned methods provides increased accuracy and flexibility, but in return it demands large amounts of computing resources. In other words, billions of additional mathematical operations and an increased need for processing power are required, primarily for training purposes and, more modestly, for inference. For this reason, a specialized hardware is recommended for developing such applications. For instance, high-performance GPUs [10], TPUs [188] for training, and accelerated IA platforms such as VPUs for inference [14].

Actually, the use of learned methods is sometimes exaggerated, because often handcrafted techniques can solve a problem much more efficiently and with fewer lines of code than learned ones. Handcrafted methods are based on a generic techniques and algorithms that work in the same way for any image, unlike learning-based ones. These latter are specific to the training dataset. In addition, the image resolution has an impact on the learned methods-based applications output. Indeed, high resolution images/videos must be used in order to obtain adequate performance, so this causes an increase in the amount of data to be processed, stored and transferred, which is already large.

1.2.1 Handcrafted Methods

The state-of-the-art proposes different handcrafted PoIs detection/description methods. Since 1976, many handcrafted methods of PoIs detection and description, as opposed to recent methods based on the deep learning, have been developed. Some of them are based on the local intensity change(s)[116], others rely on the use of a Hessian matrix[148] [82]. However, the detection algorithm choice depends mainly on its use case. In addition, the PoIs description, after they are detected, is a required step. It enables them to be used, as well as the salient visual information contained therein, in various PoI-based computer vision applications. Indeed, descriptor is a characteristic representation (color, intensity, scales, gradients, neighbourhood information, etc.) of the PoIs obtained by the detector. It is computed on an image area defined by the detector, this area is called the support of the description.

A PoIs descriptor must be robust to different constraints (motions, luminosity, etc), distinctive, effective, compact, and representative. There is a variety of PoIs detectors and descriptors. In this section, some examples of a local detector-descriptor algorithms, commonly used in the recent years for object tracking and pose estimation, are presented.

Harris is one of the widely used detection algorithms. It was developed in 1988 [82], and was then improved to Harris-Laplace in 2001 [144], and to Harris-affine in 2004 [145]. Features from Accelerated Segment Test (FAST) is a competing algorithm of Harris. It is efficient for the PoIs detection but it has a high computational complexity. Also, it can be used for PoIs description. Other algorithms exist, which combine detection and description, such as the Scale Invariant Feature Transform (SIFT)[132], the Speeded-up Robust Feature (SURF) partly inspired by the SIFT[16] or the Oriented FAST and Rotated BRIEF (ORB)[183]. This latter is based on FAST PoIs detection and Binary Robust Independent Elementary Features (BRIEF) description [183]. ORB provides fast detection and good algorithmic performance [64]. These algorithms are presented with more details in annexe A. In addition, a large variety of PoIs description methods have been also proposed such as DAISY descriptor. It is inspired by SIFT descriptor and aims to accelerate the computation time and to better deal with several types of invariance [209]. Also, for the same objectives as DAISY, there is Cheng descriptor which is presented in [37] and is based on a multi-size support regions centered on the PoIs. As well as the affine photometric model used in the inertial-aided KLT algorithm in [100].

Actually, to detect, describe and match PoIs, there are different algorithms as listed above. The choice of the good algorithm depends on the application requirements, as well as the different algorithms characteristics, such as their change invariances and computational complexity. For example, in embedded systems, the most important constraints for choosing a PoIs detection/description algorithm are the computational complexity, computation time and memory. To evaluate the different detection/description algorithms, different comparative surveys were conducted [121] [112] [105].

For this purpose, they used specific comparison metrics such as the number of detected PoIs, detection/description computation time, detector repeatability, point matching rate, etc. Table 1.2 summarizes the analysis of the experiments conducted in [121] [112] [105]. The results of these works assert that the SIFT descriptor is the most robust to affine transformations, followed by the SURF, BRIEF, DAISY and ORB descriptors. Moreover, Fast, Harris, Harris-Laplace, and Harris-affine algorithms are not robust to this type of transformation. For rotations, the ORB, SIFT, SURF, Fast, Harris, Harris-Laplace, and Harris-affine are robust to such changes. But SIFT remains the most robust followed by ORB, SURF, Harris-affine, and DAISY.

Algorithm	Invariance	Descriptor Size (byte)
Detector		
Harris	Rotation, Translation	NP
Harris-Laplace	Rotation, Translation, Scale	NP
Harris-affine	Rotation, Translation, Scale, Affine transformation	NP
Fast	Rotation, Translation, Scale	NP
Descriptor		
DAISY	Affine transformation, Translation	13 [219]
BRIEF	Affine transformation, Translation	32 [157]
Detector & Descriptor		
SIFT	Rotation, Translation, Scale, Affine transformation	128 [157]
SURF	Rotation, Translation, Scale, Affine transformation	64 [157]
ORB	Rotation, Translation, Scale, Affine transformation	32 [157]

TABLE 1.2: Performance characteristics: table of the different algorithms treated and the storage size of each descriptor vector.
(NP: Non Pertinent)

The choice of the relevant algorithm in terms of invariance or robustness is related to the application's requirements. For instance, in the case of embedded mobile robotics used for SLAM systems, robustness to rapid translation and rotation, and scale changes is assumed to be important. Indeed, these systems are vulnerable to blur caused by vibrations and rapid motions caused by fast rotation and scale changes. However, it is important to keep in mind that they have limited processing resources, which also have to be considered when choosing appropriate algorithm. These issues are addressed in different papers in terms of runtime, as well as in terms of descriptor size which allows predicting the memory required for each algorithm [206][105][12]. Actually, both SIFT and ORB algorithms remain among the potential candidates for this type of systems, thanks to their large number of detected PoIs and their large invariances to these movements. Although, these two algorithms are different in terms of their computation properties. More precisely, ORB algorithm is less complex and faster than SIFT [12]. Thus, the most relevant detection and description algorithm for embedded mobile robotics applications is ORB algorithm. It has good robustness and good performance against such applications requirements, as well as a less complex and faster processing.

Indeed, it is possible to have different combinations between detection/description algorithms to make the system more robust and improve its accuracy [17] [85]. For example, coupling the Harris detector with the ORB descriptor (rBRIEF) in the case of pure rotations, as well as the Harris detector with the BRIEF descriptor to have an invariance against viewpoint changes [85]. Figure 1.10 generally summarizes the trade-off between computational complexity, expressed in terms of computing time, and the storage required by each algorithm (descriptor) [85] [121]. This storage depends on the number of PoIs covered by each algorithm and the size of the descriptor.

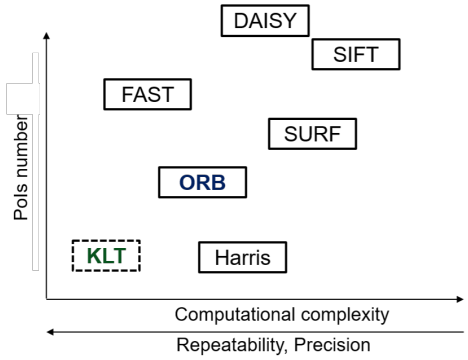

FIGURE 1.10: Trade-off between the different characteristics of the cited algorithms (KLT here represent Shi Tomasi PoI detection).

1.2.2 Learned Methods

In the previous sections, several relevant PoIs detectors-descriptors in the literature have been presented. These are generally handcrafted approaches.In this section methods that attempt to avoid handcrafted ones, using algorithms that enable the learning of the PoIs detectors and their descriptors are presented.

Learned detection-description methods are proposed to ensure a better optimization of the data used. However, they usually require massive computing power for learning and inference, *eg.* for object recognition, object extraction or object tracking applications. They also require voluminous datasets that depend on each application classes. Furthermore, the size of the learning model, the possibility of implementing parallelism, etc. influences complexity of learned methods-based applications. Learning-based algorithms are generally either supervised or unsupervised [43]. Based on the various benchmarks made in literature [12], it is not obvious to identify a handcrafted or learned technique suitable for all kinds of applications. This is mainly dependent on the targeted application and the expected implementation pattern. In the following, the focus will be on deep learning-based local invariant features.

Deep learning is about the training and inference of Convolutional Neural Networks (CNNs) with multiple layers (*i.e.* deep). This is an automatic learning of features at several abstraction levels. For this purpose, the input and output are mapped directly from data without completely depending on the user defined features. Thus, deep learning models consist of a succession of convolution layers and intermediate layers. The latter's role is to introduce redundancy through training, compact or average information and to prevent over-fitting. The system is trained iteratively by back-propagation, using training data samples called batches. During the inference (or at the time of the test), the CNN processes the input, which is the image, and produces the result for which it was trained. These models have recently revolutionized the computer vision field, so many different methods have been proposed for local descriptors based on deep learning or algorithms for end-to-end PoIs detection/description based also on deep learning such as [160][225][36].

Example : MatchNet [225] it consists in learning a deep network for patch representation and a network for robust feature comparison, both jointly. Indeed in MatchNet, each patch passes through a convolutional network to generate a fixed-dimensional representation that looks like SIFT. The representation comparison is done using a distance measurement learned and implemented as a set of fully connected layers, unlike SIFT where two descriptors are compared in the feature space using the Euclidean distance. Figure 1.11 illustrates the MatchNet architecture.

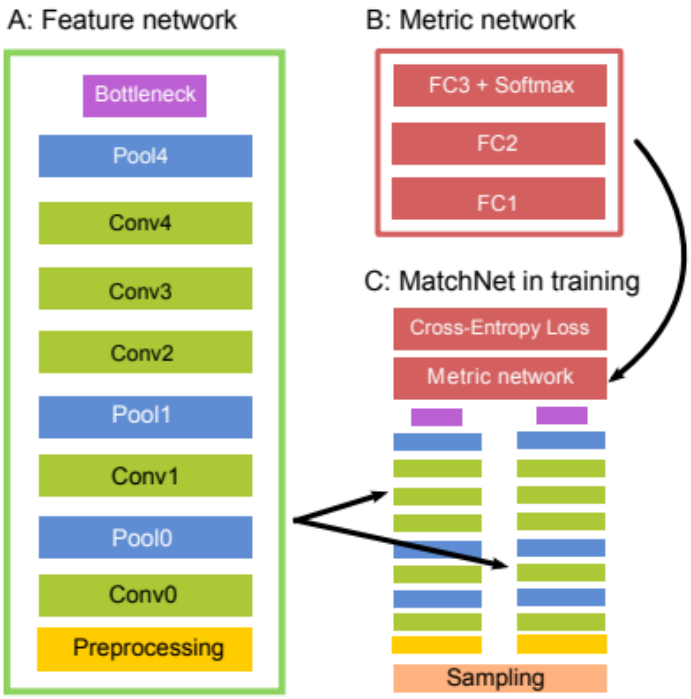

FIGURE 1.11: Global workflow of MatchNet system [225]

The MatchNet architecture is composed of three main blocks. First block A which represents the feature network, it is used for feature encoding, with an optional bottleneck layer to reduce the size of the features. Secondly, there is Block B, known as the Metric Network. It is used for the characteristic comparison. And the latest one is block C, which shows MatchNet in training. Here, the feature network is applied in the form of two "towers" on pairs of patches with common parameters. The output of the two towers is concatenated as an input to the metric network. The entire network is jointly trained on pairs of labelled patch pairs produced from the sampler to minimize the loss of cross entropy. In prediction, the two sub-networks (A and B) are used in a practical way in a double-level process pipeline.

Example : DEep Local Feature (DELF) [160] it is a new local detector/descriptor adapted to a large-scale recognition system (+1M images). Its principle is illustrated in Figure 1.12. First, and as indicated in the highlighted green part of the diagram, extraction and selection of features is made in order to assign Attention Scores and select the features with the highest scores.

The proposed retrieval system can be decomposed into four main blocks: (1) dense localized feature extraction, (2) keypoint selection, (3) dimensionality reduction and (4) indexing and retrieval. DELF for database images is indexed offline. Feature extraction from an image is done by applying Fully Convolutional Network (FCN), in particular FCN is taken from the ResNet50 model and trained on ImageNet. Then, instead of using all founded features, a pre-selection step is proposed. It consists in applying a 2-layer CNN to attribute weights to the features and finding the most relevant ones. The originality of the method lies in the fact that the selection of points of interest is made after their description. The size of each descriptor is reduced with PCA (Principal Components Analysis), and then a codebook of visual words can be generated to deal with large scale collection.

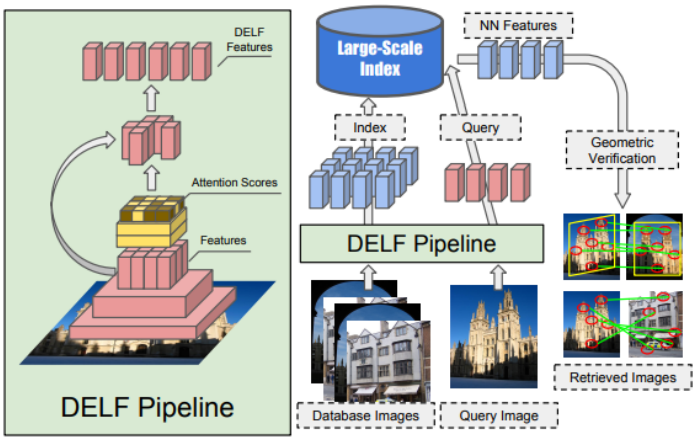

FIGURE 1.12: Global workflow of DELF system [160]

1.2.3 Points-of-Interest Matching

Matching aims to establish correspondences between two images using different techniques, with or without detection and/or description of features. In this section, different image matching approaches are presented and discussed.

There are different image matching methods in the literature [101][102][134], which can be divided into two categories: area-based methods and FT-based methods. In fact, the area-based methods may be called correlation-like methods, template matching [101] or image registration methods [134]. Not usually based on the image features extraction, they involve image registration and establish dense pixel-to-pixel matches based directly on the pixel intensity of the entire image. Such methods match two images by maximizing the similarities between sliding windows of predefined size or even the whole images [101][4][13][172]. Actually, area-based methods are more suitable for use in medical or remote sensing image registration. Because such images have a widely varied appearance due to the various imaging sensors, as well as they are often less textured. This makes them unusable by FT-based methods. However, this type of method suffers from geometric transformations and complex local deformations. Thus, FT-based methods are the most adapted ones for the system proposed in this thesis.

In FT-based methods, once the PoIs are detected and their associated local descriptors are determined, the image matching can be performed through PoI matching, from at least one image to another, using feature (PoIs)-based techniques. The purpose of this step is to find the observations of the same point among the detected PoIs, *i.e.* to determine which points in one image correspond to which points in another image.

As discussed in [101] and [134], PoIs-based methods can be subdivided into indirect and direct matching, corresponding to the use and non-use of local image descriptors. The first category is based on the use of spatial geometric relationships and optimization methods between two sets of PoIs. However, for the second one, it uses an alternative method based on the comparison of the detected PoIs descriptors. This typically consists of a matching using the measured similarity, according to a computed distance between two descriptors, in the measuring space. Among the most well-known metrics are the distance from the city block (standard L-1), the Mahalanobis distance, the Minkowski distance [83], etc. In addition, specifically for non-binary characteristics, the Euclidean distance is also used, and for binary characteristics, it is the Hamming distance. Subsequently, this is followed by a match sorting that allows to eliminate sets of presumed matches using additional local

and/or global geometric constraints. In fact, it is possible to have mismatches which are an incorrect matches due to background clutter or because the PoI is not detected in the other image. This will ultimately result in an inaccurate construction of reality. So, to avoid this, there are robust methods that are used to eliminate the *outliers*, for example the methods based on filtering of geometric constraints (for example RANdom SAmple Consensus (RANSAC) [65] method presented in appendix C), using the fundamental matrix and its alternatives.

Similar to PoIs detection/description, it is possible to perform matching using learning processes. Indeed, matching using training can be roughly classified into image-based training and point-based training. The first one is based directly on tasks without attempting to detect a salient image structure (*e.g.* PoIs, etc.) first. Generally, this learning focuses on image registration, stereoscopic matching, and camera localization. Whereas for point-based learning, it is done by applying operations on the sets of detected PoIs. Generally, it is used for classification, segmentation [33][174], and registration [195][127].

1.3 Visual Tracking

Pose estimation is an important task in different navigation methods. In particular, in FT-based methods, this task is performed immediately after feature detection/description and feature matching. In order to address this topic, firstly, *Structure from Movement* (SfM) is presented in section 1.3.1 since it is a visual technique for retrieving the camera's 3D poses from successive motions and a known set of 2D images. It is a general imaging technique, on which other more recent techniques have been based or inspired (VO and visual SLAM). Then, basic methods used to compute the pose and the trajectory are described. In particular, feature correspondences-based methods (2D-to-2D, 3D-to-3D and 3D-to-2D feature correspondences) are discussed in section 1.3.2.

1.3.1 Structure from Motion

SfM can be calculated in different ways. This depends on different factors, such as the number and type of cameras used, and whether the images are captured under controlled conditions. In the case of a single calibrated camera, the 3D structure and camera motion can only be recovered *up to scale*. *Up to scale* means that it is possible to resize the structure and amplitude of the camera motion while maintaining the observations. To compute the real scale of the structure and motion in global units, it is necessary to have additional information such as: the size of an object in the scene, and the information from another sensor, for example, an odometer or an inertial sensor. So, the process generally consists in finding the correspondences (points) between the scene and using multiview geometry to recover the scene and the 3D pose. Additional BA steps [210] are used to refine the SfM by minimizing the reprojection error.

In the simple case where two fixed cameras or a moving camera (the first and second images are considered as camera 1 and camera 2, respectively), the SfM algorithm assumes that camera 1 is at the origin and its optical axis is located along the z axis (figure 1.13).

Firstly, the SfM requires points matching between images either by using feature matching or by using point tracking from image 1 to image 2. For example, the KLT is used effectively for points tracking in the case of small camera movements.

Then, to compute the current pose relative to the previous one, it is necessary to compute the fundamental matrix, using the corresponding points found in the previous step of the computation. This matrix is used to describe the epipolar geometry of the two poses (current and previous) and it

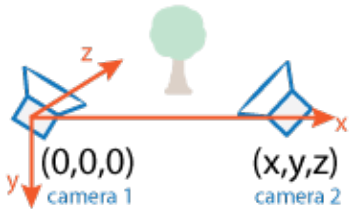

FIGURE 1.13: Structure from Motion from two views

connects a point in the current image to an epipolar line in the previous image, this is discussed with more details in appendix B.

Afterward, the 3D pose of the matched points is determined using triangulation. Indeed, since the pose is to scale, the structure is calculated with the right shape but not the actual size.

1.3.2 Motion Estimation Techniques

As explained in [190][70][71] as well as presented in the section before SfM and Motion estimation have a common feature, which is pose estimation. In fact, this latter is based on the computation of the camera displacement between the current image I_k and the previous image I_{k-1}. So, the complete camera trajectory is constructed by concatenating all these individual movements. This computation is done by using the correspondences between two sets of features f_k and f_{k-1} of two images I_k and I_{k-1} between two moments k and $k-1$. Depending on the type of these correspondences (2D or 3D), there are three different methods:

2D-to-2D Correspondences Generally, the geometric dependence between two images I_k and I_{k-1} is described by translation t and rotation R parameters. Figure 1.14 summarizes the workflow of the 2D-to-2D-based method. C_k and C_{k-1} represent the camera pose at time k and $k-1$, respectively.

$$T_k = \begin{bmatrix} R_k & t_k \\ 0 & 1 \end{bmatrix} \tag{1.1}$$

T_k, R_k and t_k represents the transformations, rotation, translation among the camera poses between time k and k_{-1}, respectively.

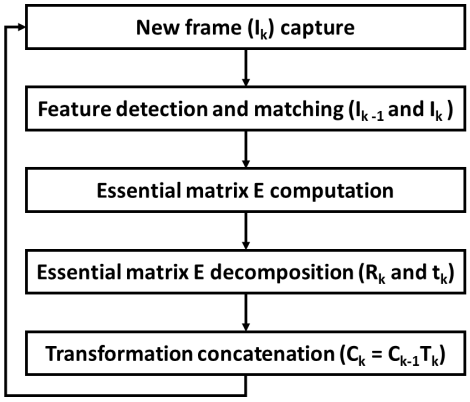

FIGURE 1.14: An overview of the 2D-to-2D algorithm [190]

In fact, these camera motion parameters are represented, up to scale factor unknown for translation, by a matrix known as the essential matrix and designed by E:

$$E \simeq tR \tag{1.2}$$

where $t = [t_x, t_y, t_z]^T$

$$t = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{1.3}$$

For a given essential matrix it is possible to have four different solutions (R, t). Assuming that the decomposition of E into singular values (SVD) [] is given by $E = U\Sigma V^T$. Where U and V are two orthogonal matrices of size 3x3 and Σ is a diagonal matrix given by :

$$t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{1.4}$$

The four solutions are:

$$R = U(\pm W^{-1})V^T t = U(\pm W^{-1})\Sigma V^T \tag{1.5}$$

Where

$$W = \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.6}$$

Thus the correct solution is selected by using the triangulation of a point, and choosing the solution where the point is in front of both cameras. Then, this rotation and translation must be optimized based on minimizing the reprojection error (equation 1.9).

In 2D-to-2D motion estimation, the absolute scale represents an input to be computed in order to be able to recover correctly the trajectory of an image sequence. In fact, as discussed in [70], an image sequence trajectory is a concatenation of the different transformations $T_{0:n}$. So, to calculate the absolute scale value, the appropriate relative scales must be computed. This is because the absolute scale of the translation cannot be computed directly from only two images. To do this, it is possible to apply the triangulation of the 3D points X_{k-1} and X_k of two successive pairs of images (figure 1.15). Then, a relative distance between any combination of two 3D points can be computed based on the

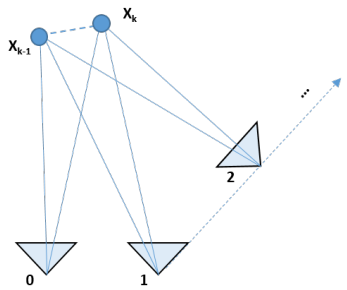

FIGURE 1.15: Motion and relative scale estimation

corresponding 3D points. Subsequently, the appropriate scale can be determined from the distance ratio r between a point pair at X_{k-1} and a point pair at X_k:

$$r = \frac{\left\| X_{k-1,i} - X_{k-1,j} \right\|}{\left\| X_{k,i} - X_{k,j} \right\|} \tag{1.7}$$

Aiming to ensure the robustness of the method, it is necessary to calculate the scale ratio for several point pairs as well as the mean or the median in case of the presence of outliers. Consequently, this ratio is used to scale the translation vector t. It should be noted that the relative scale computation requires that the characteristics be tracked over several images (at least over three images). In addition, it is possible to determine the scale by exploiting the trifocal constraint between the correspondences in three characteristic views [84] rather than using the triangulation of 3D points.

3D-to-3D Correspondences The 3D-to-3D features correspondences is used particularly in the case of stereo-vision [70]. As presented on figure 1.16, this technique determines the camera motion T_k by

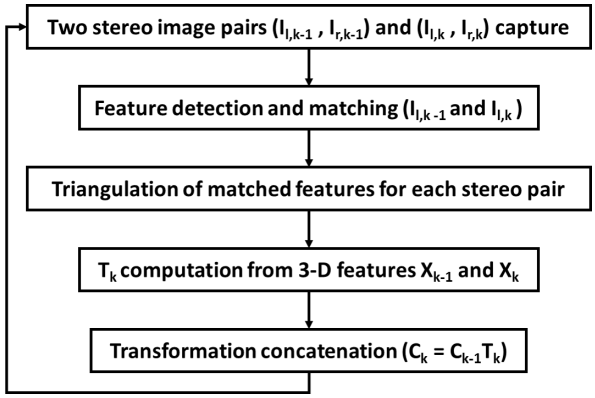

FIGURE 1.16: An overview of the 3D-to-3D algorithm [190]

computing the alignment transformation of the two 3D feature sets. Thus, the objective is to find the motion T_k that minimizes the distance L_2 between the two 3D feature sets:

$$argmin_{T_k} \sum_i \left\| \tilde{X}_k^i - T_k \tilde{X}_{k-1}^i \right\| \tag{1.8}$$

where the subscript i denotes the first feature, and $\tilde{X}_k$, $\tilde{X}_{k-1}$ are the homogeneous coordinates of the 3D points. The solution requires at least three 3D-3D non-collinear matches that can be used for a robust estimation even with outliers. So, in this case the possible way to solve the problem is to compute the translation according to the difference between the centroids of the 3D feature sets and the rotation component using the Singular Value Decomposition (SVD) method [79] [190].

3D-to-2D Correspondences The 3D-to-2D motion estimation (figure 1.17) is considered more accurate than 3D-to-3D motion estimation according to the work of Nister and al. [159]. This comes back to the error type to minimize, image reprojection error (equation 1.9) versus position of the 3D-to-3D features (equation 1.8). Thus, the transformation T_k is calculated by the 3D-to-2D correspondences X_{k-1} and p_k. X_{k-1} is estimated either from stereoscopic data or, in the monocular case, from the triangulation of the p_{k-1} and p_{k-2} measurements, which requires image matches between three views. So in this case the general problem formulation is to determine T_k that reduces the image reprojection error:

$$argmin_{T_k} \sum_i \left\| p_k^i - \hat{p}_{k-1}^i \right\|^2 \tag{1.9}$$

where p^i_{k-1} is the reprojection of the 3D point X^i_{k-1} into image I_k according to the transformation T_k.

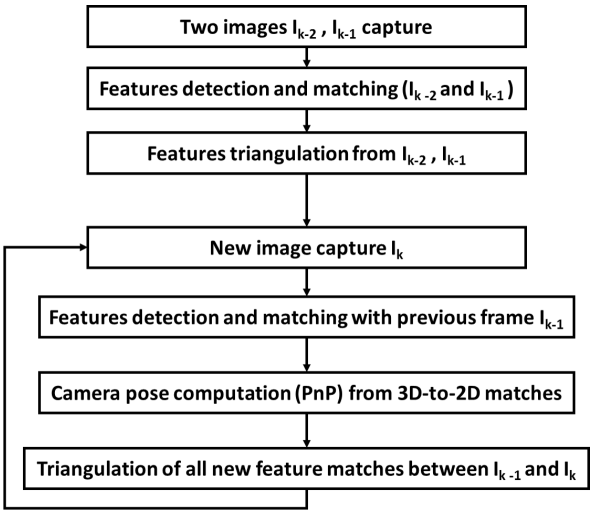

FIGURE 1.17: An overview of the 3D-to-2D algorithm [190]

This problem is known as Perspective from n Points (PnP), and it has many different solutions in the literature [149]. However, the minimal situation requires three 3D-to-2D correspondences [66]. In this case the problem is called the three-point perspective (P3P) and returns four solutions. In the case of 3D-to-2D, P3P is the standard method for robust motion estimation in case of outliers [66][71].

In the monocular case, the pose estimation based on 3D-2D alternative matching requires the triangulation of 3D points. This procedure is known as SfM and it is explained in more details above. Indeed, the initial set of 3D points and the first motion transformation are calculated from 2D-to-2D feature correspondences based on two views. Then, the successive transformations are calculated from the 3D-to-2D feature correspondences. In order to perform this process successfully, the points must be tracked on several images, at least three images. New features are triangulated anew at each calculated transformation and are added to the set of 3D features. The main objective of this method is to have a consistent and accurate set of triangulated 3D features, and to create 3D-to-2D features that correspond to at least three adjacent frames.

1.3.3 KLT Tracking

Kanade-Lucas-Tomasi (KLT) [202] is one of the most widely used optical flow algorithms in the literature. It is based on the assumption that the image flow is locally smooth as well as the image processing is performed only on a few pixels of it. It is considered as a sparse optical flow algorithm used to track features between consecutive images. In the case of optical flow methods for features tracking, when a pixel moves from the (x, y) position at time t to the $(x + dx, y + dy)$ position at time $t + dt$ (figure 1.18), its grayscale value does not change and remains constant.

Thus, the KLT features tracking algorithm supposes constant grayscale values. Consequently, the main KLT formula is defined as follows:

$$\mathrm{I}(x + dx,\ y + dy,\ t + dt) = \mathrm{I}(x, y, t) \tag{1.10}$$

'I' refers to image.

As Taylor's expansion is applied to the first order, equation 1.10 becomes:

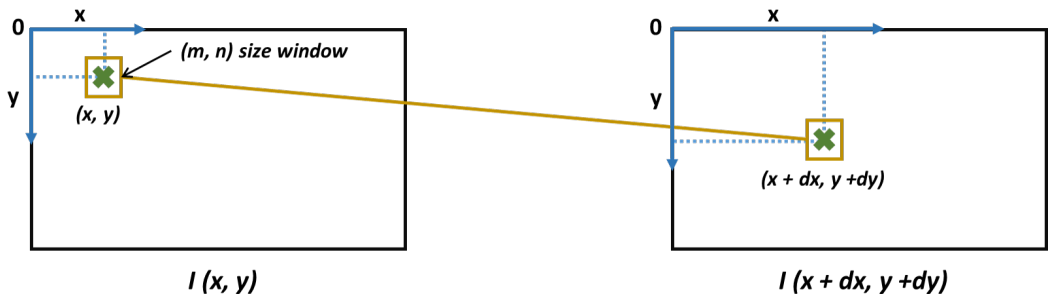

FIGURE 1.18: KLT feature tracking algorithm [202] diagram

$$\mathrm{I}(x+dx,\ y+dy,\ t+dt) \approx \mathrm{I}(x,y,t) + \frac{\partial \mathrm{I}}{\partial x}\delta x + \frac{\partial \mathrm{I}}{\partial y}\delta y + \frac{\partial \mathrm{I}}{\partial t}\delta t \tag{1.11}$$

so according to both equations 1.10 and 1.11 the following is obtained:

$$\frac{\partial \mathrm{I}}{\partial x}\mathrm{d}x + \frac{\partial \mathrm{I}}{\partial y}\mathrm{d}y + \frac{\partial \mathrm{I}}{\partial t}\mathrm{d}t = 0 \tag{1.12}$$

which gives the optical flow constraint equation by dividing the equation 1.12 by $\mathrm{d}t$:

$$\frac{\partial \mathrm{I}}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial \mathrm{I}}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{\partial \mathrm{I}}{\partial t} \tag{1.13}$$

where $\frac{\mathrm{d}x}{\mathrm{d}t}$ represents the displacement velocity of the pixel along the x-axis, and $\frac{\mathrm{d}y}{\mathrm{d}t}$ represents the displacement velocity along the y-axis. These two values are denoted by u and v, respectively. In addition, $\frac{\partial \mathrm{I}}{\partial x}$ and $\frac{\partial \mathrm{I}}{\partial y}$ are the pixel gradients in the x and y directions, respectively, and are labeled I_x and I_y, respectively. While $\frac{\partial \mathrm{I}}{\partial t}$ represents the grayscale evolution of the image over time and is referred to as I_t. So the expression 1.13 can be written as:

$$\begin{bmatrix}\mathrm{I}_x\ \mathrm{I}_y\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix} = -\mathrm{I}_t \tag{1.14}$$

The expression 1.14 above is a linear equation with two variables that can only be solved by a single pixel. The KLT algorithm considers that in a window whose size is (m, n) the motion of the pixels around the main pixel is identical, so we obtain $m \times n$ number of equations:

$$\begin{bmatrix}\mathrm{I}_x\ \mathrm{I}_y\end{bmatrix}_k\begin{bmatrix}u\\ v\end{bmatrix} = -\mathrm{I}_{tk}\ ;\quad with: k = 1, ..., mn \tag{1.15}$$

By defining:

$$\mathrm{A} = \begin{bmatrix}\begin{bmatrix}\mathrm{I}_x\ \mathrm{I}_y\end{bmatrix}_1\\ .\\ .\\ .\\ \begin{bmatrix}\mathrm{I}_x\ \mathrm{I}_y\end{bmatrix}_k\end{bmatrix},\ \mathrm{b} = \begin{bmatrix}\mathrm{I}_{t1}\\ .\\ .\\ .\\ \mathrm{I}_{tk}\end{bmatrix} \tag{1.16}$$

the equation (1.15) becomes:

$$\mathrm{A}\begin{bmatrix} u \\ v \end{bmatrix} = -\mathrm{b} \tag{1.17}$$

To obtain the components u and v of the pixel displacement velocity, the least squares solution of the over-determined linear equation is calculated as follows:

$$\begin{bmatrix} u \\ v \end{bmatrix}^{*} = -(\mathrm{A}^{T}\mathrm{A})^{-1}\mathrm{A}^{T}\mathrm{b} \tag{1.18}$$

So the pixel motion vector can be identified, as well as the position of the tracking point in the image can be computed. Generally, optical flow tracking is fast, especially, KLT algorithm due to the limited pixel number used as well as the image points number.

1.4 Vision-based Localization Systems Overview

Visual SLAM and VO are expected to run in real-time on an ordered sequence of images acquired from a fixed camera configuration (*i.e.* one or two particular cameras), whereas SfM approaches frequently use an unordered set of images often calculated in the cloud with little or no time constraints and can use different cameras.

Since in this thesis we are interested in monocular systems, we mention below the most known monocular competitive localization methods [205][113], including tracking, in the literature, *i.e.* EKF MonoSLAM [108], PTAM [117] and ORB-SLAM [153].

1.4.1 EKF MonoSLAM

In the early 2000s, A. Davison conducted historical work [108] [46] aimed at introducing vision into the SLAM named MonoSLAM. It uses image features to represent landmarks on the map. It iteratively updates the probability density of the feature depth by frame-by-frame matching to recover their 3D positions, thus it initializes a sparse feature-based map, and updates the complete state vector (robot pose plus 3D features pose) in an EKF.

Basically, as shown on figure 1.19 EKF monoslam consists of four main steps:

- Initialization: first of all, the initialization phase is carried out. During this step the camera is calibrated to ensure and improve the computation accuracy, as well as initial landmarks are computed to allow to begin the next computation process.
- Prediction: next is the prediction step. In this step the system state vector is described by: position, orientation (using a quaternion), linear velocities, and angular velocities. These parameters are used to calculate the kinematic model of the camera movement and consequently to predict its trajectory.
- Measurement & Tracking: once the model has been computed, the PoIs management is performed. This is based on the detection and processing of new landmarks. These are then tracked.
- Updates & Correction: in fact the difference between camera observation and prediction is called Innovation. The latter is the basis for this last step. The correction process updates the system status vector with the final position, as well as increases its accuracy by selecting the appropriate matched PoIs to retain according to their quality.

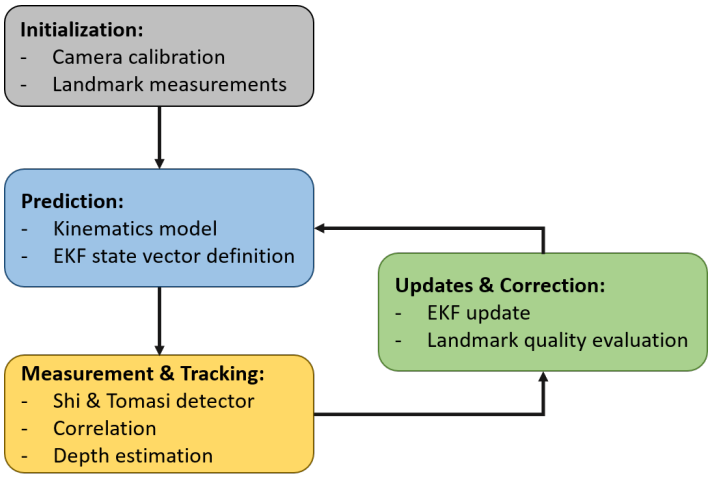

FIGURE 1.19: EKF MonoSLAM overview [46]

Many works have been based on this EKF monoSLAM to implement other improved SLAMs such as the work of J. Montiel [147] who proposes an EKF MonoSLAM that uses the inverse depth instead of the depth in the feature pose filter step, and which was improved in 2010 with the work with J. Civera [39].

1.4.2 Parallel Tracking And Mapping (PTAM)

PTAM is an algorithm originally designed for augmented reality applications and has proven its efficiency in real-time in small environments [117]. It is a feature-based algorithm and it is developed to estimate the pose of a camera in an unknown environment. As shown in figure 1.20, PTAM is

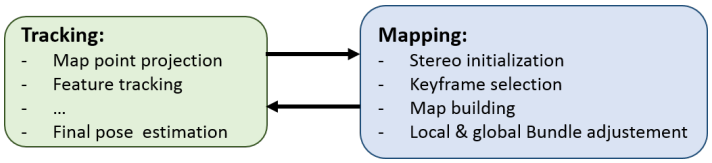

FIGURE 1.20: Brief overview of the main PTAM features [117]

considered the first work to separate tracking and mapping into two distinct processing threads, handled in parallel, generating a 3D map of the observed characteristic points. It is characterized by its support on the keyframes processing by the BA in the mapping thread. In fact, the map building process consists of 2 steps: firstly, it is built based on stereo technology which captures two images thanks to the camera's horizontal motion. Then, this map is continuously updated and refined using the newly selected keyframes. Secondly, the map can be built and optimized through the combination of the local and global BA [210].

Furthermore, the PTAM provides map points corresponding to FAST corners matched by patch correlation, which makes the points useful only for tracking but not for place recognition. The PTAM does not detect large loops, and relocalization is based on the correlation of keyframes in low resolution, which gives a low invariance to the viewpoint.

1.4.3 ORB-SLAM

More recently, a monocular SLAM approach with interesting properties has emerged as ORB-SLAM [153]. As it is depicted in figure 1.21, an ORB-SLAM system is based on three processing threads [9]:

- Tracking thread: where mainly the ORB PoIs extraction and tracking from one image to another is handled. This is accomplished by estimating the camera pose either by re-localization in the eventuality that the system is lost, or by using the previous frame in the case of regular system functioning.
- Local Mapping thread: in the first thread, a decision is made on the need to insert KeyFrame (KF). According to this, for each KF, the local mapping inserts the selected KF and starts the optimization of the system map (map points, KFs, visibility graphs, *etc*) and the implementation of the local BA.
- Loop Closing thread: this thread takes the last KF processed by the local mapping, and tries to detect and close loops. This is achieved by performing the following main functions: loop detection, similarity transformation analysis, loop fusion and essential graph optimization.

Its main properties are summed up in: it operates in real-time, in small and large indoor and outdoor environments. The system generates a compact and trackable long-term map and it is robust to severe motion clutter. Also, it allows wide baseline loop closing and relocalization based on indexing approaches, and it includes full automatic initialization. It has been recently optimized to deal with stereoscopic and RGB-D contents [154]. However, despite the algorithmic quality of ORB-SLAM and

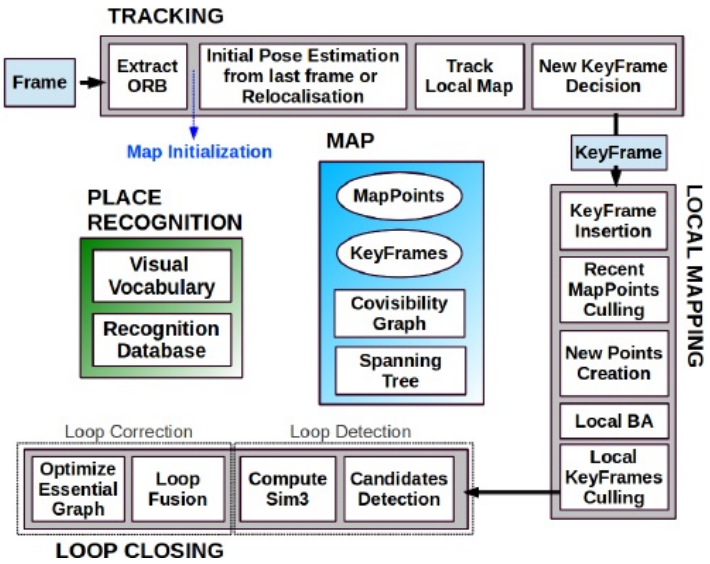

FIGURE 1.21: ORB-SLAM overview [153]

its robustness in different navigation environment scenarios, it still suffers from time consuming and computational complexity. Moreover, as with any other vision-based navigation and pose estimation method, its quality decreases dramatically in the case of difficult vision where the images to be processed are not useful. New improvements are then proposed to overcome these problems, including the various data fusion, especially visual and inertial data fusion [154][155], presented in section 2.2.1 of chapter 2.

1.5 Conclusions

This chapter describes the different approaches and methods available for visual navigation. Furthermore, the majority of visual navigation systems requires two main computation steps, firstly image

pre-processing, then pose estimation for tracking and localization. In this chapter, the different visual navigation classifications was presented with a focus on the most used ones which include odometry and SLAM approaches classification, and optimization and filter-based approaches classification.

Subsequently, as PoIs are among the most generic and accurate image features, the focus was on introducing and analysing relevant image processing and PoIs extraction methods. Also, the distinction of the ORB detector-descriptor as the most suitable algorithm for resource-limited systems was presented, since it can detect a sufficient number of PoIs for navigation application with the required invariances and interesting performances. In addition, this study helps to identify the KLT algorithm among the accurate visual tracking algorithm in case of short movements between two successive images; as well as the suitable visual tracking algorithm for embedded systems with limited resources. In fact, this algorithm based on optical flow process is lightweight because it does not require a heavy processing as for PoIs description. Next, various pose estimation methods was presented, focusing on the most commonly used methods in visual navigation and tracking systems including VO and VSLAM, as well as those based on feature correspondences.

Finally, a brief overview of the most widespread literature methods was presented. It illustrates the different previous vision-based algorithms and methods, and it discusses the different advantages and limitations of visual systems based on VO and VSLAM. Actually, these visual navigation, tracking, and localization systems provide accurate and reliable results. However, they are influenced by different environmental constraints, such as dark environments or motions that blur the image, which significantly reduces their robustness and causes the system to stall. In addition, the vision-based tracking and localization techniques suffer from the accumulation of errors due to the use of the dead-reckoning concept. Since VO is mainly based on this latter, it has a higher drift rate than VSLAM which combines dead-reckoning and BA to improve the localization accuracy but increases the computing charge and the optimization step complexity. Moreover, it should not be forgotten that a monocular visual system suffers from scale drift over time due to the use of a single camera. More precisely, VO systems are the more influenced ones by this issue than VSLAM systems. It is difficult to propagate the scale factor throughout the process because, with only one camera, this scale is not observable during the pose estimation process. Therefore, the scale factor becomes the direct reason for errors and drifts accumulated over time, especially when many PoIs are suddenly lost between two successive images. Thus, it is necessary either to use more than one camera (*e.g.* stereovision) which increases the computation complexity, or to combine the vision with other complementary data obtained from other sensors to improve the performance of the pose estimation and tracking.

Next chapter focuses on the joint use of inertial and visual data using the IMU/Camera coupling. Especially, it introduces and discusses the visual inertial navigation systems (VINS), including pose estimation for tracking and localization in odometry and/or SLAM methods. Different VINS categories will be defined, as well as the most relevant works of the state-of-the-art will be presented and discussed.

Chapter 2

Visual Inertial Navigation Systems

Multi-sensor fusion technologies are increasingly used, especially in navigation and tracking systems. This aims to compensate the estimated error of each sensor. Recently, research are getting particularly attracted by the use of visual-inertial fusion, based on low cost camera-IMU coupling. To better assist the rest of our thesis, it is helpful to introduce the IMUs. These latter are devices containing 3-axis gyroscope and 3-axis accelerometer (figure 2.1) [41]. They are self-contained sys-

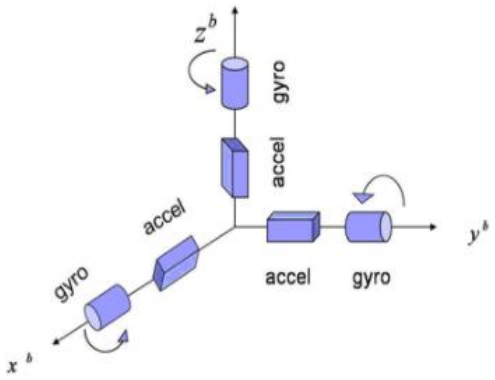

FIGURE 2.1: Conventional IMU (six dimensions)

tems that measure linear and angular motion of an object or vehicle. These units are mainly used for navigation purposes where the pose (position and orientation) of a device is of interest. In our times, many inertial sensors are based on Micro-Electro-Mechanical-Systems (MEMS) technology, in particular accelerometers and gyroscopes. These MEMS sensor functionality relies mainly on simple mechanical principles:

- MEMS accelerometer measures acceleration by measuring the change in capacity. Its microstructure, illustrated in figure 2.2, consists of a mass attached to a spring that is constrained to move only in one direction, and fixed outside plates. So, when acceleration in a particular direction is applied, the mass moves and the capacitance between the plates and the mass changes. This variation in capacitance is measured, processed and assigned to a specific acceleration rate.

- MEMS gyroscope uses the Coriolis effect to measure the angular velocity. Therefore, as shown in figure 2.2, when a mass is moving in a particular direction at a specific speed and an external angular velocity is applied, as shown by the green arrow, a force occurs, as illustrated by the blue and red arrows. This force causes the mass to move perpendicularly. Similarly to the accelerometer, this motion leads to a change in capacitance which will be measured, processed and will represent a specific angular velocity.

The MEMS components are small, light, inexpensive and have short start-up times and low power consumption. However, this kind of sensors suffers from reduced accuracy and bias instability, which are the main cause for the drift in standalone MEMS inertial navigation systems [221][91].

Officially, the IMUs are only the sensors, however, these are frequently connected with a specific software like sensor fusion. Actually, this unit is commonly used to collect inertial data that allow to track the position and orientation by the dead reckoning methods [224]. So, for this purpose, the acceleration is integrated successively to get velocity and position, respectively. Likewise, angular rate is also integrated to get the orientation with respect to the sensor/body framework. This integration process generates an accumulation of noise and estimation error, especially during long-term operation, which decreases the measurement accuracy. Nevertheless, one major benefit of using IMU is

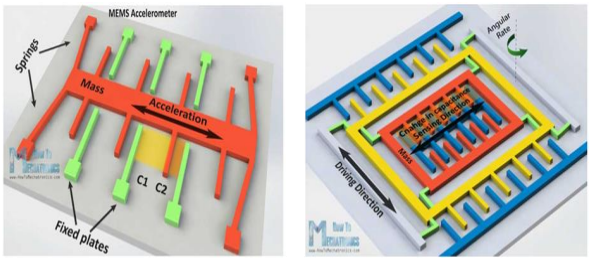

FIGURE 2.2: The micro-structure of the IMU MEMS's devices : Accelerometer & Gyroscope [140]

that it can produce pose measurements in almost any navigational context and hence it does not require any interaction with external environment to work. Unlike the camera, for example, which can no longer provide useful information in dark environments or in blurred motions. There is a wide variety of IMUs available according to the needs of each application. The price of a unit strongly depends on its sensitivity, accuracy and purpose. The IMU can cost in the order of a few euros, like those mounted in cell phones, for instance IMU chip BNO055 proposed by Bosch which consists of a $3 \times 4.5 \times 0.5mm$ chip and consumes about $5.2mA$ [192] (figure 2.3 (a)), to hundreds of euros such as ADIS16490BMLZ [57] (figure 2.3 (b)) which is proposed by Analog Devices Inc. and used for reliable avionics, precision instrumentation, *etc*.

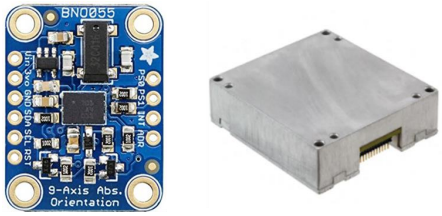

FIGURE 2.3: Inertial Measurement Unit examples : (a) Bosch IMU chip BNO055, (b) ADIS IMU (ADIS16490BMLZ)

Actually, camera and IMU sensors complement each other in many ways. On one hand, IMU measurements generally provide local acceleration and angular velocity, both in three axes. Currently, this sensor type are evolving in order to reduce its costs as well as its weight and size. Indeed the IMU gives good information on short term motion, however it has often been the subject of uncertainties because of its perturbations and biases. On the other hand, the camera provides a rich information, so it eliminates long term drift and allows large scale mapping and localization. Recently, modern high-performance, lightweight, small, low-cost and energy-efficient cameras have been successfully brought to the industrial market. The systems based on the fusion of IMU and camera data are known as Visual-Inertial Navigation Systems (VINS), and they are used in different application fields such as Micro-Aerial Vehicles (MAV) navigation [49][53], mobile Augmented Reality (AR) [80][5] and Virtual Reality (VR). There are many methods that combine IMU and camera to handle the VINS, including Visual-Inertial SLAM (VI SLAM), which handles VINS with loop closure, and Visual-Inertial Odometry (VIO). Depending on how IMU measurements are integrated, these methods can be classified into two approaches: loosely coupled fusion and tightly coupled fusion. Furthermore, orthogonally to the previous classification, these methods can also be divided

into filter-based methods and optimization-based methods.

Similarly to the visual navigation and tracking systems presented in the previous chapter, pose estimation is one of the main tasks of VINS. Due to the use of inertial data in addition to the visual data, the processing of this task is changing. Additional computation process are used for inertial data pre-processing and data fusion (section 2.2). Nowadays, there are different useful VINS algorithms for both VI SLAM and VIO. These algorithms are designed to be applied specifically to each VINS class.

This chapter addresses the VINS topic. Based on the literature, VINS are subdivided into two parallel classifications, depending to their back-end optimization type or to their sensors coupling type (figure 2.4). These classifications are firstly described in section 2.1. Subsequently, an overview associated to VINS as well as the pose estimation methods review are discussed in section 2.2, according to the back-end optimization classification. Before concluding, the VINS challenges and new trends are presented in section 2.2.4. And finally the conclusion is presented in section 2.3.

2.1 Systems Classification

As in visual navigation systems presented in chapter 1, VINS can be divided into different categories, mainly: filter- and optimization-based methods, and SLAM and VO methods. In VINS, inertial data is combined with visual data, so the latter category is known as Visual-Inertial SLAM (VISLAM) and Visual-Inertial Odometry (VIO). This data combination requires specific processing techniques, making the methods of VINS and algorithms partially different from those of visual navigation systems. Moreover, another particularity concerns the way in which the different sensors are combined: tightly and loosely coupled. In this section, these classifications are discussed with a focus on the main differences compared to pure visual systems. Firstly in section 2.1.1, the filter- and optimization-based categories are addressed by introducing the VI SLAM and VIO methods. Then in section 2.1.2, the different sensor coupling methods are discussed.

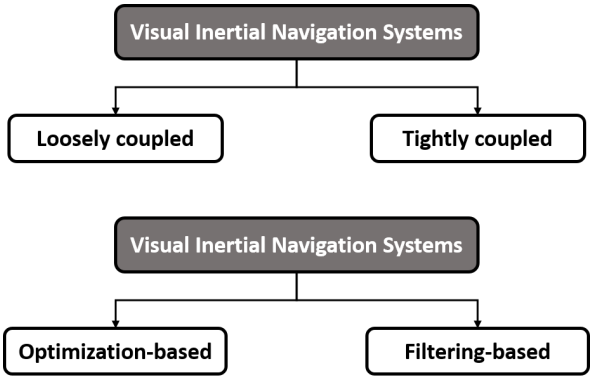

FIGURE 2.4: Overview of common VINS categories

2.1.1 Back-end Optimization Types

As discussed in [95], [32] and [146], filter-based methods are among the initial methods used to solve VIO and VI SLAM problems [166]. In this kind of approach, proprioceptive sensor measurements, such as IMUs, are used to compute the preliminary distribution of the system pose. Whereas measurements from exteroceptive sensors, such as cameras, are used for the construction of the probability distribution. Generally, in VIO case [93][20], inertial measurements (linear and angular velocities)

are used to compute the dynamic model of the system. The latter are considered as an input for the prediction step that predicts the vehicle motion. For visual measurements, features such as PoIs or others, are used for the measurement/observational model computation. This model is used to update the predictions in the update step. Filter-based VI SLAM [177][231] follows the same principle as VIO. However in these cases the models are non-linear in the majority of cases. These filter-based processing steps are defined as follows:

- Dynamic model(in case of non-linear system):

$$x_I = f(x_{t-1}, u_t) + w_t \tag{2.1}$$

where: u_t: the control vector w_t: the process noise, $w_t \sim N(0, Q_t)$ with Q_t is the variance.

- IMU status is expressed within a 16-size vector:

$$x_I = \begin{bmatrix} {}^{I}_{W}q^T & {}^{W}p_I^T & {}^{W}v_I^T & b_g^T & b_a^T \end{bmatrix}^T \tag{2.2}$$

where:
${}^{I}_{W}q^T$: the quaternion rotated from the world frame to the IMU frame
${}^{W}p_I^T$: the position of the world coordinate system
${}^{W}v_I^T$: the speed of the world coordinate system
b_g^T & b_a^T: the gyroscope bias and the accelerometer bias, respectively

- System prediction $x_{t|t-1}$ and measurement/observation z_t:

$$z_t = h(x_t) + n_t \tag{2.3}$$

$$x_{t|t-1} = f(x_{t-1}, u_t) \tag{2.4}$$

And the propagated covariance matrix is expressed as:

$$P_{t|t-1} = F_t P_{t-1} F_t + Q_t \tag{2.5}$$

where:

$$F_t = \frac{\partial f}{\partial x}|_{x_t, u_t} \tag{2.6}$$

- System update equations:

$$y_t = z_t h(x_{t|t-1}) \tag{2.7}$$

$$S_t = H_t P_{t|t-1} H_t^T + R_t \tag{2.8}$$

$$H_t = \frac{\partial h}{\partial x}|_{x_t} \tag{2.9}$$

Even if filter-based methods allow to perform system state estimation with high accuracy, they suffer from potential introduction of significant errors due to the linearization of non-linear measurements reducing system performance.

On the other hand, optimization-based methods allow the system to perform the pose estimation task using both visual and inertial measurements in a joint optimization process. For this purpose,

a non-linear least squares problem (Bundle Adjustment (BA) [210]) is solved on a set of data. BA process aims at refining a visual reconstruction to produce jointly optimal estimates of the 3D structure and viewing parameters (camera pose and/or calibration). This processing technique allows to outperform filter-based methods in terms of accuracy but with a higher computational cost. Furthermore, among the drawbacks of optimization-based methods, the high rate of inertial measurements increases the number of variables in the optimization, resulting in slow operation. Consequently, this issue makes these methods less suitable, or even unsuitable, for real-time optimization. To overcome these constraints, various works have been proposed. For example in [69], the VIO system is providing a preintegration approach to IMU measurements. This preintegration approach addresses the important stream of IMU measurements by combining them between a pair of keyframes into a single motion constraint. Mono VI SLAM [177], OKVIS (stereo) [124] and VI ORB SLAM (monocular) [155] are among the relevant optimization-based literature works. They ensure remarkable accuracy and interesting memory optimization, but they still have higher computational cost than filter-based systems. Finally, optimization-based methods offer better localization accuracy and lower memory consumption, whereas for filter-based methods, the advantages are mainly concentrated on computing resources [32]. Therefore, when choosing between a filter-based or optimization-based method, a trade-off must be made between accuracy and other constraints such as computing time and cost, depending on the final system application.

2.1.2 Loose and Tight Integration

VINS are multi-sensor systems, since they are composed mainly of camera for visual information and IMU for inertial measurements. There are different system configurations to fuse inertial and visual measurements. It is possible to divide them into loose and tight coupling-based systems (figures 2.5 and 2.6), in addition to the possibility of having hybrid systems that combine the tight and loose coupling [231]. This type of classification is orthogonal to the first one explained above, *i.e.* it is possible to have a loose coupling- and optimization-based VINS at the same time, or a loose coupling and filter-based VINS, *etc*.

Concretely, in a loose coupling system, filter-based or optimization-based approach, pose and orientation are computed by combining the estimation from two autonomous sub-systems, *i.e.* camera and IMU. The separately determined measurements are then combined afterwards to refine the system position and orientation (*i.e.* pose) [64][237]. Although loose integration is computationally efficient thanks to the reduction in computational complexity [95][146], it suffers from data inconsistency. Using a loose coupling approach involves separate processing of visual and inertial data. This leads to potential loss of information and thus reduces system performance. As shown in the table 2.1, the majority of high-performance systems are based on tight coupling.

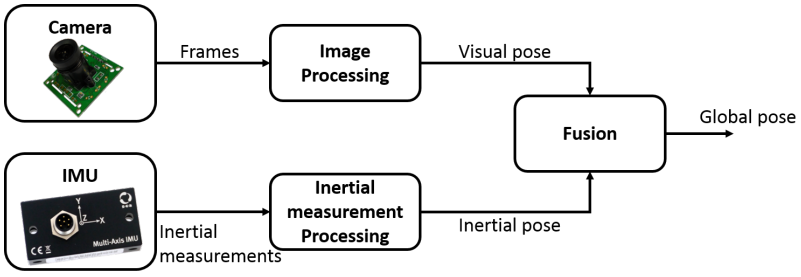

FIGURE 2.5: Loosely-coupled IMU/camera overview

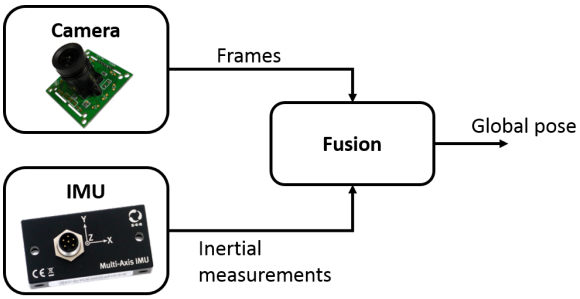

FIGURE 2.6: Tightly-coupled IMU/camera overview

In contrast, the tight coupling approach fuses visual and inertial data at an early stage, using a single high-order estimation filter, to achieve better accuracy (figure 2.6). Visual measurements can be obtained by detecting and tracking images PoIs, while inertial data can be the raw IMU measurements. Tightly coupled approaches perform a direct and systematic fusion of visual and inertial measurements and commonly provide better accuracy results. However, it leads to significant computational and implementation constraints, since the sensors frequencies are very different. Indeed, the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) are among the suggested solutions to be used to solve this problem [118] [229] (section 2.2.2). Tight integration is more precise than the loosely one but it is more complicated and requires more efforts for its execution and implementation [99, 158, 193, 231].

2.2 Overview of VINS

Most VINS based on filtering or optimization (VIO, VI SLAM, tracking, etc.) have several common processing steps, pose estimation is among the most common. Pose estimation methods used in this work take into account inertial measurements processing as well as inertial and visual data fusion. This aims at ensuring a correct and accurate pose estimation. Consequently, many works dealing with this subject are proposed in the literature. Table 2.1 gives a larger overview of recent state-of-the-art methods that are all tested on a common benchmark (EuRoC dataset [24]).

Year	Method	Back-End Approach	Fusion Type	Application
2015	OKVIS [124]	optimization-based	tightly-based	MAV
2015	ROVIO [20]	filter-based	tightly-based	Robotics
2017	VINS Mono [175]	optimization-based	tightly-based	MAV, AR
2017	VI ORB SLAM [155]	optimization-based	tightly-based	MAV
2017	Adaptive VI SLAM [168]	optimization-based	tightly-based	AR
2018	Trifo-VIO [235]	filter-based	tightly-based	Robotics
2018	R-VIO [93]	filter-based	tightly-based	AR
2019	VI EKF SLAM [177]	filter-based	tightly-based	MAV

TABLE 2.1: Classification of state-of-the-art competitive VINS algorithms (MAV: Micro Aerial Vehicles, AR: Augmented Reality)

In this section, an overview of the relevant literature work in both system categories, either optimization-based or filter-based methods, is discussed. These two categories are most concerned with the way of integrating in the computations and fusing inertial and visual data. Besides, an overview of some of the most relevant works is described.

2.2.1 Optimization-based Approaches

Adaptive VI SLAM presented in [168] is among the relevant optimization-based literature VI SLAM. In this work, the system initialization is based on inertial measurements. Indeed, the method proposed is a VI SLAM system, it consists of two odometry versions: an optimization-based VIO and a fast optical flow-based VO for real-time pose estimation, which are combined with the existing ORB SLAM [153] system. In order to provide a dynamically Adaptive VI SLAM, an adaptive execution module is also proposed in [168]. This module selects the most suitable odometer method between VIO and VO, according to the inertial-based system motion. The Adaptive VI SLAM [168] evaluation provides an effective performance in terms of tracking time and RMSE. In addition, S. Leutenegger, P. Furgale *et al.* proposed a VI SLAM system [124] called OKVIS, which is a widespread literature VI SLAM example. In this work the initialization step is processed by computing the preliminary state estimation of the system using the IMU measurements. Below are presented some of the most common optimization-based systems in the literature.

VI ORB SLAM [155] is one of remarkable VI SLAM works in literature. As depicted on figure 2.7, it is a monocular system based on the visual ORB SLAM [153] and presented in section 1. VI ORB SLAM can be considered among the tightly coupled algorithms that are optimization-based. It contains a sparse ORB SLAM front-end, a graphical optimization back-end, loop closing and relocalization. Unlike visual ORB SLAM where initialization is only vision-based, VI ORB SLAM proposes

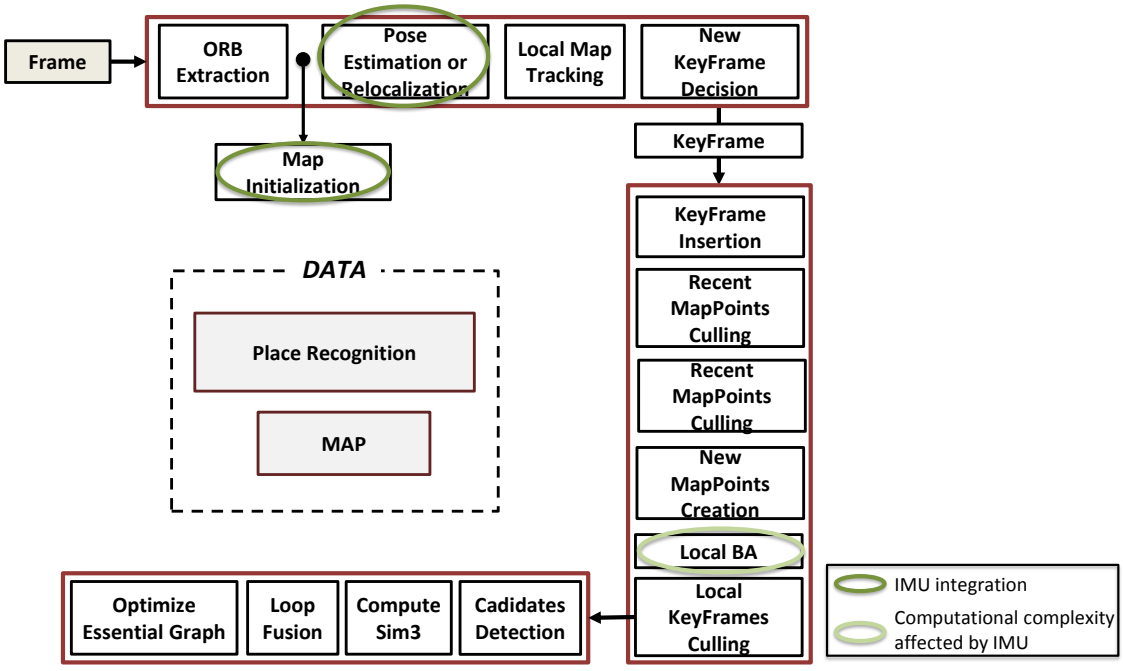

FIGURE 2.7: Functional diagram of VI ORB SLAM [155], focusing on the exploitation of inertial information in the global visual chain

a new method of IMU initialization, in parallel to vision-based one [155]. It is performed while a visual SLAM is running (ORB SLAM) and it consists of four steps:

1. Estimating gyroscope biases;
2. Approximating scale and gravity (without taking into account accelerometer bias);

3. Estimating accelerometer bias (refining scale and direction of gravity);

4. Estimating velocity.

As soon as the initialization step is done, inertial measurements are integrated in the processing workflow of the tracking stage, mainly for pose estimation (dark green circle in figure 2.7). Hence, as shown in figure 2.8, the optimization evolution in the VI ORB SLAM tracking thread is performed using inertial measurements. First of all, it starts with the optimization of frame j linked by an IMU constraint to the last keyframe i (figure 2.8 (a)). Then the result of the latter is used as a prerequisite for the next optimization (figure 2.8 (b)). As the two frames j and j + 1 are linked by an IMU constraint, and having frame j before the previous optimization, frames j and j + 1 are optimized when tracking the next frame j+1 (figure 2.8 (c)). The optimization ends with the marginalization of frame j and the result is used as a prerequisite for the next optimization figure 2.8 (d)).

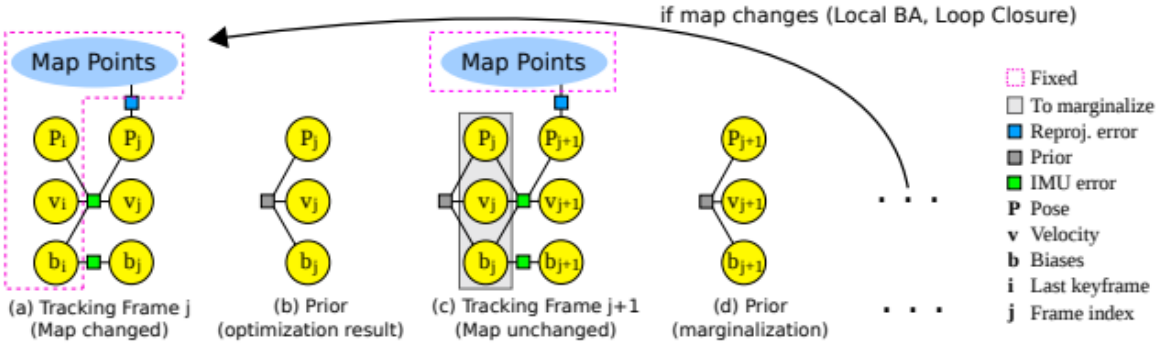

FIGURE 2.8: Evolution of the optimization in the Tracking thread of VI ORB SLAM [155]

It is the same for the local mapping, this thread is also affected by the IMU integration, especially in terms of computational complexity. Actually, the local mapping module uses the Local (LBA) to optimize the N last keyframes and all points observed on these keyframes after inserting a new keyframe. These Local Maps are then recovered according to the time series of the keyframe. So, as shown in figure 2.9, contrary to the ORB SLAM, in VI ORB SLAM the fixed window links the N+1 keyframes and the co-visibility graph.

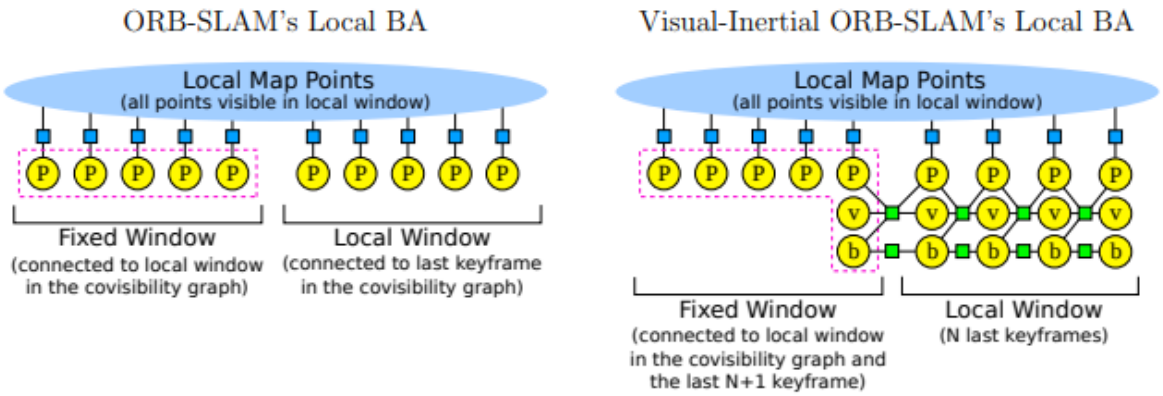

FIGURE 2.9: Comparison of Local BA between original ORB SLAM (left) and Visual-Inertial ORB SLAM (right). The local window in Visual-Inertial ORB SLAM is retrieved by temporal order of keyframes, while in ORB SLAM is retrieved using the covisibility graph [155]

VI ORB SLAM is characterized by its drift-free localization strategy [155], unlike visual odometry approaches where drift is boundless. In addition, the experiments listed in [155] show that monocular VI ORB SLAM recovers the scale metric with high precision and achieves better accuracy than stereo VIO. Especially, when it is continuously localized in the same environment. However, the main limitation of VI ORB SLAM is that the IMU initialization is based on visual initialization using a monocular visual ORB SLAM. Also, it can take more than 10 seconds for the scale convergence. However, this issue can be a problem for navigation systems that require scale estimates right at the beginning. To this aim, [175] provides "VINS Mono", presented below, as a recent optimization-based VI SLAM.

VINS Mono [175][175], called VINS Mono, this is an innovative VI SLAM method. It is characterized by an innovative front-end interface that uses the KLT optical flow to track the Harris corner, while the back-end interface uses a sliding window for non-linear optimization. In this work, initialization is performed using a loosely coupled sensor fusion that gives the initial values, then aligns the IMU preintegration metric and SfM results to obtain scale, gravity, velocity, and even bias. Actually,

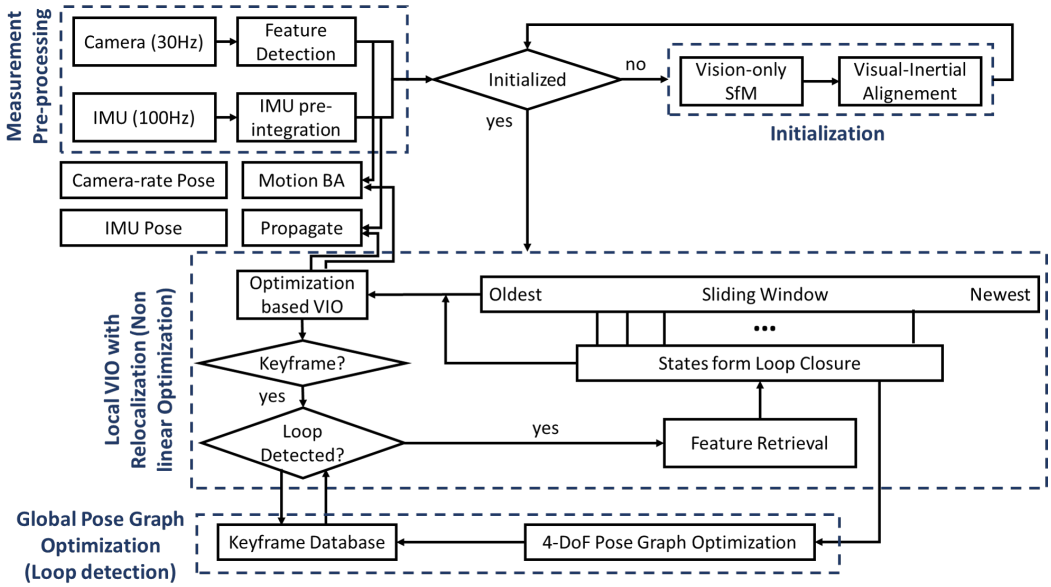

FIGURE 2.10: VINS Mono System framework, reproduced from [175]

as illustrated on figure 2.10, the system starts with the initialization procedure. The latter provides all the required values to start the non-linear optimization-based VIO. The VIO with relocalization modules tightly fuses the pre-integrated IMU measurements, feature observations and redetected loop closure features. Finally, the global pose graph optimization performs a global optimization to eliminate drift and achieve the reuse goal.

Monocular VIO [92], as it is illustrated on figure 2.11, is a monocular VIO system based on fixed-lag optimization. This optimization approach consists on maintaining a local map of fixed size and marginalizing out past states and measurements. Indeed, the use of the fixed-lag setting allows to optimize only recent observations and parameters. This results in a significant computational cost reduction. Monocular VIO proposes a framework based mainly on three major stages (figure 2.11): feature detection, initialization and structure from motion including the non-linear optimization based on inertial-visual fusion. This framework optimizes noisy inertial measurements and visual data

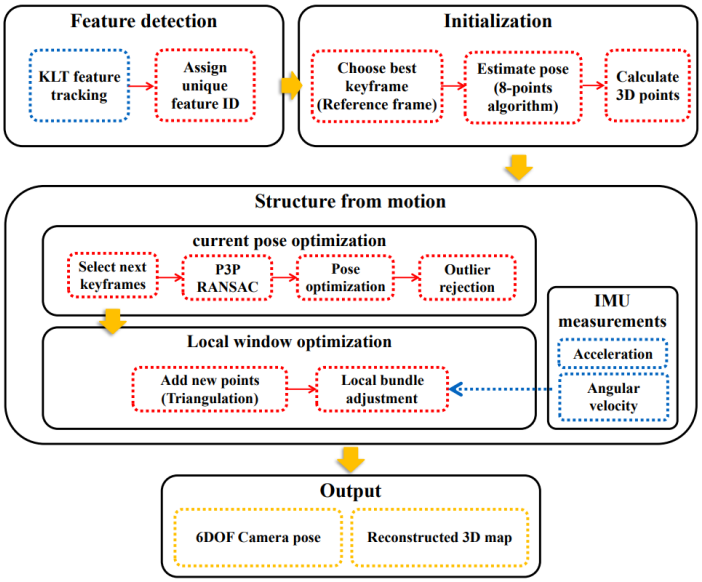

FIGURE 2.11: An illustration of the flowchart of the Monocular VIO algorithm proposed in [92]

directly in a single cost function. The latter consists of the inconsistency of transformations and parameters from IMU to camera, such as IMU biases, camera poses and feature positions. Updating these parameters is done progressively over time as new observations become available. For tracking, [92] uses the KLT algorithm, based on the Harris detector, which reduces the computational cost and makes the system more suitable for real-time applications.

2.2.2 Filter-based Approaches

Similarly to optimization-based systems, there are various filter-based odometry, tracking and SLAM works. In this section, a selection of the well-known filter-based systems is presented. Recently, M. Quan *et al.* propose an interesting filter-based VI SLAM algorithm known as Accurate VI EKF SLAM [177]. It provides competitive results, they are more robust and precise in different environments and conditions, with no need to pass by only visual initialization. It is initialized using the initial EKF VIO estimation using IMU and vision measurements processing.

In addition, another well-known work in the literature is R-VIO [93]. It is a VIO system based on the Multi-State Constraint Kalman Filter (MSCKF). For the initialization, the states are relative to a local reference frame and start from zero without the need for post-alignment to a fixed global frame. In addition, R-VIO provides a gain in terms of computation charge, but its quality accuracy is not really impressive. In the following, the main steps of Accurate VI EKF SLAM are described, as well as the R-VIO workflow is discussed.

Accurate VI EKF SLAM: in [177], a monocular VI SLAM system based on tight coupling is proposed and considered among the filter-based algorithms. This accurate VI EKF SLAM is able to estimate the pose and motion in an accurate, robust, long-term and high frequency way. This system works by launching parallel threads of processing. It consists of three threads: the first one is the EKF VIO with the feedback mechanism. The second one is the thread in charge of building the map and executing the LBA. Then, the last one is the loop closing thread. These last two threads are used mainly to construct a consistent global map.

The Accurate VI EKF state vector is composed of the IMU pose and a set of landmarks. As shown in figure 2.12, the EKF uses multiple landmarks (inliers) (red crosses on figure 2.12) simultaneously to estimate the camera pose. However, the MSCKF used in R-VIO [93] (presented below), it uses a single landmark **x**, so the camera pose is estimated using several previous camera poses where the landmark **x** has been observed.

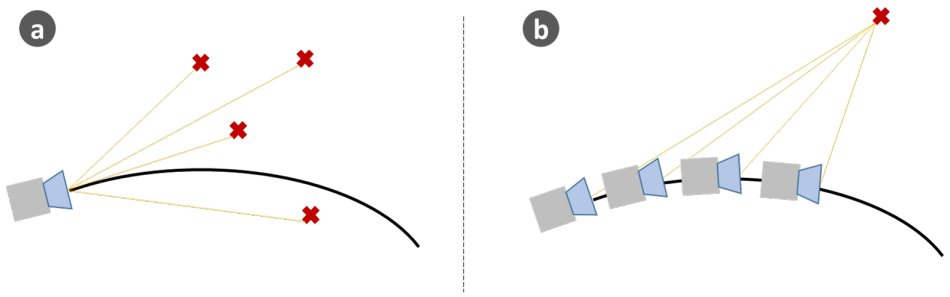

FIGURE 2.12: a) EKF pose estimation, b) MSCKF pose estimation

The VI EKF SLAM state vector is presented as follow:

$$x_k = \begin{bmatrix} x_{I_k}^T & x_L^T \end{bmatrix}^T \tag{2.10}$$

$$x_I = \begin{bmatrix} {}^W_I p^T & {}^W_I q^T & {}^W_I v^T & b_a^T & b_g^T \end{bmatrix}^T \tag{2.11}$$

$$x_L = \begin{bmatrix} {}^W f_1^T & \dots & {}^W f_M^T \end{bmatrix}^T \tag{2.12}$$

with:
x_I: the IMU pose in IMU frame $\{I\}$ (eq. 2.15)
x_L: the landmarks poses
${}^W f_i$: inverse depth landmark representation ($fi = (x_i, y_i, z_i, \theta_i \phi_i \rho_i)$)
(x_i, y_i, z_i): the camera center position from which PoI was first observed
$[\theta_i \phi_i]$: the unit ray directing towards the point from the center of the camera
ρ: inverse depth of the camera along the unit ray

This state vector representation in EKF VIO imposes a high computational complexity. Subsequently, as in VI ORB SLAM, after the pose estimation is carried out, a keyframe is selected to perform the BA technique. This technique is generally used to build accurate and consistent global maps. However, it requires a high execution time. That is why keyframe-based processing is advantageous, as it reduces the execution time of the BA. In addition, without the loop closure, the estimation error is boundless as it is mentioned below in the R-VIO. Thus, the VI EKF SLAM is terminated by the loop closure module which decreases the accumulated drift when passing through an already mapped environment. Since Accurate VI EKF SLAM is a feature-based algorithm, it is subject to failure or quality deterioration in complex environments, mainly the poorly textured ones. Also, like any visual-inertial algorithm, inertial measurement integration requires an additional computing cost.

R-VIO: in [93], authors propose a VIO algorithm called Robocentric Visual-Inertial Odometry (R-VIO), allowing the system to be localized in the global frame using its previous positions in the reference frames. R-VIO is listed among the VIO algorithms that are based on the filtering approaches; more precisely, it is one of the algorithms that uses MSCKF algorithm [150]. The latter is a classic

algorithm inspired from EKF and it is used in VI systems. This technique is characterized by its state vector x_k composed of the global frame coordinates, the IMU pose and a window of relative camera's poses calculated between the last N frames of reference. Actually, the system maintains a sliding window w_k^T of the camera's past frames in the state vector, and uses landmark measurements to impose probabilistic constraints on these frames.

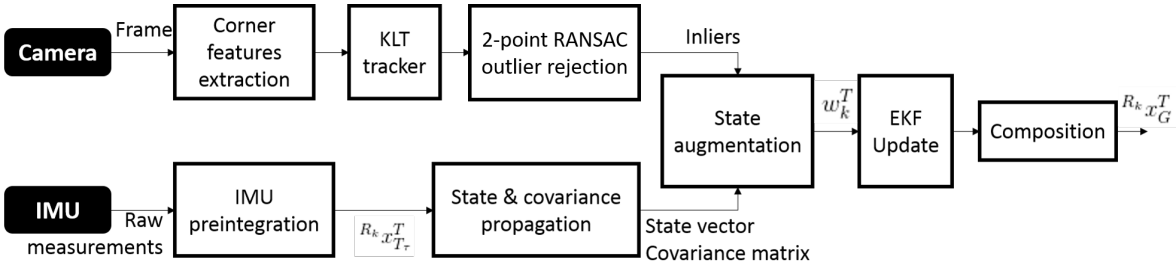

FIGURE 2.13: Block Diagram of MSCKF's workflow process in R-VIO

As depicted in the figure 2.13, the R-VIO's workflow contains an image processing stage where the PoIs are extracted, tracked and filtered using 2-point RANSAC outlier rejection. This aims to conserve only inliers with good tracking information. The filter's measurement is composed of these sorted inliers. In parallel, there is an IMU data processing stage: after the IMU preintegration, the system state and covariance matrix are computed and propagated in order to output the filter's prediction. In the state augmentation, local poses are computed for each frame where an inlier is correctly tracked. Then a window of camera's past frames is passed to EKF update stage to prepare the state and covariance matrix used to estimate the global pose.

Generally, the R-VIO state vector x_k is represented as follows:

$$x_k = \begin{bmatrix} {}^{R_k}x_G^T & {}^{R_k}x_{I_\tau}^T & w_k^T \end{bmatrix}^T \tag{2.13}$$

$${}^{R_k}x_G^T = \begin{bmatrix} {}^{k}_{G}q^T & {}^{R_k}p_G^T & {}^{R_k}g^T \end{bmatrix}^T \tag{2.14}$$

$${}^{R_k}x_{I_\tau}^T = \begin{bmatrix} {}^{\tau}_{k}q^T & {}^{R_k}p_{I_\tau}^T & v_{I_\tau}^T & b_{g_\tau}^T & b_{a_\tau}^T \end{bmatrix}^T \tag{2.15}$$

$$w_k^T = \begin{bmatrix} {}^{2}_{1}q^T & {}^{R_1}p_{R_2}^T & \dots & {}^{i}_{i-1}q^T & {}^{R_{i-1}}p_{R_i}^T & \dots & {}^{N}_{N-1}q^T & {}^{R_{N-1}}p_{R_N}^T \end{bmatrix}^T \tag{2.16}$$

with:

${}^{R_k}x_{I_\tau}^T$: the IMU pose in IMU frame $\{I_\tau\}$

${}^{R_k}x_G^T$: the state vector of the global frame in the robocentric frame of reference $\{R_k\}$(the starting frame)

w_k^T: the window of the relative poses between the last N robocentric frames of reference

${}^{k}_{G}q$: the rotation from $\{R_k\}$ to $\{G\}$

${}^{R_k}p_G$: the position of $\{G\}$ in $\{R_k\}$

${}^{R_k}g$: the local gravity

${}^{\tau}_{k}q, {}^{R_k}p_{I_\tau}$: the relative rotation and translation of system from $\{R_k\}$ to $\{I_\tau\}$

v_{I_τ}: the system velocity in $\{I_\tau\}$

b_{a_τ}, b_{g_τ}: the accelerometer and gyroscope bias, respectively Thus, the R-VIO method updates the state only with landmarks that are moved out of the field of view of the current frame, or with visual measurements taken on frames that are removed from the state vector. This means that not all currently available measurements are used to update the state. In addition, the linearization errors make the filter inconsistent. To solve this problem, research has been ongoing for a long-time. In 2009, the idea of the first Jacobian estimates (FEJ) [96] was adopted to improve the consistency of the MSCKF [126] [125]. After a few years Guoquan P. Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis proposed the OC methodology [125] which was intended for OC-VINS development [89] [88] [119]. Up to now and even with the current work of R-VIO, this problem persists despite conserving the appropriate observability properties independently of the linearization points. Furthermore, since the R-VIO is an odometry algorithm, it does not allow loop closure to be performed either by mapping or place recognition. However, this is an important feature to ensure a limited error VINS performance for an accurate long-term pose estimation. As an alternative, VI SLAM system can produce more accurate and more precise results thanks to using mapping and loop closing.

2.2.3 Optimization-based vs Filter-based Approaches

Filter-based approaches provide pose estimation, using a filter that relies on inertial measurements and image features. To improve the computational efficiency in this kind of systems, interference processes are delayed until the last stage of the system, and the filter's past states are marginalized. The latter has a certain number of drawbacks, mainly it leads to problems of consistency. The EKF algorithm is one of the most common filters used for data fusion. It consists mainly of two major steps: prediction and update. EKF performs a first-order linearization around the current mean and covariance at each step. Therefore, it is reliable only for systems that have a Gaussian model with limited non-linearity. Also, its computational complexity increases quadratically with the number of tracked characteristics in the state vector. Hence, its accuracy is affected or limited. In order to overcome these problems and to obtain better accuracy, the UKF algorithm can be used instead of EKF, specifically for highly non-linear systems. It does not compute Jacobian matrices. However, despite the above advantage, UKF is still costly for computing power, so it is difficult to implement it on embedded systems, such as UAVs. Another algorithm is used for filter-based visual-inertial systems, it is the MSCKF. The latter proceeds by constraining the measurements by a stochastically cloned pose in a sliding window. This algorithm suffers from a noise gain in the unobservable subspace direction, that impacts the consistency of the system state estimation.

Furthermore, optimization-based approaches use a non-linear optimization to reduce directly any errors between the motion obtained from the integrated inertial measurements, and the camera motion obtained using standard reprojection error reduction. These approaches can outperform filter-based ones in terms of accuracy, due to their ability to linearize current and past states. However, optimization-based approaches can suffer from different problems, depending on the optimization method used. These problems are mainly related to computational complexity, which complicates the implementation of these approaches on systems with limited resources. Indeed, these computational costs can be reduced in different ways. Among the most well known and used techniques are the possibility of processing only keyframes, the use of sliding windows or the use of incremental smoothing (updating only small variables subsets).

Most of relevant filter- or optimization-based literature methods are evaluated and compared using EuroC dataset [24]; we will come back in the analyze of these evaluations in section in section 4.1 of chapter 1. This dataset gives different sequences with different difficulty levels of the navigation

environment. Table 2.2 illustrates the comparison of the estimated trajectory accuracy claimed by the relevant literature works. In the most of state-of-the-art works, the method accuracy is analyzed using the criterion of the Root Mean Square Error (RMSE) compared to the ground truth. According to evaluation reported in table 2.2, the most accurate method in most difficult cases is VI EKF SLAM, followed by VI ORB SLAM and Adaptive VI SLAM. However, these last two algorithms are the most vulnerable, since they fail in the most difficult conditions, such as V103 and V203 EuRoC sequences (section 4.1), which reduces their reliability and robustness. Other methods listed in table 2.2 are all robust in difficult cases but they are less accurate (high RMSE) compared to VI EKF SLAM even in easy cases (R-VIO is the least accurate while VINS Mono is the most accurate in the remaining algorithms). This reduces the benefit of their use if the application's purpose is to have good accuracy (small RMSE).

Sequ.	**Methods**							
	OKVIS	ROVIO	VINS Mono	VI ORB SLAM	VI EKF SLAM	Adaptive VI SLAM	Trifo-VIO	R-VIO
V101	0.084	0.15	0.048	**0.027**	0.047	0.096	0.06	0.08
V102	0.16	0.19	0.048	**0.028**	0.041	0.059	0.07	0.16
V103	0.21	0.17	0.17	fail	0.081	**0.068**	0.13	0.14
V201	0.13	0.28	0.054	0.032	**0.029**	0.066	0.065	0.22
V202	0.17	0.60	0.10	**0.041**	0.044	0.073	0.12	0.31
V203	0.26	0.18	0.15	0.074	**0.058**	fail	0.15	0.44
MH03	0.26	0.40	0.07	0.87	0.046	**0.039**	0.24	0.36
MH04	0.34	0.88	0.09	0.22	**0.088**	0.092	0.12	1.04
MH05	0.44	1.26	0.14	0.082	**0.064**	0.086	0.18	0.86

TABLE 2.2: Estimation accuracy (RMSE) of different approaches on the EuRoC MAV dataset

To conclude, optimization-based methods outperform filter-based methods mainly with excellent localization accuracy, while the main advantages of filter-based methods are in terms of computing resources. Choosing the right method type to use for a given system/application can be difficult and involves making trade-offs between accuracy and computational resources.

2.2.4 Challenges Associated to VINS and Development Trends

Recently, VINS have made significant improvements in their quality and performance. However, this is still not enough to solve all the challenges that this field may face. Among the main of these challenges are: firstly, tracking problems related to difficult navigation environments, such as poor lighting, vibrations and difficult motions. Secondly, localization and semantic mapping issues linked to the outdoor navigation environment constraints, etc. The progress achieved in deep learning is assumed to handle these various challenges, and to enhance the navigation efficiency of the majority of current VINS, especially those based on traditional methods that use geometric features like PoIs, lines, etc. Moreover, it is not easy to detect, represent and track different objects coexisting in the same space, in real time, using a mobile navigation system. Another VINS challenge is due to the used sensors type, this is part of their improvement requirements. While cameras and IMUs represent an interesting combination for VINS, other sensors exist and provide more effective aid, according to system and environment requirements. For example, lightweight and low-cost LiDARs may work better in environments with poor lighting conditions [77][87], event cameras [128][129] could provide more effective support for dynamic motion, etc. [238][151].

Consequently, today's research is tending to be focused specifically on addressing these challenges and developing more efficient solutions in terms of computational complexity and resources. However, it should not be forgotten that using deep learning continues to have the drawback of its huge computing cost and overhead. Thus, the integration and use of different sensors are one of the

current research topics concerning VINS. Lightness and miniaturization have a great importance for new VINS. They allow them to operate on small devices, which have limited computing resources, such as embedded systems, telephones, drones or smart glasses, etc. Indeed, significant and motivating results have been obtained with Microsoft Hololens [75], Intel RealSense and Google Tango [97]. Consequently, future applications and systems will tend to be as embedded as possible. Moreover, the use of several sensors is much more beneficial than the use of a single one. One sensor cannot correctly detect environmental information, making state estimation subject to great uncertainty. A multi-sensor system improves the accuracy and completeness of a system since the sensors work together to correctly detect the necessary processing information.

2.3 Conclusions

In this chapter, methods of VINS, as well as their classification were presented. The focus was on the two main classifications: filter- and optimization-based methods and the loosely- and tightly-coupled methods. Following the optimization- and filter-based approaches, different pose estimation techniques were discussed and illustrated using the most known literature works. In addition, a comparison between the two categories is done in order to explain the advantages and inconveniences of each one.

While there has been significant progress in recent years concerning VINS, there are still many challenges to be addressed and which are currently open for discussion:

- *Robust localization*: even if current VINS are able to provide accurate motion tracking, they are not robust enough for long-term, large-scale and safety-critical deployments, such as autonomous driving, in part because of resource constraints. Therefore, even by effectively integrating loop closures or building and using new cards, it remains difficult to obtain persistent VINS in environments with difficult conditions such as poor lighting and unfavourable movements.

- *Upgrades with various aid sensors*: although optical cameras are considered the perfect aid source for the IMU in many applications, other aid sensors may be more appropriate for certain environments and motions. For example, acoustic sonar can be used instead in underwater environments; low-cost lightweight LiDARs can work better in environments, such as environments with poor light conditions; and event cameras for a more accurate capturing of dynamic motions. In this context, it is essential to study in detail the VINS extensions that can use different aid sources for handheld applications.

As inertial and visual sensors increasingly become ubiquitous, VINS have undergone significant research efforts and progress over the past decade, encouraging an increasing number of innovative applications in practice. Moreover, due to the specific sensor properties and application type, it is not easy to develop VINS algorithms from scratch without understanding the advantages and disadvantages of the existing literature approaches. Especially, each method has its own objective and does not necessarily proceed in the same way to carry out required processing at different stages of a VINS.

After having presented and compared the main methods dedicated to VINS, the latter architecture and hardware integration is also an interesting topic. Therefore, in the next chapter, the implementation part and hardware integration of these VINS will be discussed with a main focus on embedded systems.

Chapter 3

Embedded Tracking Systems

Traditionally, computer vision applications used to be executed on powerful and fixed workstations. Today devices such as a PC (even laptop) designed for long life expectancy, with a constraining form factor and a high power consumption are now an issue to run flexible, volatile applications that are not too greedy in terms of power efficiency. Electronic systems evolved to became compact, self-sufficient, connected and capable to interact with their environment. Trends in embedded system design consists in conceiving architecture able to cop concurrently with general purpose process while achieving more specialized task on dedicated hardware accelerator. Recently, the concept of Cyber Physical Systems (CPS) appeared *a.k.a* embedded systems [86]. In particular, the aim of CPS is to deal with the heterogeneity of the available resources. Thus, tasks are distributed all over a bunch of devices achieving different steps of the whole application process.

In general, a CPS connects the physical world with the virtual world via sensors or actuators. The system usually consists of various elements that work together to carry out missions and activities. These physical elements interact with the environment through a network or other communication technologies. Thus, CPS concept covers all devices from sensors to embedded systems and remote computing stations. The standard CPS scheme is shown in figure 3.1.

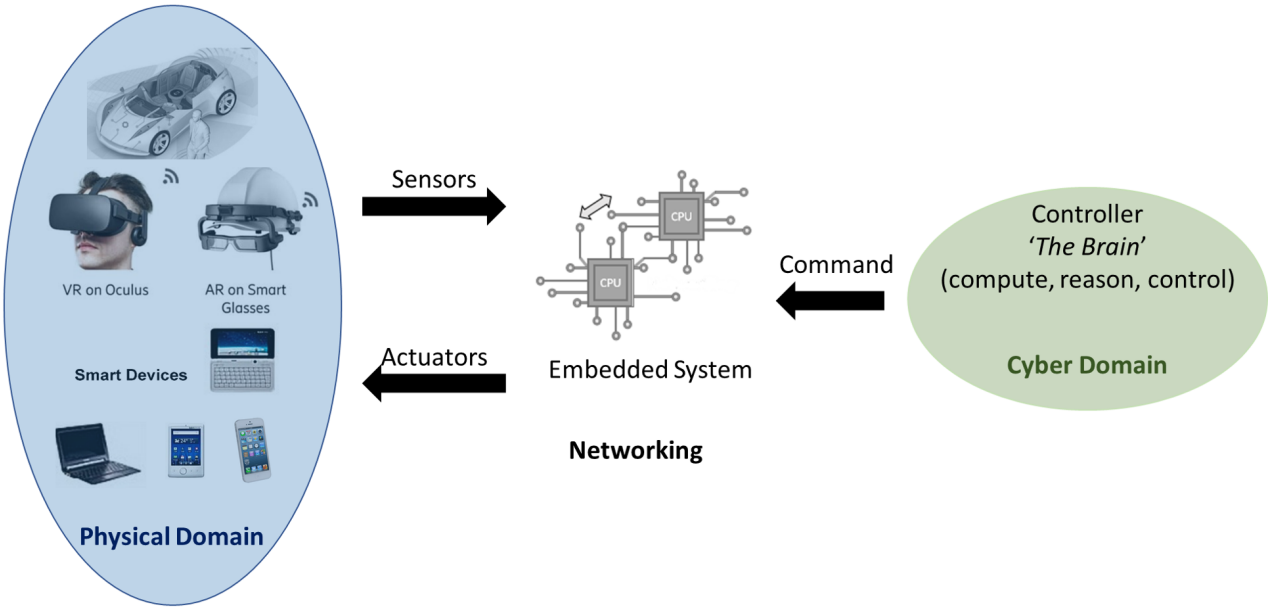

FIGURE 3.1: CPS standard scheme

Modern CPS are present in various fields such as: automotive, avionics systems, intelligent and advanced building, Internet of Things (IoT), medical systems, automated and robotic manufacturing, AR devices and many others fields. They can operate on large-scale systems, that have many and varied uses, covering a wide range of applications and increase their importance in our daily lives. Furthermore, CPS are characterized also by the price decrease compared to electronic systems a few years ago. Therefore, they are successfully and widely used in modern systems. Today, CPS performs specific tasks in real time, without the need for huge and powerful motherboards or cloud computing.

However, one of today's major challenges for scientists and engineers is how to design modern CPS with a higher degree of autonomy and independence. They should be automatic or even autonomous, allowing for greater mobility, portability and performance in any real-time environment. Acquisition, analysis and understanding of the environment must be properly satisfied as well. In

order to carry out its mission, the CPS must supervise and control physical processes in real time. Therefore, it is critical that all components of the system perform as fast and accurately as possible in order to operate together.

As embedded systems are one of the major link of a CPS, this chapter focuses the constraints met by designers when developing a device such as a VINS. Firstly, we briefly define what is an embedded system and trade-offs that have to be made in terms of power efficiency. Then, we focus on the diversity of recent architectures and components used to built an efficient system. In particular, the section 3.1 illustrates this latter by providing metrics. Then, in section 3.2, different existing embedded computing architectures, especially those used for visual-inertial navigation and tracking computation, are presented. Subsequently, in section 3.3, the difficulties of designing an embedded tracking system are explained, including the co-design approach, and the cutting edge VINS and their implementation are described.

3.1 Definition and Features

An embedded system is a special purpose computer system, programmed and controlled using a Real-Time Operating System (RTOS). It must meet specific requirements relying on its dedicated tasks. In this section, we introduce the different embedded systems classification, then we explain their main design challenges and performance metrics.

3.1.1 Embedded System Classification

Embedded systems can be defined according to their features and functionality requirements. Obviously, performances and power consumption may differ depending on targeted application:

Autonomy The autonomy discussed here deals with processing and computations made by the system. In fact, autonomous embedded systems are devices that receive inputs (analog or digital), process incoming data and perform several dedicated tasks, and produce the final results. All of this is done autonomously without the need to give part or all of the processing to one or more external processors. The following are some examples of autonomous embedded systems: video game consoles, digital cameras, Digital Audio players, *etc*.

Real-time This refers to systems able to deal with bounded task execution time. As time is a critical constraint, output accuracy and precision are affected. There are two types of real-time embedded systems: *the hard real-time systems*, where there is no flexibility or tolerance in time constraints. *i.e.* the smallest error caused by the non-respect of the deadline or the inaccuracy of the result can cause catastrophic consequences. *Soft real-time systems* tolerates precise tolerance delays. The response must be given on time but with a tolerance within acceptable limits.

Network-Connected This embedded system category is connected to the Local Area Network (LAN), Wide Area Network (WAN), Internet, cellular communication network, *etc.*, in order to access and use resources. For instance, 5G is a recent network connection used to ensure this communication between embedded systems. In fact, 5G or fifth-generation cellular communications technology is an ongoing topic that affects all industries, especially embedded systems and specifically the IoT. The latter covers the description, development, and use of connected devices.

The growing number of connected devices in the various industrial fields, from agriculture to pharmaceutical production and the automotive industry, represents a main factor driving 5G. These industries rely primarily on connected devices for a large amount of data collection that makes work processes more efficient, increases productivity and continuously improves products and services for customers. 5G enables the efficient management of data generated by the IoT, as well as providing the near-instant communication required for critical services such as robotic-assisted surgery and autonomous navigation. Among the most likely application areas to be affected by the future hyper-connected 5G, there are AR/VR, Autonomous vehicles and intelligent infrastructure. For example, in 2017, around 60-80% of cars sold worldwide have telematics installed, which is expected to increase to 90% by 2020 for cars connected to the internet [141]. In addition, smart highways, networks, properties and other smart infrastructure investments would involve sensor technology to support the development not only of autonomous vehicle ecosystems, but of smart cities in general.

Mobile/Wearable Embedded Systems Mobile embedded systems are used in mobile embedded devices such as mobile phones, smart glasses, car/robot/drone remote controlled or not, etc. These embedded systems are constrained by resources (mainly memory) and power consumption.

FIGURE 3.2: Mobile embedded systems examples

Architecture differs depending on system purpose and targeted application domain. Nowadays, wearable systems are increasingly used and requested by users, given their utility in daily life and ability to be adapted to different conditions. Indeed, they are mainly multi-sensor, mobile/portable, autonomous and often connected embedded systems. The multiplicity of systems sensors improves data gathering and decision making. Moreover, a wise choice of sensors and processing algorithm enables to target specific domain.

Embedded systems enables to collect data provided by sensors such as IMU and camera [64][162][231]. Simple filtering functions can also be applied to these raw data but early stages of more complicated processing (e.g. autonomous navigation, AR...) can be partially integrated. Thus, these systems provide more intelligible and compact data that can save bandwidth, latency and energy. As connectivity is the corner stone of all embedded systems, and data are massively shared especially in CPS, it is inconceivable to dispense with standard. However, it comes at the cost of power consumption and thus of autonomy. Then next section precisely covers the design constraints and associated metrics.

3.1.2 Design Challenges and Performance Metrics

As presented in [61], currently general-purpose microprocessors have reached their limit in energy efficiency. The manufacture process in semiconductor Integrated Circuit (IC) device fabrication was

7nm in 2018 and will reach 5nm in 2020 [106]. However, Complementary Metal Oxide Semiconductor (CMOS) scaling no longer provides efficiency gains proportional to the increase in transistor density. Nowadays, industrial and academic works and studies are more and more focusing on the design of specialized and optimized accelerators and the use of computing resources heterogeneity. System architects have to find a trade-off between these three criteria: performance, power consumption and energy efficiency. Since those systems still have to perform general tasks [62], devices such as General Purpose Graphics Processing Unit (GPGPU) or even last-generation Field Programmable Gate Arrays (FPGA) are becoming more and more attractive to designers. Although they are greedy in terms of power consumption, they offer a combination of general purpose System-On-Chip (SoC) and dedicated accelerators for image processing and highly parallel re-configurable processing (GPU res. programmable logic resources). Today SoC such as those smartphone built-in represents best trade-offs between versatility versus power consumption. All sensors are connected to the chip in a centralized manner.

Thus, the designer faces three major constraints, ensuring the architecture versatility between general purpose and specialized tasks, limited global power consumption and sustaining high computing capability on dedicated times-lot. Firstly, the general-purpose/specialized capability of an embedded system is one of the early issues raised at the beginning of its hardware integration and its implementation. Actually, sensors can be connected together with the same processing part support centrally, as well as they can be distributed to different parts of the system, according to the processing requirements. In fact, the SoC architecture, presented below in section 3.2.1, is the perfect candidate to deal with.

In terms of power consumption, generally a SoC consumes between a few *milliWatts* and a dozen *Watts*. For example, a cutting edge smartphone such as the Nokia 7 from HMD Global, the company claims a two-day autonomy (with an integrated battery of 3800mAh). This figures covers the execution of a range of various application involving signal processing, screen display, camera acquisition and associated processing, modem data exchange and so on, *etc.* Thus, the addition of all powers consumed by dedicated resources. In the case of VI SLAM/VIO, a research team recently design and implement a chip dedicated to VIO Navigation system [203]. Indeed, among the main challenges of designing such chip is to perform these tasks with the lowest possible power consumption. The Nano Drones autonomous navigation proposed in [203] consumes between 2mW and 24mW, depending on its operating conditions, making it the first odometry system based on visual-inertial fusion that requires such low power consumption. This SoC is called Navion and is presented later in this chapter. Furthermore, traditionally IC provider always gives typical performance metrics to illustrate computing capabilities of their chip. For generalist tasks, millions of Instructions Per Second (MIPS) or Instructions Per Cycle (IPC) are ones of them. But these metrics are directly linked to the Instruction Set-Architecture (ISA) thus they are discriminant. Nowadays, specific metrics are promoted to characterize accelerator efficiency. Trillion of Floating Point Operation per Second (TFLOPS) for a given data width, Giga Pixel per second for video, etc. are also used. In 2016, NVIDIA introduces a new performance measurement, the Deep Learning Tera Operations Per Second (DL FLOPS) computing figure, it refers to the ability of processing an image classification (AlexNet) using deep Convolutional Neural Network (CNN).

These metrics are sometimes controversial, because of their dependency to the processing kernel

and architecture implementation. They represent a kind of theoretical peak performance not achievable for real-world applications. Thus, some benchmark providers propose typical kernel implementation that can be implemented on candidate architecture and sort them by performance and efficiency in terms of energy (BDti [103], Spec2006 [42], etc). In fact, power and energy aspects consider *Watt* and *Joule* metrics, respectively. More realistic measures allow the efficiency to be computed using, for instance, operation per second in power consumption and/or energy delay product (*e.g.* TOPS/Watt), time processing $\times$ power consumed depending on executed kernels or applications, as well as a new mJoule/inference or Frame per Second (fps) metrics, which has been introduced to measure efficiency of CNN.

According to the previous discussion, the embedded systems constraints are mainly related to the system's specialization level, power consumption and performance. These constraints are interconnected. Particularly in tracking embedded systems, which is the main subject in this thesis, the challenge is to design an efficient system using a minimum power consumption, using different components (specialized/generalist architectures). Thereafter, the different computing architectures, classified into three types: SoCs, IP Modules, and ASICs, are presented.

3.2 Embedded Computing Architectures

The choice of a well-fitted architecture relies on the respect of application requirements. Designers translate these latter into throughput, latency metrics under power constraint. The targeted host architecture ensures the execution of the application on monolithic or heteregeneous ressources. As there is a wide range of embedded devices and components, this section focuses on the description of mainstream available architectures and the trends in IC chip for embedded VINS. First SoCs are presented and illustrated using the examples commonly used in embedded navigation systems, IP modules and ASIC architectures.

3.2.1 System on Chip (SoC)

This type of architecture is able to execute many applications. By design, a SoC is commonly a general purpose platform with small dedicated accelerators. Ressources are interconnected together via dedicated communication media (e.g. bus, crossbar, Network-On-Chip) and data are shared through a specific memory architecture. A software programmer is able to developp applications and/or kernels by using typical compiling toolchain and can also use accelerators with dedicated instructions. The purpose of the SoC is then to meet the performance, the power consumption, and the surface area constraints (in the case of circuit design) set at the time of specification. SoCs are used in different products and in different ways, such as MultiProcessor SoC (MPSoC). The latter platform is capable of managing high-level applications and taking advantage of tasks, processes and thread parallelism.

Over the past ten years, ARM has become the leader as embedded processors provider and has achieved a 95% mobile market share since Intel left in 2016. They offer the most energy-efficient solutions, from microcontroller units (for a wide range of IoT markets worldwide), portable devices and ADAS to high-end industrial computing [7]. Supplied as a core license, ARM products are integrated into many leading SoCs (TI, Qualcomm, ATMEL, etc.). Today, "big.Little" ARM architecture is the reference for supporting different types of workloads while maintaining low energy consumption. It consists of a combination of small low-energy flexible cores and large high-performance flexible

cores (*e.g.* 4 cortex A55 + 4 cortex A75). It should also be noted that the RISC-V architecture has appeared in recent years. RISC-V is a free and open-source ISA from *U.C. Berkeley* [217]. More than 100 companies now support this initiative. The keywords of this new ISA are longevity, portability and reliability since a frozen ISA (less than 50 instructions) is provided and the extensions are well classified (multiply and divide, atomic, single precision floating point, double precision floating point, compressed instructions) for a total of 200 instructions.

Among the well-known architectures, the following ones can be listed:

NVIDIA Tegra Xavier SoC, this architecture has the leading role in bringing GPGPU devices to the embedded world.

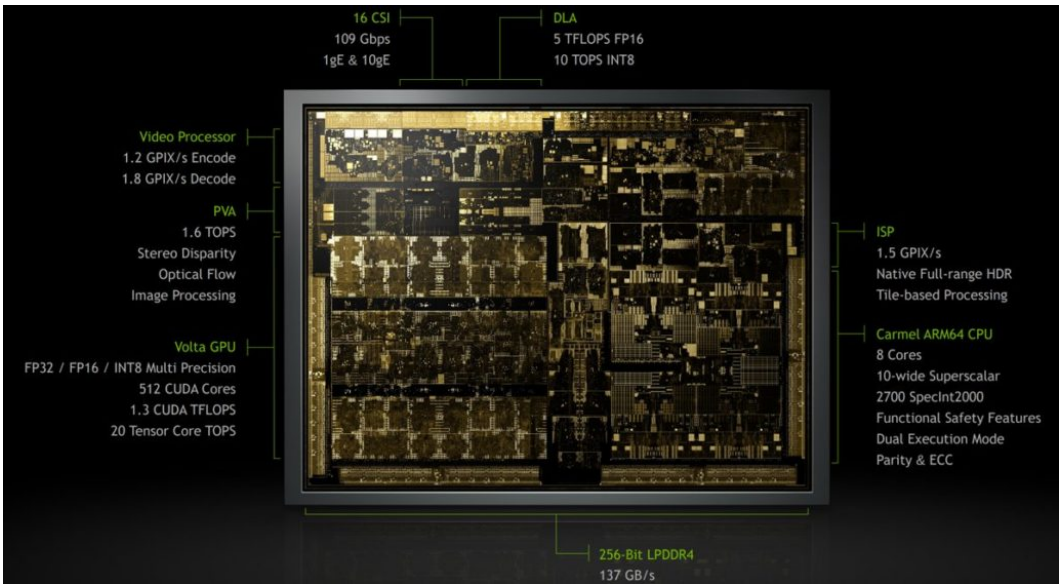

FIGURE 3.3: NVIDIA Tegra DRIVE Xavier SoC [161]

Proposed in 2018, NVIDIA Tegra device called Xavier [51], shown in figure 3.3, is composed of 9 billion transistors. This kind of SoC consists of 8-core CPU Carmel ARM64 10-wide superscalar with functional safety features plus parity and ECC suitable for autonomous driving. In addition, there exists a Packet Video Audio (PVA), this unit is used for processing computer vision tasks as filtering and detection algorithms. Two identical instances are implemented that can works independently or in lockstep mode. Each unit contains an ARM cortex R5 processor, a DMA unit, two memory unit plus two vector processing units (7 ways VLIW). Customizable logic is also available. Operations can be performed on 32×8, 16×16 or 8×32 bit data vector. Furthermore, this NVIDIA device includes a new 512-core Volta GPU. This architecture has been designed for machine learning market and is optimized for inference over training process. It includes 8 stream multiprocessors with individual $128KB$ L1 cache and a shared $512MB$ L2 cache. There is also 512 CUDA tensor cores. Besides, there exist a 8K HDR video processors, as well as a Drive Letter Access (DLA) which is able to achieve $5,7DLTOPS(FP16)$ with a configuration/block, an input activation and filters weigh, a convolutional core, a post processing unit, and an interface with memories (SDRAM and internal RAM).

Xilinx Zynq UltraScale+ MPSoC, this device offers 64-bit processor scalability, and combine real-time control with software and hardware drivers for graphics, video, waveshaping and package processing. Based on a common real-time processor and programmable logic platform,

three distinct variants include: dual application processor (CG) devices, GPU and quad application processor (EG) devices, and video codec (EV) devices; creating unlimited possibilities for applications such as wireless 5G, Advanced Driver Assistance Systems (ADAS) and the Industrial Internet of Things (IIoT).

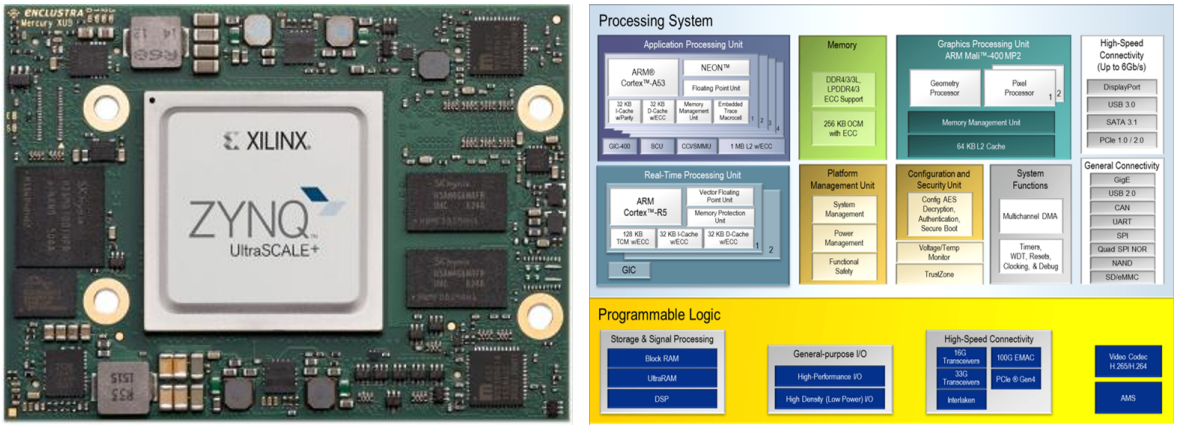

FIGURE 3.4: Xilinx Zynq UltraScale+ MPSoC Overview

The Xilinx Zynq UltraScale+, presented in figure 3.4, is characterized by various features. Firstly, this device is composed of an innovative ARM+ FPGA architecture for differentiation, analysis and control. Moreover, it includes middleware, stacks, accelerators, IP ecosystem, and extended operating system. In addition, this device involves an integration providing de facto a fully programmable platform. Also, it can achieve up to 5x system-level performance per Watt on Zynq-7000 SoC. It is designed to offer lower system power consumption along with most flexible and scalable platform for optimal reuse and time-to-market. Furthermore, the Xilinx Zynq UltraScale+ is characterised by its advanced design tools, C/C++ and open CL design abstractions and also widest range of software and hardware design tools and reference designs.

3.2.2 Intellectual Property (IP) Modules

When application execution time becomes a difficult constraint, and acceleration of critical tasks is required, some custom IPs are used to reduce processing time, especially for audio, video, image, radio frequency tasks, *etc*. IP provider proposes components such as DSP, GPU, ISP, *etc*. They are programmable targets and are implemented to met critical deadlines. Recently, Network Processing Unit (NPU) has been introduced in patent by [3]. *Huawei* and *Apple* announced the integration of industrial NPU chips in their SoC Kirin and A11 Bionic, respectively (Huawei's NPU is presented in [72]). Whereas *Qualcomm* enlights its Neural Processing Engine, which is a flexible combination between CPU, DSP, and GPU. Among the relevant IPs used for embedded systems, the following can be presented:

Digital Signal Processor (DSP) is a microprocessor optimized for digital signal processing (filtering, compression, extraction, *etc.*) with typical Arithmetic and Logic Unit (ALU). There are various DSP producers in the market including: Qualcomm with its Hexagon DSP family, STmicroeletronics, Analog Device and others; so several DSP solutions are available, such as CEVA Platform XM6 [31], Tensilica DSP from Cadence [25], Snapdragon 845's Hexagon 685 DSP from Qualcomm Hexagon [176] (figure 3.5) and others.

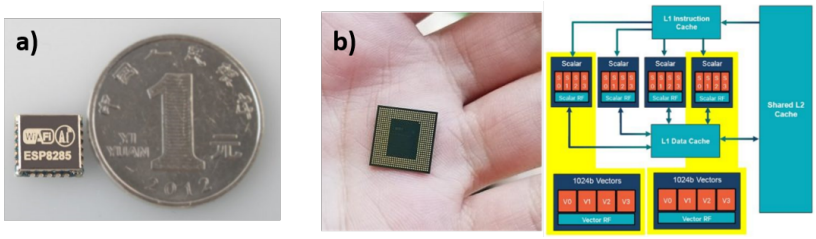

FIGURE 3.5: a) Tensilica L106 DSP, b) Snapdragon 845's Hexagon 685 DSP

Graphics Processing Unit (GPU) these power efficient IPs are dedicated to image, video, rendering processing and targeted for tablets, smartphones, wearable AR/VR, IoT, smart TV, and automotive devices. In the case of ARM-G76, the architecture consists of unified shader, dual texture unit, a blender and tile access, depth and stencil manager, and memory. The 20 shaders cores include three execution engines (with eight execution lanes for each), making a total of 480 execution lanes. The execution units have 8-bit integer dot product support for improved machine learning performance. In a similar vein, Adreno architecture is the GPU solution from *Qualcomm* to provide efficient SoC. Last generation is able to support software features such as OpenGL ES 3.2, OpenCL 2.0 and DirectX 12.

Image Signal Processor (ISP) is a microprocessor specially used for image processing in digital cameras, mobile phones or other devices. An ISP uses parallelism at the pixel level to increase its processing speed and energy efficiency. One can cite ARM Mali C71 (figure 3.6), Qualcomm Spectra, etc.

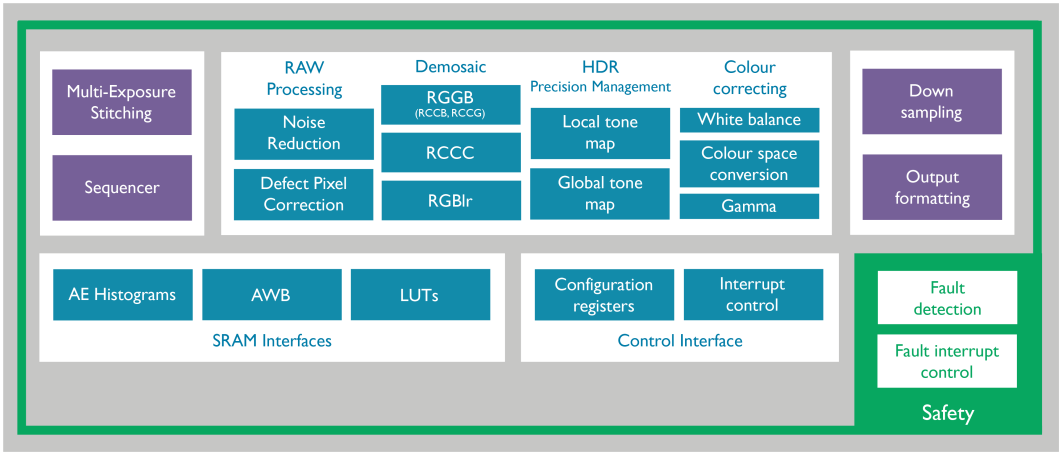

FIGURE 3.6: Functional block diagram of ARM Mali C71 ISP [8]

Field Programmable Gate Arrays (FPGA) traditionally, FPGA devices were used only for prototyping and/or making a proof of concept for hardware accelerators before the taping out. FPGA enables the device reprogramming of hardware logic cells to perform different functionalities. It can take advantage of fine-grain data parallelism by enabling the concurrent execution of different data processing chain.

These last years, FPGA device and their attractiveness for customers has changed since these devices consists of both a hardware logic part and a more generalist software processing system part, available on the same silicon, such as Zynq UltraScale+ MPSoC [222]. Thus, it is possible to

design high efficient hardware components and make easier data exchange with more generalist processing such as multiprocessor platform in the same device. Nowadays, in Xilinx latest product, a.k.a. Adaptative Compute Acceleration Platform (ACAP) called Versal device includes a dual-core ARM A72, dual-core Cortex ARM R5, DSP, Interfaces IPs settled next to a logic part. Furthermore, *Intel* company recently acquires *Altera* FPGA Manufacturer [104] and proposes the Arria 10 device. Others manufacturers such as *Microsemi* or *Lattice Semiconductor* only provides FPGA with logic cells.

Neural Processing Unit (NPU) is primarily an artificial intelligence accelerator. It is a specialized circuit that implements all the control logic and arithmetic required to execute machine learning algorithms. It is often multi-core designs that run on predictive models such as Artificial Neural Networks (ANNs) or Random Forests (RFs), focusing on low-precision arithmetic, new data flow architectures or in-memory computing capacity. Actually, to perform learning and neural processing tasks, general purpose CPUs are the least suitable because they are not designed for massively parallelized execution. GPUs and DSPs are much better choices, but even then there is plenty of room for improvements.

Since the term NPU has been recently introduced, manufacturors decline their own version. HiSilicon/Huawei created the term NPU while Apple publicly uses the term NE/neural engine. Other IP providers such as Cadence/Tensilica have chosen to call their processor a Neural Network DSP (Vision C5) and Imagination Technologies (2NX Series) uses the term Neural Network Accelerator (NNA).

3.2.3 Optimized Implementation IC for Sensors Integration and Processing

An ASIC is an electronic device that integrates, on the same chip, all the active elements required for an electronic function or combination to be performed. Indeed, it is exclusively dedicated to a certain application and to a specific user. ASIC devices are optimized in terms of performance, power consumption and occupied area (energy efficiency). Several chips focusing on sensors integration for tracking, localization and navigation have been proposed and designed in academic or industrial works. Firstly, the Holographic Chip: Intel HPU [75] is a custom multiprocessor (used as a co-processor) called the Holographic Processing Unit, or HPU. It is in charge of the integration of all embedded sensors (IMU, custom ToF depth sensor, Head-tracking cameras, IR camera, etc.) through several interfaces: MIPI, CSI/DSI, I2C, PCI. This Taiwan Semiconductor Manufacturing Company (TSMC) has made a $28nm$ co-processor has $65M$ logic gates and occupies $144mm^2$. It consists of 24 Tensilica DSP cores. It has around $8MB$ of SRAM, and an additional layer of $1GB$ of low-power DDR3 RAM. HPU offers a trillion of calculations per second. Claimed as low-power, it consumes $10W$ for handle gesture and environment sensing.

A. Suleiman *et al.* present in [204] a new compact and dedicated navigation chip, named Navion chip (figure 3.7). It is a customized IC targeting VIO and is intented to fit in a more compact system such as nano/micro aerial vehicle and VR/AR on portable devices. The chip uses inertial measurements and mono/stereo images to estimate the drone's trajectory and a 3D map of the environment. Several optimizations are performed to minimize chip power and footprint for low-power applications, while maintaining accuracy. Authors announced an average power budget of only $24mW$ while processing from 28 to $171fps$. Die process is $65nm$ CMOS, the chip occupies $20mm^2$ and presents a fully integrated VIO implementation.

In closing, the offer in terms of embedded computing architecture is large and diverse. This is a competitive industry that is evolving rapidly and continuously. However, these evolutions are

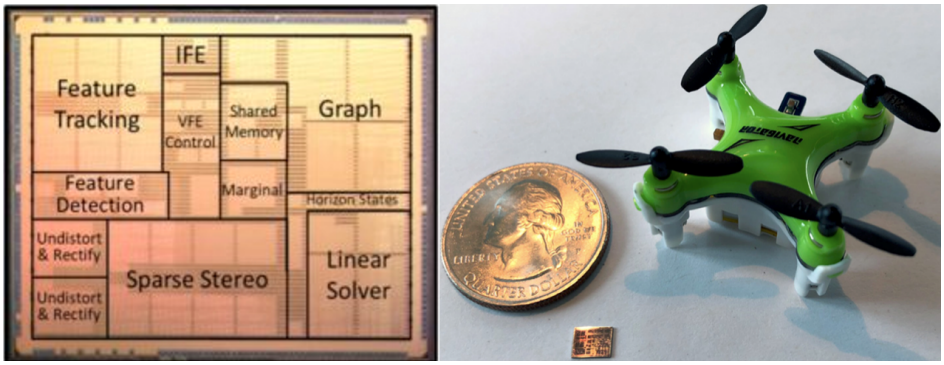

FIGURE 3.7: Navion chip architecture

always focused on improving performance while respecting the constraints of embedded systems and maintaining, or even improving, the system's task quality. From the discussion made in this section, it is noticeable that today a SoC is capable of hosting visual-inertial data fusion and object tracking algorithms thanks to a MPSoC and dedicated IPs. However, we also noticed that there is still a room for improvement especially for VINS. In order to make pending issues explicits, the following section focuses on the state of the art in motion estimation and pose tracking for embedded VIO/VI SLAM.

3.3 VI Navigation Embedded Systems

Recently, technological developments for autonomous navigation systems have received a lot of attention, especially for miniaturized systems (*e.g.* nano/pico air vehicles [135][220], on-board glasses [6], etc.). Particularly, motion estimation and pose tracking, in a given environment, are a crucial processing steps. Other applications also require system's pose and motion estimation, in automotive and VR/AR [34] (obstacle avoidance, environment mapping). Because of the large amount of sensor data to process, robotics research field covers all these challenges.Due to the use of multiples sensors, systems must sustain high processing throughput, keep acceptable performances and can even take advantages in terms of robustification. VI SLAM/VIO, discussed in chapter 2, are among the most relevant topics widely discussed today in the literature. In this section, specific constraints related to VI SLAM and VIO implementation on embedded architectures are presented in section 3.3.1. Afterwards, a brief overview of the main literature works dealing with this topic is discussed (section 3.3.2).

3.3.1 Specific Constraints to VIO/VI SLAM Implementation

Implementing embedded system applications, especially tracking and navigation-based ones, including SLAM and VIO, imposes specific constraints. It concerns synchronization between sensors data rate and the processing part, response time *i. e* low latency (reactive) and high cadency (must provides reliable output in short time) and the highly constrained power consumption budget and form factor.

Synchronization. IMU and camera sensors data must be integrated over time. Specific synchronization mechanism are dedicated to sensor acquisition. Moreover, as data rate differ, strategies for

data integration and delaying are also required to merge relevant information (*e. g* movement, pose) in the process pipeline.

Response time. This feature may vary according to the considered approach. In the case of VIO, the system ouputs a pose at the frequency of process tick even if inaccurate or false. In the case of VI SLAM, even if a fused pose is available, a failure can occur due to the closing loop process. Therefore, both VIO and VI SLAM, in case of KeyFrame(KF)-based methods, compute the final pose just for KFs. Thus, the computation time is reduced, and the response time is also decreased.

Frequency constraint. Compared to a complete VI SLAM system [153][155], VIO does not have a loop closure, which occurs at a lower rate and can be off-loaded into the cloud. Thus, a VIO system can provide motion estimation at a higher frequency, which is an essential requirement for autonomous navigation robots, drones or high-speed vehicles, as well as for AR/VR devices.

Highly constrained power consumption budget and form factor. The real-time implementation of these methods requires relatively powerful computing architectures. However, high-embeddability systems such as nano/pico drones or UAVs have high embedded constraints. In fact, these miniature systems have limitations in terms of both, power budget and form factor. For example, in the quadrotor presented in [130] (figure 3.8) where the processor used is Qualcomm Snapdragon 801, the same one is used for smartphones, and it consumes about 3*W* of power [120].

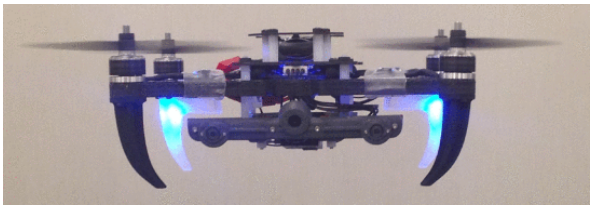

FIGURE 3.8: The smallest drone available on the market that uses VIO to estimate its own position announced by Qualcomm [203]. (Figure captured from [130])

3.3.2 Overview of Existing VIO/VI SLAM Systems

There are various works that propose different solutions for embedded localization and navigation systems, including tracking, based on IMU/camera coupling. Table 3.1 presents the relevant literature embedded system works. In the following, the multi-sensor embedded systems design constraints and co-design methodology are discussed. Then, a brief overview of the remarkable state-of-the-art contributions is presented in order to illustrate the co-design concept and its various stages.

	System	Method	Application
2019	Navion [203] [204]	VIO	Nano drones
2018	EMoVI-SLAM [211]	VI SLAM	head-gear
2018	VINS-Mono [175]	VIO	Drone
2018	PIRVS [231]	VI SLAM	Autonous vehicles
2017	VIO-on-Chip [232]	VIO	Nano Drones
2017	Blind cane [63]	VIO	Walking cane
2016	MAV-VIO [2]	VIO	Quadrotor MAV
2014	On-board VIO [236]	VIO	Micro-MAV

TABLE 3.1: State of the Art of competitive embedded VINS methods

Recent works that propose navigation embedded systems are based on a specific methodology that aims to find the trade-off between the algorithms quality and the embedded computation architecture constraints. This methodology is called 'co-design'. In the following paragraphs, this approach is presented and illustrated using the Zhengdong's work [232]. The latter proposes a specific co-design approach that target an on-chip implementation of a VIO system.

Multi-Sensor System/Architecture Design Constraint: Co-Design.

In order to design an optimal embedded SLAM/VI SLAM/VIO system it is necessary to deal with the hardware configuration and algorithms in a cooperative way. Hence the use of co-design methodologies and tools is crucial and essential.

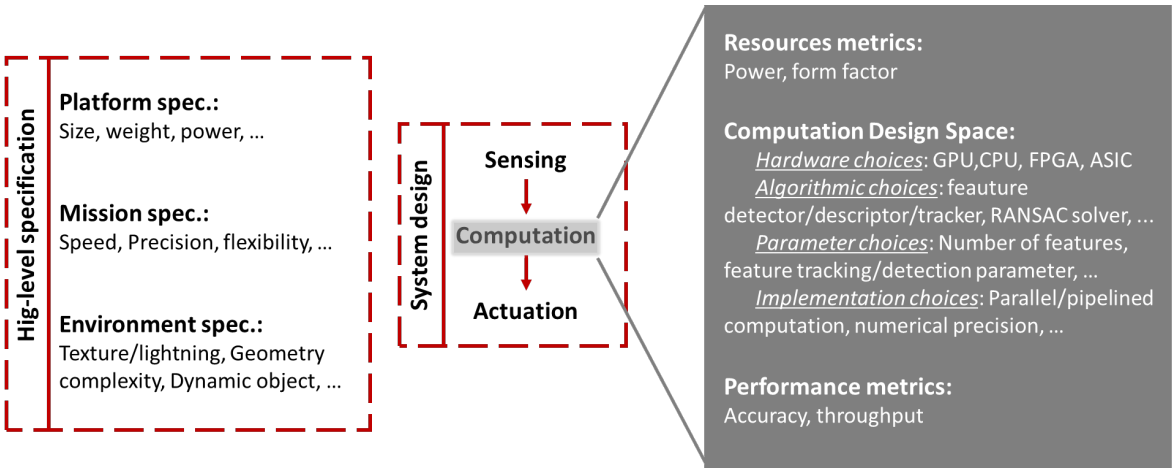

FIGURE 3.9: Co-design method overview (algorithm-hardware trade-off) (inspired from [232])

In fact, the algorithm-hardware co-design process consists firstly in listing the high-level specifications initially defined. These specs often concern the description of the platform, the mission and the environment, as it is illustrated in figure 3.9. Afterwards, the design choice must be made regarding on-board detection, computation and implementation. Here, the objective is to estimate the pose and motion for localization and tracking using inertial and visual data, *i.e.* using the IMU and camera as sensors. So the main focus is on the computation. Actually, the first step in co-design is to switch from high level specifications to the constraints implicated on the available computing resources, and the required VIO performance. This concerns mainly power, form factor, and computation rate, etc (figure 3.9). Performance and computing resources are so dependent that to improve performance it is recommended to use more computing resources. Next, the second step is to make the choices for the design of the VIO system in order to define a "design space". These choices should be made taking into account the performance-resources trade-off, so as to best fulfill the performance requirements by using also the available computing resources. The purpose of the third step is to explore the design space in order to find the optimal solution to conceive a system that achieves the required performance-resource trade-off. This is by evaluating the system empirically. After all, it is the validation step of the final system, which represents the result of the co-design described before.

Generally, an optimal SLAM/VI SLAM/VIO system must be able to provide: 1) accurate sensor time stamping by hardware synchronization; 2) accurate large-scale calibration; and 3) an easy-to-use CPU and embedded platform. Recently, various embedded development kits have been developed and released to the market by different providers (e.g. ODROID based on Samsung SoC, TX1/TX2

based on Nvidia SoC, Intrinsyc based on Qualcomm SoC). However, their use as platforms that support different sensors is difficult and requires time and expertise in this field. However, nowadays different specialized development solutions exist for SLAM, VI SLAM and VIO methods, and which simplify the algorithmic concept evaluation and validation tasks. For example, Google Tango [138] which is a monocular system, Microsoft HoloLens [75] which has HPU access limitations and is not easy to use as a computing platform, and it is a Windows platform only, as well as Apple ARKit [5] which is limited to Apple hardware and the iOS platform only. Recently, Google proposed Google ARCore [80] characterized by its potential to enable SLAM on hundreds of millions of Android devices, but which may have a limitation regarding the form factor of the device.

Examples of systems based on the Co-Design approach

Following the previous discussion, we will present herein some of the relevant works that illustrate the embedded VINS implementation and the Co-Design application regarding to the two approaches: VIO and VI SLAM.

VIO Embedded Implementation exp. 1 : Zhengdong's work [232] describes resources by two power consumption [232] and form factor criterion. This later represents the weight and size available for the computation unit. The form factor depends on energy consumption, for instance, in the case of high energy consumption system, a large battery is required. Or, when the system's energy consumption does not exceed $1W$, then it is not mandatory to have a fan. So, it is useful to minimize the size and therefore to optimize the form factor.

For the second step, they specify a performance feature (table 3.2). Firstly, they rely on the accuracy which describes the VIO drift and error estimation. Secondly, they proceed via the rate which measures the potential rate for processing sensor data and computing a state estimation. In [232] this rate is split in two parts: the front-end throughput which must be high enough to adapt to the camera's frame rate. And the back-end throughput which should be higher than the keyframe (KF) rate, and high enough to ensure that the landmarks are tracked on consecutive KF. In addition, a

		Design goal	Hign-level specs
Resources	power	2W	power, endurance
	forme factor	-	size, weight
Performance	estimation error	25cm	accuracy
	front-end throughput	20fps	speed, agility
	back-end throughput	5 fps	speed, agility

TABLE 3.2: Performance-Resources Trade-off Specification [232]

specific terminology was used in this step to describe the defined design space $D = H\ x\ A\ x\ I\ x\ P$, it is presented as follow: D for design space, H for hardware, A for algorithm, I for implementation choices, and P for parameter choices. In [232], the hardware choices are limited to embedded CPU and FPGA, because of the very limited power budget of the desired application, which is a nano aerial vehicle.

Then the third step, which is the exploration of the design space, the authors of this paper propose a new strategy called Iterative Splitting Co-design (ISC). This technique allows finding the right trade-off between resources and desired performance. It consists in splitting this research into two steps. The first attempts focus on minimizing resources by conserving the desired estimation error, that is using a selection of well-adapted algorithms and parameter choices. Then, the second ones

focus on maintaining the desired throughput, as much as possible, by adjusting implementation choices and hardware parameters. Figure 3.10 summarizes the main steps of the proposed technique ISC.

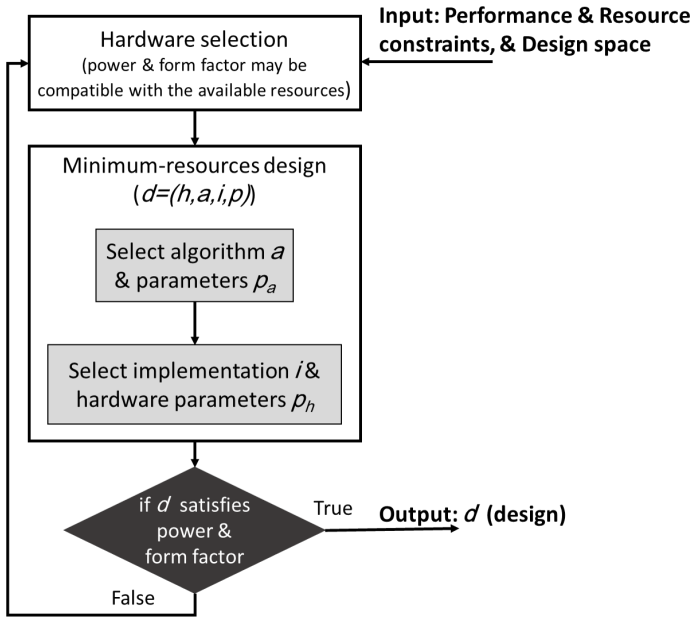

FIGURE 3.10: Overview of the ISC algorithm operating diagram (inspired from [232])

Finaly, for the final VIO design and validation step of [232], an example of the proposed ISC is executed. The hardware choices are embedded ARM Cortex-A 15 and FPGA, and the dataset used for test is EuRoC dataset [24]. As a result, [232] presents a new VIO system on an FPGA, as a specialized equipment, using the co-design process. With this proposal, the authors reaffirm the importance of co-design to maintain accuracy and throughput, as well as to minimize energy consumption (2*W* energy budget) of ultra-embedded systems, *e.g.* nanodrones.

VIO Embedded Implementation exp. 2 : Based on [232], the same authors continued their research to propose an energy-efficient accelerator for VIO. The estimate of the drone trajectory and a 3D map of the environment are obtained according to the pipeline described on figure 3.11, which has three main components: Vision Front-End (VFE), IMU Front-End (IFE), and Back-End (BE). VFE produces landmarks and their corresponding feature coordinates. It processes input mono/stereo frames to track, in all frames, several landmarks based on their 2D projections in the images. IFE is in charge to calculate, at each KF, the measurement preintegration between KFs i and j to produce the relative rotation $(\Delta\tilde{R}_{ij})$, the velocity $(\Delta\tilde{v}_{ij})$, and the position $(\Delta\tilde{p}_{ij})$. And BE aims to fuse VFE and IFE outputs by solving a non-linear optimization problem. In fact, the BE must perform this optimization to find its best solution, which is the best KF state that satisfies the different factors and minimize any discrepancies. The factors used here has the objective of describing the relationship and constraints between states in the sliding window, they are about: vision, IMU, and marginalization.

The algorithmic choices made in this paper [203] are based on the co-design concept, hence they respect the compromise between different criteria such as accuracy, memory and computational complexity. Consequently, in VFE, the chosen algorithms are Lucas-Kanade optical flow (KLT) for feature tracking, Shi-Tomasi for feature detection, and 2-point RANSAC or 3-point RANSAC for mono RANSAC or stereo RANSAC, respectively, for geometric verification performing. However, in BE the

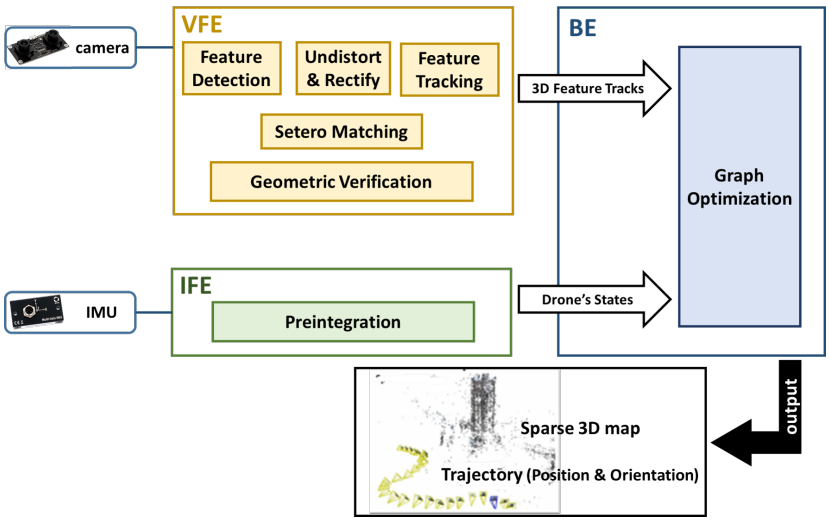

FIGURE 3.11: Navion [203] VIO pipeline

chosen method for optimization is a non-linear factor-graph-based optimization instead of filtering-based optimization (*e. g* EKF-based optimization). As a result, the VIO proposed gains in memory and ensure an absolute translation error estimated by $0.16m$ (using factor-graph optimization method).

Moreover, in an effort to reduce energy consumption and footprint, [203] has carried out various optimizations as image compression, as well as it has integrated the whole VIO system fully on one chip, as it is illustrated on figure 3.12. These eliminates the costly off-chip processing and storage. To confirm the accuracy maintain of this proposition, the system was evaluated on different architecture, a desktop Intel Xeon E5-2667 CPU and an embedded ARM Cortex-A15 CPU, using the EuRoC dataset. On the two architectures, Xeon and ARM CPUs, the trajectory error is estimated by 0.22%, while for Navion the trajectory error is estimated by 0.28%. With these results, the trajectory error increases by 6% due to losses in image compression step and using fixed point arithmetic. In detail, this error increases more in difficult EuRoC sequences (explained in section 4.1). Besides, the execution throughput of Navion is much more efficient than in other methods.

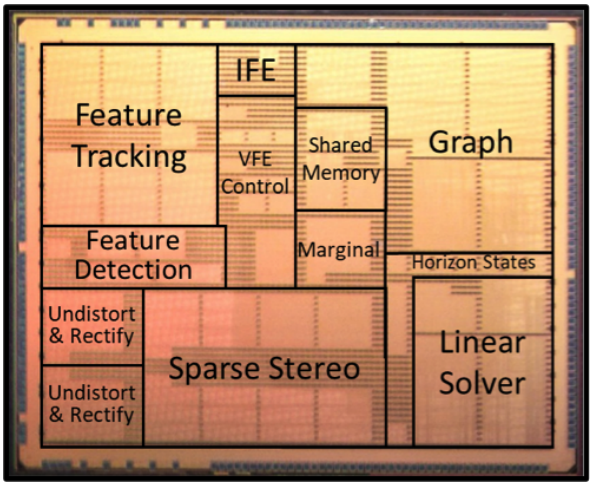

FIGURE 3.12: Photo of Navion chip (die) [203]

In short, this work presents the scaling up of VIO for nano/pico drones. It improves the VIO on an

FPGA, and proposes a new ASIC solution (considered the first) fully integrated, called Navion. This chipboard is manufactured in 65nm CMOS technology. The configuration of Navion relies on several programmable parameters, such as the KF rate, horizon size and number of feature tracks, which can significantly influence throughput, accuracy and energy efficiency. In fact, Navion is characterized by its capacity to give interesting results using visual-inertial data fusion. This is thanks to its suitability for different types of environment. In addition , this VIO system provides efficient and real-time processing. For example, in the case of EuRoC dataset where the frame rate is 20fps and the system power consumption is in an average of 2mW at 1V. While for the most problematic navigation cases, where stereo-vision images (size of 752x480) are acquired at a frame-rate up to 171fp and the IMU measurements are generated at a frequency up to 52kHz, the power consumption is estimated in an average of 24mW at 1V. These results makes it suitable for critical applications such as autonomous navigation, mapping and portable AR/VR.

VI SLAM Embedded Implementation exp. 1 : In addition, as explained before (chapters 1 and 2), SLAM remains a well-known navigation method for ego-motion tracking, that add to odometry the loop closing step. Until today there is still no enough works on embedded systems that integrates a complete VI SLAM. In the following, EMoVI SLAM [211] proposes a solution based on ORB-SLAM that considers a loosely coupling between a camera and an IMU. The pose generated by visual ORB SLAM and the pose computed from the IMU inertial measurements are combined using UKF filter to estimate the final pose. Also, in this work the authors proposed a method of computing scale and updating it, which is similar to the method used in VI ORB SLAM (figure 3.13). Finally, an embedded portability of EMoVI-SLAM was proposed (figure 3.14).

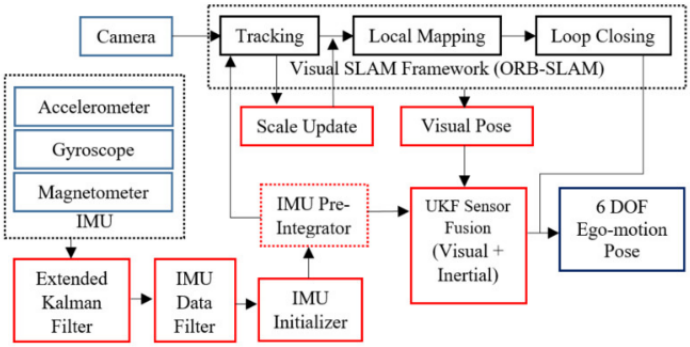

FIGURE 3.13: EMoVI-SLAM diagram (from [211])

FIGURE 3.14: Wearable EMoVI-SLAM (a head-gear and a processing unit tied on waist) (from [211])

As the focus in this section is on the embedded system implementation, the EMoVI-SLAM embedded implementation is described. The sensors used for this system are a camera with a fish-eye lens at $30fps$ and a 9-axis IMU at $200Hz$. The board used for the system implementation and evaluation is composed of a $1.92GHz$ processor and a $2GB$ DDR3L RAM memory. However, reducing the system's power consumption also allows the memory consumption reduction by optimizing the SLAM parameters configuration. In addition, EMoVI-SLAM has a maximum and minimum errors estimated by $0.234m$ and $0.006m$, respectively, on EuRoC dataset, with a power consumption estimated by $0.20A - 0.35A$ at $5V$.

VI SLAM Embedded Implementation exp. 2 : In 2018, another embedded VI SLAM was proposed, called PerceptIn Robotics Vision System (PIRVS) (figure 3.15) [231]. It provides a VI SLAM system characterized by a flexible sensor fusion and hardware co-design. It is composed of two OmniVision global shutter CMOS image sensors capturing 640×480 resolution color images at $60fps$, and a commercial grade InvenSense IMU reporting inertial measurements at $200Hz$. The chips used in this device are a dualcore ARM Cortex A72 and quad-core ARM Cortex A53 SoC, which also has an embedded ARM Mali-T860MP4 GPU with four shader cores with shared hierarchical tiler. The PIRVS has also a $2GB$ RAM and $8GB$ internal storage. In addition, this device provides two interfaces for external communication: 1) a 3-pin GPIO interface to connect with any robotics device (*e.g.*, ground robot, drone, *etc*.), 2) standard USB 3.0.

FIGURE 3.15: The embedded VI SLAM PIRVS package [231]

In this work, the tracking step is based on a tightly IMU/Camera fusion. In fact, after detecting and describing the features, the tracking is involves using the EKF algorithm. The EKF prediction step is done using the IMU inertial measurements, the EKF measurement step is calculated by exploring the visual information (a set of correspondences between 2D image features an 3D map points), and finally the EKF update is applied when both visual information and inertial measurement are available. The mapping is executed in parallel to the tracking step.

Moreover, PIRVS approach (PerceptIn Robotics Vision System) [231] offers the possibility of being loosely coupled to additional sensors, as GPS in figure 3.16. This is through the interfaces for external communication, such as the USB 3.0 which allows the PIRVS to function as a "plug-and-play" sensor, making it more flexible.

Briefly, the MAV proposed in [50] estimates trajectory alignment to the ground truth, then it computes the position RMSE over the aligned trajectory. The complexity of the datasets varies in terms of trajectory length, light dynamics, and illumination conditions. However, the best accuracy

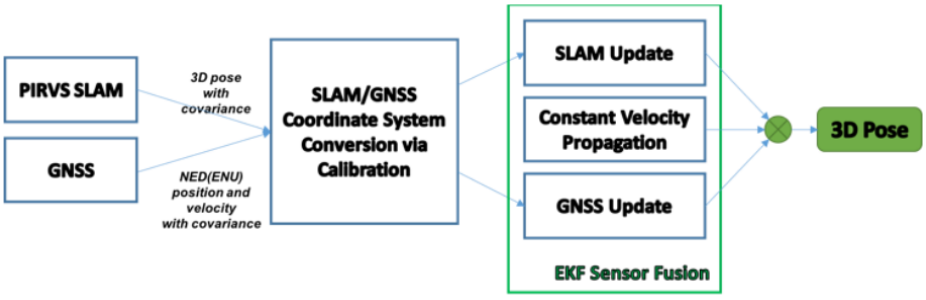

FIGURE 3.16: Loose coupling of PIRVS/GPS (PIRVS Loosely Coupled Sensor Fusion Example) [231]

results are obtained for VINS-Mono [175], with a loop closure. It gives an average of $0.12m$ in absolute translation error when considering Up Board (64 bit Intel Atom CPU). This error reaches $0.16m$ for VINS-Mono on ODROID XU4 (Samsung Exynos 5422 system in an ARM big.Little architecture 4 A7 at $1.5GHz$ and 4 A15 at $2.0GHz$).

To conclude, the literature contains different implementations of VIO or VI SLAM systems which confirms the interest of using a heterogeneous embedded computing architecture. This enables specific functions to be accelerated using dedicated architectures and computing accelerators. Consequently, enabling the implementation by targeting selectively kernels/tasks requires a precise and detailed knowledge of the application and is always a tedious work. Optimizing synchronization of sensors, data exchange, refine the process while fitting the hardware resources utilization comes at the price of performance degradation. These latter must remain acceptable in the targeted application domain.

3.4 Conclusion

This chapter addressed the navigation topic for embedded systems, including tracking and VIO/VI SLAM. Firstly, definition and features of embedded systems were discussed. Then, architecture and functionality of components which composes integrated circuits have been described. In fact, the focus was particularly on devices used for processing VINS. Depending on the system aims and use, the computing architecture choice is done to respect its specific constraints and design challenges, and to ensure its outputs accuracy and robustness. Furthermore, increasing the number of sensors leads to an increase in the processing complexity required to perform the data integration and fusion steps. Embedded energy efficient architecture are optimized and specialized but can lose accuracy whereas mainstream architecture are able to run various tasks on heterogeneous resources with accuracy at the cost of power.

To illustrate these issues, first, the co-design was explained, then relevant literature works was described and discussed. Actually, these works are mainly based on inertial-visual fusion, thus they are based on IMU/camera coupling. Their application's fields and use-cases are different, but all of them perform the visual-inertial pose estimation step for localization, tracking and navigation, using either VIO or VI SLAM methods. Figure 3.17 presents a trade-off between the main VI navigation embedded systems constraints, which are power consumption, integration density and processing accuracy and complexity, for different embedded systems implementations levels as smartphone, headset mounted, drone devices and autonomous vehicle. In fact, data fusion must produce an accurate and reliable navigation results (position, orientation, *etc.*), even the fact that the sensors are

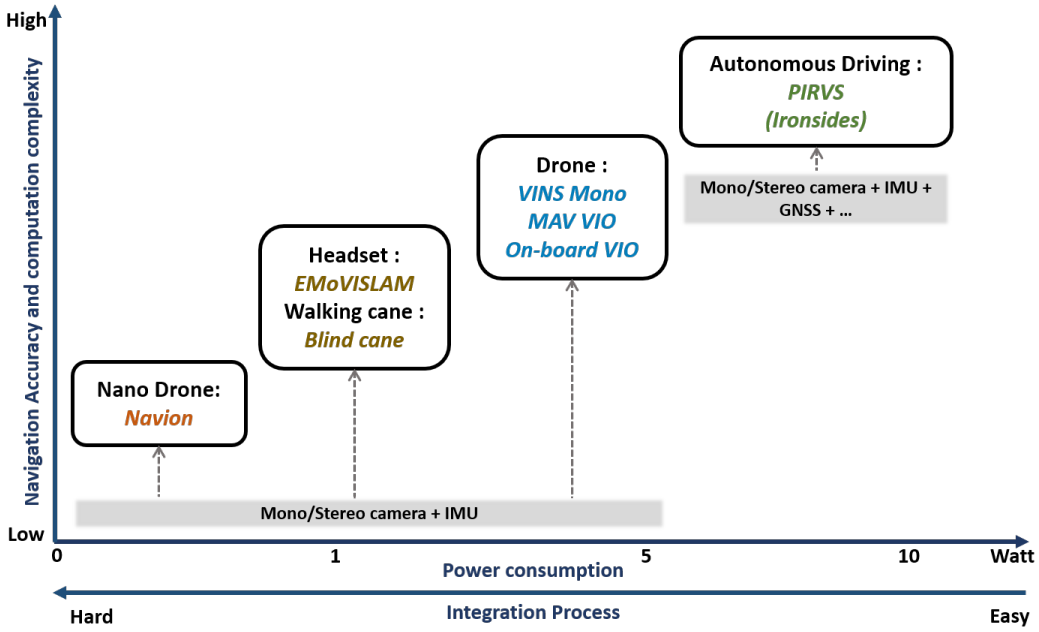

FIGURE 3.17: A classification of embedded tracking systems

multiple and various. Also, the applications and hardware resources determinism must be guaranteed for obvious reasons of responsibility in the case of an accident and compliance with automotive standards. Thus, the number of integrated circuits, as well as the size and number of electronic cards, increase accordingly. Considering autonomous mobile robot, autonomous nano drone, smartphones, *etc.*, the need of accuracy and autonomy are combined with real time constraints, *i.e.* they have to be able to rely on a dynamic/responsive global localization system and interactions support with others robots/vehicles (specific infrastructure and equipment as Vehicle-to-Infrastructure for example).

This chapter also shows that at the cost of hardware restrictions and algorithmic adaptation optimization, an acceptable accuracy can be achieved by the embedded systems. When combining with external positioning systems, a very precise location is even possible [231][211][203]. In any case, the design challenge will lie in the implementation and synchronization of all available localization sources available. Actually, taking position regarding the state-of-the-art presented in this chapter, as well as in previous ones, and taking benefit from these different studies and discussions, the proposed solution in this thesis is a visual-inertial tracking method. It is an adaptive approach to its navigation environment, designed for embedded systems (MAVs, drones, *etc.*). It works within the framework of the SLAM method, precisely, relies on ORB SLAM back-end. As discussed in this chapter, the Co-Design is considered to be one of the most optimal integration methods for this type of system. It allows us to seek a trade-off between algorithmic quality and embedded architecture constraints. This is one of the issues that we are interested particularly in our research. Moreover, it is possible to foresee MPSoC (FPGA) for our solution integration, which includes GPU acceleration for complex functions such as point detection/description, pose estimation, EKF-based visual-inertial data fusion, etc. Indeed, this kind of architecture solution is the most suitable for MAV and UAV applications in which we are first interested to test this work. Thus, a more constraining implementation inspired by the work done in Navion [203] can also be foreseen in the future. In the next chapter our thesis proposal is presented and discussed.

Chapter 4

Context Adaptive Visual-Inertial SLAM

Efficiency, accuracy and robustness represent a trade-off to be taken into account when developing an embedded SLAM system. Therefore, in order to achieve real-time pose, it must be optimized. As presented in the literature, the optimizations applied to SLAM allow its implementation on embedded platforms. However, these optimizations do not address the system's robustness and its accuracy in the different navigation conditions. For these reasons, this thesis proposal is to develop an adaptive, embedded and robust tracking method, which can integrate a SLAM system. Thus, it is based on the ORB SLAM methodology, it is among the well-known SLAM research in the literature and provides interesting accuracy results. The ORB SLAM source codes are open-source; therefore, they are widely used when comparing many different SLAM research works. Thanks to the execution control module, the proposed system has the ability to analyze the environment and navigation conditions in order to choose between two tracking approaches (figure 4.8): the visual-inertial technique based on EKF fusion algorithm and the visual technique based on KLT algorithm.

FIGURE 4.1: Context Adaptive VI SLAM workflow

In this chapter, we start in section 4.1, by analyzing the different scenes and movements types that occur during navigation. In addition, by using the different state-of-the-art VO, VIO, and SLAMs, and their evaluation results, we can detect problematic cases, *i.e.* those where SLAM does not perform successfully. Therefore, this study allowed us to identify the different navigation environments and to classify them into two main categories: easy, and difficult environments. Also, it leads us to choose appropriate metrics that will be employed in our execution control module in order to evaluate the environment and the execution status of our system, so that the most adapted tracking method can be correctly chosen. Then, we describe the Context Adaptive Visual-Inertial SLAM, starting with the description of its global workflow, in section 4.2, followed by the discussion about its different processing components in section 4.3, namely our implementation of KLT-ORB tracking, EKF VI tracking and execution control module for managing the entire system.

4.1 Navigation Environments Analysis

The main VINS challenge is their robustness to the various navigation environment constraints. A VINS can easily stall if the navigation conditions are too severe (scene condition, motions type, *etc*). Thus, the use of a relevant dataset may provide these constraints to check and validate the system in all possible environments. In fact, they may differ depending on the targeted applicative domain. As it is a cornerstone for any approach dealing with ego motion in embedded multisensory system, dataset must satisfy: application domain (automotive, robotic, drone, wearable device (VR/AR), *etc*.), objective/goal (function, indoor/outdoor, corner case (night, weather, dust...), *etc*.), configuration/setup (sensors type, position, resolution, precision, calibration data, framerate, sensor synchronization, *etc*.), ground truth (pose/map), size of dataset, and diversity, and back-annotation/labels.

With the democratization of mobile systems, many academic works focus on the building of dataset used for benchmark. These benchmarks have emerged for multi-purpose applications, including localization, tracking and mapping. Most of these benchmarks focus on images (sequences, monocular or stereoscopic, High Definition (HD) panoramas) that are geo-referenced. In addition, the newest ones may have additional information such as semantic data, inertial data or long-term

data, as well as variable conditions such as lighting (day/night), weather (sun/rain/snow), and season (summer/winter).

4.1.1 Main Datasets in Literature

Several datasets exist in the literature. The main ones used for localization and tracking purposes are described in tables 4.1 and 4.2. In these tables, it is possible to observe the main differences between the listed datasets, which are mainly related to the sensors types, the system and environment for dataset generation, the ground truth form, *etc*.

Oxford Robotcar [136] a public and common multimodal dataset used for image based localization. This dataset contains LiDAR point cloud, geo-localized stereo images, both captured from a terrestrial vehicle and in different lighting conditions as shown on figure 4.2, and RTK-corrected GPS/INS trajectories. A recent version, *RobotCar Seasons* [189], is proposed to deal with long-term localization.

FIGURE 4.2: Sequence example of the Oxford Robotcar dataset

TorontoCity [216] gathers both airborne and ground HD images and LiDAR, GPS and semantic information, vector maps, 3D models, in the coverage of a large city.

7-Scenes [194] represents a collection of tracked RGB-D camera frames (figure 4.3), such as the TUM RGB-D benchmark [60] which contains RGB-D data and ground truth for the evaluation of visual odometry and visual SLAM systems.

KITTI [142] composed of geolocalized stereo images and LiDAR data, collected while driving terrestrial vehicle around the mid-size city, in rural areas and on highways, with different dynamic objects (pedestrians, vehicles, *etc*.) and different scenes types (different light and shadow conditions) as depicted in figure 4.4. It was complemented by aerial information in [139].

FIGURE 4.3: Sequence example of the 7 scenes dataset

FIGURE 4.4: Sequence example of the KITTI dataset

EuRoC [24] recorded in the context of the European Robotics Challenge (EuRoC), to assess the contestant's visual-inertial SLAM and 3D reconstruction capabilities on Micro Aerial Vehicles (MAV). In addition, the position ground truth instruments was recorded with a Leica total station, and 6D pose ground truth are recorded with a Vicon. EuRoC MAV provides dataset in two different environments with three different difficulty levels (easy, medium and difficult) for each one: an indoor room called the "Vicon environment" designated as Vxxx, and a higher and more spacious machine hall known as "Machine Hall environment" and referred as MHxx. First "x" in Vxxx refers to the dataset sequence number, for Vicon environment there is two sequences sets (V1 and V2), each one is composed of three levels, the subsequent "xx" denotes the difficulty level (V101, V102, V103, V201, V202, and V203). The same for Machine Hall environment for which "xx" denotes also the dataset level (MH01, MH02, MH03 and MH04). The difference between these three difficulty levels depend mainly on the the motion types and scene conditions (subsection 4.1.2).

Shortname	Configuration			Ground Truth		Sensor Information											
	Year	Platform	Envir.	Pose	Map	IMU	GPS	Labels	2D Lidar	3D Lidar	Mono	Stereo	Omni	RGBD	Event	Radar	Sonar
UZH-FPV Drone Racing [48]	2019	UAV	Indoor, Outdoor	O		O					O				O		
Rosario Dataset [170]	2018	Mob	Terrain	O		O	O					O					
KAIST Day/Night[38]	2018	Veh	Urban	O		O	O	O		O	O						
Complex Urban [110]	2018	Veh	Urban	O	O	O	O		O	O							
Multi Vech Event [208]	2018	Veh	Urban	O		O	O			O		O			O (stereo)		
RPG-event [152]	2017	UAV / Hand	Indoor	O		O					O				O		
Robot @ Home [184]	2017	Mob	Indoor	O	O			O	O					O			
Zurich Urban MAV [137]	2017	UAV	Urban	O		O	O				O						
Chilean Underground [122]	2017	Mob	Terrain (Underground)	O						O		O				O	
Agricultural robot [35]	2017	Mob	Terrain	O			O	O		O	O			O			
Beach Rover [90]	2017	Mob	Terrain	O		O	O			O	O	O		O			
EuRoc [24]	2016	UAV	Indoor	O	O	O					O	O					
Cityscape [40]	2016	Veh	Urban	O			O	O			O						
Solar-UAV [109]	2016	UAV	Terrain	O	O	O	O		O								
Oxford-robotcar [136]	2016	Veh	Urban	O		O	O			O	O	O					
NCLT [28]	2016	Mob	Urban	O		O	O			O			O				

TABLE 4.1: Collection of odometry and SLAM datasets with individual data and sensor configuration details (Veh = Vehicule, Mob = Mobile) (1/2)

Shortname	Configuration			Ground Truth		Sensor Information											
	Year	Platform	Envir.	Pose	Map	IMU	GPS	Labels	2D Lidar	3D Lidar	Mono	Stereo	Omni	RGBD	Event	Radar	Sonar
KITTI[139]	2013	Veh	Urban	O		O	O	O	O	O	O	O					
Canadian Planetary [29]	2013	Mob	Terrain	O		O	O		O	O		O					
TUM-RGBD [**sturm12iros**]	2012	Hand / Mob	Indoor	O		O								O			
Devon Island Rover [74]	2012	Mob	Terrain	O			O					O					
UTIAS Multi-Robot [123]	2011	Mob	Urban	O				O									
Ford Campus [165]	2011	Veh	Urban	O		O	O		O	O	O		O				
San francisco [199]	2011	Veh	Urban	O		O	O	O		O	O		O				
Annotated-laser [226]	2011	Veh	Urban	O				O	O		O						
MIT-DARPA-Urban [94]	2010	Veh	Urban	O		O	O	O	O	O	O						
Marulan [167]	2010	Mob	Terrain	O		O	O		O		O					O	
NewCollege [196]	2009	Mob	Urban	O			O		O			O					
Rawseeds-indoor [30]	2009	Mob	Indoor	O	O	O			O		O	O	O				O
Rawseeds-outdoor [171]	2009	Mob	Urban	O	O	O	O		O		O	O	O				O

TABLE 4.2: Collection of odometry and SLAM datasets with individual data and sensor configuration details (Veh = Vehicule, Mob = Mobile) (2/2)

4.1.2 EuRoC MAV Dataset Analysis

Considering the datasets presented in the former section and according to this thesis proposal's purposes, the focus will be, through this work, on the EuRoC MAV dataset [24], because it provides various sequences of inertial and visual data, captured by drone, as well as the ground truth. Thus, this allows to evaluate the proposed system and to compare it with different literature works, since EuRoC is the database widely used by the computer vision community.

As explained above, EuRoC MAV dataset provides two types of dataset sequences:

- The first one (MHxx) is specifically meant to evaluate VI SLAM algorithms in a real industrial context. It was acquired in a low textured environment, characterized by reflective surfaces and many black areas. MHxx data is composed by a 3D position ground truth using the Leica multistation in a machine hall at ETH Zurich (figure 4.5).

FIGURE 4.5: The ETH Machine Hall environment (MHxx) [24]

- The second one (Vxxx) was captured in a room where different objects and obstacles had been placed to increase the texture and make the environment more challenging. It contains a 6D pose ground truth acquired using the Vicon motion capture system, and additional accurate 3D point cloud of the environment (figure 4.6), taken by the Leica 3D laser scanner.

FIGURE 4.6: The Vicon environment (Vxx) [24]

All datasets were recorded using an AscTec Firefly MAV equipped with visual inertial sensor as shown in figure 4.7. This MAV consists of a stereo camera Wide Video Graphics Array (WVGA) with global shutter, at $20fps$, IMU sensor MEMS with sampling rate $200Hz$.

In order to identify the most constraining motions and scene type, each sequence is analyzed and characterized according to the two factors that influence the dataset levels. In the following part, the synthesis of each factor for the eleven dataset sequences is presented. The scene can be described

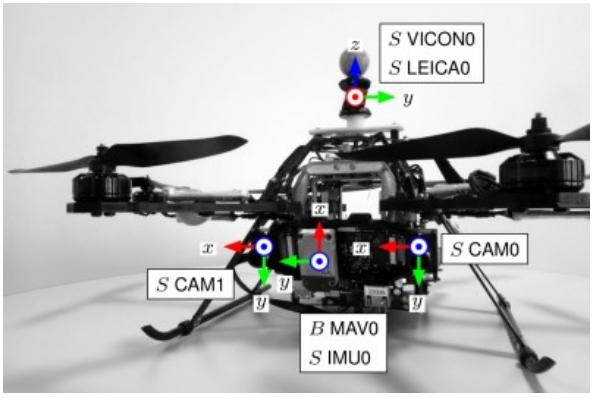

FIGURE 4.7: The Asctec Firefly hex-rotor helicopter for data acquisition [24]

by its luminosity, brightness, texture, exposure, *etc.*, whereas, motion is characterized by its type (rotation or/and translation), its velocity (linear or/and angular) and intensity. Table 4.3 lists the main scene characteristics of each dataset sequence. According to the analysis shown in this table,

Seq.	Level	Luminosity	Texture	Exposition
MHxx	Easy	Good without dark spots	Many objects, Various forms	N/A
	Medium	Good without dark spots		N/A
	Difficult	Artificial lighting with many dark spots		N/A
V1xx	Easy	Good without dark spots	Many geometric objects, Poor texture	N/A
	Medium	Good without dark spots		Low
	Difficult	Backlit scenes		Under/Over-exposed pictures
V2xx	Easy	The best lighting without dark spots	Many geometric objects, Good texture (with some exceptions)	N/A
	Medium	Artificial lighting with a few dark spots		N/A
	Difficult	Artificial lighting with many dark spots		A few under/over - exposed pictures

TABLE 4.3: Scene characteristics in the different EuRoC sequences

easy levels are mainly characterized by good luminosity and texture conditions, and without any exposition problem. Medium level is relatively more complicated than easy level: that's is because of exposition and lightning problems. Finally, difficult dataset level encompasses the majority of the scene problems. In particular, this level suffers from a lot of dark spots as well as a luminosity problem arising from lightning conditions and under/over-exposed frames.

In addition, table 4.4 shows the motion characteristics of various levels of different EuRoC dataset sequences. In this analysis, the system motion is divided into six motion type: roll, pitch, yaw, strafing, surging and elevation. A motion is considered difficult, when it is composed of several types of movements, when it is fast, and especially when it is a combination of these two criteria (fast and composite motion). The latter cause a high speed and abrupt motion as well as vibrations that lead to image blur.

To resume, following the different dataset sequences analyses presented above, we propose to simplify the EuRoC dataset sequences categorization into two major difficulty levels: easy and difficult navigation environment, as there is no great difference between the medium and difficult levels. In fact, for both levels the problems are mainly related to the scene, such as lighting, and to the motion type, such as vibrations. Table 4.5 shows the compendium of all the the main characteristics of these categories.

Seq.	Level	Roll	Pitch	Yaw	Strafing	Surging	Elevation
MHxx	Easy	Motion twice as slow as the difficult sequence with small changes in field of view angle (($\simeq 0.44m/s$ *and* $0.22rad/s$))					
	Medium	Fast and smooth motions with Yaw change and a speed twice faster than easy sequences (($\simeq 0.99m/s$ *and* $0.29rad/s$))					
	Difficult	No abrupt motion changes or any significant pitch changes, with random motions and a speed twice as fast as easy sequences (($\simeq 0.93m/s$ *and* $0.24rad/s$))					
V1xx	Easy	+	+	+	+	+	+
	Medium	+	+	+++	+	+++	+
	Difficult	+	+++	+++	+	++	+
V2xx	Easy	+	+	+	+	+	+
	Medium	+	++	+++	+	++	+
	Difficult	+	+++	+++	+	++	+

TABLE 4.4: Motion characteristics in the different EuRoC sequences (Crosses 'x' represent the motion occurrence level in the sequence in incremental order)

Navigation Environment	EuRoC Dataset	Description
Easy	V101, V201, MH01, MH02	**scene**: no dark spots, good texture, no exposure problem **motion**: small changes in FoV, slow motion (roll, pitch, yaw, *etc.*), slight vibration
Difficult	V102, V202, V203, V204, MH03, MH04	**scene**: many dark spots, various texture level, exposition problems, artificial lighting **motion**: random fast motion, abrupt motion, lot of pitch/yaw combination

TABLE 4.5: EuRoC dataset environment categorization

4.2 Context Adaptive Visual-Inertial SLAM Workflow

The proposal is developed within the framework of the SLAM method, in particular ORB SLAM. It focuses on tracking thread, where we provide two different approaches, independently implemented each one in a thread and both based on ORB PoIs. However, the Local Mapping and Loop Closing threads are the same as in ORB SLAM. Actually, as depicted in figure 4.1, our proposed system first analyzes the environment in order to choose the appropriate tracking method. Next, according to the chosen approach, an initialization step is carried out before starting the chosen tracking approach. Afterwards, the system interfaces with other threads following the same process as in ORB SLAM. The transition from tracking to Local Mapping and Loop Closing is performed through keyframe. Therefore, whenever a keyframe is selected, the following threads are executed.

In this section, the overall solution workflow, the transitions and the connections between its different components are discussed. To start, the initialization and tracking steps are presented. Then, local mapping and loop closure processes are described.

4.2.1 System Initialization

The system starts in visual-inertial mode, in order to ensure tracking at system launch, simultaneously with the first analysis of the motion and the surrounding visual scene. Afterwards, depending on the control module's decision, one of the two proposed tracking approaches is executed: either remain using visual-inertial tracking, or move into visual tracking. In both cases it is necessary to proceed through the initialization step. This is a required process for SLAM: it consists in preparing a first set of map points for non-linear optimization and loop closure. Firstly, the initialization process starts by extracting the ORB PoIs in the current image k and matching them with the reference image r. Depending on the number of matches, either the next step is executed or the current frame becomes the reference frame and the same process is repeated. Then, the second phase is to check

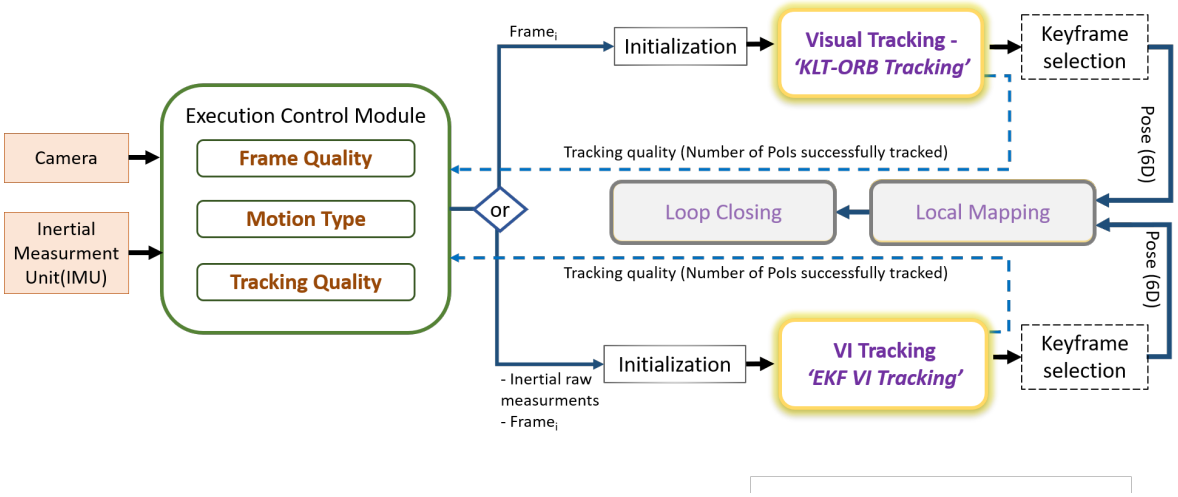

FIGURE 4.8: Context Adaptive VI SLAM overview

the parallax of each match in order to choose a set of feature matches $\mathcal{F}$ that have enough parallax. Provided that the size of $\mathcal{F}$ is greater than a given threshold, these feature matches $\mathcal{F}$ are triangulated, and then the next step (tracking process) is executed. In this work, the initialization step is less time consuming compared to other SLAMs, such as ORB SLAM. This is thanks to the proposed tracking methods. In EKF VI tracking mode, initialization step requires a maximum of 10 frames and in KLT-ORB tracking mode, the required number of frames is a maximum of 100 frames.

4.2.2 Tracking

At this stage, the tracking method is already selected (figure 4.8). Then, according to the chosen tracking approach, the execution and computation mode changes in order to maintain the consistence continuity of the following functions for the rest of the SLAM process.

KLT-ORB Tracking. The challenge of KLT tracking using ORB PoIs is the map point creation without having to detect PoIs on every frame (as long as there is no large motion between them). Indeed, after ORB PoIs detection, KLT tracking is executed in order to provide tracking from one frame to another, using only these ORB PoIs, as long as possible. Practically, the number of detected PoIs is limited to 500 and a PoIs re-detection is expected when this number is below 400. So to remain consistent with the map and to provide the correct functioning of the others parts of SLAM (local mapping & loop closure processes), the map point creation and the map updating are performed at the end of the visual tracking process, using the PoIs successfully tracked by the KLT algorithm. This provides gains related to PoIs detection/description decrease. Afterwards, subsequent tracking steps are performed by adapting the same ORB SLAM functions.

EKF VI Tracking. For EKF VI tracking, the process is different from KLT-ORB tracking. The only common thing done is the map points creation and the map update. The reason of this proceeding, is to ensure that the map will be up to date, when switching to visual tracking, in order to perform correctly the KLT-ORB SLAM.

In the proposed EKF VIO, the current system state $X_{B_k} = \left[q_{B_k}^{W^T}\ p_{B_k}^{W^T}\ v_{B_k}^{W^T}\ b_{g_k}^{T}\ b_{a_k}^{T}\right]^T$ is estimated using visual and inertial data. In our solution, the processed systems are embedded ones, so that number of PoIs to be computed is important since they are part of the state vector. Therefore, in

order to provide real-time performance, based on the visual EKF SLAM[185] and 1-point RANSAC [39] methods, it is a good idea to consider only the features visible in the current image. In fact, at the beginning of the EKF VIO process, the number of PoIs is limited to an average of 10. Subsequently, during the tracking, these points are sorted from one image to another in order to keep only the points that are still in the camera's FoV. When a PoI is not reliable enough, according to the score threshold computed during the sorting process, or when it is deleted, the system detects and reintegrates up to 5 new PoIs. Hence, based on the experiments (chapter 5), the maximum number of PoIs in a X_k state vector is up to 25 PoIs and the average number is 9 PoIs.

Keyframe Selection Decision This phase is the last tracking task, it comes after the map point creation and the map update. This phase consists in testing each image that has successfully completed the tracking process against three main criteria, inspired from the research of [153], [155] and [177]:

1. the time gap between two keyframes must be greater than a certain threshold. Indeed, an IMU provides precise and valuable measurements only when it comes to the short term, otherwise the measurements are not accurate enough. This criteria provides the accuracy and reliability of the system.

2. non-linear back-end optimization processing must be fully done. This criteria gives the possibility to have as many keyframes as possible, consequently it improves the motion tracking accuracy.

3. the rotation angle between the last keyframe and the current frame is also a critera for selecting keyframes, *i.e.* if the rotation angle is above a certain threshold so that image is considered as a keyframe. This criterion has the advantage of insuring the reconstruction of a globally coherent map.

By satisfying one of these criteria, the current image is selected as the keyframe and can then be used for local mapping and loop closure. The purpose of this step is not to use all the images in the SLAM, but to be limited to the keyframes.

4.2.3 Local Mapping & Loop Closure

Once a keyframe is selected and inserted, the system searches for new matches with the local map, then updates the covisibility graph. Subsequently, depending on the tracking information, the map points that are improperly triangulated are eliminated. Moreover, the current keyframe's ORB PoIs that are not matched, are made unable (prevented from finding) to find new matches with the rest of the connected keyframes in the covisibility graph. This is done in order to build new map points. Subsequently, non-linear optimization based on the local BA is performed to optimize the local map. Once completed, redundant keyframes are eliminated in order to make the factor graph more concise.

Loop closure performs place recognition via the DBoW2 [76] functions using it in the same way as [153] and [155]. So when a new loop is detected, a Sim(3) optimization and a full BA are performed to eliminate the accumulated drift. In the thesis's proposal, the interest is not to improve these two processes, but to ensure the consistency of the overall system workflow and its functioning.

4.3 Context Adaptive Visual-Inertial Tracking Components

After explaining the global workflow of the Context Adaptive VI SLAM, we will especially focus, in this section, on the proposed Context Adaptive VI Tracking components which are execution control module, EKF VI tracking and KLT-ORB tracking. As shown in figure 4.9, first the system and tracking status are checked. Then, the system motion is measured in order to quantify the camera's Field of View (FoV) status, finally, the required number of PoIs is verified. Visual-inertial EKF tracking is selected mainly when the visual data is likely to be corrupted, either because of fast motions, scenes with low texture or brightness problems, *etc*. So the system uses the EKF fusion of inertial and visual measurements to ensure a successful and robust tracking. This method is less computationally intensive than other tracking methods based on data fusion, but it is even more computationally intensive than a vision-based method. This is due to the additional cost of processing inertial measurements and their use for the pose computation. Therefore, it is advantageous to alternate in some cases to the visual KLT-based tracking, especially for the easier navigation environments (table 4.4). The visual KLT tracking algorithm is selected when there are no large motion, the frame quality is sufficient (*e.g.* textured scenes, without blur) and the previous tracking status is satisfying.

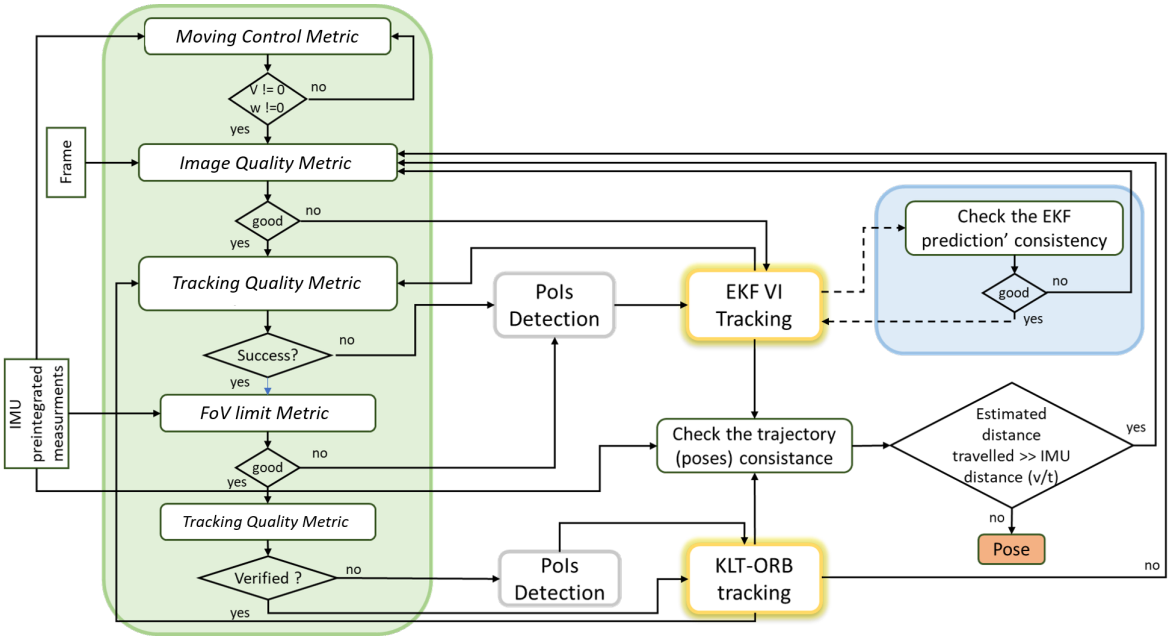

FIGURE 4.9: The adaptive execution control flowchart (v: velocity, w: angular velocity, t: time)

Throughout this section, the various components of the proposed method are discussed. First, the execution control module is introduced, by outlining its various metrics and discussing its functioning. Next, the visual tracking based on the KLT algorithm and the visual-inertial tracking based on EKF are presented along with their various operating steps.

4.3.1 Execution Control Module

To improve the SLAM's robustness while, making at the same time its implementation easier on embedded architectures, lead us to design a solution that switches between two different tracking techniques. The system is able to select the relevant tracking mode depending on current navigation environment characteristics. The goal is to reduce computational complexity and to ensure tracking in the most constraining environments. This is in order to choose the appropriate tracking method

for the appropriate motions and external environment. Therefore, in order to produce an optimal correct pose, two control modes are proposed: global and local control, using various control metrics and combinations.

Global Control Metrics

The global control, represented by green box on figure 4.9, enables the system status to be checked before choosing the tracking and pose computation methods to be performed, as well as the final pose accuracy and the overall trajectory consistency to be verified. The main metrics employed for this execution control module are:

- Moving Control Metric The controller analyzes the system moving state (mobile or immobile) in order to avoid the pose re-computation. Or, the IMU velocity information (the linear velocity v and the angular velocity w) are used, and system is considered mobile as soon as its velocity is non-zero ($v \neq 0$ and/or $w \neq 0$). In fact, recalculating the pose when the system is not moving (immobile system) causes a loss of processing resources, and can also leads to errors, due primarily to accumulated noises, in the pose computation for the next step (when the system starts moving again). So, when system is immobile, neither of the two tracking methods is executed and the state of the system remains the same as after the last pose computation.

- Image Quality Metric The image quality is evaluated before starting the tracking process. This evaluation is carried out by computing the histogram of the pixel intensity values for each frame. Thus we can determine whether the image has a correctly exposed histogram so it can be used as the input to extract new PoIs and execute the purely visual tracking, or it has an underexposed or overexposed histogram then this frame is not mineable for extracting correct PoIs. Consequently, it is necessary to use inertial data and therefore go through visual-inertial tracking. In fact, based on pixel intensity histogram, once more than 80% of the frame is dark, or more than 83% of it is white, the frame is considered as underexposed or overexposed, respectively.

- Field of View Limit Metric After the inertial data (measured between two frames) preintegration, explained latter, the IMU can give a 6D pose for each frame: a 3D translation plus a 3D rotation (figure 4.10), which allow to quantify the system's motion between two frames (two final poses), in terms of rotational angles as well as angular and linear velocities. According to the detected motion

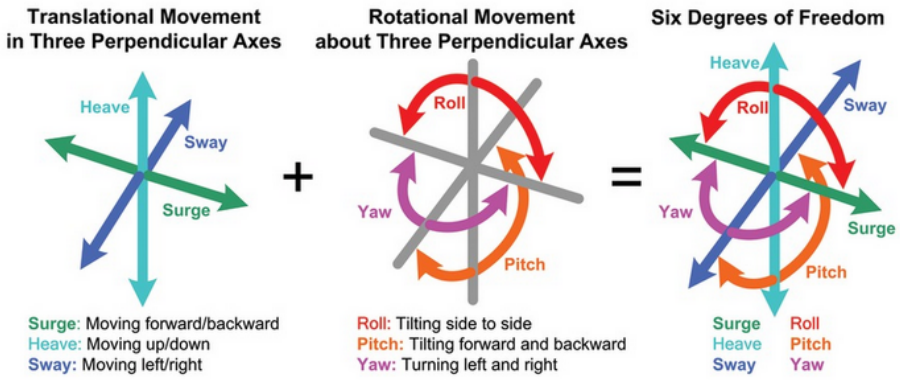

FIGURE 4.10: The six DoF movement composition

type and magnitude, the module can recognize if the FoV is changed or not. Therefore, it allows to know if there is enough visual information to continue the tracking by referring to the previous frame. In fact, if the FoV has changed, the new image no longer contains the same information as the previous one, nor the minimum information required to visually calculate the pose without getting lost. So, the control module select the visual-inertial tracking.

Generally, for mobile embedded systems, such as smart glasses and drones (MAV), rotation is the most influencing motion. It is possible to be roll, pitch, yaw or a combination of these three rotations. Thanks to the EuRoC MAV dataset analysis, the rotation thresholds are defined by fixing a rotation angle maximum for each axis, as follow (figure 4.11): roll angle < 41°, pitch angle < 61° and yaw angle < 79°. Also, the angular velocity impacts the rotational motion. According to the EuRoC dataset and the literature works, the problematic angular velocity starts from $\simeq 0.62 rad/s$.

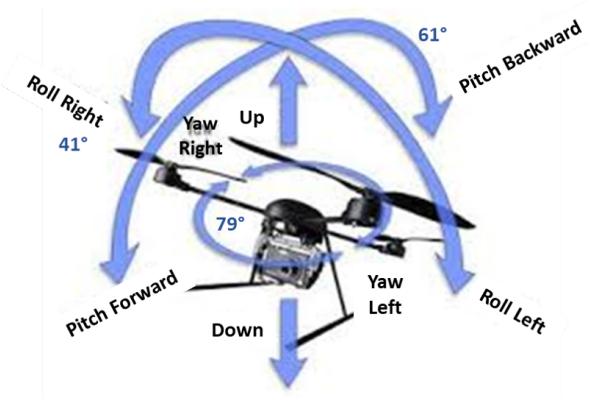

FIGURE 4.11: Rotation thresholds (estimated from head and drone motion analysis)

The translation constraints are not trivial, as well. According to the motion characterization in section 4.1, the displacement becomes problematic mainly when the linear velocity exceeds $\simeq 0.75 m/s$. Moreover, the brutality and rapidity of the movements (translation or rotation), as well as the vibrations, can cause a motion blur in the frame. Therefore, this motion analysis can detect this phenomenon and avoid its impact on the system accuracy and robustness, as well.

- Tracking Quality Metric (sufficient PoIs) Using this metric, the execution control module checks the system state by analyzing the previous tracking status, relying on current and previous frame PoIs matching. For this purpose, the required metric is the number of PoIs correctly tracked in the previous frame. In fact, the analysis of the literature leads to set the threshold for the number of successfully tracked PoIs mainly at 90% of the overall number of detected PoIs [100][187].

- Trajectory Consistency Metric This control metric is applied after chosen the tracking method and calculated the pose. The objective of this verification is to detect erroneous poses in order to exclude them from the subsequent pose computation. Bad poses can cause errors in the next tracking processing steps. Also, this test allows to remove outliers from the trajectory and to smooth it. The trajectory consistency metric is the error calculation between the distance travelled, calculated from the IMU speed, and the displacement duration and the distance calculated between the current pose and the previous one.

Local Control Metric

In addition to the measurements and tests presented above, there is a local control. This metric is inspired from [107] and [197] and called EKF Consistency Metric. As presented in figure 4.9 (blue box), the local control is applied to EKF VI tracking method. That will facilitate the earlier detection of the pose computation problems, thus avoiding expensive computations. For visual-inertial EKF tracking, the consistency of the current EKF fusion filter is evaluated. The metric used is Normalized Deviation Squared (NDS), which deals with the measurement prediction [107]. It evaluates the consistency of the measurement prediction (made by the EKF filter) on a previous measurements sample. At this level, if the filter produces non-coherent measurement predictions, the final pose is not calculated and the system moves to the next image [197][107] .

Control Module Functioning

The control module provides the previously described metrics in a sequenced testing process (green block in figure 4.9). These tests ascertain whether or not each metric's thresholds are satisfied in order to generate a Boolean response (*e.g.* 0 or 1) that finally enables the most appropriate tracking mode to be performed.

At the start, the control module analyzes the present state of the system using the moving control metric. Generally, if the system is in stationary mode, it keeps its last estimated pose, otherwise it can go on the other measuring tests. This allows to optimize the processing time by avoiding recalculating the same pose several times. Secondly, if the system is moving then the image quality is checked. Actually, a histogram analysis is performed, at each frame received, in order to evaluate the frame quality predominantly based on its luminosity. Once the frame is identified as potentially problematic, the system switches directly to visual-inertial tracking while waiting for the next frame. Using the frame quality metric ensures the robustness of the system and avoids tracking failure due to dark or overexposed locations, for example. Following these two previous tests, which mainly optimize computing performance, the control module checks the FoV, that above all enables to improve the tracking robustness and quality. As it is explained before, the FoV metric depends mainly on the rotational angle, angular velocity and/or linear velocity in order to maintain the tracking continuity in problematic cases. In fact, if the system is moving, the current frame is correct and the FoV thresholds are satisfied, the system is considered navigating in a difficult context and the EKF VI tracking is activated. Otherwise the system switch to visual KLT-ORB Tracking assuming that the navigation context is easy. Next control metric, which is the tracking quality metric, is only applied at the last step of the control module, when we have already verified that the system is moving, the current frame is correctly exposed and the FoV is not changed. It is primarily used to verify the previous tracking success rate in order to identify whether ORB PoIs need to be re-detected before executing the KLT-ORB visual tracking or not. In contrast to the foregoing tests, the pose estimate consistency can be tested in an offline mode using the local control metric inspired from [107] and [197]. Actually, this test is only applied in order to improve the EKF VI pose quality relying especially on the EKF prediction.

4.3.2 Visual KLT-ORB Tracking

As presented on figure 4.12, the proposed visual tracking method is based on the KLT algorithm [202], triangulation and pose estimation, according to the section 1.1, it is considered as a sparse

optical flow algorithm used to track features between consecutive images. Specifically, it is the pyramidal KLT algorithm. Its benefits are more obvious when the computation error is more important, as in a high image resolution or an important distance travelled by the camera. Indeed, it improves tracking processing accuracy and robustness by reducing the computation error. The KLT pyramidal algorithm consists in detecting images at different resolutions, *i.e.* from the smallest resolution of the image to the resolution of the original image. Here, the number of the used pyramids is initially configured in three layers in order to respect the effects of real time problems. The KLT algorithm calculates the pixel motion starting from the pyramid with the lowest resolution, then applying the scale transformation. Each pixel motion computed is considered as the initial value for the following layer. The accuracy increases from one pyramid layer to another until the original image where finally the pixel motion is computed.

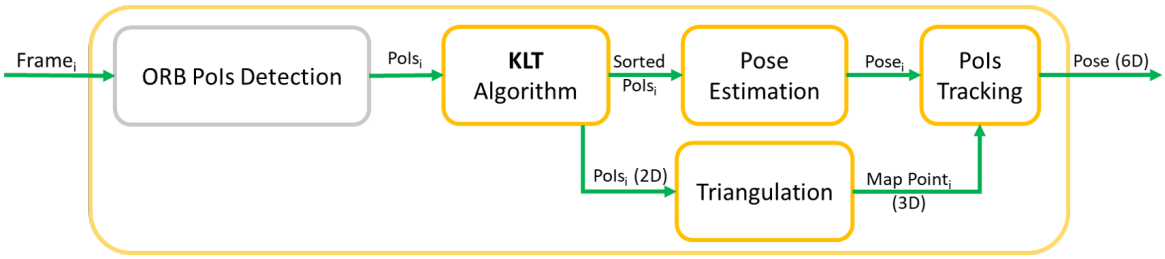

FIGURE 4.12: Visual KLT-ORB Tracking workflow

Compared to other tracking methods, optical flow is among the fast ones. In particular, when the number of PoIs is low, the optical flow is more efficient. Therefore, only the matched PoIs are used to track the current frame. In fact, only successfully tracked PoIs are saved with the current frame that will be set as the reference frame for the subsequent process. Nevertheless, it should not be forgotten that the KLT optical flow method is influenced by changes in grayscale and may be inaccurate if the grayscale invariance assumption is not respected, for example when the neighboring environment changes or the camera's exposure parameters change. This is controlled relying on the image quality metric. In fact, frames where more than 80% of their pixels are black or white are eliminated from tracking processing.

After applying KLT algorithm, the visual pose estimation is performed. The relative pose estimation between two frames can be performed by epipolar geometry [180] using 2D-2D image matching (section 1.3 / appendix B). In this work, such a computation is only performed during the initialization step. Moreover, using ORB SLAM as a platform to integrate the proposed optical flow tracking leads to retaining one of already integrated pose calculation modes, namely the pose estimation using only the last computed pose and a constant velocity motion model [153] : *current frame pose = velocity motion model * last frame pose*. Indeed, a motion model represents the uniform linear motion matrix of each processed frame. It is computed on the basis of the pose estimation just updated in the current frame and relative transformation of the previous frame : *velocity motion model = current frame pose * last frame transformation*, and it is used for the next pose computation. Consequently, the PoI's 2D-3D projection is no longer performed as in original ORB SLAM. However, one must keep in mind, in the ORB SLAM workflow the PoIs matching, pose computation and map update and optimization are done based on this reprojection [153]. Therefore, since the proposed solution relies on the same principle as ORB SLAM, except for the tracking and pose estimation, these measurements must be provided. Thus another way to calculate this 2D-3D reprojection is suggested. In the proposed system, after the PoIs detection and as long as the KLT visual tracking conditions are still valid (mainly the number of successfully tracked PoIs), the PoIs successfully tracked by KLT are sorted and

the map is updated by recalculating the tringulation and the 2D-3D reprojection of these points only. Thereupon, if the next frame still meets the criteria to be processed by the KLT VO method, then it is not necessary to match the PoIs of the previous frame with the local map to validate the PoIs of the tracking. However, when the number of the PoIs successfully tracked is below the threshold, then the tracking fails, and the current frame leaves the VO module to return to the VIO module.

4.3.3 EKF Visual-Inertial Tracking

Most literature studies prove that the visual and inertial data combination provides robust and accurate tracking, even in difficult environments. EKF is among the most widely used algorithms for this purpose, especially in embedded systems. It can calculate a pose by relying on the preintegrated IMU data and the camera's visual data. To begin the process, since the frequency of the two sensors (IMU and camera) differs, the IMU measurements are preprocessed using the inertial measurement preintegration technique in order to be able to use them with the camera's data. Figure 4.13 presents the proposed EKF VI Tracking workflow based on IMU preintegration, ORB PoIs detection and EKF VIO process which consisting of prediction, measurement and update steps. In the following, IMU preintegration is discussed, then the different EKF VIO process steps are described.

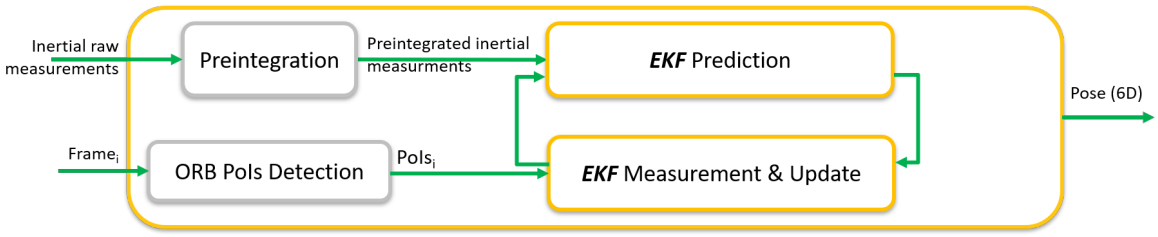

FIGURE 4.13: EKF VI Tracking workflow

IMU Preintegration Process

First the IMU preintegration is performed, the EKF algorithm will receive the resulting IMU measures and frames as inputs. As illustrated on figure 4.14 red ellipses identifies inertial measurements between two captured frames. In fact, the IMU frequency is more higher than the camera frame rate. To avoid frequent integration of IMU measurements, the IMU preintegration approach proposed in [69], is applied. In fact, all the IMU accelerationa_k and angular velocity ω_k between frames i and j are integrated to compute the IMU preintegration Δ_{ij} (blue square shown in figure 4.13). It is mainly composed of the preintegrated rotation $\Delta\tilde{q}_{ij}$, displacement $\Delta\tilde{p}_{ij}$ and velocity $\Delta\tilde{v}_{ij}$.

Inspired by [177], these terms are expressed as follows (equations 4.1, 4.2 and 4.3):

$$\Delta\tilde{q}_{ij} = \prod_{k=i}^{j-1} \begin{bmatrix} 1 \\ \frac{1}{2}(\tilde{\omega}_k - b_{g_i})\Delta t \end{bmatrix} \tag{4.1}$$

$$\Delta\tilde{p}_{ij} = \sum_{k=i}^{j-1} (\Delta\tilde{v}_{ik}\Delta t + \frac{1}{2}\Delta\tilde{R}_{ik}(\tilde{a}_k - b_{a_i})\Delta t^2) \tag{4.2}$$

$$\Delta\tilde{v}_{ij} = \sum_{k=i}^{j-1} \Delta\tilde{R}_{ik}(\tilde{a}_k - b_{a_i})\Delta t \tag{4.3}$$

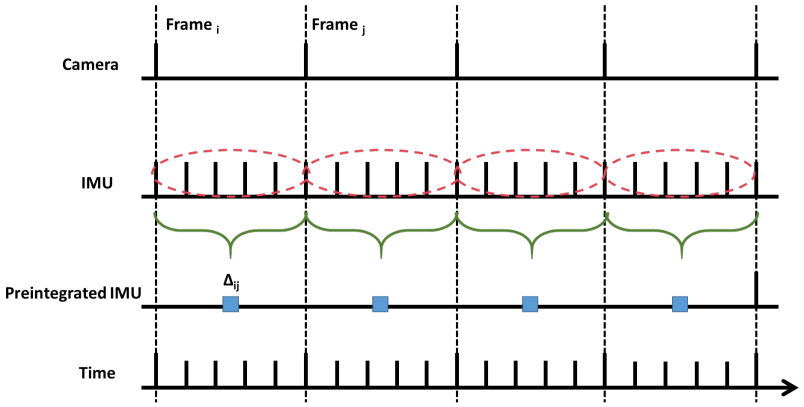

FIGURE 4.14: IMU preintegration diagram

Using first-order expansion and based on geometric constraints, the preintegrated IMU measurement residual $\begin{bmatrix} r_{\Delta q_{ij}}^T & r_{\Delta p_{ij}}^T & r_{\Delta v_{ij}}^T & r_{\Delta b_{g_{ij}}}^T & r_{\Delta b_{a_{ij}}}^T \end{bmatrix}^T \in \mathbb{R}^{15}$, used for pose optimization, is expressed as:

$$r_{\Delta q_{ij}}^T = \left(\Delta\tilde{q}_{ij}(\tilde{b}_{g_i}) \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\frac{\partial \Delta\tilde{q}_{ij}}{\partial b_g}\delta b_g \end{bmatrix} \right)^{-1} \otimes q_{B_i}^{W^{-1}} \otimes q_{B_j}^W \tag{4.4}$$

$$r_{\Delta p_{ij}}^T = R_{B_i}^{W^T}(p_{B_j}^W - p_{B_i}^W - v_{B_i}^W \Delta t_{ij} - \frac{1}{2} g^W \Delta t_{ij}^2) - \left[\Delta\tilde{p}_{ij}(\bar{b}_{gi}, \bar{b}_{ai}) + \frac{\partial \Delta\tilde{p}_{ij}}{\partial b_g}\delta b_g + \frac{\partial \Delta\tilde{p}_{ij}}{\partial b_a}\delta b_a \right] \tag{4.5}$$

$$r_{\Delta v_{ij}}^T = R_{B_i}^{W^T}(v_{B_j}^W - v_{B_i}^W - g^W \Delta t_{ij}) - \left[\Delta\tilde{v}_{ij}(\bar{b}_{gi}, \bar{b}_{ai}) + \frac{\partial \Delta\tilde{v}_{ij}}{\partial b_g}\delta b_g + \frac{\partial \Delta\tilde{v}_{ij}}{\partial b_a}\delta b_a \right] \tag{4.6}$$

with Jacobian $\frac{\partial \Delta(.)}{\partial b_.}$ describing the impact of bias estimation change on IMU preintegrated measures.

Therefore, according to the IMU preintegration functions described above, the constraints between two frames can be expressed using the inertial data. The current image j is considered irrelevant or unimportant for the SLAM process, when the camera motion is slightly changed and the parallax between the two images i and j is slightly weak. This motion quantity is derived from the IMU preintegration value. For this image type, specifically when the Δ_{ij} value (figure 4.14) is below the threshold, the camera pose is estimated using the visual tracking module and this image is considered as not important for SLAM and skipped.

EKF VIO Process

The EKF VIO process is based on: Prediction, Measurement and Update, but first, the studied system state vector must be defined. Below, the system state vector composition is described. Then, the different EKF steps, as used in the proposed method, are presented.

State Vector Composition The system is represented by the state vector X_k. The latter is composed of the IMU state X_{B_k} and a set of the landmarks that correspond to the tracked features X_{L_k}:

$$X_k = \begin{bmatrix} X_{B_k}^T, X_{L_k}^T \end{bmatrix}^T \tag{4.7}$$

$$X_{B_k} = \left[q_{B_k}^{W^T} \; p_{B_k}^{W^T} \; v_{B_k}^{W^T} \; b_{g_k}^T \; b_{a_k}^T \right]^T \tag{4.8}$$

$$X_{L_k} = \left[f_1^{W^T} \; \dots \; f_m^{W^T} \right]^T \tag{4.9}$$

with:

$q_{B_k}^W$: the unit quaternion that represents the rotation of the reference frame B_k to the word frame W;

$p_{B_k}^W \in \mathbb{R}^3$: the $3D$ position in the reference B_k with respect to W;

$v_{B_k}^W \in \mathbb{R}^3$: the $3D$ speed of the reference B_k with respect to W;

b_g and b_a: the additive bias of the gyroscope and accelerometer, respectively;

$f_l^W = [x_l \; y_l \; z_l \; \theta_l \; \phi_l \; \rho_l]^T$: the l^{th} landmark inverse depth coordinates [147]. It is composed of the camera position $(x_l, y_l, z_l)^T$ where the l^{th} landmark was firstly observed, the azimuth θ_l and the elevation ϕ_l angle that define the unit radius (expressed in the global frame) from the camera center $(x_l, y_l, z_l)^T$ to the l^{th} landmark, and ρ_l is its inverse depth within the unit ray.

Therefore, following 4.7, the EKF error state vector is expressed as:

$$X_k = \left[\delta q_{B_k}^{W^T} \; \delta p_{B_k}^{W^T} \; \delta v_{B_k}^{W^T} \; \delta b_{g_k}^T \; \delta b_{a_k}^T \; \delta f_1^{W^T} \; \dots \; \delta f_m^{W^T} \right]^T \tag{4.10}$$

where the standard additive error: $x = \tilde{x} + \delta x$ is used for the 3D position, velocity, biases, and landmarks.

Prediction - IMU Model IMU measures a system's angular velocity $\hat{w}$ and acceleration $\hat{a}$ with respect to the inertial frame $\{B\}$. These measurements are assumed to be affected by zero mean Gaussian white noise η and bias b that varies slowly over time:

$$\hat{\omega} = \omega + b_g + \eta_g \tag{4.11}$$

$$\hat{a} = a + b_a + \eta_a \tag{4.12}$$

In practice, the IMU provides measurements at defined times. To facilitate EKF prediction, the IMU propagation model is given directly in discrete time, which provides the necessary derivatives when calculating in close form. Subsequently, the discrete-time IMU state propagation model $X_{B_{k|k-1}} = f_k(X_{B_{k-1}})$ is expressed using the measured acceleration $\hat{a}_{k-1}$ and the angular velocity $\hat{\omega}_{k-1}$ obtained by IMU:

$$\tilde{q}_{B_{k|k-1}}^W = \tilde{q}_{B_{k-1}}^W \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(\hat{\omega}_{k-1} - \hat{b}_{g_{k-1}})\Delta t \end{bmatrix} \tag{4.13}$$

$$\tilde{p}_{B_{k|k-1}}^W = \tilde{p}_{B_{k-1}}^W + \tilde{v}_{B_{k|k-1}}^W \Delta t + \frac{1}{2} g^W \Delta t^2 + \frac{1}{2} \tilde{R}_{B_{k-1}}^W (\hat{a}_{k-1} - \tilde{b}_{a_{k-1}}) \Delta t^2 \tag{4.14}$$

$$\tilde{v}_{B_{k|k-1}}^W = \tilde{v}_{B_{k-1}}^W + \frac{1}{2} g^W \Delta t + \tilde{R}_{B_{k-1}}^W (\hat{a}_{k-1} - \tilde{b}_{a_{k-1}}) \Delta t \tag{4.15}$$

$$\tilde{b}_{g_{k|k-1}} = \tilde{b}_{g_{k-1}} \tag{4.16}$$

$$\tilde{b}_{a_{k|k-1}} = \tilde{b}_{a_{k-1}} \tag{4.17}$$

where g^W the gravity vector expressed in the global frame W.

In this work, the slow random walk of the inertial biases is ignored, therefore the biases b_a and b_g are considered fixed and are estimated as part of the system state.

The first-order discrete-time linearized IMU error state propagation model is:

$$\delta X_{B_{k|k-1}} = \Phi_k \delta X_{B_{k-1}} + G_k \eta_B \tag{4.18}$$

where $\eta_B = [\eta_a^T \ \ \eta_g^T]^T$ the system noise, its associated diagonal covariance Q is expressed as follows:

$$Q = \begin{bmatrix} \frac{\sigma_a}{\Delta t} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \frac{\sigma_g}{\Delta t} \end{bmatrix} \tag{4.19}$$

where:

σ_a: accelerometer covariance.

σ_g: gyroscope covariance.

Φ_k and G_k: the $f_k(.)$ Jacobian matrix with regard to the IMU state and the system noise Jacobian matrix, respectively.

Therefore, the covariance matrix is propagated as follows:

$$P_{k|k-1} = \begin{bmatrix} P_{B_{k|k-1}} & P_{BL_{k|k-1}} \\ P_{BL_{k|k-1}} & P_{L_{k|k-1}} \end{bmatrix} = \begin{bmatrix} \Phi_k P_{B_{k-1}} \Phi_k^T + G_k Q G_k^T & \Phi_k P_{BL_{k-1}} \\ P_{BL_{k-1}} \Phi_k^T & P_{L_{k-1}} \end{bmatrix} \tag{4.20}$$

where $P_{L_{k-1}}$ is the landmarks covariance matrix (it depends on the camera covariance parameters).

Measurement Model The inverse depth representation [147] is applied here to represent features in order to immediately use the new ones. This representation improves the linearity of the measurement equations, and the management of low parallax features. Thus, it enhances the accuracy of the system. The inverse depth representation f_l^W for the l_{th} landmark can be transformed into corresponding $EuclideanXYZ$ coordinates $y_l^W(f_l^W)$ as :

$$y_l^W(f_l^W) = \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} + \frac{1}{\rho_l} m(\theta_l, \phi_l) \tag{4.21}$$

where $m(\theta_l, \phi_l)$ is defined as:

$$m(\theta_l, \phi_l) = \begin{bmatrix} cos\phi_l sin\theta_l \\ -sin\phi_l \\ cos\phi_l cos\theta_l \end{bmatrix} \tag{4.22}$$

Therefore, the measurement model that describes the projection of the l_{th} landmark, wich is represented in inverse depth coordinates, to the k_{th} image is:

$$z_{kl} = h_{kl}(X_{B_{k|k-1}}, f_l^W) + \sigma_{kl} \tag{4.23}$$

$$z_{kl} = \pi((R_{B_k}^W R_C^B)^T (y_l^W(f_l^W) - p_{B_k}^W - R_{b_k}^W p_C^B)) + \sigma_{kl} \tag{4.24}$$

where σ_{kl} is the measurement noise having the covariance Σ_{kl}, and π is the projection function determined by the camera's intrinsic parameters, previously known from the calibration. According to this measurement model, the reprojection error is calculated as follows:

$$r_{kl} = z_{kl} - h_{kl}(\tilde{X}_{B_{k|k-1}}, \tilde{f}_l^W) \tag{4.25}$$

and its linearized approximation is:

$$r_{kl} \simeq H_{B_{lk}} \delta X_{B_k} + H_{f_{lk}} \delta f_l^W + \sigma_{kl} = H_{kl} \delta X_k + \sigma_{kl} \tag{4.26}$$

where the $H_{B_{kl}}$ and $H_{f_{kl}}$ matrices are derived from the measurement model $h_{kl}(X_{B_k}, f_{W_l})$, with respect to the IMU state estimation and the position of the l_{th} landmark respectively. Thus, the Jacobian measurement matrix H_{kl} is defined as:

$$H_{kl} = \begin{bmatrix} H_{B_{kl}} \, 0 \ldots H_{f_{kl}} \, 0 \ldots \end{bmatrix} \tag{4.27}$$

Update The estimated state update is performed by stacking the m individual residual measurements r_{kl} at time step k together to form a single residual vector $r_k = [r_{k1}^T \ldots r_{kl}^T \ldots r_{km}^T]^T$ of $2m \times 1$ expressed as:

$$r_k = H_k \delta X_k + \sigma_{kl} \tag{4.28}$$

Similarly, the measurement Jacobians are also combined into a single measurement matrix of $2m \times n$ as: $H_k = [H_{k1}^T \ldots H_{kl}^T \ldots H_{km}^T]^T$. Afterwards, the full EKF state and covariance matrix are updated as follows:

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + \Sigma_{\sigma_k})^{-1} \tag{4.29}$$

$$\tilde{X}_{k|k} = \tilde{X}_{k|k-1} \circ K_k r_k \tag{4.30}$$

$$P_{k|k} = (I_{16+6m} - K_k H_k) P_{k|k-1} \tag{4.31}$$

with:

$\Sigma_{\sigma k}$ (equation 4.29): the stacked covariance matrix of $2m \times 2m$ of visual measurements

$\circ$ (operator in equation 4.30): equivalent to the operator $\oplus$ for orientation and vector addition for other state.

According to system state prediction based on IMU measurement, the features used for the update are first detected, by the ORB detector, then matched. Subsequently, the *Mahalanobis* distance $d = r_{kl}^T (H_{kl} P_{k|k-1} H_{kl}^T + \Sigma_{\sigma_{kl}})^{-1} r_{kl}$ is calculated to select the matched features that will be employed for the update. In fact, only features whose d is below the given threshold are considered inliers and are therefore used for the EKF update. In addition, the 1-point RANSAC method [39] is applied here to find reliable inliers.

4.4 Conclusion

In this chapter, the proposed Context Adaptive VI SLAM for mobile embedded systems is detailed. First of all, the navigation environment analysis is a step that has proven to be crucial for a robust tracking and SLAM development. This analysis allows, on the one hand, to identify problematic

navigation environments that require a greater effort in terms of computing time and capacity, on the other hand, to find the cases where navigation may be more easily insured and constraints relaxed. Thus, this analysis promotes robust tracking based on combining different algorithmic methods.

After discussing the navigation environments, the overall workflow of the Context Adaptive VI SLAM is explained, outlining the way in which the two proposed tracking methods, EKF VI Tracking and Visual KLT-ORB, interface with the rest of the SLAM processes. As a reminder, this work relies on the ORB SLAM and is only focused on the first tracking thread, where PoIs tracking and pose estimation are performed. Whereas, the ORB SLAM Keyframe selection, as well as its last two threads, Local Mapping and Loop Closing, are used without any modification. So, subsequently, the main Context Adaptive VI Tracking components are explained. First, the execution control module is presented. This module is used to analyze inertial data, image quality, PoI number, and tracking feedback, between the current and previous frames, in order to identify the motion type (rotation, translation) and the visual data quality. It also aims to choose the most adapted tracking method for the current navigation environment. In this work the choice is made between the visual approach based on the optical flow via the KLT algorithm, and the visual-inertial approach based on the data fusion using the EKF algorithm.

Once the navigation environment is analyzed, the two tracking methods used are presented. Thus, the implemented visual tracking is described. This method allows faster real-time tracking, especially in an easy navigation context (low motion and textured scenes), which is beneficial for mobile embedded systems. In fact, in most works, a visual tracking in SLAM system is performed by detecting PoIs and describing them for each frame. This approach requires a significant execution time and computation cost. For instance in ORB SLAM, computing the detected PoIs description takes at least $10ms$. This explains the benefits of using our proposed optical flow-based VO. It allows to quickly calculate the relative pose between the current frame and the previous frame, thanks to the decrease in the frequency of PoI detection/description processing. Especially, in the case of relatively low translation and rotation motions, and also if mapping and keyframe creation are not required, there is no need to re-detect a high number of PoIs so the processing runtime is reduced. Thus, the visual KLT-ORB tracking reduces the complexity of pose computation and localization for SLAM. In addition, since KLT algorithm employs 2D points, in our proposal we go through the triangulation process using 2D-3D reprojection in order to provide the map, update it and maintain its coherence across frames and from one approach to another. Simultaneously to this process, the pose estimation task performed herein is characterized by the implementation of the motion model with respect to the previous image calculation, only. This is an advantageous method, in term of computation complexity, compared to other visual methods involving the epipolar geometry computation.

Nevertheless, when navigation environment is difficult, the tracking is performed by fusing inertial and visual data using EKF algorithm. This accurate and efficient tracking technique is explained. In our proposal, the state vector used in the EKF VI system contains, besides position, rotation, speed and bias, landmarks that help to improve the accuracy of the pose estimation. In order to reduce these landmarks accumulation and the system computational complexity, we limit the number of landmarks integrated in the state vector to only 5 ones, in total. In addition, these landmarks are updated every frame processed (removal of lost landmarks and integration of the new selected landmarks). Despite of the expensive overhead of computation as compared to pure visual method, it is possible to minimize this time and take advantage of it for a robust SLAM implementation on embedded mobile systems. This is achieved by alternating between the two methods, thanks to the execution control module. The latter aims at analyzing the navigation environment and, depending

on the motion type and scene conditions, the suitable tracking approach is selected and executed.

In order to assess the performances and efficiency of the proposed solution, various tests are carried out, within defined experimental settings and using dedicated evaluation methods and metrics. They are presented and discussed in the next chapter (chapter 5).

Chapter 5

Experiments and Results

This last chapter presents the evaluation of the individual components and the global proposal. For this purpose, various experiments and different comparisons, with relevant state-of-the-art works, were performed. These evaluations were done by focusing on two main points. The first point deals with the quantification of the quality and robustness. The second point concerns the profiling and the execution time of the main function of the proposed system. The current chapter is organized as follows: the experimental environment as well as the result analysis and appraisal methodology are presented in section 5.1. Then, the presentation of the trajectory evaluation methodology is addressed in section 5.2. Subsequently, the systems evaluation is addressed starting by ORB SLAM odometry [153]. The latter is analyzed and appraised, in section 5.3.1. After this, the KLT-ORB tracking assessment is performed in section 5.3.2. This evaluation allows the identification of the most problematic cases for vision-only tracking. In particular, the goal is to define and refine thresholds trigger to choose the relevant tracking method. Following the same process, section 5.3.3 presents and discusses the quality and profiling evaluation results of the EKF VI tracking. Lastly, in section 5.3.4, the Context Adaptive VI Tracking, including execution control module, is evaluated, then compared with other relevant state-of-the-art works in terms of accuracy, robustness and runtime performance.

5.1 Experimental & Evaluation Environment

This section introduces the environment in which the experiments are performed. The evaluation methodology is described, followed by the the system's implementation details. At the end of this section, the required dataset to be used is specified. In order to evaluate this thesis proposal, first, ORB SLAM method [153] evaluation is presented. Then, our proposed solution is assessed by focusing, separately, on the two proposed approaches: KLT-ORB and EKF VI tracking. Afterwards, the overall system is evaluated, integrating all its components: KLT-ORB tracking, EKF VI tracking, and the execution control module. Both, the ORB SLAM and the proposed solution evaluations are conducted according to the implementation protocol detailed below and using the dataset sequences described later.

The proposed system is implemented in a PC environment (figure 5.1), using OpenCV3 library [23] mainly for image processing, Eigen3 library for linear algebra (matrices, vectors, numerical solvers, *etc*.) and the Robotic Operating System (ROS) indigo [179], as all development was done on Ubuntu 14.04 Operating System (OS). Table 5.1 provides the hardware and software specification.

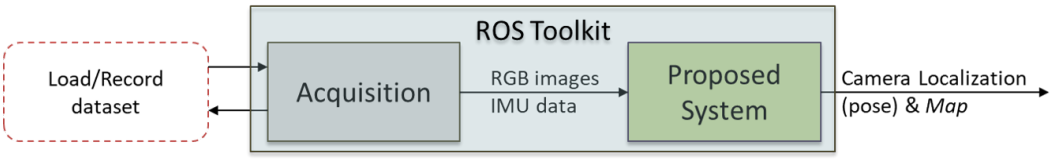

FIGURE 5.1: Experimental environment structural architecture

Hadware specification		Software Specification	
CPU	intel core i7	OpenCV version	3.0 version
RAM	8GB	Eigen version	3.1 version
OS	Ubuntu 14.04	ROS version	ROS indigo

TABLE 5.1: Experimental Environment

The experiments were performed using various sequences of KITTI's [139] and EuRoC's [24] datasets (subsection 4.1 of chapter 4). The KITTI dataset only provides visual data (images), which is characterized by simple and non-abrupt linear motions over a longer distance (mainly important 2D motions with less vibration). Indeed, this visual data is about car navigation motions in an outdoor context, with different dynamic elements such as vehicles, cyclists and pedestrians, as well as different light and shadow conditions. The use of the KITTI dataset is primarily intended to analyze the initial behaviour of visual SLAM, and reach a conclusion regarding the main problems encountered during the different experiments, including the scale estimation and the non-determinism issues.

EuRoC dataset provides inertial data (IMU measurements), in addition to visual data (images of size : [752 x 480]). This is especially useful for evaluating visual-inertial and visual tracking, and SLAM systems in the same conditions. Moreover, EuRoC dataset is widely used in literature for benchmarking different SLAM systems. For these reasons, the EuRoC dataset is the focus of the presented experiments.

5.2 Trajectory Evaluation

The accuracy of tracking methods is quantified by evaluating the estimate trajectory with respect to the ground truth. However, this is not straightforward and easy task. In fact, the estimated trajectory and the ground truth are often expressed in different reference frames, therefore they cannot be compared directly. Subsequently, a pre-process of trajectory alignment must be carried out. Furthermore, a trajectory contains different poses values (position and rotation) at different times making it a large collection of data. Consequently, the way to concisely summarize whole trajectory information into precise and accurate metrics is not trivial. There are many processes used to evaluate the estimated trajectories. In this section, the commonly trajectory evaluation techniques and error metrics are discussed. In order to be able to address this issue, non-determinism of the system and scale estimation problem are first discussed.

5.2.1 Non-Determinism

The non-determinism issue relates to the behaviors of the multi-threading system implementation and the heavy use of RANSAC [154][111]. As an example, figure 5.2 illustrates the non-deterministic nature of the ORB SLAM system on each trajectory axis (x, y, z), the discontinuity of some trajectories, such as in run 6 (green trajectory), reflects poses that were not computed during these runs for these frames. Based on ORB SLAM analysis, the non-determinism can generate an additional error margin estimated at $\simeq 1m$ (expressed in RMSE). To overcome this problem, several works [19][44][111][154] point out the need to run the non-deterministic system five to ten times, and to calculate its mean, maximum and minimum error. In our evaluations the system is running ten times.

5.2.2 Scale Estimation

The KLT-based VO trajectories, generated using the KITTI dataset, are plotted to illustrate the scaling effect. Figures 5.3 and 5.4 demonstrate the scale estimation impact (graph errors and overlaps) on the magnitude of the estimated poses.

The issue with scale estimation remains a problematic subject that is not well discussed in literature, in particular the methods used in visual-inertial based systems such as R-VIO [93], VI Mono-SLAM [175] and Mono Visual EKF SLAM [185]. Each proposed scale estimation method is adapted to

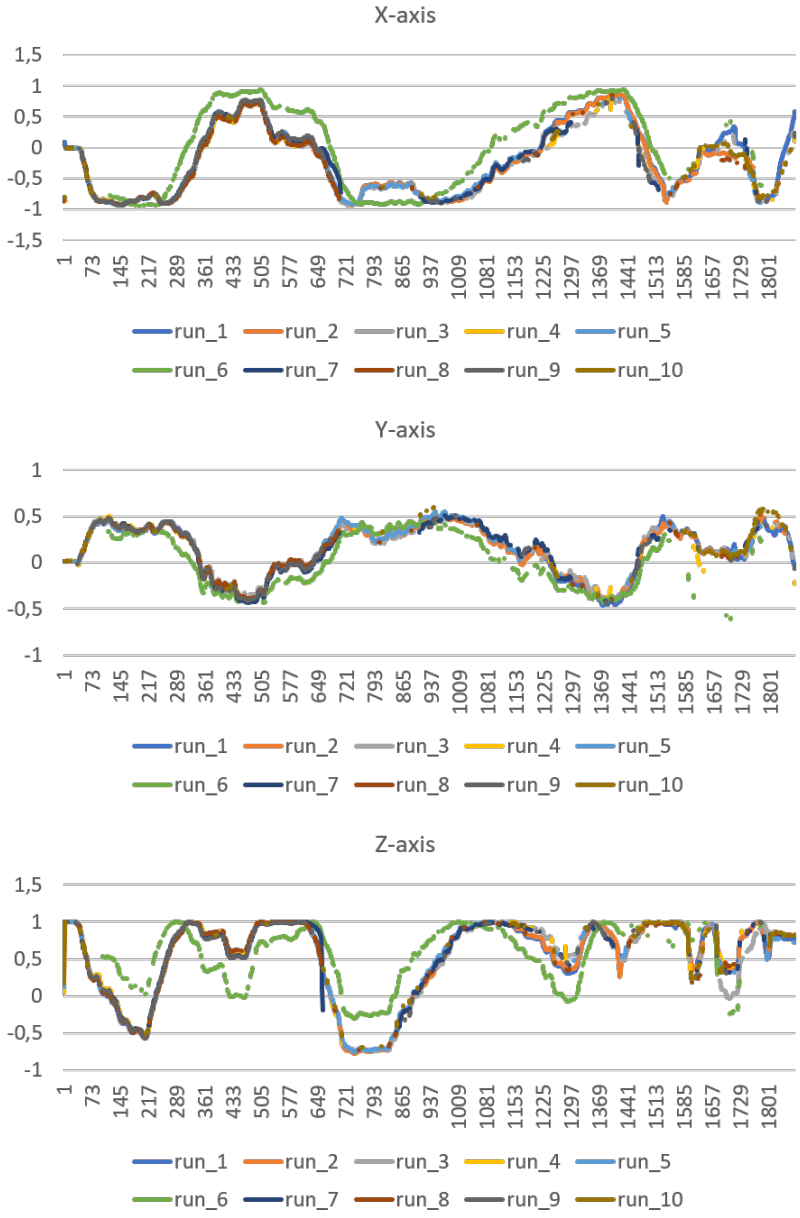

FIGURE 5.2: The comparison of numerous runs of ORB SLAM (*m*) (horizontal axis shows the IDs of the processed frames)

the intended method and its configuration; therefore, it cannot be considered as a universal solution. These methods can be performed using, for example, random constants, or rate-based estimates between poses from each sensor independently [198] (for loosely coupled multi-sensor systems based mainly on the Kalman filter). For previously listed solutions, trajectories (by axis) are estimated to find the most relevant method. A specific interest is given to the EKF VI system.

While tracing the first experiments' graphs, taking into account the complete dataset sequence covering a distance of 58 *meters* in 144 *seconds*, suspicious behaviors regarding the evolution of the graphs' amplitudes were observed, as well as a drift at the end of some of trajectories. After making an axis by axis comparison (figure 5.5 on left) between the ground truth and the estimated EKF VI system trajectory, it can clearly be seen that the trajectory suffers from a scale estimation problem. The tracks obtained, based on the distance computation between two successive frames are depicted

FIGURE 5.3: KLT-based Visual Odometry (with and without scale) *vs.* KITTI 02 ground truth trajectories (*m*)

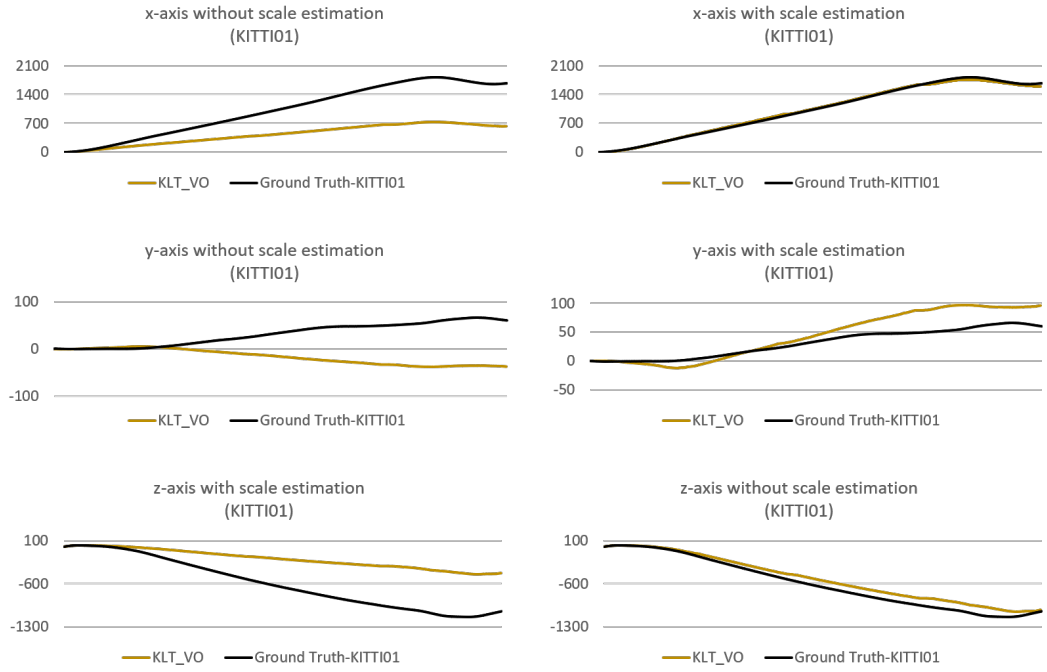

FIGURE 5.4: KLT-based Visual Odometry (with and without scale) *vs.* KITTI 01 ground truth trajectories (*m*)

on figure 5.5 (on the right). The results provided by this method are inconsistent, noisy and imprecise compared to the ground truth.

The scale estimation, based on the distance computation between two inertial poses, is also tested using the inertial data obtained from the IMU preintegration measurements. As shown in figure 5.6,

FIGURE 5.5: On the left: EKF VI system without any scale estimation *vs.* EuRoC dataset (V101) ground truth; On the right: EKF VI system with scale estimation, based on the distance between two images, *vs.* EuRoC (V101 & MH01) dataset ground truth (*m*) (horizontal axis shows the IDs of the processed frames)

the results from these experiments are rather improved from those obtained in the previous ones illustrated in figure 5.5; however, there are still discrepancies and inconsistencies (especially at mid-stream).

The approach proposed mainly for loosely coupled systems [198], based on the rate computed between the visual pose coming from the camera and the inertial pose coming from the IMU, is also assessed. However, for a tightly coupled systems computing a final fused (visual-inertial) pose, this scale estimation is obtained by considering the rate between the pose derived from the EKF measurement step, which is the inertial pose obtained using IMU measurements, and the EKF final pose which is the visual-inertial fused pose (section 4.3.3 of chapter 4). However, since this method is based on estimated pose, it is influenced by its variations. Therefore, a thresholds are used in order to filter out the scale to avoid the highest number of errors as explained in [198], see figure 5.7. As presented, the obtained trajectory has approximately the same behavior as the trajectory estimated using the scale estimation based on the pre-integrated IMU measurements, but provides a higher correspondence to the ground truth, in terms of the three trajectory axis (x, y, z).

In our solution, considering the visual results of the above tests, the last cited scaling estimation method is applied, inspired from [198] and based on the ratio between inertial and fused poses. This method gives meaningful trajectories, although it remains insufficiently accurate to evaluate the system and calculate absolute and relative errors (ATE and RPE presented, see subsection 5.2.4). Actually, other literature works often use ground truth for trajectory scaling. Thus, they produce better

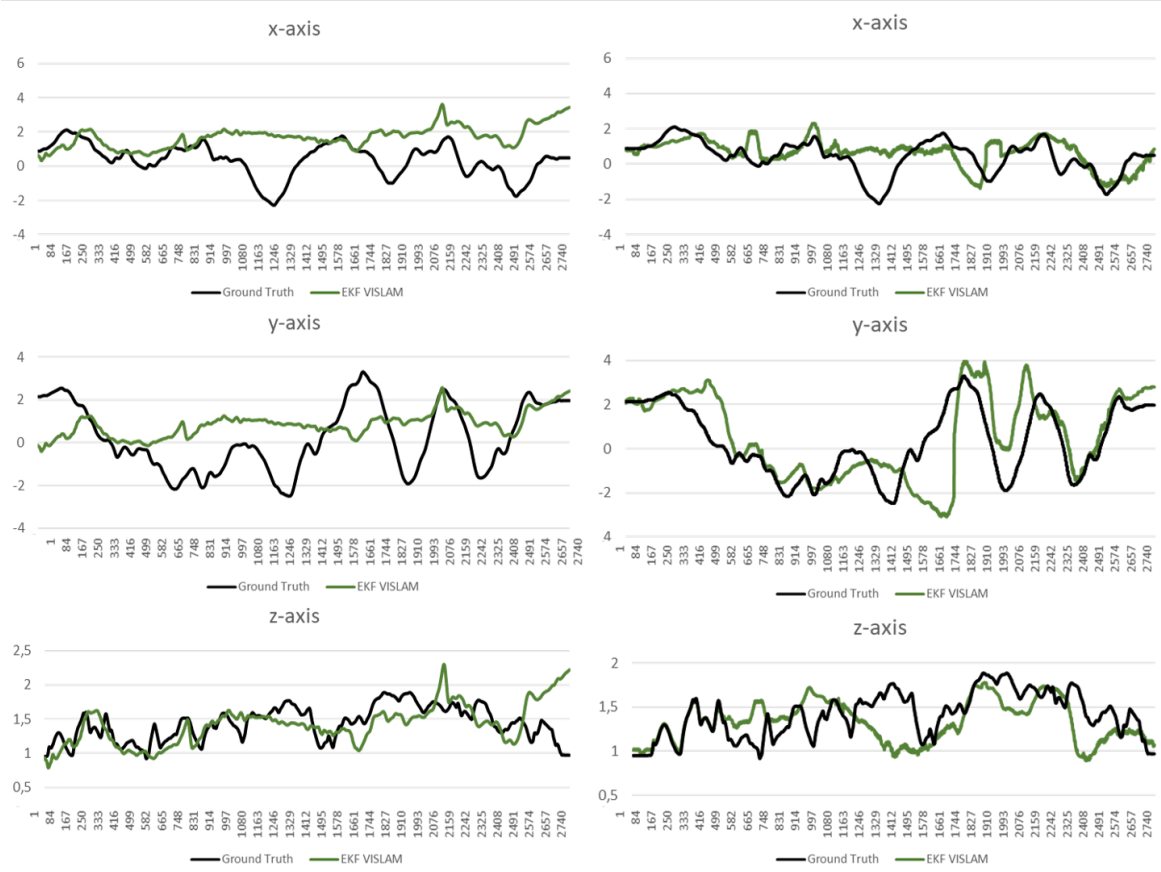

FIGURE 5.6: On the left: EKF VI system without any scale estimation *vs.* EuRoC dataset (V101) ground truth; On the right: EKF VI system with scale estimation, based on the IMU pose computed between two images, *vs.* EuRoC dataset (V101) ground truth (m) (horizontal axis shows the IDs of the processed frames)

dynamic systems that avoid the incidence of scale estimation errors on the results accuracy. Therefore, using only this last scaling estimation processing, comparing our solution with other works would not be so meaningful. Thus an additional pre-processing of the result trajectories is provided in order to evaluate our system, so called trajectory alignment is explained in next section.

5.2.3 Trajectory Alignment

With the appearance of numerous algorithms for VO/ VIO/ VSLAM/ VI SLAM, these last years, the use of a common benchmarking pipeline and metrics becomes crucial to fairly compare and evaluate those different approaches. In fact, the quantitative assessment and comparison of an estimated trajectory $\hat{P} = \{\hat{p}_i\}_{i=0}^{N-1}$ with the ground truth $P_{GT} = \{p_{GT_i}\}_{i=0}^{N-1}$ is neither straightforward nor simple. These data contain pieces of potentially missing information and typically have different sampling rates and sizes. In addition, both estimated trajectory and ground truth should be considered in the same reference frame, which is particularly difficult when the scale factor is unknown (*e.g.* VO). These tasks require trajectory pre-processing so that it may be expressed according to the same ground truth reference frame [230]. Different techniques have been proposed in literature. However, their objective is generally the same. They essentially consist to determine the transformation $f(.)$, with respect to the ground truth reference frame, in order to find an *equivalent* aligned

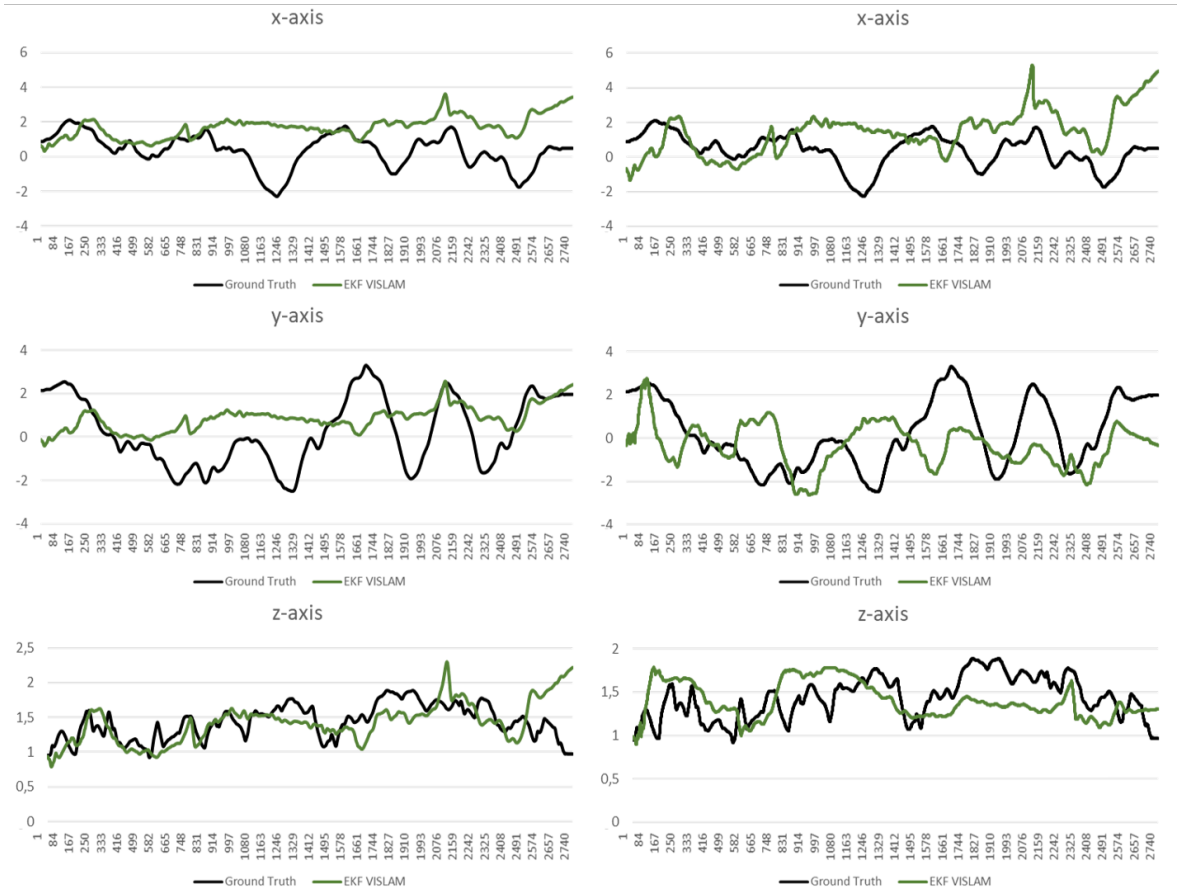

FIGURE 5.7: On the left: EKF VI system without any scale estimation *vs.* EuRoC dataset (V101) ground truth; on the right: EKF VI system with scale estimation, based on the rate between inertial and fused poses, *vs.* EuRoC dataset (V101) ground truth (m)

trajectory $\hat{P}' = f(\hat{P})$, and compute the evaluation errors. This pre-processing is known as "*Trajectory Alignment*".

The most-used technique [230] is available as an open-source toolbox [181] and provides a tutorial to quantitatively evaluate the quality of an estimated trajectory. According to this work, there are different trajectory alignment types. They are based, mainly, on the similarity, rigid body, or yaw-only rigid body transformations, that are suitable to monocular, stereo and VI systems, respectively. Yaw-only rigid body transformation is the common used method for VI systems. It consists of one DoF rotation around the gravity axis z plus three DoFs translation that represents the four unobservable DoFs for VI systems [114]. Indeed, as explained in [230], in a visual-inertial system, the pre-integrated inertial measurements rely on the rotation around only one axis R_z, namely the gravitational axis z, thus the rotation is expressed using only single parameter θ:

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

Practically, there are two possibilities to provide the trajectory alignment. This process is either performed taking into account all estimated states, or even single state [55]. In the first case, the process tends to provide a lower amount of error by seeking for a transformation using all the estimates and

ground truth measurements. In the second case, the computations give an error estimation that increases over time. This technique aims to compute the rigid body transformation or the yaw-only transformation, for example, using only a single state, which can be the first state. Moreover, both techniques assumed that the trajectory evaluation can be based only on the translational components "*trans*" (position (x, y, z)) of the estimated trajectory and the ground truth poses [186][234].

Various relevant VO/ VIO/ VSLAM/ VI SLAM state-of-the-art works [93][231][55][177] apply the [230] tutorial to align and assess their results. In order to be comparable with these systems, we also carried out our evaluations relying on the [230][234] pipeline. This is performed by focusing on the use of all the states to perform the yaw-only rigid body, applying a fixed scale s (presented later in equation 5.2) at 1, for visual-inertial systems and similarity alignment, estimating a scale according to the *Umeyama* method [212], for monocular visual systems, in addition to computing errors on the basis of the translation component considering all the three axis (x, y, z). This approach requires two main steps: time-alignment and geometrical-alignment.

Time-Alignment Because the ground truth and estimated trajectories do not usually have the same timestamps, these trajectories are aligned temporally in order to ensure a correct temporal association. There is currently no method that addresses the temporal faulty association. To find the ground truth that corresponds to the estimated pose at time t, most tools take a naive matching strategy and simply use the closest ground truth [181][213]. Thanks to this step, the used data will be temporally consistent for the remaining steps.

Geometrical-Alignment After the time-alignment, the geometrical-alignment is performed as in [230] and is available as an open source tool in [181]. This processing is carried out joining the estimated positions $\{\hat{p}_i\}_{i=0}^{N-1}$ and the ground truth $\{p_{GT_i}\}_{i=0}^{N-1}$, considering that all states have the same uncertainty since the covariance is not determined. The geometrical-alignment aims to compute the transformation $S' = \{s', R', t'\}$ that satisfies the differentiation below:

$$S' = argmin_{S=\{s,R,t\}} \sum_{s=0}^{N-1} \left\| p_{GT_i} - sR\hat{p}_i - t \right\| \tag{5.2}$$

where s is a scalar ($s = 1$ if the scale is known, like in stereo and inertial setup), $R \in SO(3)$ and $t \in \mathbb{R}^3$ are the system rotation matrix and translation, respectively.

The equation 5.2 is solved using the *Umeyama* approach [212] discussed in [230] and based on the SVD method to obtain s', R' and $t' = mean_N(p_{GT}) - s'R'mean_N(\hat{p})$. Subsequently, the aligned estimated trajectory (position $\{\hat{p}_i\}_{i=0}^{N-1}$ and rotation $\{\hat{R}_i\}_{i=0}^{N-1}$ components) is calculated according to the computed transformation elements $S' = \{s', R', t'\}$:

$$\hat{p}'_i = s'R'\hat{p}_i + t', \quad \hat{R}'_i = R'\hat{R}_i, \tag{5.3}$$

Furthermore, in case where the yaw-only rigid body transformation is used, the *Umeyama* rotation computation needs to be adapted. Thus, according to equation 5.2 R' is expressed as:

$$R' = argmin_{R \in SO(3)} \left\| P - R\hat{P} \right\|_F^2 \tag{5.4}$$

where:

$P = [r_0, r_1, ..., r_{N-1}]$, $r_i = p_{GT_i} - \mu_P$ and $\mu_P = \frac{1}{N}\sum_{i=0}^{N-1} p_{GT_i} = mean_N(p_{GT})$

$\hat{P} = [\hat{r}_0, \hat{r}_1, ..., \hat{r}_{N-1}]$, $\hat{r}_i = \hat{p}_i - \mu_{\hat{p}}$ and $\mu_{\hat{p}} = \frac{1}{N}\sum_{i=0}^{N-1}\hat{p}_i = mean_N(\hat{p})$

$\left\|.\right\|_F$ is the Frobenius norm.
The Frobinus expression computed in equation 5.4 can be written also as follows:

$$\left\|P - R\hat{P}\right\|_F^2 = trace(PP^T + \hat{P}\hat{P}^T - 2R\hat{P}P^T) \tag{5.5}$$

therefore, equation 5.4 is similar to:

$$R' = argmax_{R\in SO(3)} trace(R\hat{P}P^T) \tag{5.6}$$

Hence, if at this step we are only interested in the yaw-only rigid body transformation, then we will only need to identify one parameter θ' that allow the aligned rotation R'_z computation (equation 5.1):

$$\theta' = argmax_\theta (p_{12} - p_{21})sin\theta + (p_{11} - p_{22})cos\theta \tag{5.7}$$

where p_{ij} is the coefficient of $\hat{P}P^T$.

Actually, such an approach is inherently problematic. In fact, as stated in [234], the aligned trajectory $\{\hat{p}'_i\}_{i=0}^{N-1}$ is computed using the estimate $\{\hat{p}_i\}_{i=0}^{N-1}$ and the ground truth $\{p_{GT}\}_{i=0}^{N-1}$, and then used once again when evaluating the estimates. Therefore, the Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) metrics (defined in section 5.2.4) are affected. Despite these inconsistency, this alignment technique is commonly used to evaluate several prominent studies in the literature. Likewise, in this thesis we were led to respect this consensus in order to evaluate our solution and to compare it with other literature approaches [93][231][177].

5.2.4 Evaluation Metrics

The accuracy evaluation is achieved considering the most common error metrics employed for VO/ VIO/ VSLAM/ VI SLAM and positioning systems appraisal: ATE and RPE [230][228][201]. Let $\{\hat{p}'_i\}_{i=0}^{N-1} \in SE(3)$ is a sequence of poses from the aligned estimated trajectory and $\{p_{GT}\}_{i=0}^{N-1} \in SE(3)$ is a sequence of the ground truth trajectory. These sequences are assumed to be time-synchronized, equally sampled, and equally long N.

ATE: computes the distance between the estimated aligned poses $\hat{p}'_i$ of the camera and its ground truth p_{GT}, expressed in *meters* (figure 5.8). This metric is more suitable for SLAM evaluation since it allows an assessment of the overall consistency of the estimated trajectory with respect to the ground truth. Here, the ATE is computed focusing on the translational component *trans* (*i.e.* the 3D position considering all axis (x, y, z)) and expressed as a *mean*, *median* and *RMSE*:

$$\Delta\, trans(\hat{p}'_i) = trans(p_{GT_i}) - \Delta R_i(trans(\hat{p}'_i)^T) \tag{5.8}$$

where $\Delta R_i = R_{GT_i}(\hat{R}'_i)^T$ is the rotational error between the aligned trajectory and the ground truth, R_{GT_i} and $trans(p_{GT_i})$ are the rotation matrix and the 3D position of a given ground truth pose i, respectively, $\hat{R}'_i$ and $trans(\hat{p}'_i)$ are the rotation matrix and the translational component of the estimate trajectory i, respectively.

The Root Mean Square Error (RMSE) is computed by averaging all possible time intervals of the translational component *"trans"*, equation 5.8:

$$ATE_{pos}\ (RMSE)\ =\ (\ \frac{1}{N}\sum_{i=1}^{N}\ \|\ \Delta\ trans(\hat{p}'_i)\ \|^2\ )^{1/2} \tag{5.9}$$

Additionally, ATE can be expressed in other forms such as the mean and the median values:

$$ATE_{pos}\ (mean)\ =\ \frac{1}{N}\sum_{i=1}^{N}\ \|\ \Delta\ trans(\hat{p}'_i)\ \| \tag{5.10}$$

$$ATE_{pos}\ (median)\ =\ median(\ \sum_{i=1}^{N}\ \|\ \Delta\ trans(\hat{p}'_i)\ \|\ ), i \in (1, N) \tag{5.11}$$

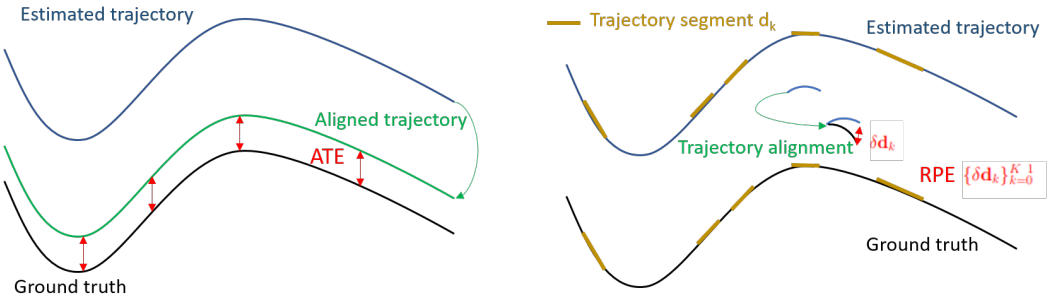

FIGURE 5.8: Illustrations of ATE and RPE (inspired from [230])

RPE: measures the difference between the estimated motion and the actual ground truth motion. As depicted on figure 5.8, this metric is used to evaluate the local accuracy of the estimated trajectory over a fixed time interval Δt with a limited number K of poses pair $\{d_k\}_{k=0}^{K-1}$, $(d_k = \{\hat{p}_j, \hat{p}_l\}, l > j)$. Considering the translational component (position (x, y, z)), the RPE is defined as follows:

$$\delta trans(p_k)\ =\ \left\|trans(p_l) - \delta R_k trans(\hat{p}'_l)\right\|_2 \tag{5.12}$$

with δR_k is the rotational error between the aligned trajectory and the ground truth.

For all the pairs of state, which represent the sub-trajectories, the 3D pose RPE is :

$$RPE_{pos}\ =\ trans(\{\delta p_k\}_{k=0}^{K-1}) \tag{5.13}$$

RPE metric provides a better representation of the trajectory drift over time, which is useful for the VO/ VIO systems evaluation.

5.3 Quality Evaluation & Runtime Performance

The results of the ORB SLAM tracking assessment are the first to be presented in this section, as ORB SLAM is the baseline for this work. Next, the quality and runtime performance evaluation results of the main parts of the proposed Context Adaptive VI Tracking, KLT-ORB Tracking and EKF VI Tracking are discussed, then our proposed Adaptive Tracking is evaluated. Following this, a

comparative analysis of the thesis's solution versus other works is provided. All evaluations relating to this work are described and analyzed according to the workflow explained previously in sections 5.1 and 5.2. The quality evaluation is based on the RPE and ATE metrics measurements, and the runtime evaluation is performed thank to a profiling step that enable to quantify required memory and processinng time required for each method. Since we are interested in tracking, both metrics are strongly linked.

5.3.1 ORB SLAM Odometry

In order to evaluate our solution, the monocular ORB SLAM odometry, and in particular the tracking step, must be characterized. To begin this process, the qualitative evaluation is presented and the different error metrics introduced previously are analyzed. After this, the ORB SLAM performance in terms of runtime analysis of the main features is discussed.

Figures 5.9, 5.10 and 5.11 illustrate the top view of the aligned ORB SLAM's odometry trajectory according to the (x, y) plane, played on different EuRoC dataset levels (one run on each dataset sequence (V201, MH03 and V203)) and compared to the ground truth [181].

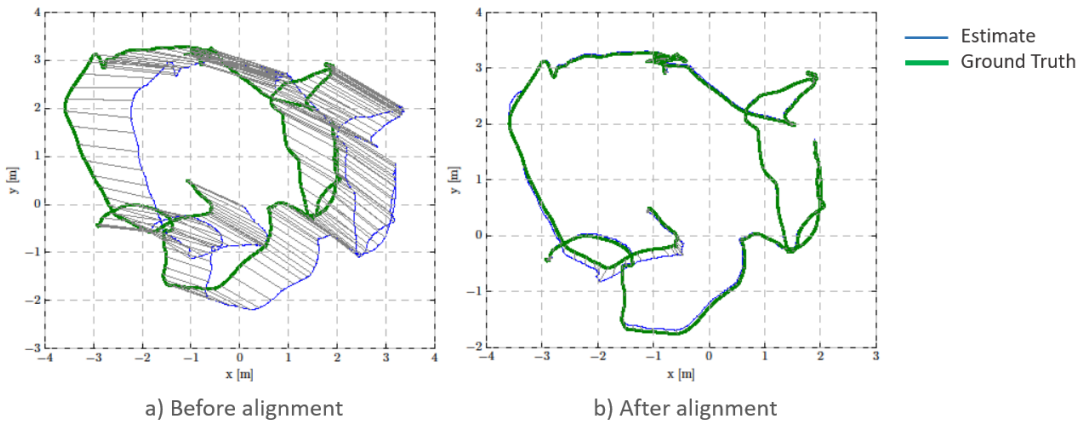

FIGURE 5.9: ORB SLAM (Odometry) aligned trajectory *vs.* EuRoC dataset ground truth (V201) (*m*) (gray lines represent the states correspondences with a step of ten poses)

ORB SLAM succeeds in ensuring navigation in both easy (*e.g.* V201) and difficult (*e.g.* MH03, V203) environments. In the easy navigation environment (figure 5.9) ORB SLAM can perform tracking with an ATE of maximum $0.38m$, in term of RMSE, and without any troubles, since the vision conditions are satisfying to its operating mode. By contrast, in difficult navigation environments, tracking becomes increasingly complicated (figure 5.11) and can easily fail in many scenarios. Such problems are mainly due to dark environments, and fast and abrupt motions (especially rotations). Table 5.2 shows ATE metrics of ORB SLAM tracking on different EuRoC sequences using a mean of ten runs in order to avoid the non-determinism problem (section 5.2.1). The ATE values increase in a significant way from easy to difficult navigation environment. In fact, the accuracy of ORB SLAM odometry in difficult environments can be up to five times less accurate, in terms of RMSE, than its performance in easy environments, like in V203 environment where the accuracy is up to $1.69m$. Moreover, the RMSE value of the ATE exceeds its median value, *i.e.* the estimated trajectory encountered more outliers along the tracking. In addition, figure 5.12 shows the ORB SLAM odometry RPE evolution, for different played datasets and thus different traveled distances. Generally, ORB SLAM tracking trajectory has a small RPE value for small distances that gradually increases along

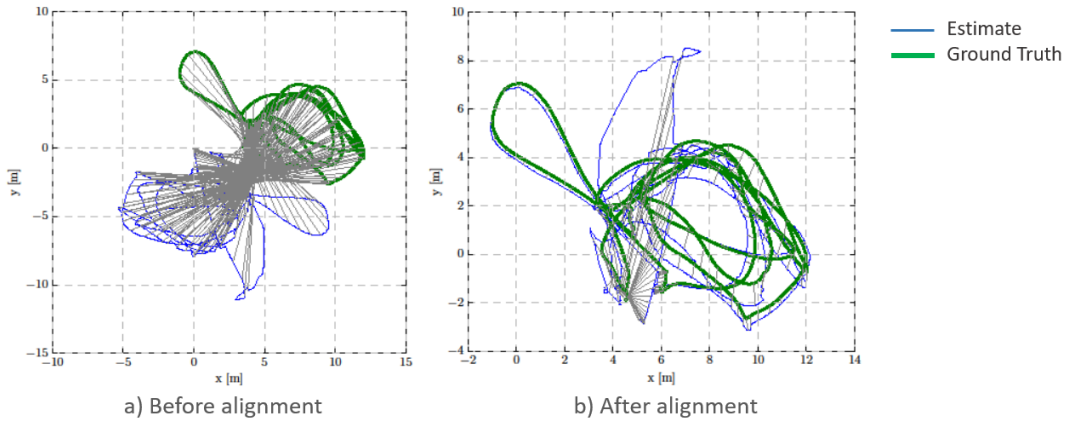

FIGURE 5.10: ORB SLAM (Odometry) aligned trajectory *vs.* EuRoC dataset ground truth (MH03) (m) (gray lines represent the states correspondences with a step of ten poses)

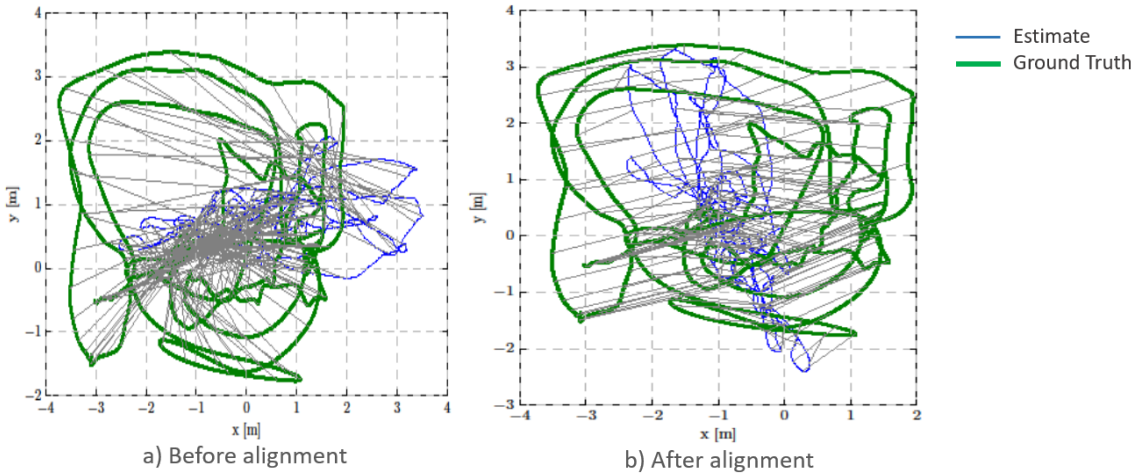

FIGURE 5.11: ORB SLAM (Odometry) aligned trajectory *vs.* EuRoC dataset ground truth (V203) (m) (gray lines represent the states correspondences with a step of ten poses)

the length of the sub-trajectories. In V201 for example, for a distance traveled of $\delta d = 3m$ the RPE is approximately $0.04m$, in term of RMSE, and for a distance traveled of $\delta d = 18m$ the RPE is up to $0.1m$. In particular, for difficult navigation environment (*e.g.* V203), the ORB SLAM trajectory RPE grows dramatically between the beginning and the end of the curve, where the travelled distance length is more important, exceeding $1m$.

Dataset	ATE		
	Mean	Median	RMSE
V101	0.88	0.28	0.38
V201	0.62	0.02	0.26
V203	1.54	1.53	1.69
MH01	0.70	0.33	0.15
MH03	0.48	0.43	0.31
MH04	1.82	0.98	1.13

TABLE 5.2: ORB SLAM (Odometry) evaluation using ATE metrics (m)

The ORB SLAM profiling, focus on the pose estimation and tracking features, in addition to the keypoint management and the total image processing. As shown in table 5.3, the execution time

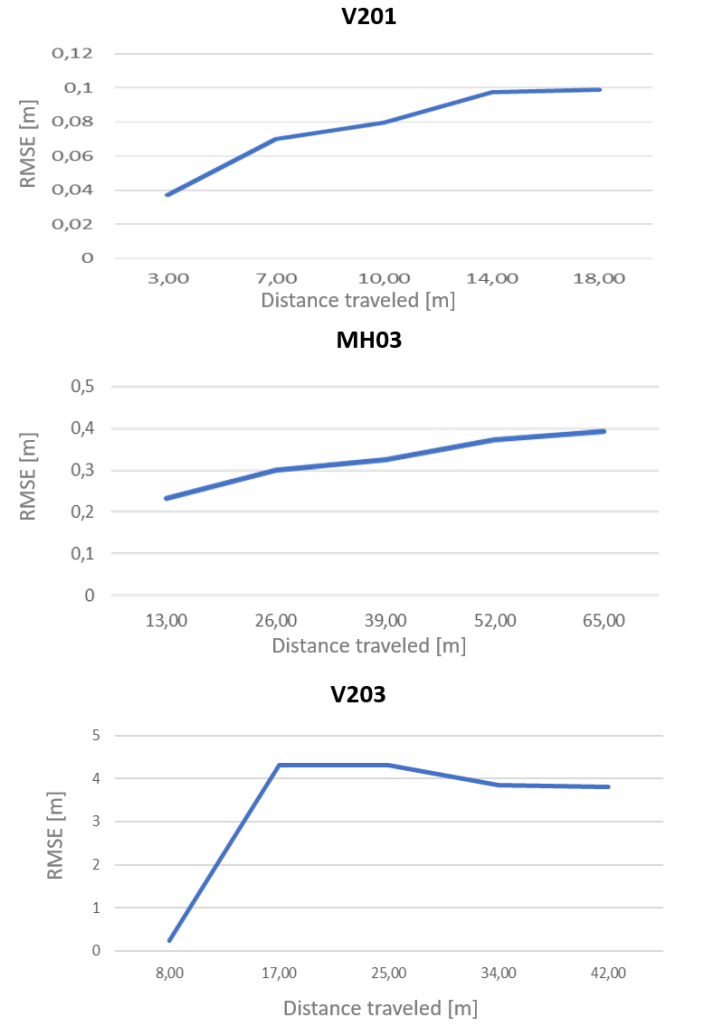

FIGURE 5.12: RPE translation of ORB SLAM (odometry), played on different EuRoC dataset levels, for different sub-trajectory lengths

varies between the different dataset sequences. This is mainly due to the different level of complexity in the navigation environment. Tracking takes longer for Vicon (Vxx) dataset, because this type of dataset shows faster and abrupt rotational motions even in good texture and lighting. For the keypoint management, the Machine Hall (MHxx) dataset is the most computationally constraining because of the dataset structure, which is characterized, by weak texture and sway motions.

Dataset sequence	V101	V201	V202	MH01	MH03	MH04	Occurrence
Keypoint Management	26	25	24	28	29	32	all sequence frames
Pose Estimation & Tracking	11	13	17	12	18	22	
Total image processing	39	41	48	44	50	55	

TABLE 5.3: The main ORB SLAM tracking features execution time ($ms/frame$)

The processing time depends on PoIs detection occurrence and the number of PoIs detected. In ORB SLAM the ORB PoIs detection is performed for all processed images, and detects approximately 1000 PoIs per image, which imposes an additional computational overhead and increases the execution time. In particular, keypoint management and tracking are the functions that are the most influenced by this additional computation. This is among the improvements made in the proposed

adaptive method.

5.3.2 KLT-ORB Tracking

The proposed KLT-ORB tracking is based on a specific keypoint management. ORB PoIs (500 PoIs), employed in tracking processing, are detected on the first frame and are reused on the next one, after being sorted. This sorting is carried out based on the PoI status after KLT algorithm. A PoI is either considered as an input and is saved or is no longer used and is deleted. This sorting continues until a predefined minimum threshold is reached (80% of the number of PoIs detected initially). After reaching this threshold, the re-detection of the PoIs, using the ORB detector, is performed in order to maintain tracking stability. However, KLT-ORB tracking cannot achieve successful long-distance tracking, therefore it does not provide continuous processing across the whole dataset sequences. In fact, when KLT-ORB tracking is executed, it can provide tracking in several (easiest) parts of the sequence, mainly at the beginning of the navigation. Whereas, for the rest of the sequence, it keeps returning the last estimated pose, or zero if it gets lost (figure 5.13). Consequently, this appraisal is based only on the successful KLT-ORB tracking parts of the trajectories, numbering four in the case depicted and highlighted in various colors on figure 5.13.

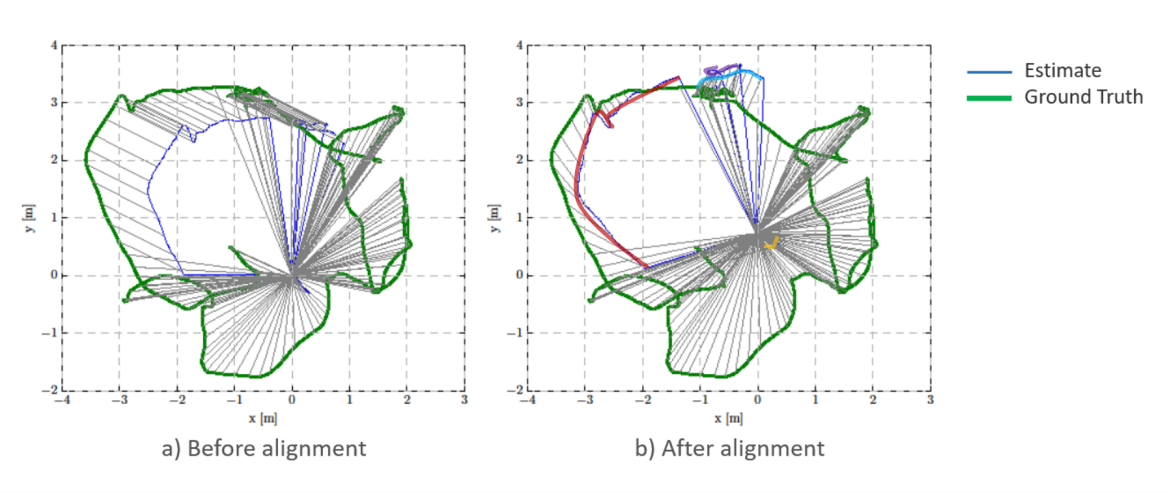

FIGURE 5.13: KLT-ORB tracking aligned trajectory *vs.* EuRoC dataset ground truth (V201) (*m*)
(gray lines represent the states correspondences with a step of ten poses)

The obtained trajectories are evaluated computing the RPE considering median, mean and RMSE errors. On curve depicted in figure 5.13, we can notice that the KLT-ORB tracking fails, starting from the last part of the dataset, this is because the instability of the motion at this part compared to the first part. In fact the motion stability at the beginnings of the dataset enables the system initialization and allows more accurate and more robust tracking. However, for the rest of the dataset, the visual KLT-ORB tracking faces more difficulty for initialization, consequently, is not able to ensure a precise and successful tracking for longer distances. The visual KLT-ORB tracking is sensitive to the complex and various motion changes that a sequence can have, as well as to the long displacement lengths.

As a reminder the interest of implementing visual KLT-ORB tracking is to use it when navigating in easy environments (section 4.1), *i.e.* environments with homogeneous, slow and simple movements. Based on about twenty tests on different EuRoC dataset sequences (mainly easy sequences), the KLT-ORB tracking successfully manages to navigate and track the short sequence distances (before tracking is lost and relocation not fixed). This visual tracking is achieved with a motivating RPE

(RMSE), estimated at an average of $\simeq 0.2m$ for the majority of dataset sequences used (table 5.4). Hence, the robustness of this approach is estimated by $\simeq 25\%$ to $\simeq 30\%$ of the global dataset ($\simeq 800$ used frames out of the total of $\simeq 2714$ dataset frames).

Dataset	KLT-ORB tracking		
	Mean	Median	RMSE
V101	0.18	0.067	0.22
V201	0.15	0.075	0.21
V203	failed	failed	failed
MH01	0.11	0.038	0.19
MH03	0.27	0.10	0.25
MH04	failed	failed	failed

TABLE 5.4: KLT-ORB tracking evaluation using the RPE (m)

Table 5.4 represents the RPE of the tracked sections for various dataset sequences. According to these measurements, the KLT-ORB tracking behaves correctly on the travelled distances, since the KLT algorithm is adapted to the movement type performed during this period, reaching at maximum a RPE of $0.25m$, in term of RMSE, when tracking in a difficult environment such as MH03. However, the KLT-ORB technique is not robust enough to track across all sequences in the dataset, thus ATE cannot be computed.

Statistics on the processing time by the main functionalities of the proposed solution, are given for the KLT-ORB tracking profiling. Table 5.5 presents the tracking runtime results expressed in *seconds*. The KLT-ORB tracking thread works at an average frequency of $\simeq 32Hz$. This is mainly due to the use of the keypoint management and the KLT algorithm. ORB PoIs detection occurrence is reduced,

	ORB SLAM	
	Keypoint Management	Pose Estimation & Tracking
Mean Time	29	18
Occurrences	191	191
	KLT-ORB tracking	
	Keypoint Management	Pose Tracking
Mean Time	24	6
Occurrences	31	191

TABLE 5.5: KLT-ORB tracking execution time evaluation using EuRoC dataset ($ms/frame$)

which also decreases the total execution time compared to ORB SLAM.

5.3.3 EKF Visual-Inertial Tracking

In this section, EKF VI tracking is evaluated. First, the quality of the results is studied, then the profiling process is described. As for ORB SLAM odometry, this method is evaluated on different EuRoC sequences and using every time a mean curve from a tenth of runs. Figures 5.14, 5.15 and 5.16 illustrate a top view of different examples of the EKF VI tracking process results, in different dataset levels, compared to ground truth.

	ATE		
	Mean	Median	RMSE
V101	0.109	0.096	0.15
V201	0.12	0.104	0.15
V203	0.54	0.53	0.69
MH01	0.098	0.071	0.17
MH03	0.28	0.23	0.39
MH04	0.29	0.30	0.65

TABLE 5.6: EKF VI tracking evaluation using RPE and ATE (m)

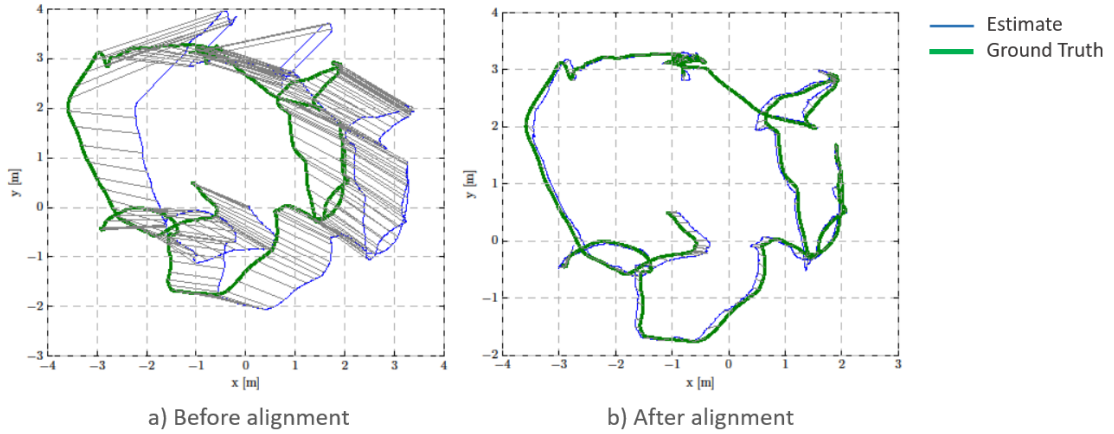

FIGURE 5.14: EKF VI tracking odometry (aligned trajectory) *vs.* EuRoC dataset (V201) ground truth (*m*) (gray lines represent the states correspondences with a step of ten poses)

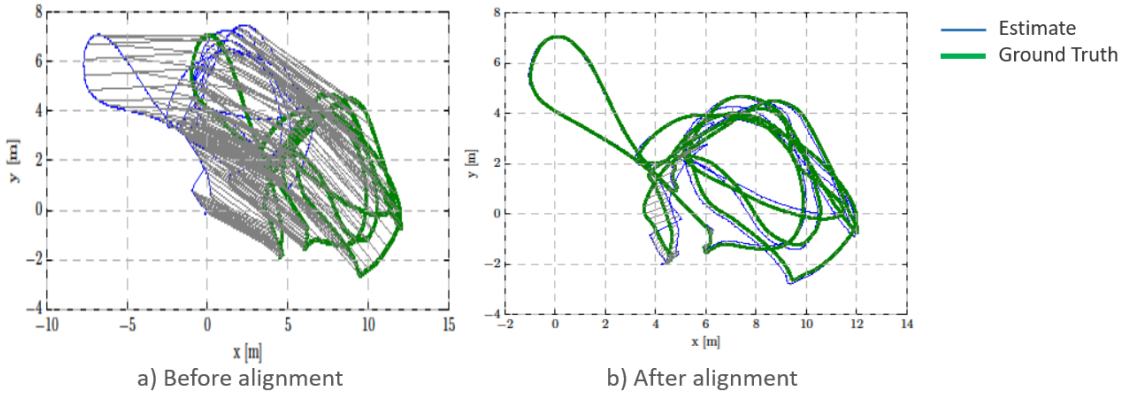

FIGURE 5.15: EKF VI tracking odometry (aligned trajectory) *vs.* EuRoC dataset (MH03) ground truth (*m*) (gray lines represent the states correspondences with a step of ten poses)

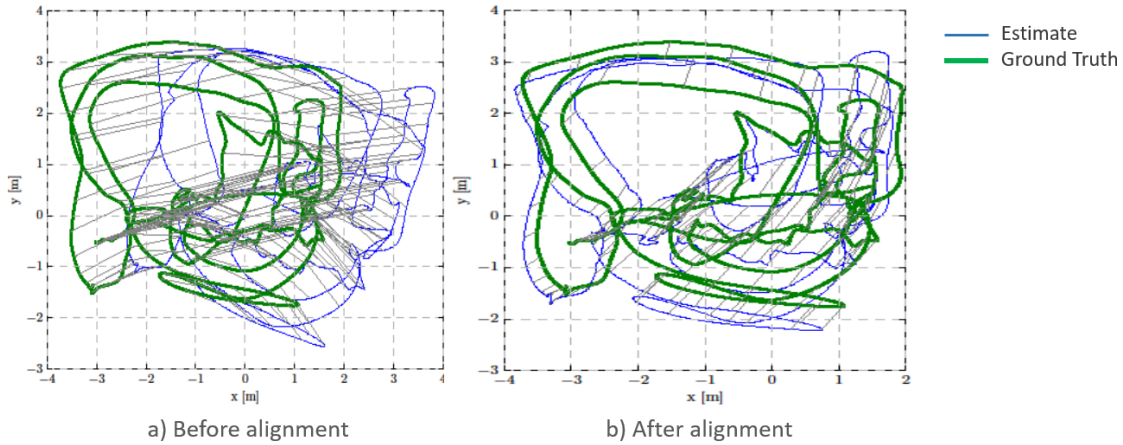

FIGURE 5.16: EKF VI tracking odometry (aligned trajectory) *vs.* EuRoC dataset (V203) ground truth (*m*) (gray lines represent the states correspondences with a step of ten poses)

As presented in table 5.6, RMSE of the EKF VI tracking ATE is in an average of $0.2m$ for easy dataset sequences and $0.6m$ for the most difficult ones (see section 4.1.2 of chapter 4) thanks to the EKF implementation method used for visual and inertial data fusion. However, EKF VI tracking may suffer from some discrepancies and an increase in the number of errors mainly in the middle and at the end of the navigation. These discrepancies are more visible over long distances ($>$ 70 *meters*, sequences in grey cells on the table 5.6), they lead to ATE estimated at a minimum of $0.39m$, in term of RMSE. However, over medium or short distances (between 30 *meters* and 70 *meters*, sequences in white cells on the table 5.6), these deviations decrease and provide an estimated error of $0.17m$ maximum. Based on the results shown in table 5.6, we can conclude that the proposed EKF VI approach has an efficient accuracy over short to medium distances. Furthermore, it also provides strong performance even during fast motions and abrupt rotations (difficult sequences).

In addition, figure 5.17 shows the behavior of the EKF VI' RPE as a function of different sub-trajectory lengths. In an easy environment, the RPE values are small and their evolution is more stable over the different distances, for example in V201 the error increases from $\simeq 0.07$ for a distance traveled of $3m$ to $0.1m$ for a distance traveled of $14m$. However, for difficult environments, these values start to increase more quickly (*e.g.* from $0.09m$ to $2m$) but are still less important when compared with those of ORB SLAM odometry.

As previously described, EKF VI tracking is made up of three basic parts: *Prediction*, in which inertial measurements are processed and pre-integrated in order to predict the system state; *Measurement*, in which frames are processed to compute the measured state and how many new PoIs need to be detected (if any) and added to the system status computation is decided; and *Update*, in which the final state and pose estimation is performed. Since our thesis system is a tight coupling, the parts involving *Measurement* and *Update* intersect at different computing levels. For profiling evaluation, these parts are thus performed as a single large function that represents about 70% of the overall EKF VI tracking computation complexity. Whereas the *Prediction* part depends mainly on the IMU and the initial configuration as well as the feedback of the previous system state. Table 5.7 shows the different execution times required for various EuRoC sequences. Moreover, the number of PoIs (min, max and mean) considered during the system computation are also given.

Dataset	Steps			Number of PoIs		
	Prediction	Measurement & Update	Global System	Mean	Max	Min
V101	2.6	25	35	9	25	6
V201	2.4	24	35	12	24	8
V202	2.5	24	36	10	25	5
MH01	3.5	27	38	8	22	5
MH03	4.5	29	40	9	24	7
MH04	4.5	29	39	9	24	5

TABLE 5.7: EKF VI tracking execution time evaluation ($ms/frame$) & PoIs number statistic

The scene characteristics and the motion behavior have a direct impact on the number of detected PoIs, this number varies between 5 and 25 PoIs at maximum. Any lost PoI requires additional processing and therefore additional complexity and computing time. Indeed, this issue leads to a selective renewal of PoIs, or even a total re-detection of new ORB PoIs in order to integrate them instead of the lost ones. As a consequence, a computational load is required to model these new points in a way that makes them suitable for use in the rest of the EKF processing. Loosing PoIs is frequent occurrence in difficult navigation environments (*e.g.* EuRoC MH03 and V203 dataset sequences, *etc.*). The additional computational load, caused by the re-detection and modelling of PoIs, is estimated to be at least $3ms$ more than the total computational load for tracking in an easy environment (*e.g.* $35ms$ in V101 and $39ms$ in MH04).

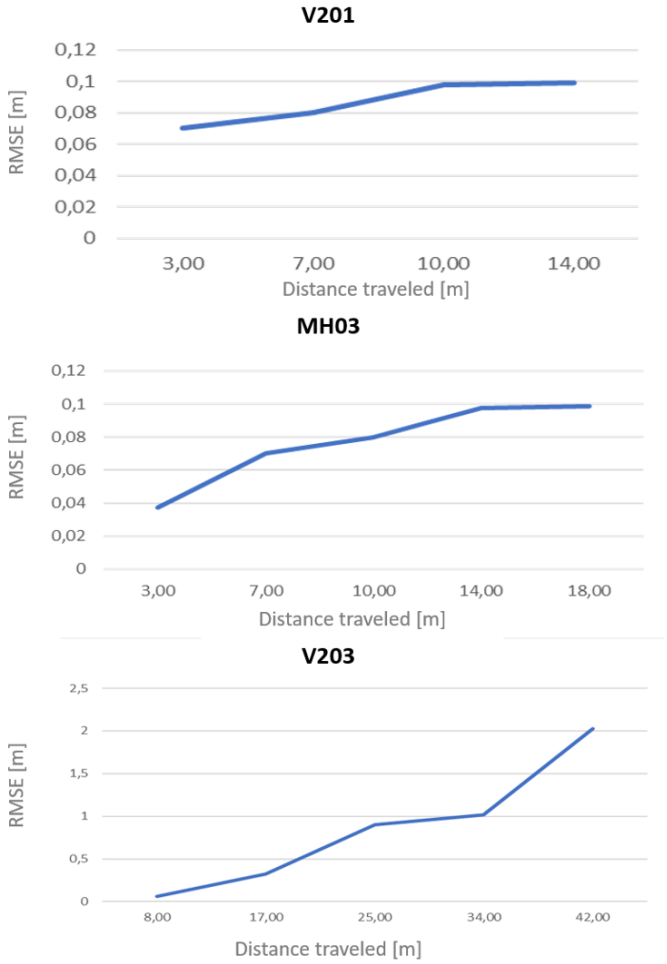

FIGURE 5.17: RPE translation of ORB SLAM (odometry), played on different EuRoC dataset levels, for different sub-trajectory lengths

5.3.4 Context Adaptive VI Tracking

In this section our overall solution named Context Adaptive VI Tracking, is evaluated. At startup, the control module checks the initial system state. Generally, if the system is in stationary mode, the tracking starts with the KLT-ORB tracking. If the system is already in motion, the EKF VI tracking method is executed. Afterwards, using the execution control module, the frame quality, the camera's FoV and the tracking quality/status for the previous image are verified.

Table 5.8 gives an overview of the effect of these control metrics on system trajectory accuracy (expressed in terms RMSE of ATE and execution time), by providing an example of various configuration cases that illustrate the benefit of using all control metrics together. Omitting one or more criteria, as illustrated in table 5.8, where FoV is disabled for example, the tracking quality is deteriorating by approximately $0.03m$ in term of RMSE of ATE. In particular, when navigating in difficult dataset sequences, tracking can failed completely. The system keeps tracking only in cases without underexposed or overexposed and dark scenes (*e.g.* V102, V201 and V202). However, by activating the additional environmental control to keep a minimum of the FoV of the previous frame (computing the rotation and translation between two frames), the proposed system provides accurate

tracking and navigation even in a difficult conditions. Furthermore, applying the trajectory consistency metric does not have significant effect on the overall system quality. It slightly enhances the trajectory accuracy in particular within an easy navigation context. Therefore, using all proposed control parameters: in motion, image quality, PoI cheking and FoV, enable to achieve an acceptable ATE in term of RMSE. It provides an average RMSE accuracy of $0.06m$ in easy dataset sequences, and ensures a robust tracking and an average RMSE accuracy of $0.29m$ in difficult sequences.

Control Metric		Metric State									threshold's range
Moving		on	-	-	-	-	-	-	on	on	v = 0 or v $\neq$ 0
Image Quality		-	on	-	-	-	-	on	on	on	> 80% black pix.
PoI check		-	-	on	-	-	on	on	on	on	90% successful PoIs tracking
FoV		-	-	-	on	on	on	-	on	on	$\theta \geq 41 or \alpha \geq 61 or \beta \geq 79$
Traj. consistency		-	-	-	-	on	on	on	-	on	[1.3 - 0.8]
Results in terms of RMSE(ATE) (cm) & Execution time (ms)											
(m)	easy	0.113	0.086	0.089	0.078	0.080	0.076	0.100	0.070	0.063	
	difficult	failure	failure	failure	0.289	0.289	0.290	0.291	0.290	0.290	
(ms)	easy	32.00	31.00	31.20	32.30	32.20	31.8	31.50	31.63	31.43	
	difficult	failure	failure	failure	36.40	36.40	36.70	35.40	35.49	35.47	

TABLE 5.8: Context Adaptive VI Tracking accuracy evaluation (ATE) in various configuration cases

Another issue, concerning the use of the control module, is the switching management between the visual and visual-inertial approaches, which can impact the overall system quality. When switching from one mode to the other, the PoIs remaining from the previous tracking mode are always sorted before tracking the current frame. This helps to ensure the consistency of the whole systems by linking the information between both tracking approaches.

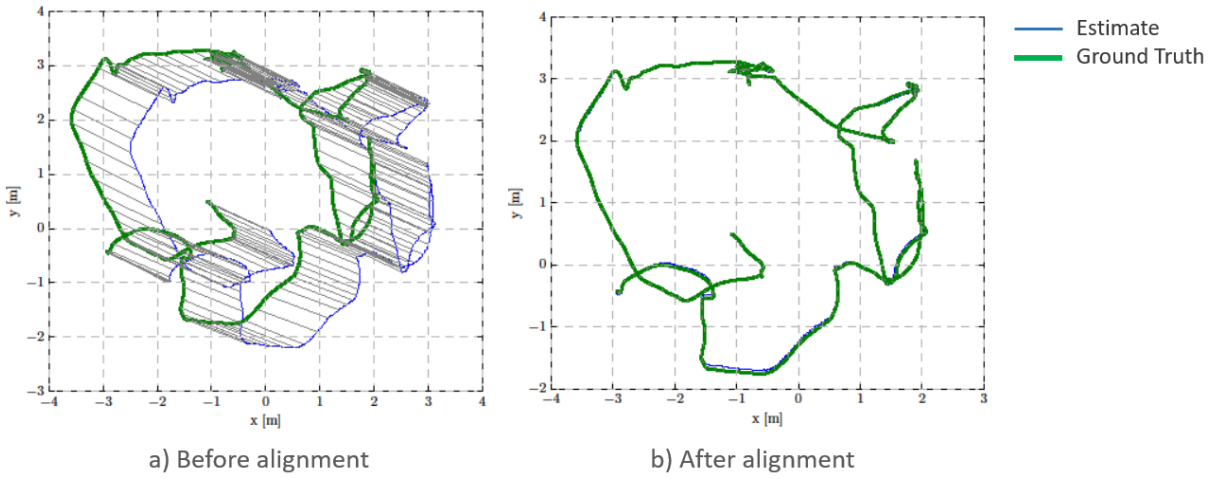

FIGURE 5.18: Context Adaptive VI Tracking (before and after alignment) *vs.* EuRoC dataset (V101) ground truth (m) (gray lines represent the states correspondences with a step of ten poses)

Figures 5.18, 5.19 and 5.20 illustrate a top view of the Context Adaptive VI tracking played on different dataset levels and compared to the ground truth. As presented on table 5.9, the required time for the switching areas, during which each system performs its initialization process in order to resume correctly the tracking, varies between approximately $0.25s$ and $2.7s$ depending on whether control module switching from KLT-ORB tracking to EKF VI tracking or the reverse. Figure 5.21 shows an example of the Context Adaptive VI tracking compared to the ground truth and highlights the switching steps. The circles drawn on this figure identify the switching areas during which each

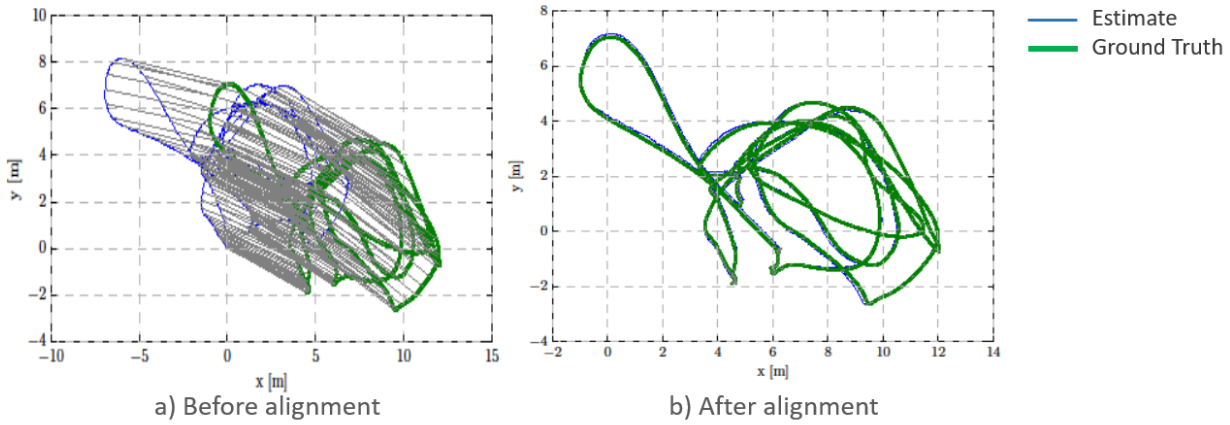

FIGURE 5.19: Context Adaptive VI Tracking (before and after alignment) *vs.* EuRoC dataset (MH03) ground truth (*m*) (gray lines represent the states correspondences with a step of ten poses)

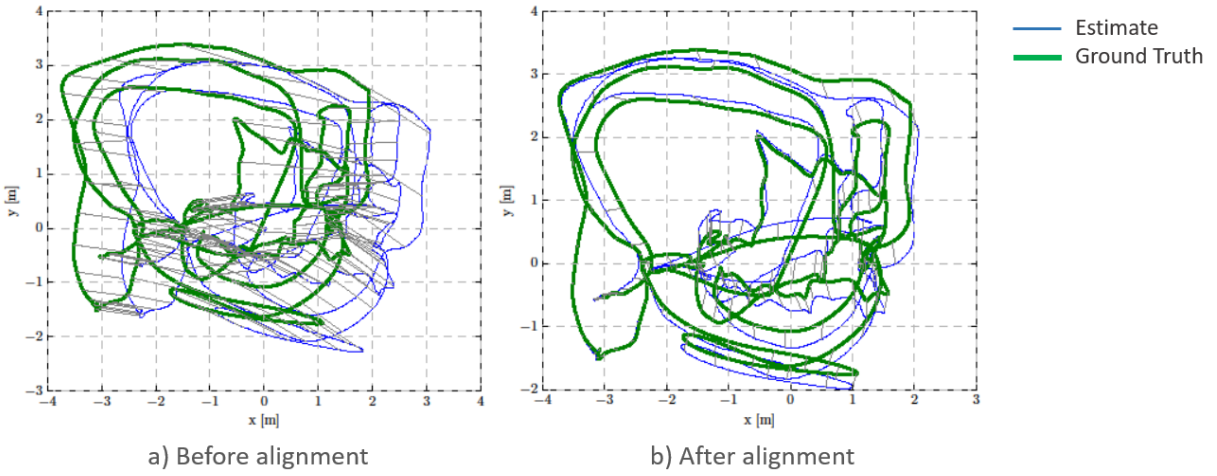

FIGURE 5.20: Context Adaptive VI Tracking (before and after alignment) *vs.* EuRoC dataset (V203) ground truth (*m*) (gray lines represent the states correspondences with a step of ten poses)

	EKF VI to KLT-ORB tracking		KLT-ORB to EKF VI tracking	
Dataset level	Easy (30 to 70 *meters*)	Difficult (>70 *meters*)	Easy (30 to 70 *meters*)	Difficult (>70 *meters*)
max. number of frames	10	110	6	26
max. execution time (s)	0.24	2.64	0.2	1.1
occurrence	≥ 5	≤ 3	≥ 5	≤ 3

TABLE 5.9: Statistics on mode switching

system performs its initialization process in order to resume correctly the tracking. Executing KLT-ORB tracking requires an initialization step that can process up to 100 frames, unlike EKF VI tracking which only needs about 6 frames to properly resume tracking. During the initialization step of both methods, the system continues to produce a pose that is not inevitably accurate, since it relies on the last state estimate and calculated by a new computation process. Switching between these two tracking methods can provide some tracking errors, observed mainly when comparing the aligned trajectory to the ground truth per axis (*i.e.* the difference between the aligned trajectory (x, y or z) and ground truth). For example, the case depicted on figure 5.21, where these errors can reach at a maximum $0.27m$ by axis, and are compensated by the accuracy of the system afterwards, particularly

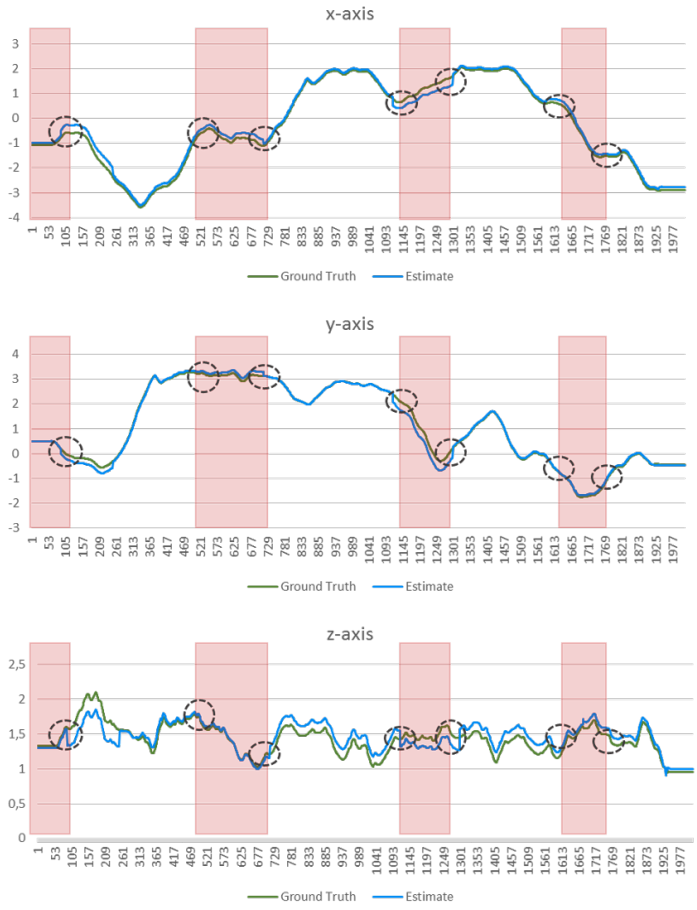

FIGURE 5.21: Context Adaptive VI Tracking *vs* EuRoC dataset ground truth (V201)(m). The circles mark the switching areas between the two modes: the KLT-ORB method and the EKF VI method. The red rectangles denote the trajectory parts that are from the KLT-ORB method. The others are the trajectory parts that are from EKF VI method (horizontal axis shows the IDs of the processed frames)

when distances travelled grow (figure 5.22). Actually, EKF VI tracking is used more than KLT-ORB tracking. This is mainly due to the complicated nature of the dataset (EuRoC) sequences, which favors the former one. The used rate of EKF VI tracking, ensures navigation continuity and therefore the robustness of the Context Adaptive system.

As we can see on figure 5.22, the RPE values increase from short distances to longer distances, this can be due also to the transition mode error. However, this error can be recovered along the travelled distance especially in easy and less difficult environments like V201, V101. For example, in MH03, the RPE value drops from 0.33m for a distance traveled of 52m to 0.27m for a longer distance traveled (65m).

After the previous discussion, we can see that switching between two relevant tracking methods does not degrade in a meaningful manner tracking accuracy, and provides comparable outcomes with those of the state-of-the-art: ORB SLAM [153], R-VIO [93], VINS [175], OKVIS [124], VI-EKF SLAM[178], AVIO[168], EKF VIO [177] and ROVIO [20]. Figures 5.23 and 5.24 illustrate different tracking methods comparison using different difficulty levels of EuRoC dataset sequences (MH03 and V203). All the compared systems show approximately similar performances to the ground truth; except the ORB SLAM odometry, which deviates on a part of the trajectory and presents difficulties, in particular when it is about a difficult navigation environment (*e.g.* V203). Such problems confirm

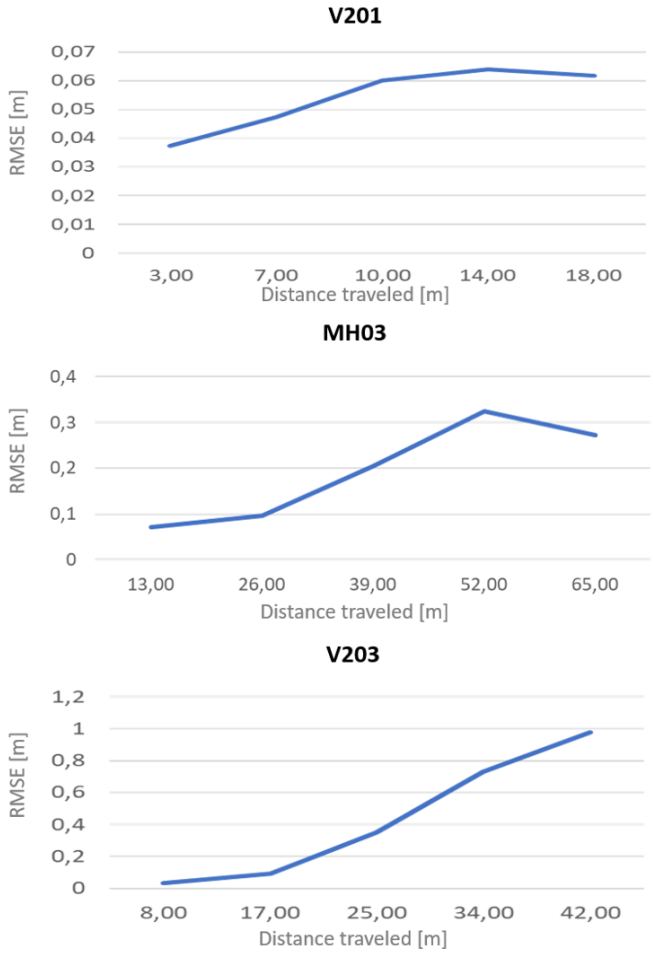

FIGURE 5.22: RPE translation of ORB SLAM (odometry), played on different EuRoC dataset levels, for different sub-trajectory lengths

the uncertainty of the odometry of ORB SLAM compared to VIO methods, which comes from various causes, including the use of a monocular camera and the need for other SLAM sections (Loop Closing and Local Mapping) to improve its localization accuracy, as shown latter on figure 5.25. The results presented in table 5.10 describe, the measured quality of each method according to our evaluation methodology and table 5.11 gives the different state-of-the-art systems results as they are mentioned in there original papers, in terms of ATE, with regard to the different EuRoC dataset sequences. ORB SLAM resulted the highest error values, especially in the case of difficult environments where the RMSE of ATE can reach $1.69m$, for example. Among other systems, our proposed adaptive method shows an accuracy that is, in many cases, similar or higher than the EKF VI tracking results. Furthermore, our Context-Adaptive VI tracking performs comparably to the VINS [175] and AVIO [168], and even better in some cases, like in R-VIO [93]. For example, in all easy sequences, our proposed solution outperforms R-VIO accuracy by about $0.02m$ minimum. This is achieved thanks to the execution control module, that, depending on the navigation context conditions, allows alternation between the following two approaches: KLT-ORB tracking which is adapted for easy environment (table 5.4), and EKF VI tracking, dedicated mainly to medium and difficult navigation contexts (table 5.6). Moreover, our Context Adaptive VI tracking compared to the adaptative method proposed in

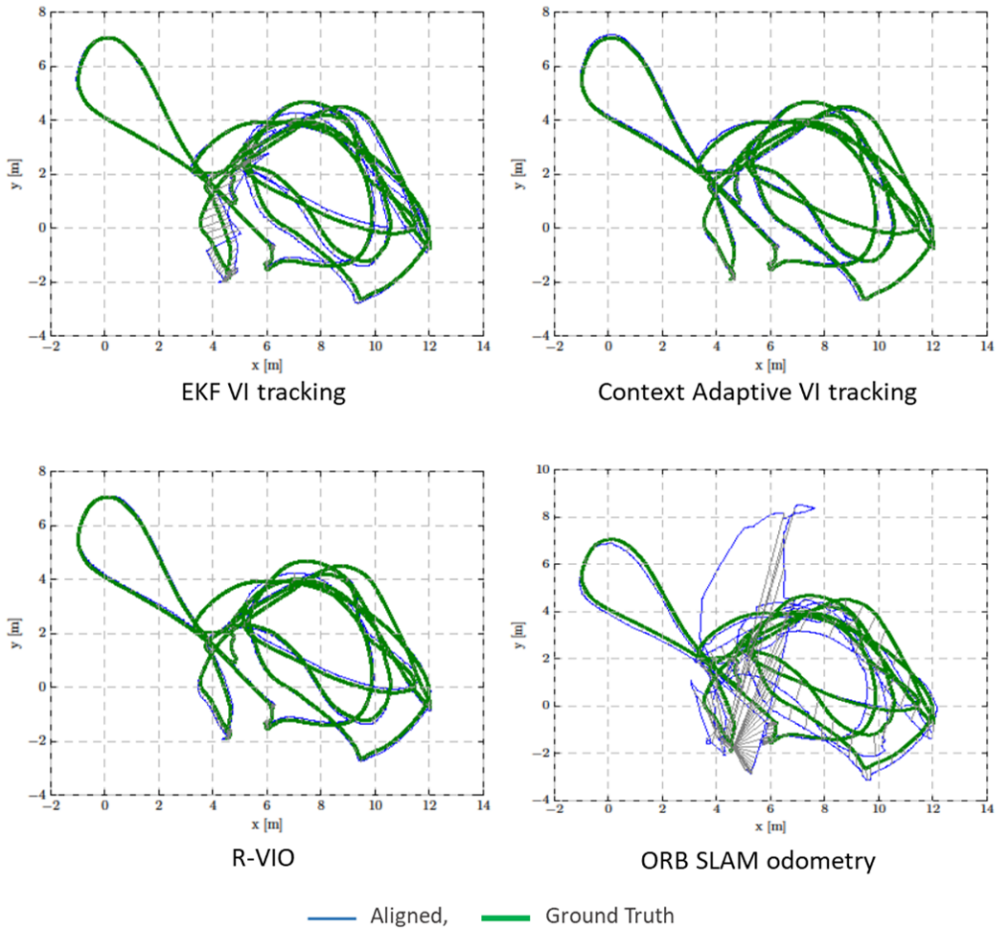

FIGURE 5.23: Trajectories comparison of different systems (ORB SLAM (Odometry), R-VIO, EKF VI tracking and the proposed system: Context Adaptive VI Tracking) on an easy-medium EuRoC dataset (MH03) (m) (gray lines represent the states correspondences with a step of ten poses)

[168], presents slightly less accurate results of $0.015m$, in terms of RMSE of ATE, in easy environments, but more robust in difficult one. The latter is also managed by an adaptive module based mainly on IMU measurements and successfully tracked PoIs number. This module alternates between visual and visual-inertial approaches, according to predefined operating levels. Therefore, for a fully controlled and adaptive operating mode, this system [168] gets easily lost in the difficult dataset sequences and does not allow continuous tracking and SLAM on the whole navigation track.

Dataset		ATE (*RMSE*) (m)			
		ORB SLAM	R-VIO [93]	proposed EKF VIO	proposed method
Easy	V101	0.38	0.10	0.15	0.083
	V201	0.26	0.16	0.15	0.063
	MH01	0.15	0.18	0.17	0.098
Difficult	V203	1.69	0.98	0.69	0.16
	MH03	0.31	0.48	0.39	0.14
	MH04	1.13	0.47	0.65	0.29

TABLE 5.10: Context Adaptive VI Tracking (tracking part) trajectory errors evaluation and comparison with various literature systems (measured values)

As presented in the previous sections, the PoIs detection frequency, in the two tracking approaches used for the proposed Context Adaptive VI Tracking, decreases remarkably compared to

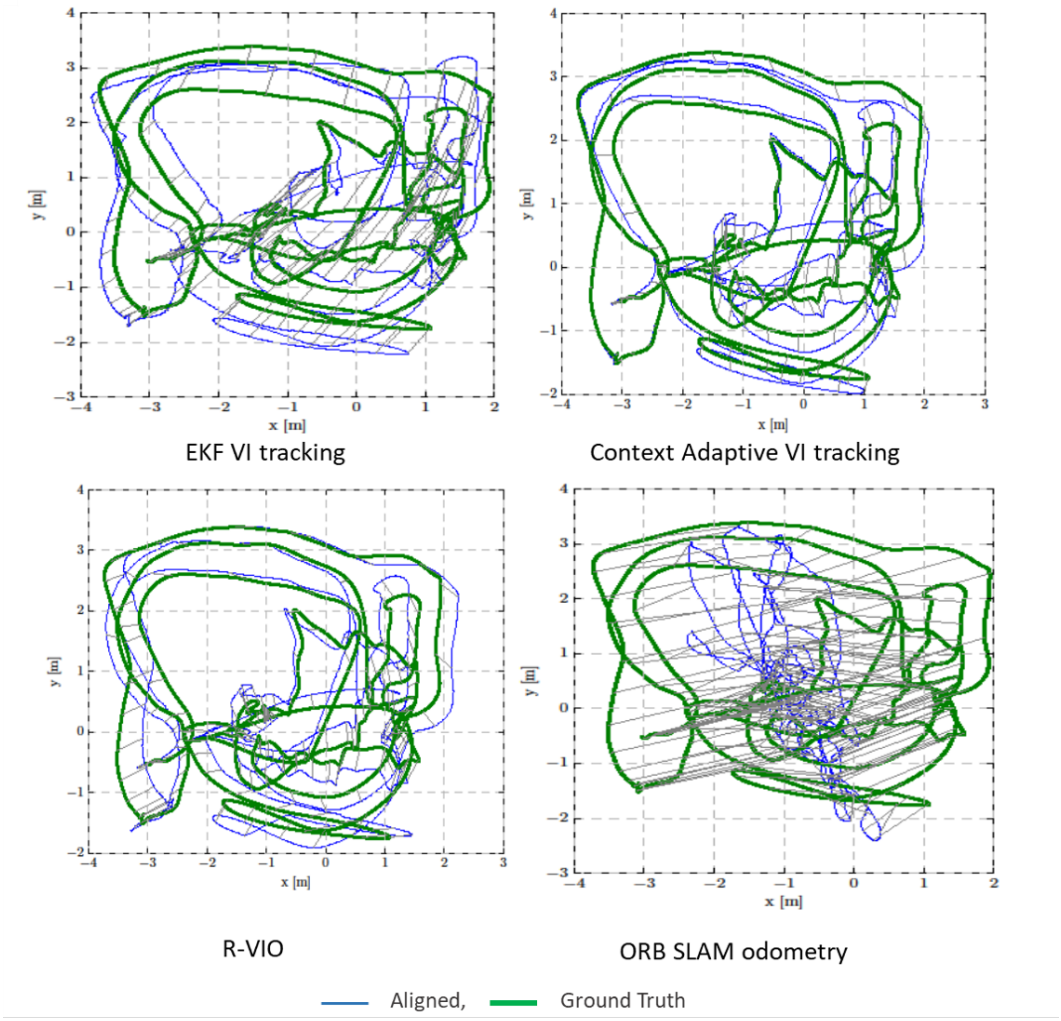

FIGURE 5.24: Trajectories comparison of different systems (ORB SLAM (Odometry), R-VIO, EKF VI tracking and the proposed system: Context Adaptive VI Tracking) on a difficult EuRoC dataset (MH03) (*m*) (gray lines represent the states correspondences with a step of ten poses)

Dataset		ATE (*RMSE*) (m)					
		VINS[175]	OKVIS[124]	VI KF-SLAM[178]	AVIO[168]	EKF VIO [177]	ROVIO [20]
Easy	V101	0.079	0.094	0.11	0.094	0.088	0.15
	V201	0.080	0.090	0.062	0.066	0.082	0.28
	MH01	0.15	0.33	0.16	0.036	0.17	0.24
Difficult	V203	0.16	0.17	0.17	failed	0.18	0.60
	MH03	0.22	0.25	0.25	failed	0.30	0.40
	MH04	0.32	0.27	0.29	failed	0.44	0.88

TABLE 5.11: Context Adaptive VI Tracking (tracking part) trajectory errors evaluation and comparison with various literature systems (original papers values)

ORB SLAM. In fact, the PoIs detection function occurrence and the number of PoIs required for the pose computation are both reduced. Thus, the PoIs are not systematically detected in all sequence frames and the number of detected PoIs varies depending on the system status and the executed method. This technique enables the decrease of computational workload for the Context Adaptive VI Tracking. As presented in table 5.12, the total reduction due to these improvements is estimated to be more than 16% (depending on the executed sequence and compared to other works).

By switching between both tracking methods, the computation time and overall time for each function is reduced. As shown in table 5.13, Keypoint Management, including different processes

Dataset	Evaluated system	PoI Detection Occurrence	PoI number
Easy	ORB SLAM	100% sequence frames	1000
	R-VIO	100% sequence frames	200 PoI
	proposed KLT-ORB VO	$\simeq$ 17% sequence frames	500 PoI
	proposed EKF VIO	100% sequence frames	$\simeq$ 5 PoI
	proposed method	$\simeq$ 74%	500 or 5 PoI
	VINS [175]	100%	100 - 300 PoI minimum
	OKVIS [124]	100%	400 PoI
	AVIO [168]	100%	500 PoI
	EKF VIO [177]	100%	50 or 1000 PoI
Difficult	ORB SLAM	100% sequence frames	1000 PoI
	R-VIO	100% sequence frames	200 PoI
	KLT-ORB based SLAM	$\simeq$ 22% sequence frames (or lost)	500 PoI
	EKF VI based SLAM	100% sequence frames	$\simeq$ 5 PoI
	proposed method	$\simeq$ 87%	500 or 5 PoI
	VINS [175]	100%	100 - 300 PoI minimum
	OKVIS [124]	100%	400 PoI
	AVIO [168]	100%	500 PoI
	EKF VIO [177]	100%	50 or 1000 PoI

TABLE 5.12: Comparison of PoIs management in the various methods analysed

needed for the visual data computation, such as PoIs detection and optical flow computing or the Inverse Depth representation and state vector insertion processing, takes 24.31$ms/frame$ in easy navigation environment and 24.07$ms/frame$ in difficult navigation environment. This represents a gain of 2.93$ms/frame$ minimum compared to ORB SLAM, notably thanks to reducing the number of detected PoIs. In addition, Pose Estimation & Tracking, which handles all necessary steps for the pose computation and tracking tasks, according to each method (either from visual measurements and following the motion model process, or by using the EKF VI process), had approximately similar execution time, compared to the EKF VIO, and a higher gain compared to ORB SLAM. This is estimated at 5.6$ms/frame$ in easy sequences and 10.41$ms/frame$ in difficult sequences.

Dataset	Evaluated system	functions's mean execution time	
		Keypoint Management	Pose Estimation & Tracking
Easy	ORB SLAM	27.24	12.72
	R-VIO	21.38	12.90
	proposed KLT-ORB VO	24.00	6.20
	proposed EKF VIO	24.60	8.00
	proposed method	24.31	7.12
	AVIO [168]	NA	19.9
	EKF VIO [177]	NA	12.00
Difficult	ORB SLAM	30.02 or lost (failure)	22.31 or lost (failure)
	R-VIO	23.86	13.41
	proposed KLT-ORB VO	lost (failure)	lost (failure)
	proposed EKF VIO	26.30	10.11
	proposed method	24.07	11.40
	AVIO [168]	NA	failure
	EKF VIO [177]	NA	13.00

TABLE 5.13: Execution time evaluation and comparison using EuRoC dataset ($ms/frame$) (NA: Not Available)

Summarizing, as presented in table 5.12 and 5.13, the computation complexity, especially that related to PoIs processing, and execution time of the Context Adaptive VI Tracking is significantly improved. Using the EKF VI algorithm on 80% of the used dataset reduces the complexity of the system's computation compared to ORB SLAM [227][169]. This is especially true in regard to the computational load where the approach has a gain of 25% on average, thanks to the reduction of the ORB PoIs detection frequency and thus the number of ORB PoIs. The use of the KLT-ORB tracking saved about 20% in execution time over all of the dataset sequences.

Before concluding, it is wise to introduce an analysis of the Context Adaptive VI SLAM, in order

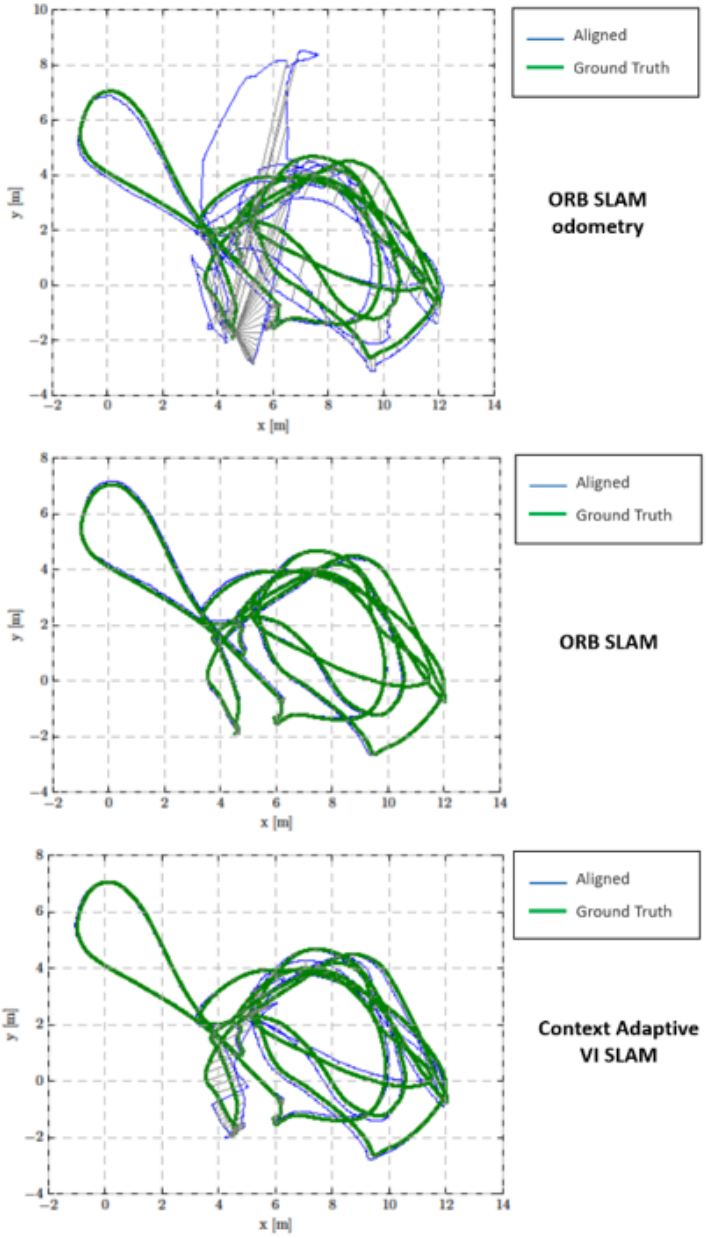

FIGURE 5.25: ORB SLAM odometry, ORB SLAM and Context Adaptive VI SLAM comparison (MH03) (gray lines represent the states correspondences with a step of ten poses)

to check how our adaptive tracking method works with the rest of the ORB SLAM components. Figure 5.25 illustrates the comparison between ORB SLAM odometry, ORB SLAM and Context Adaptive VI SLAM and ground truth executed on MH03 EuRoC sequence.

As we can see on figure 5.25, the ORB SLAM results are more accurate than those of ORB SLAM odometry. The reason for this is the use of the ORB SLAM's complementary threads: Local mapping and Loop closure, thus proving their importance in ensuring the consistency of the estimated trajectory, especially in visual SLAM. Furthermore, ORB SLAM and Context Adaptive VI SLAM have about the same behavior at the beginning of the navigation. However, from the middle and towards the end of the sequence, the proposed SLAM begins to encounter difficulties causing discrepancies

Dataset	proposed SLAM	ORB SLAM
	ATE	ATE
V101	0.09	**0.08**
V201	**0.08**	0.10
V203	**0.62**	0.65
MH01	0.12	**0.09**
MH03	0.18	**0.14**
MH04	**0.55**	0.59

TABLE 5.14: ORB SLAM and Context Adaptive VI SLAM evaluation illustrated using the RMSE of the ATE (m)

in the trajectory. These deviations are more important, compared to the ORB SLAM, due to the map optimization that requires a sufficient number of PoIs. Whereas, in our proposal, this number is reduced by more than 50% of the required amount. Despite the above challenge, Context Adaptive VI SLAM provides results consistent with those of ORB SLAM. As shown in table 5.14, our proposal SLAM represents accuracy estimated in terms of the average RMSEs of the easy sequences by $0.09m$ and by $0.45m$ for difficult navigation sequences; this means a loss in RMSE accuracy estimated in average of $0.01m$, approximately, compared to ORB SLAM, mainly in easy navigation environments.

5.4 Discussion & Conclusion

This chapter aimed to describe experiments performed to demonstrate the relevancy of the proposed Context Adaptive VI Tracking. The experimental environment was introduced as well as the main issues met in the different experiments, were discussed. The non-determinism and the scale estimation problems were explained. To address the non-determinism problem, we relied on the state-of-the-art technique executing the systems ten times, then use the average of the results for evaluation purposes. Scale estimation is an issue little addressed in the present literature, it poses many difficulties regarding the pose estimation. An adapted scale estimation approach was implemented in this work to compute the final camera poses, based on the rate between inertial and fused visual-inertial poses. In addition, a section of this work focuses on the trajectory alignment process and discusses the evaluation error metrics [230] [181] [234]. In our thesis, we performed the trajectory alignment applying the yaw-only rigid body and similarity transformations that are suitable for visual-inertial and monocular systems, respectively, and we quantified the system quality relying on the system trajectory accuracy and using the two most common metrics: ATE and RPE. Actually, trajectory alignment is a trajectory pre-processing used to enable the system quality evaluation and comparison with different state-of-the-art tracking methods: ORB SLAM [153], R-VIO [93], VINS [175], OKVIS [124], VI-EKF SLAM[178], AVIO[168], EKF VIO [177] and ROVIO [20]. After the previous discussions, our solution and its different components was evaluated. But, since ORB SLAM is considered to be a reference in the domain, it has been taken as a starting point for this work and evaluated in a first step. Our proposal was appraised by evaluating each component separately: the KLT-ORB Tracking and the EKF VI Tracking, and then the whole Context Adaptive Tracking system. The results of our performance assessment show, that the KLT-ORB tracking provides a short execution time and gives a sufficient accuracy, but also it lacks of robustness in difficult environments (with long distances and/or characterized by abrupt and rapid motions). The KLT-ORB tracking unquestionably provides a significant computational cost reduction thanks to the decrease of ORB PoIs redetection frequency and the use of the simple KLT visual algorithm. Furthermore, EKF VI tracking has been proven to be a robust method in most navigation conditions. However, it requires an additionnal computing workload due to the integration of IMU measurements. We proposed a leverage

that consists in reducing the number of PoIs detected (500 or 5 PoIs) and used for pose estimation. The performance results of the proposed system (the Context Adaptive VI Tracking) was presented, analyzed in terms of trajectory accuracy, computational load and computational time, and compared to other state-of-the-art works' results. Actually, Context Adaptive Tracking provides, an average ATE of $0.08m$ in easy navigation environments and of $0.19m$ in difficult environments. It provides a continuous and robust tracking in different navigation environments, contrary to some known literature approaches such as ORB SLAM [153] or the adaptive visual-inertial approach proposed in [168]. In addition, the proposed system accuracy is similar to most inertial visual approaches, however, it better exceeds them in the case of difficult navigation by at least a $0.02m$ error improvement. It can achieve an average frame-rate processing between $32fps$ and $29fps$, depending on the navigation environment (easy or difficult), this represents an execution time gain estimated by $1ms$ and $1.8ms$, when navigating in easy context and difficult context, respectively, compared to R-VIO [93]. Therefore, our proposal differs from other works by ensuring the system accuracy and even improving it and also reducing the computational load and execution time. This performance is due to our different contributions, mainly the algorithms used and their adequacy, the control module metrics and its operation, and the overall system implementation process. Moreover, the Context Adaptive SLAM was also assessed in order to verify its behavior regarding the rest of ORB SLAM components (Local Mapping and Loop Closing). Therefore, the Context Adaptive SLAM provides an accuracy loss estimated only by less than $0.01m$ comparing to ORB SLAM accuracy, especially for easy navigation environments. After all the previous evaluations and discussion, the interest of the proposed execution control module in the proposed context adaptive tracking, using two different approaches, is confirmed regarding the system robustness, accuracy and respect of the embedded system constraints.

Conclusion and Perspectives

Summary of the contributions

This research study proposes a robust visual-inertial tracking method suitable for mobile embedded systems implementation such as MAVs [175][93]. On one hand, tracking algorithms employed in different navigation systems are constrained by the navigation context conditions, especially those used in vision based navigation systems. For example, these systems are sensitive to black scenes with low texture, fast camera motion causing blur images, *etc*. Therefore, inertial measurement are used to improve visual tracking systems. This requires more computational complexity to perform visual-inertial fusion for tracking.

On the other hand, embedded systems are resources constrained. They are limited by computational power, on-board memory and sensor payload. Thus, they cannot handle the straightforward implementation of high computational complexity tracking algorithms. Current thesis develops and evaluates a Context Adaptive VI Tracking for embedded integration. It switches between two tracking methods: visual KLT-ORB tracking and EKF VI tracking, thanks to the execution control module that analyses the navigation context (motion, scene, previous tracking quality, and tracking consistency) and enables the switch to the appropriate tracking method.

First, a study of the state-of-the-art of navigation and tracking techniques based on computer vision and visual-inertial fusion models was put forward. Then, a discussion on the several embedded computing architectures has been also carried out. For designing adequate embedded systems, the hardware/software co-design methodology was explained as well. In addition, concerning tracking and localization, SLAM and odometry methods were discussed with emphasis on embedded implementations based on monocular vision and visual-inertial fusion. Besides this, an analysis of the navigation environments was performed. The focus was on the identification of the most problematic issues that impact the tracking robustness. This study allowed to choose the required control metrics and to configure the proposed execution control module.

After the previous discussion, our solution was presented and detailed. It consists of three main components, which are two different tracking methods: KLT-ORB tracking and EKF VI tracking, and an execution control module. The proposed system switches between these two tracking techniques according to the navigation environment analysis performed using the execution control module. The latter uses different metrics to alternate between visual and visual-inertial tracking such as system motion (linear and angular velocity), system FoV (traveled distances and rotational angles) and previous tracking quality (number of tracked PoI from last frame). Therefore, our proposition is named: "The Context Adaptive VI Tracking".

Subsequently, our proposed system is evaluated. Throughout our appraisals, several challenging issues were highlighted, in particular when comparing the estimated trajectories to the ground truth and computing estimating accuracy errors. These aspects include primarily the non-determinism, the scale estimation and the trajectory alignment. Firstly, to deal with the non-determinism issue, each approach is run ten times and the estimation errors are computed on the average of these runs.

Next, for scale estimation, we analyzed several methods proposed in the literature and we took inspiration from an approach devoted especially to visual-inertial systems [198]. The scale estimation solution, proposed in this work, consists in computing a ratio between the pure inertial pose and the final fused pose. Subsequently, to evaluate our estimate trajectories regarding the ground truth and compare them with other literature solutions, we have to get the estimation and the ground truth into the same reference frame using an alignment technique commonly used in the state-of-the-art systems, in particular those based on visual-inertial fusion. Therefore, in this thesis, we applied the Zhang and Scaramuzza trajectory alignment [230] based on the *Umeyama* method [212] and available as an open source tool [181].

The Context Adaptive VI Tracking assessment is performed taking into account all the issues addressed above. Our proposal ensures a robust tracking even in the difficult and easy navigation environment. Relying on different EuRoC dataset sequences levels, as one of the most relevant (visual-inertial) dataset used in VINS literature giving a diversity of navigation environments configurations, our proposed system provides an accuracy, expressed in terms of ATE's RMSE, between 0.063 and 0.098*meters* at maximum in easy environments (V101 and MH01, respectively), and between 0.14 and 0.29*meters* in difficult environments (MH03 and MH04, respectively). This is reached by maintaining an average frame-rate processing of $32fps$ in easy environments and $29fps$ in difficult ones. These results are obtained thanks to the efficiency of the chosen tracking algorithms, the execution control module and the PoIs management that significantly reduces the occurrence of PoIs detection and also limits the number of the detected PoIs to either 5 or 500 PoIs. Our primary contributions to achieve these results were:

(i) analysis of different types of navigation context, and identification of the more complex and problematic navigation conditions;

(ii) development of an execution control module allowing a switch between two tracking methods, according to various parameters: the system's navigation environment type (image quality), in motion, FoV and the current tracking operating status (PoI cheking);

(iii) PoIs management methodology performed between different processing functions, that saves memory and computational time by limiting the number of the PoIs at 500 or 5 PoIs;

(iv) development of two tracking approaches, appropriate to the embedded implementation, as well as their adaptation to the remaining ORB SLAM processing parts: KLT-ORB and EKF VI tracking.

However, integrating the multi-method solution on an embedded architecture might require difficult engineering and implementation, particularly, in terms of latency required for the switching management between the two tracking methods. For now, this latency can reach 6 *seconds* in case of medium-difficult tracking environment (> 70 *meters*).

Perspectives

The research presented in this thesis offers potential extensions and directions for further work. These future developments aim to extend the contributions provided in this thesis and to improve the practicability of the proposed solution in real world applications. In the following paragraphs, our outlook perspectives are introduced.

Improving the scale estimation process Accurate scale estimation is a challenging research problem in visual and visual-inertial tracking. Experimental results have illustrated the benefit of using visual and inertial data for the scale estimation. In particular, the scale estimation method based on IMU and vision-inertial fused poses. As mentioned in chapter 5, this estimation technique varies among the time, because it relies on the final estimated pose which can be erroneous. Future research can be undertaken to develop better scale estimation strategies using visual and inertial data independently of final pose processing, even with tightly coupled sensors. This is intended to avoid the large variations of scale, as well as the additional noise and errors inherent to this.

Improving the trajectory consistency metrics Future studies can also expand the metrics employed for the execution control module. Experimental results have demonstrated the importance of the execution control module and its controlling metrics used to alternate between the two proposed tracking approaches. Following these results, future studies can also improve the existing metrics, especially, trajectory consistency metric. They already can define a new global metric, instead of local NDS metric, that refines the final pose estimation independently of the chosen tracking method. Therefore, the tracking accuracy can be enhanced.

Hardware-constrained integration of the Context Adaptive Visual-Inertial Tracking The Context Adaptive Visual-Inertial Tracking can be employed in larger navigation systems. In particular, micro robotic vehicles, which have limitation of computation complexity, memory, space and power resources. The proposed method has been developed to be implemented on such embedded systems and to respect the embedded systems requirements and constraints. Therefore, the future work could focus on porting the solution on an embedded Multi-Processor SoC (MPSoC) architecture of the Xilinx family, *Zynq UltraScale+ MPSoC* for example, including accelerators and dedicated components, such as GPU, video CODEC, *etc*. In addition, the flexible I/Os and processing power of the Zynq UltraScale+ MPSoC requires very little hardware beyond the MPSoC itself, other than the sensors and the external memory. The performance/watt measurement of this device is about 3 times better compared to a CPU-based system using silicon from a leading competitor [1]. Subsequently, this embedded implementation will enable to assess the switching latency between the two tracking methods and analyze its behavior, in order to improve it in a constrained context. Also it will allow to evaluate the accuracy and the robustness of the proposed tracking in a real embedded context, and its efficiency in terms of the computation complexity and the required execution time.

Appendix A

Detector/Descriptors Examples

A.1 Example 1: SIFT & SURF

PoI Detection In 1999, in order to overcome the detectors invariances problems, Lowe proposed the Scale Invariant Feature Transform (SIFT) detector [132]. It is characterized by its invariance to changes in scale, image rotation and some lighting conditions, and its better robustness to viewpoint changes. The PoI detection using this method requires, first of all, a convolution between the image and a set of Gaussian filters at different scales:

$$L(\mathrm{x}, \mathrm{œ}) = \mathrm{G}(\mathrm{x}, \mathrm{œ}) * \mathrm{I}(\mathrm{x}) \tag{A.1}$$

where:

x= (x, y): the PoIs coordinates

$L(\mathrm{x}, \mathrm{œ})$: the image filtered by a Gaussian

$I(\mathrm{x})$: the image to be processed

$G(x, y, \sigma)$: Gaussian filter

σ : scale

After this smoothing, Gaussian difference (DoG) images are then calculated:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{A.2}$$

where k is a fixed multiplier factor of the algorithm depending on the smoothness of the space discretization of the desired scales (generally equal to$\sqrt{2}$) (figure A.1).

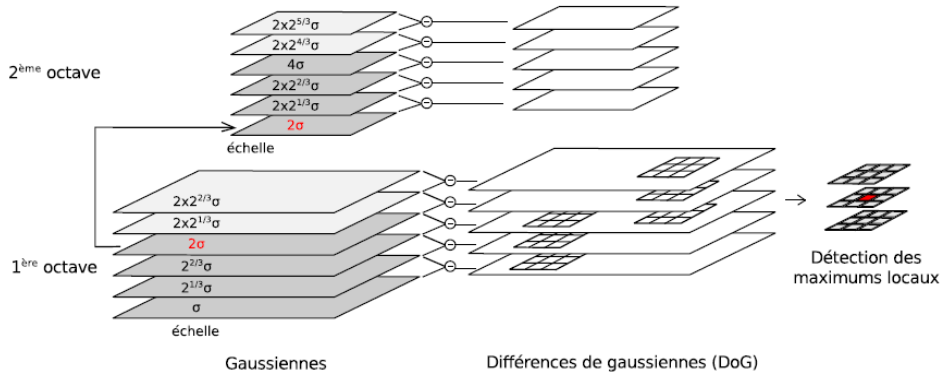

FIGURE A.1: Illustration of the detection based on a DoG of the SIFT algorithm

Once the resulting images of Gaussian differences are created, the identification of PoI is done by calculating local extrema. Indeed, a point is considered as a PoI if its value is maximum in relation to all the 26 points surrounding it. Figure A.1 illustrates this selection of PoI.

Afterwards, to better refine this identification, the interpolation of the neighbourhood color components is applied to improve the positioning of these PoI. Thus, low-contrast points, as well as those on the image borders, are discarded. This detector produces a very large number of PoI compared to all other detectors thanks to its detection at different scales and resolutions. But this does not prevent it from having a disadvantage that comes from the computational complexity and the time consumption [206][105], which makes it heavy and inefficient for low-power systems, as the embedded mobile navigation systems especially embedded tracking systems treated in this thesis (chapter 3). In 2006, Herbert Bay and *al.* proposed an accelerated alternative SIFT detector algorithm, it is the Speed-Up Robust Feature detector (SURF) which is faster but not as robust as SIFT [164].

SURF [15][16] is based on the same concept as the SIFT detector. It is an algorithm that is characterized by its invariance to scale and rotation changes. It is based on the approximations by the second order partial derivatives matrix of the Hessian matrix $(2x2)$. To accelerate the computation, SURF uses a precomputation on the integral image, which makes it different from the SIFT detector, but both still have a long computation time, especially compared to ORB which is presented later. *i.e.* in [206] the detection time per point is 1.63 $.10^{-3}ms$ and 0.47 $.10^{-3}ms$ for SIFT and ORB detectors, respectively.

PoI Description SIFT descriptor is based on oriented gradient histograms. It was proposed by Lowe [132] [131] to couple the SIFT detector with a well adapted descriptor. In fact, after detecting the PoI, a list of their coordinates and their characteristic scales are provided. Describing a PoI (x) using the SIFT descriptor begins by dividing the neighbourhood of each PoI into 16 blocks of 4×4 pixels.Then in each block, a histogram is constructed using the orientations $\theta(x, œ)$ computation at each uncoupled x point (figure A.2-(a)). These are weighted by the amplitude $m(x, œ))$ of the gradient and by the convolution with a Gaussian function (figure A.2-(b)).

This histogram contains 36 graduations (classes) each one representing an angle of 10 degrees. The figure A.2 illustrates the histogram construction and the dominant orientations identification:

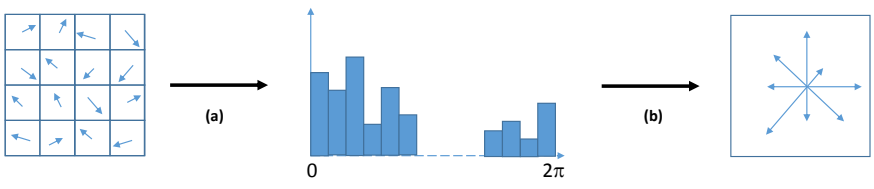

FIGURE A.2: Histogram construction

A.2 Example 2: ORB, FAST & BRIEF

PoI Detection Oriented FAST and Rotated BRIEF (ORB) [183] is a method that combines the Features from Accelerated Segment Test (FAST) detector and the Binary Robust Independent Elementary Features (BRIEF) descriptor by modifying them to improve their performance. It was developed in 2011 by Ethan Rublee et *al.*, this algorithm is an efficient alternative to SIFT and SURF, especially in terms of performance and computation cost, which are a critical condition for the embedded mobile

navigation systems development, and especially for our proposed embedded tracking system proposed in this thesis. In fact, FAST is a corner detection algorithm proposed in 2006 by Edward Rosten and Tom Drummond [182] [191]. It was developed to satisfy the requirements of real-time applications as the moving SLAM robot with limited computer resources. The FAST detector is based on the use of a circle around the central pixel (pixel studied) (figure A.3).

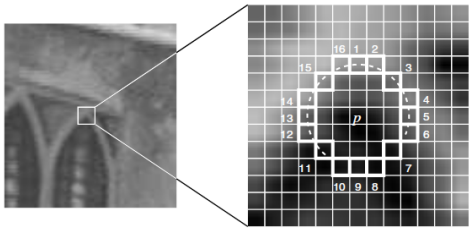

FIGURE A.3: Illustration of the pixel to be tested (central pixel) and the 16 pixels of the circle [182]

First of all, the pixel to be tested p must be selected and the 16 pixels circle around it must be taken into consideration. Then, a threshold t is set to perform the test. A pixel p is considered a PoI (corner) if there is a set of n pixels adjacent to the 16 pixel circle (the white dotted circle in figure A.3) brighter than $I_p + t$ ("bright" pixels) or darker than $I_p - t$ ("dark" pixels) (I_p: the intensity of the pixel to test). n is chosen equal to 12 pixels, this amounts to the use of the high-speed test which excludes a significant number of non-corners and consequently it only examines the four pixels 1, 5, 9 and 13 (1 and 9 are first examined if they are brighter or darker. If this is the case, pixels 5 and 3 are examined). So to have p as a corner, three of the four pixels must be brighter than I_{p+t} or darker than I_{p-t}. If not, then p can't be a corner. This procedure is applied to all pixels of the image.

This algorithm is not safe from any limitations. First of all, the algorithm does not work very well when $n < 12$, because the number of detected PoI is very high. Then, the speed of the algorithm depends on the sequence in which the pixel test is performed [182]. In terms of invariances, the FAST detector is non-invariant in terms of scale and rotation changes. To ensure invariance to scale changes, Ethan Rublee and *al.* [183] proposed producing a scale pyramid and extracting the PoI from it. These are found by applying the FAST detector, then filtered by calling Harris [82] at each scale. However, for rotation invariance Ethan Rublee et *al.* [183] proposed to use the intensity centroid. Its value allows assigning an orientation to a corner by assuming that the intensity of the corner is shifted from its center. Following these improvements, the new detection algorithm on which the ORB is based is called oFAST (oriented FAST) [183]. The latter is known for its lightness in terms of computational complexity as well as for its rapid computation time for PoI detection as in [206] and PoI description and matching which is illustrated in [105] and explained in follow, especially when compared to the other algorithms mentioned above.

PoI Description For description ORB is based on BRIEF descriptor. In fact BRIEF [26] is a binary PoI descriptor. It is based on fixing a set of N point pairs and calculating the difference between their intensity values two by two to allow them to be compared using the following test $\tau(p, x, y)$:

$$\tau(p, x, y) := \begin{cases} 1 & \text{si} \quad p(x) < p(y) \\ 0 & \text{sinon} \end{cases} \tag{A.3}$$

where $p(x)$ is the intensity over the entire pixel $x(u, v)^T$.

The BRIEF descriptor vector size is $\frac{N}{8}$ ($N = 128, 256$ ou 512). This descriptor is known for its rapid implementation. As a result, it is effective in real-time applications. However, it fails to have the invariance to rotations. The rotated BRIEF (rBRIEF) is the extension of the BRIEF descriptor that solves this problem of rotation changes. It was proposed as part of the ORB detection/description algorithm. Thus, it is one of the most appropriate and widely used description algorithms for embedded mobile navigation systems.

Appendix B

Epipolar Geometry

B.1 Epipolar geometry

Epipolar geometry refers to the set of geometric relationships that are applied between two images independently of the rigid scene. This is a well-known concept used in both vision and specifically computer vision. In this sub-section the main notions used later for the motion computation between images and pose are presented: the fundamental and essential matrices.

B.1.1 Fundamental Matrix

Let $\hat{q}_1$ and $\hat{q}_2$ be two observations (expressed in homogeneous coordinates) of the same 3D point Q having homogeneous coordinates Q. In this study, these two observations are considered two points of interest detected respectively in frames f_1 and f_2 presented by the process presented in section 1.2. Using epipolar geometry, it is possible to identify a point $\hat{q}_1$ that is positioned on an epipolar line $l_2 \sim F\hat{q}_1$ in the second camera (figure B.1). As well, if $\hat{q}_1$ and $\hat{q}_2$ are the same 3D point observations, then the following relationship is verified:

$$\hat{q}_2^T F \hat{q}_1 = 0 \tag{B.1}$$

The fundamental matrix F is a 3×3 matrix with rank 2. It only depends on the intrinsic and extrinsic parameters of the cameras and is therefore independent of the scene. The epipoles e_1 and e_2 are the projection of the optical centres in the two images, and they constitute the core of the fundamental matrix $Fe_1 = 0$ and $F^T e_2 = 0$.

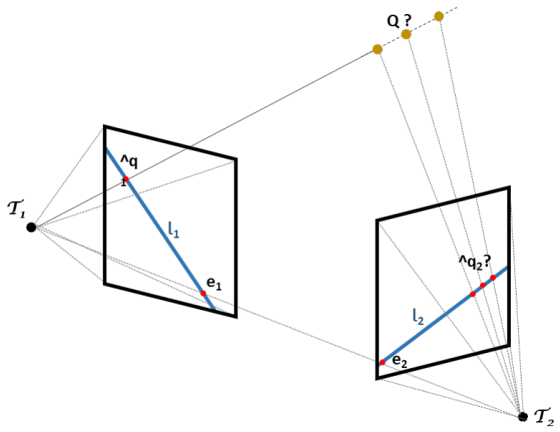

FIGURE B.1: Epipolar Geometry

This matrix is useful because it allows to calculate the cameras' projection P_i matrices. So, in the case of estimating the metric displacement between two images (two cameras), it is possible to use the essential matrix directly without computing the fundamental matrix. In addition, if the cameras are calibrated, it is possible to estimate the essential matrix from the fundamental matrix.

B.1.2 Essential Matrix

In the same way as the fundamental matrix, the essential matrix links two homologous points $\hat{d_1}$ and $\hat{d_2}$, expressed this time in the main plane of each camera (and not in the image frame): $\hat{d_i} = K_i^{-1}\hat{q_i}$ is the camera's calibration matrix c_i.

The essential matrix provides the following bilinear relationship:

$$\hat{d_2}^T E \hat{d_1} = 0 \tag{B.2}$$

where $E \sim K_2^T F K_1$ is the essential matrix. It has only five degrees of freedom that correspond to the extrinsic parameters of inter-image displacement (six degrees of freedom less one due to the uncertainty of the scale factor).

Appendix C

RANSAC

C.1 Filtering for robust estimation: RANSAC

In practice, when matching, some correspondences are very different from reality and are considered as erroneous or *outliers*. These data should not be taken into account for the displacement estimate, which is later explained, as they would significantly reduce this estimate's quality. To solve this problem, different approaches have been proposed such as M-estimators [98] and RANdom SAmple Consensus (RANSAC) [67]. Here we are interested in the latter method.

The RANSAC algorithm proposed by Fischler and Bolles [67] is a general parameter estimation approach that is developed to deal with a large proportion of outliers in the input data. Unlike other common robust estimation techniques such as M-estimators and least-median squares that have been adopted by the computer vision community from the statistical literature, RANSAC was developed by the computer vision community. It is a paradigm based on a simple principle: searching for the largest set of data compatible with a parametric model. Moreover, in order to ensure that the system to be solved is sufficiently constrained, a minimum number of data is used to calculate the model parameters, instead of using all available data, and this is depending on the number of degrees of freedom of the model. Lastly, the research performed is not exhaustive but is often done on a sampling approach in order to reduce its computational cost.

RANSAC is an iterative algorithm based on two steps: a) selecting a minimum random sample of matches to estimate the parameters of the geometric model, and b) constructing a subset of matches that is compatible with that geometry. This set is called a *consensus* set. The consistency is measured by the size of the consensus set (when the original version of RANSAC is performed) or by using a more sophisticated model fit measure (M SAC, M LESAC). The search is stopped when a consensus set of sufficient size is found, or when a sufficient number of runs have been performed to ensure the reliability of the solution.

N represents the number of RANSAC iterations, it is selected to be sufficiently high to ensure that the probability p (usually set at 0.99) of at least one of the random sample sets not having an outlier. Let u be the probability that a selected data point is an outlier and $v = 1 - u$ the probability of observing an outlier. N iterations of the minimum number of points indicated m are required, when

$$1 - p = (1 - u^m)^N \tag{C.1}$$

then after doing some computational manipulations, N is

$$N = \frac{\log(1 - p)}{\log(1 - (1 - v)^m)} \tag{C.2}$$

Bibliography

[1] In:

[2] D. Abeywardena et al. “Fast, on-board, model-aided visual-inertial odometry system for quadrotor micro aerial vehicles”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1530–1537.

[3] Palmer et al. *NEURAL PROCESSING UNIT*. Patent No.: US 8,655,815 B2. 2014. URL: https://patents.google.com/patent/US8655815.

[4] R. J. Althof, M. G. J. Wind, and J. T. Dobbins. “A rapid and automatic image registration algorithm with subpixel accuracy”. In: *IEEE Transactions on Medical Imaging* 16.3 (1997), pp. 308–316.

[5] Apple. *ARKit*. Website. 2018. URL: https://developer.apple.com/arkit/.

[6] A. Arcaya-Jordan, A. Pegatoquet, and A. Castagnetti. “Smart Connected Glasses for Drowsiness Detection: a System-Level Modeling Approach”. In: *2019 IEEE Sensors Applications Symposium (SAS)*. 2019, pp. 1–6.

[7] ARM. *Products processors*. 2018. URL: https://www.arm.com/products/silicon-ip-cpu (visited on 10/03/2018).

[8] *ARM Mali-C71*. 2017. URL: https://www.anandtech.com/show/11293/arm-announces-mali-c71-automotive-isp (visited on 2017).

[9] Raul Mur Artal. “Real-time accurate visual slam with place recognition”. PhD thesis. Universidad de Zaragoza, 2017.

[10] Soheil Bahrampour et al. “Comparative Study of Deep Learning Software Frameworks”. In: *arXiv: Learning* (2016).

[11] T. Bailey and H. Durrant-Whyte. “Simultaneous localization and mapping (SLAM): part II”. In: *IEEE Robotics Automation Magazine* 13.3 (2006), pp. 108–117.

[12] V. Balntas et al. “HPatches: A benchmark and evaluation of handcrafted and learned local descriptors”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), pp. 1–1.

[13] D. I. Barnea and H. F. Silverman. “A Class of Algorithms for Fast Digital Image Registration”. In: *IEEE Transactions on Computers* C-21.2 (1972), pp. 179–186.

[14] Brendan Barry et al. “Always-on vision processing unit for mobile applications”. In: *IEEE Micro* 35.2 (2015), pp. 56–66.

[15] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “SURF: Speeded up robust features”. In: *Computer vision–ECCV 2006* (2006), pp. 404–417.

[16] Herbert Bay et al. “Speeded-up robust features (SURF)”. In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359.

[17] Ertuğrul BAYRAKTAR and Pınar BOYRAZ. "Analysis of feature detector and descriptor combinations with a localization experiment for various performance metrics." In: *Turkish Journal of Electrical Engineering & Computer Sciences* 25.3 (2017).

[18] Vincent gay bellile et al. "Review and classification of vision-based localisation techniques in unknown environments". In: *IET Radar, Sonar Navigation* 8 (Dec. 2014).

[19] Berta Bescos et al. "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4076–4083.

[20] M. Bloesch et al. "Robust visual inertial odometry using a direct EKF-based approach". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 298–304.

[21] Michael Bloesch et al. "CodeSLAM—learning a compact, optimisable representation for dense visual SLAM". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2560–2568.

[22] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. "Visual Navigation for Mobile Robots: A Survey". In: *Journal of Intelligent and Robotic Systems* 53 (Nov. 2008), pp. 263–296.

[23] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[24] Michael Burri et al. "The EuRoC Micro Aerial Vehicle Datasets". In: *Int. J. Rob. Res.* 35.10 (Sept. 2016), pp. 1157–1163. ISSN: 0278-3649. URL: https://doi.org/10.1177/0278364915620033.

[25] Cadence. *TIE Language - The Fast Path to High-Performance Embedded SoC Processing*. Tech. rep. URL: https://ip.cadence.com/knowledgecenter/resources/know-dip-wp.

[26] Michael Calonder et al. "BRIEF: Binary robust independent elementary features". In: *Computer Vision–ECCV 2010* (2010), pp. 778–792.

[27] Antonio Canclini et al. "Evaluation of low-complexity visual feature detectors and descriptors". In: *Digital Signal Processing (DSP), 2013 18th International Conference on*. IEEE. 2013, pp. 1–7.

[28] Nicholas Carlevaris-Bianco, Arash K. Ushani, and Ryan M. Eustice. "University of Michigan North Campus long-term vision and lidar dataset". In: *International Journal of Robotics Research* 35.9 (2015), pp. 1023–1035.

[29] Kieran Carroll. "The Early History of Canadian Planetary Exploration". In: Oct. 2019.

[30] Simone Ceriani et al. "Rawseeds Ground Truth Collection Systems for Indoor Self-Localization and Mapping". In: 27.4 (2009). ISSN: 0929-5593. URL: https://doi.org/10.1007/s10514-009-9156-5.

[31] *CEVA XM6 Product Note*. CEVA, Inc., May 2017.

[32] Chen Chang et al. "A Review of Visual-Inertial Simultaneous Localization and Mapping from Filtering-Based and Optimization-Based Perspectives". In: *Robotics* 7 (Aug. 2018), p. 45.

[33] R. Q. Charles et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 77–85.

[34] D. Chatzopoulos et al. "Mobile Augmented Reality Survey: From Where We Are to Where We Go". In: *IEEE Access* 5 (2017), pp. 6917–6950.

[35] Nived Chebrolu et al. "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields". In: *The International Journal of Robotics Research* 36 (July 2017), p. 027836491772051.

[36] L. Chen et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2018), pp. 834–848.

[37] Hong Cheng et al. "A deformable local image descriptor". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[38] Y. Choi et al. "KAIST Multi-Spectral Day/Night Data Set for Autonomous and Assisted Driving". In: *IEEE Transactions on Intelligent Transportation Systems* 19.3 (2018), pp. 934–948.

[39] J. Civera et al. "1-point RANSAC for EKF-based Structure from Motion". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 3498–3504.

[40] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: June 2016.

[41] Peter Corke, Jorge Lobo, and Jorge Dias. "An Introduction to Inertial and Visual Sensing". In: *I. J. Robotic Res.* 26 (June 2007), pp. 519–535. DOI: 10.1177/0278364907079279.

[42] Standard Performance Evaluation Corporation. *SPEC CPU2006 Analysis Papers: Guest Editor's Introduction*. 2018. URL: https://www.spec.org/cpu2006/publications/SIGARCH-2007-03/ (visited on 10/03/2018).

[43] Gabriela Csurka, Christopher Dance, and Martin Humenberger. *From handcrafted to deep local invariant features*. July 2018.

[44] Igor Cvišić et al. "SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles". In: *Journal of field robotics* 35.4 (2018), pp. 578–595.

[45] Quoc Khanh Dang et al. "A Virtual Blind Cane Using a Line Laser-Based Vision System and an Inertial Measurement Unit". In: *Sensors* 16.1 (2016). ISSN: 1424-8220. URL: http://www.mdpi.com/1424-8220/16/1/95.

[46] Andrew J. Davison. "Real-Time Simultaneous Localisation and Mapping with a Single Camera". In: *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*. ICCV '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 1403–. ISBN: 0-7695-1950-4. URL: http://dl.acm.org/citation.cfm?id=946247.946734.

[47] César Debeunne and Damien Vivet. "A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping". In: *Sensors* 20.7 (2020), p. 2068.

[48] J. Delmerico et al. "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.

[49] Jeffrey A. Delmerico and Davide 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.

[50] Jeffrey A. Delmerico and Davide Scaramuzza. "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots". In: *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. 2018, pp. 2502–2509. URL: https://doi.org/10.1109/ICRA.2018.8460664.

[51] Mike Demler. *XAVIER SIMPLIFIES SELF-DRIVING CARS*. Tech. rep. Microprocessor Report, The Linley Group, 2017. URL: http://www.linleygroup.com/mpr/h/article.php?id=11820.

[52] G.N. Desouza and A.C. Kak. "Vision for Mobile Robot Navigation: A Survey". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24 (Mar. 2002), pp. 237 –267.

[53] Tien Do, Luis C Carrillo-Arce, and Stergios I Roumeliotis. "High-speed autonomous quadrotor navigation through visual and inertial paths". In: *The International Journal of Robotics Research* 38.4 (2019), pp. 486–504.

[54] year = 2017 url = https://fr.wikipedia.org/wiki/Fichier:AFI$_0$5$_2$017$_D$onecle_drone$_0$02.$jpgurldate = 2018 - 12 - 20Donecletitle = Donecledrone$.

[55] D. Dornellas and M. Silva. "Stereo visual-inertial aided navigation for UAVs". In: 2019.

[56] H. Durrant-Whyte and T. Bailey. "Simultaneous localization and mapping: part I". In: *IEEE Robotics Automation Magazine* 13.2 (2006), pp. 99–110.

[57] Digi-Key Electronics. *ADIS16490BMLZ*. 2021. URL: https://www.digikey.fr/product-detail/fr/analog-devices-inc/ADIS16490BMLZ/ADIS16490BMLZ-ND/6490592.

[58] J. Engel, V. Koltun, and D. Cremers. "Direct Sparse Odometry". In: (Mar. 2018).

[59] J. Engel, T. Schöps, and D. Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: 2014.

[60] Jakob Engel, Jurgen Sturm, and Daniel Cremers. "Camera-based navigation of a low-cost quadrocopter". In: Oct. 2012, pp. 2815–2821. ISBN: 978-1-4673-1737-5.

[61] Hadi Esmaeilzadeh et al. "Dark Silicon and the End of Multicore Scaling". In: *Proceedings of the 38th Annual International Symposium on Computer Architecture*. ISCA '11. San Jose, California, USA, 2011, pp. 365–376. ISBN: 978-1-4503-0472-6. URL: http://doi.acm.org/10.1145/2000064.2000108.

[62] Hadi Esmaeilzadeh et al. "Neural Acceleration for General-Purpose Approximate Programs". In: *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO-45. Vancouver, B.C., CANADA, 2012, pp. 449–460. ISBN: 978-0-7695-4924-8. URL: https://doi.org/10.1109/MICRO.2012.48.

[63] Kun Fan et al. "Hardware implementation of a virtual blind cane on FPGA". In: *Real-time Computing and Robotics (RCAR), 2017 IEEE International Conference on*. IEEE. 2017, pp. 344–348.

[64] Wei Fang, Lianyu Zheng, and Huanjun Deng. "A motion tracking method by combining the IMU and camera in mobile devices". In: *Sensing Technology (ICST), 2016 10th International Conference on*. IEEE. 2016, pp. 1–6.

[65] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[66] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. URL: http://doi.acm.org/10.1145/358669.358692.

[67] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. URL: http://doi.acm.org/10.1145/358669.358692.

[68] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast semi-direct monocular visual odometry". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 15–22.

[69] Christian Forster et al. "IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation". In: *Robotics: Science and Systems*. 2015.

[70] Friedrich Fraundorfer and Davide Scaramuzza. "Visual odometry: Part i: The first 30 years and fundamentals". In: *IEEE Robotics and Automation Magazine* 18.4 (2011), pp. 80–92.

[71] Friedrich Fraundorfer and Davide Scaramuzza. "Visual Odometry: Part II - Matching, Robustness, and Applications". In: *IEEE Robotics Automation Magazine - IEEE ROBOT AUTOMAT* 19 (June 2012), pp. 78–90.

[72] Andrei Frumusanu. *HiSilicon Kirin 970 - Android SoC Power & Performance Overview*. 2018. URL: https://www.anandtech.com/show/12195/hisilicon-kirin-970-power-performance-overview (visited on 10/03/2018).

[73] Jorge Fuentes-Pacheco, José Ruíz Ascencio, and Juan M. Rendón-Mancha. "Visual simultaneous localization and mapping: a survey". In: *Artificial Intelligence Review* 43 (2012), pp. 55–81.

[74] Paul Furgale et al. "The Devon Island rover navigation dataset". In: *International Journal of Robotic Research - IJRR* 31 (May 2012), pp. 707–713.

[75] R. Furlan. "The future of augmented reality: Hololens - Microsoft's AR headset shines despite rough edges [$Resources_{T}oolsandToys$]". In: *IEEE Spectrum* 53.6 (2016), pp. 21–21.

[76] Dorian Galvez-Lopez and J. D. Tardos. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197.

[77] P. Geneva et al. "LIPS: LiDAR-Inertial 3D Plane SLAM". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 123–130.

[78] Jason Geng. "Structured-light 3D surface imaging: a tutorial". In: *Advances in Optics and Photonics* 3.2 (2011), pp. 128–160.

[79] G. H. Golub and C. Reinsch. "Singular value decomposition and least squares solutions". In: *Numerische Mathematik* 14.5 (1970), pp. 403–420. ISSN: 0945-3245.

[80] Google. *ARCore Fundamental Concepts*. Website. 2018. URL: https://developers.google.com/ar/discover/concepts.

[81] Giorgio Grisetti et al. "A Tutorial on Graph-Based SLAM". In: *IEEE Intelligent Transportation Systems Magazine* 2 (2010), pp. 31–43.

[82] Chris Harris and Mike Stephens. "A combined corner and edge detector." In: *Alvey vision conference*. Vol. 15. 50. Manchester, UK. 1988, pp. 10–5244.

[83] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.

[84] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0521540518.

[85] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. "Comparative evaluation of binary features". In: *Computer Vision–ECCV 2012*. Springer, 2012, pp. 759–773.

[86] Richard Helps and Scott Pack. "Cyber-physical system concepts for IT students". In: Oct. 2013, pp. 7–12.

[87] J. A. Hesch et al. "A Laser-Aided Inertial Navigation System (L-INS) for human localization in unknown indoor environments". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 5376–5382.

[88] J. A. Hesch et al. "Consistency Analysis and Improvement of Vision-aided Inertial Navigation". In: *IEEE Transactions on Robotics* 30.1 (2014), pp. 158–176.

[89] Joel A Hesch et al. "Towards consistent vision-aided inertial navigation". In: *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 559–574.

[90] Robert A Hewitt et al. "The Katwijk beach planetary rover dataset". In: *The International Journal of Robotics Research* 37.1 (2018), pp. 3–12.

[91] Jeroen D Hol. "Sensor fusion and calibration of inertial sensors, vision, ultra-wideband and GPS". PhD thesis. Linköping University Electronic Press, 2011.

[92] E. Hong and J. Lim. "Visual inertial odometry using coupled nonlinear optimization". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 6879–6885.

[93] Zheng Huai and Guoquan Huang. "Robocentric Visual-Inertial Odometry". In: Oct. 2018, pp. 6319–6326.

[94] Albert S. Huang et al. "A High-Rate, Heterogeneous Data Set From The DARPA Urban Challenge". In: 29.13 (2010). ISSN: 0278-3649. URL: https://doi.org/10.1177/0278364910384295.

[95] Guoquan Huang. "Visual-Inertial Navigation: A Concise Review". In: *2019 International Conference on Robotics and Automation (ICRA)* (2019), pp. 9572–9582.

[96] Guoquan Huang, Anastasios Mourikis, and Stergios Roumeliotis. "A First-Estimates Jacobian EKF for Improving SLAM Consistency". In: vol. 54. Mar. 2009, pp. 373–382. ISBN: 978-3-642-00195-6.

[97] Jingwei Huang et al. "3Dlite: Towards Commodity 3D Scanning for Content Creation". In: 36.6 (2017). ISSN: 0730-0301. URL: https://doi.org/10.1145/3130800.3130824.

[98] Peter J. Huber. "Robust Statistics". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1248–1251. ISBN: 978-3-642-04898-2.

[99] Myung Hwangbo, Jun-Sik Kim, and Takeo Kanade. "Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and GPU implementation". In: *The International Journal of Robotics Research* 30.14 (2011), pp. 1755–1774.

[100] Myung Hwangbo, Jun-Sik Kim, and Takeo Kanade. "Inertial-aided KLT feature tracking for a moving camera". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 1909–1916.

[101] "Image registration methods: a survey". In: *Image and Vision Computing* 21.11 (2003), pp. 977 –1000. ISSN: 0262-8856.

[102] "Image Registration Techniques: A Survey". In: (). URL: http://dx.doi.org/10.31224/osf.io/rv65c.

[103] Berkeley Design Technology Inc. *Benchmark*. 2018. URL: https://www.bdti.com/ (visited on 10/03/2018).

[104] Intel. *Intel SoCs FPGA*. 2018. URL: https://www.intel.com/content/www/us/en/products/programmable/soc.html (visited on 10/03/2018).

[105] Şahin Işık. "A comparative evaluation of well-known feature detectors and descriptors". In: *International Journal of Applied Mathematics, Electronics and Computers* 3.1 (2014), pp. 1–6.

[106] ITRS. *International Technology Roadmap for semiconductors 2.0*. Tech. rep. ITRS, 2015. URL: http://www.itrs2.net/itrs-reports.html.

[107] P. Ivanov, S. Ali-Löytty, and R. Piché. "Evaluating the consistency of estimation". In: *International Conference on Localization and GNSS 2014 (ICL-GNSS 2014)*. 2014, pp. 1–5.

[108] Andrew J Davison et al. "MonoSLAM: real-time single camera SLAM". In: *IEEE transactions on pattern analysis and machine intelligence* 29 (July 2007), pp. 1052–67.

[109] S. Jashnani et al. "Sizing and preliminary hardware testing of solar powered UAV". In: *The Egyptian Journal of Remote Sensing and Space Science* 16 (Dec. 2013). DOI: 10.1016/j.ejrs.2013.05.002.

[110] Jinyong Jeong et al. "Complex urban dataset with multi-level sensors from highly diverse urban environments". In: *The International Journal of Robotics Research* 38.6 (2019), pp. 642–657.

[111] Chris Kahlefendt. "Implementation and evaluation of monocular SLAM for an underwater robot". In: *Hamburg, January* 127 (2018).

[112] Ebrahim Karami, Siva Prasad, and Mohamed S. Shehata. "Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images". In: *CoRR* abs/1710.02726 (2017).

[113] Amey Kasar. "Benchmarking and Comparing Popular Visual SLAM Algorithms". In: *Asian Journal For Convergence In Technology (AJCT)* (2019). URL: http://www.asianssr.org/index.php/ajct/article/view/757.

[114] Jonathan Kelly and Gaurav S Sukhatme. "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration". In: *The International Journal of Robotics Research* 30.1 (2011), pp. 56–79.

[115] Hanme Kim et al. "Simultaneous mosaicing and tracking with an event camera". In: *J. Solid State Circ* 43 (2008), pp. 566–576.

[116] Les Kitchen and Azriel Rosenfeld. "Gray-level corner detection". In: *Pattern recognition letters* 1.2 (1982), pp. 95–102.

[117] Georg Klein and David Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. ISMAR '07. 2007, pp. 1–10. ISBN: 978-1-4244-1749-0. URL: https://doi.org/10.1109/ISMAR.2007.4538852.

[118] Stanisław Konatowski, Piotr Kaniewski, and Jan Matuszewski. "Comparison of Estimation Accuracy of EKF, UKF and PF Filters". In: *Annual of Navigation* 23.1 (2016), pp. 69 –87. URL: https://content.sciendo.com/view/journals/aon/23/1/article-p69.xml.

[119] Dimitrios G. Kottas et al. "On the Consistency of Vision-Aided Inertial Navigation". In: *Experimental Robotics: The 13th International Symposium on Experimental Robotics*. Ed. by Jaydev P. Desai et al. Heidelberg: Springer International Publishing, 2013, pp. 303–317. ISBN: 978-3-319-00065-7.

[120] Dimitrios G. Kottas et al. "On the Consistency of Vision-Aided Inertial Navigation". In: *Experimental Robotics: The 13th International Symposium on Experimental Robotics*. Ed. by Jaydev P. Desai et al. Heidelberg: Springer International Publishing, 2013, pp. 303–317. ISBN: 978-3-319-00065-7.

[121] Haeseong Lee et al. "Recent advances in feature detectors and descriptors". In: *IEIE Transactions on Smart Processing & Computing* 5.3 (2016), pp. 153–163.

[122] Keith Leung et al. "Chilean underground mine dataset". In: *The International Journal of Robotics Research* 36 (Jan. 2017), pp. 16–23.

[123] Keith Yu Kit Leung et al. "The UTIAS multi-robot cooperative localization and mapping dataset". In: *The International Journal of Robotics Research* 30 (2011), pp. 969 –974.

[124] S. Leutenegger et al. *Keyframe-based visual-inertial odometry using nonlinear optimization*. 2015.

[125] M. Li and A. I. Mourikis. "Improving the accuracy of EKF-based visual-inertial odometry". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 828–835.

[126] Mingyang Li and Anastasios I. Mourikis. "High-precision, consistent EKF-based visual-inertial odometry". In: *The International Journal of Robotics Research* 32.6 (2013), pp. 690–711.

[127] Rui Liao et al. "An Artificial Agent for Robust Image Registration". In: *arXiv preprint arXiv: 1611.10336* (Nov. 2016).

[128] P. Lichtsteiner, C. Posch, and T. Delbruck. "A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor". In: *IEEE Journal of Solid-State Circuits* 43.2 (2008), pp. 566–576.

[129] Shih-Chii Liu and Tobi Delbruck. "Neuromorphic sensory systems. Curr Opin Neurobio". In: *Current opinion in neurobiology* 20 (June 2010), pp. 288–95.

[130] G. Loianno et al. "Estimation, Control, and Planning for Aggressive Flight With a Small Quadrotor With a Single Camera and IMU". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 404–411.

[131] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[132] David G Lowe. "Object recognition from local scale-invariant features". In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[133] Yuncheng Lu et al. "A survey on vision-based UAV navigation". In: *Geo-spatial Information Science* 21.1 (2018), pp. 21–32.

[134] J. Ma et al. "Image Matching from Handcrafted to Deep Features: A Survey". In: *International Journal of Computer Vision* (2020), pp. 1 –57.

[135] Kevin Y. Ma et al. "Controlled Flight of a Biologically Inspired, Insect-Scale Robot". In: 340.6132 (2013), pp. 603–607.

[136] Will Maddern et al. "1 year, 1000 km: The Oxford RobotCar dataset". In: *The International Journal of Robotics Research* 36.1 (2017), pp. 3–15.

[137] András L Majdik, Charles Till, and Davide Scaramuzza. "The Zurich Urban Micro Aerial Vehicle Dataset". In: 36.3 (2017). ISSN: 0278-3649. URL: https://doi.org/10.1177/0278364917702237.

[138] Eitan Marder-Eppstein. "Project Tango". In: *ACM SIGGRAPH 2016 Real-Time Live!* SIGGRAPH '16. Anaheim, California, 2016, 40:25–40:25. ISBN: 978-1-4503-4378-7. URL: http://doi.acm.org/10.1145/2933540.2933550.

[139] Gellert Mattyus et al. "Enhancing Road Maps by Parsing Aerial Images Around the World". In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. ICCV '15. 2015, pp. 1689–1697. ISBN: 978-1-4673-8391-2. URL: http://dx.doi.org/10.1109/ICCV.2015.197.

[140] How to Mechatronics. *MEMS Accelerometer Gyroscope Magnetometer Arduino*. 2021. URL: https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/.

[141] Medium. *7 Connected Car Trends Fueling the Future*. 2018. URL: https://www.https://www.globalxetfs.com/how-can-5g-accelerate-iot/ (visited on 03/07/2018).

[142] Moritz Menze and Andreas Geiger. "Object scene flow for autonomous vehicles". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. June 2015, pp. 3061–3070.

[143] Matt Miesnieks. *How is ARCore better than ARKit?* 2017. URL: https://medium.com/6d-ai/how-is-arcore-better-than-arkit-5223e6b3e79d (visited on 01/09/2017).

[144] Krystian Mikolajczyk and Cordelia Schmid. "Indexing based on scale invariant interest points". In: *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*. Vol. 1. IEEE. 2001, pp. 525–531.

[145] Krystian Mikolajczyk and Cordelia Schmid. "Scale & affine invariant interest point detectors". In: *International journal of computer vision* 60.1 (2004), pp. 63–86.

[146] S. A. S. Mohamed et al. "A Survey on Odometry for Autonomous Navigation Systems". In: *IEEE Access* 7 (2019), pp. 97466–97486.

[147] J. M. M. Montiel, Javier Civera, and A. Davison. "Unified Inverse Depth Parametrization for Monocular SLAM". In: *Robotics: Science and Systems*. 2006.

[148] Hans P Moravec. "Towards automatic visual bbstacle avoidance". In: *International Conference on Artificial Intelligence (5th: 1977: Massachusetts Institute of Technology)*. 1977.

[149] Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. "Accurate Non-Iterative O(n) Solution to the PnP Problem". In: *IEEE International Conference on Computer Vision* (Jan. 2007).

[150] Anastasios I. Mourikis and Stergios I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation* (2007), pp. 3565–3572.

[151] E. Mueggler et al. "Continuous-Time Visual-Inertial Odometry for Event Cameras". In: *IEEE Transactions on Robotics* 34.6 (2018), pp. 1425–1440.

[152] Elias Mueggler et al. "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM". In: *The International Journal of Robotics Research* 36.2 (2017), pp. 142–149.

[153] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[154] Raul Mur-Artal and Juan D. Tardós. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In: *IEEE Transactions on Robotics* 33 (2017), pp. 1255–1262.

[155] Raúl Mur-Artal and Juan D Tardós. "Visual-inertial monocular SLAM with map reuse". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 796–803.

[156] 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.

[157] Duc-Hoang Nguyen, Minh-Triet Tran, and Vinh-Tiep Nguyen. "Comics Instance Search with Bag of Visual Words". In: *International Conference on Future Data and Security Engineering*. Springer. 2015, pp. 299–313.

[158] Janosch Nikolic et al. "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 431–437.

[159] David Nistér, Oleg Naroditsky, and James Bergen. "Visual Odometry". In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR'04)*. 2004.

[160] Hyeonwoo Noh et al. "Large-Scale Image Retrieval with Attentive Deep Local Features". In: Oct. 2017, pp. 3476–3485.

[161] *NVIDIA DRIVE SoC*. 2018. URL: https://hardware.developpez.com/actu/188558/NVIDIA-DRIVE-detaille-son-SoC-Xavier-qui-rassemble-un-processeur-ARM-a-huit-coeurs-et-un-processeur-graphique-Volta/ (visited on 2018).

[162] Hanna Nyqvist and Fredrik Gustafsson. "A high-performance tracking system based on camera and IMU". In: *Information Fusion (FUSION), 2013 16th International Conference on*. IEEE. 2013, pp. 2065–2072.

[163] N. O'Mahony et al. "Adaptive Multimodal Localisation Techniques for Mobile Robots in Unstructured Environments : A Review". In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. 2019, pp. 799–804.

[164] PM Panchal, SR Panchal, and SK Shah. "A comparison of SIFT and SURF". In: *International Journal of Innovative Research in Computer and Communication Engineering* 1.2 (2013), pp. 323–327.

[165] Gaurav Pandey, James McBride, and Ryan Eustice. "Ford Campus vision and lidar data set". In: *I. J. Robotic Res.* 30 (Oct. 2011), pp. 1543–1552.

[166] F. Pang et al. "Depth enhanced visual-inertial odometry based on Multi-State Constraint Kalman Filter". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 1761–1767.

[167] Thierry Peynot, Steve Scheding, and Sami Terho. "The Marulan Data Sets: Multi-Sensor Perception in a Natural Environment with Challenging Conditions". In: 29.13 (2010). ISSN: 0278-3649. URL: https://doi.org/10.1177/0278364910384638.

[168] Jin-Chun Piao and Shin-Dug Kim. "Adaptive Monocular Visual–Inertial SLAM for Real-Time Augmented Reality Applications in Mobile Devices". In: *Sensors* 17 (Nov. 2017), p. 2567.

[169] Jonathan Piat et al. "HW/SW co-design of a visual SLAM application". In: *Journal of Real-Time Image Processing* (Nov. 2018).

[170] Taihú Pire et al. "The Rosario dataset: Multisensor data for localization and mapping in agricultural environments". In: *The International Journal of Robotics Research* 38.6 (2019), pp. 633–641.

[171] "Place Recognition using Near and Far Visual Information". In: *IFAC Proceedings Volumes* 44.1 (2011). 18th IFAC World Congress, pp. 6822 –6828. ISSN: 1474-6670.

[172] W. K. Pratt. "Correlation Techniques of Image Registration". In: *IEEE Transactions on Aerospace and Electronic Systems* AES-10.3 (1974), pp. 353–358.

[173] INSPEX H2020 Project. *INSPEX- Integrated Smart Spatial Exploration System*. 2018. URL: http://www.inspex-ssi.eu/.

[174] Charles R. Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, 5105–5114. ISBN: 9781510860964.

[175] T. Qin, P. Li, and S. Shen. "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[176] *Qualcomm Hexagon DSP: An architecture optimized for mobile multimedia and communications*. Qualcomm Technologies, Inc., 2017.

[177] M. Quan et al. "Accurate Monocular Visual-Inertial SLAM Using a Map-Assisted EKF Approach". In: *IEEE Access* 7 (2019), pp. 34289–34300.

[178] Meixiang Quan et al. "Map-Based Visual-Inertial Monocular SLAM using Inertial assisted Kalman Filter". In: (Sept. 2017).

[179] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[180] Patrick Rives. "Visual servoing based on epipolar geometry". In: *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*. Vol. 1. IEEE. 2000, pp. 602–607.

[181] Robotics and Perception Group. *Rpg trajectory evaluation online tool*. 2018. URL: https://github.com/uzh-rpg/rpg_trajectory_evaluation.

[182] Edward Rosten and Tom Drummond. "Machine learning for high-speed corner detection". In: *Computer Vision–ECCV 2006* (2006), pp. 430–443.

[183] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2564–2571.

[184] J.R. Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. "Robot@Home, a robotic dataset for semantic mapping of home environments". In: *The International Journal of Robotics Research* (Mar. 2017), p. 027836491769564.

[185] Ludovico Russo et al. "A ROS Implementation of the Mono-Slam Algorithm". In: vol. 4. Jan. 2014, pp. 339–351. ISBN: 9781921987243.

[186] M. Salas et al. "Trajectory alignment and evaluation in SLAM: Horn's method vs alignment on the manifold". In: Robotics: Science and Systems, 2015.

[187] Imane Salhi et al. "Suivi d'objet par capteurs visuels et inertiels sur systèmes embarqués". In: *Conférence Française de Photogrammétrie et de Télédétection (CFPT 2018)*. Champs-sur-Marne, France, 2018. URL: https://hal.archives-ouvertes.fr/hal-02338377.

[188] Kaz Sato, Cliff Young, and David Patterson. "An in-depth look at Google's first Tensor Processing Unit (TPU)". In: *Google Cloud Big Data and Machine Learning Blog* 12 (2017).

[189] Torsten Sattler et al. "Benchmarking 6DOF Urban Visual Localization in Changing Conditions". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018*. 2018.

[190] D. Scaramuzza and F. Fraundorfer. "Visual Odometry [Tutorial]". In: *IEEE Robotics Automation Magazine* 18.4 (2011), pp. 80–92.

[191] Adam Schmidt, Marek Kraft, and Andrzej Kasiński. "An evaluation of image feature detectors and descriptors for robot navigation". In: *Computer Vision and Graphics* (2010), pp. 251–259.

[192] bosch sensortec. *Smart sensor: BNO055*. 2021. URL: https://www.bosch-sensortec.com/products/smart-sensors/bno055.html#order.

[193] Shaojie Shen, Nathan Michael, and Vijay Kumar. "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft MAVs". In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 5303–5310.

[194] Jamie Shotton et al. "Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images". In: June 2013, pp. 2930–2937.

[195] Martin Simonovsky et al. "A Deep Metric for Multimodal Registration". In: *19th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2016)*. Ed. by Sebastien Ourselin et al. Vol. 9902. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016. Athènes, Greece: Springer, Oct. 2016, pp. 10–18. URL: https://hal.archives-ouvertes.fr/hal-01576914.

[196] Mike Smith et al. "The New College Vision and Laser Data Set". In: *The International Journal of Robotics Research* 28.5 (2009), pp. 595–599.

[197] Joan Solà. "Consistency of the monocular EKF-SLAM algorithm for 3 different landmark parametrizations". In: *IEEE International Conference on Robotics and Automation (ICRA) 2010*. 6 pages. Anchorage, United States, May 2010, pp.3513–3518. URL: https://hal.archives-ouvertes.fr/hal-00420054.

[198] Ariane Spaenlehauer et al. "A Loosely-Coupled Approach for Metric Scale Estimation in Monocular Vision-Inertial Systems". In: *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2017)*. Nov. 2017, pp. 137–143. URL: https://hal.archives-ouvertes.fr/hal-01678915.

[199] Carol Spoon. *San Francisco Politics Books*. Version V1. 2019.

[200] Hauke Strasdat, José MM Montiel, and Andrew J Davison. "Visual SLAM: why filter?" In: *Image and Vision Computing* 30.2 (2012), pp. 65–77.

[201] Jrgen Sturm et al. "A benchmark for the evaluation of RGB-D SLAM systems". In: Oct. 2012, pp. 573–580. ISBN: 978-1-4673-1737-5.

[202] Jae Kyu Suhr. "Kanade-Lucas-Tomasi (KLT) feature tracker". In: *Computer Vision (EEE6503)* (2009), pp. 9–18.

[203] A. Suleiman et al. "Navion: A 2-mW Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones". In: *IEEE Journal of Solid-State Circuits* 54.4 (2019), pp. 1106–1119.

[204] A. Suleiman et al. "Navion: A Fully Integrated Energy-Efficient Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones". In: *2018 IEEE Symposium on VLSI Circuits*. 2018, pp. 133–134.

[205] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. "Visual SLAM algorithms: a survey from 2010 to 2016". In: *IPSJ Transactions on Computer Vision and Applications* 9 (Dec. 2017).

[206] S. A. K. Tareen and Z. Saleem. "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK". In: *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. 2018, pp. 1–10.

[207] Keisuke Tateno et al. "Cnn-slam: Real-time dense monocular slam with learned depth prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6243–6252.

[208] "The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception". In: 3 (). ISSN: 2377-3774. URL: http://dx.doi.org/10.1109/LRA.2018.2800793.

[209] Engin Tola, Vincent Lepetit, and Pascal Fua. "A fast local descriptor for dense matching". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[210] Bill Triggs et al. "Bundle Adjustment — A Modern Synthesis". In: *Vision Algorithms: Theory and Practice*. Ed. by Bill Triggs, Andrew Zisserman, and Richard Szeliski. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 298–372. ISBN: 978-3-540-44480-0.

[211] S. Ullah, B. Song, and W. Chen. "EMoVI-SLAM: Embedded Monocular Visual Inertial SLAM with Scale Update for Large Scale Mapping and Localization". In: *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 2018, pp. 880–885.

[212] S. Umeyama. "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 13 (1991), pp. 376–380.

[213] *VSlam Evaluation*. 2018. URL: https://github.com/nicolov/vslam_evaluation (visited on 2017).

[214] Joseph Walsh et al. "Deep Learning vs. Traditional Computer Vision". In: Apr. 2019. ISBN: 978-981-13-6209-5.

[215] R. Wang, M. Schwörer, and D. Cremers. "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3923–3931.

[216] Shenlong Wang et al. "TorontoCity: Seeing the World with a Million Eyes". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3028–3036.

[217] Andrew Waterman et al. *The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA*. Tech. rep. UCB/EECS-2011-62. EECS Department, University of California, Berkeley, 2011. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-62.html.

[218] David Weikersdorfer et al. "Event-based 3D SLAM with a depth-augmented dynamic vision sensor". In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 359–364.

[219] Simon Winder, Gang Hua, and Matthew Brown. "Picking the best daisy". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 178–185.

[220] Robert J. Wood et al. "Progress on "Pico" Air Vehicles". In: *Robotics Research : The 15th International Symposium ISRR*. Ed. by Henrik I. Christensen and Oussama Khatib. Cham: Springer International Publishing, 2017, pp. 3–19. ISBN: 978-3-319-29363-9.

[221] Oliver J. Woodman. "An introduction to inertial navigation". In: 2007.

[222] Xilinx. *Zynq UltraScale+ MPSoC*. Website. 2020. URL: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html.

[223] Xiao Xin Lu. "A Review of Solutions for Perspective-n-Point Problem in Camera Pose Estimation". In: *Journal of Physics: Conference Series* 1087 (Sept. 2018), p. 052009.

[224] De Xu, M. Tan, and Gang Chen. "An Improved Dead Reckoning Method for Mobile Robot with Redundant Odometry Information". In: Jan. 2003, 631 –636 vol.2. ISBN: 981-04-8364-3. DOI: 10.1109/ICARCV.2002.1238497.

[225] Xufeng Han et al. "MatchNet: Unifying feature and metric learning for patch-based matching". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3279–3286.

[226] Shao-Wen Yang, Chieh-Chih Wang, and Charles Thorpe. "The annotated laser data set for navigation in urban areas". In: *I. J. Robotic Res.* 30 (Aug. 2011), pp. 1095–1099.

[227] Georges Younes et al. "Keyframe-based monocular SLAM: Design, survey, and future directions". In: *Robotics and Autonomous Systems* 98 (Sept. 2017).

[228] Chaofan Zhang et al. "VINS-MKF: A Tightly-Coupled Multi-Keyframe Visual-Inertial Odometry for Accurate and Robust State Estimation". In: *Sensors* 18 (Nov. 2018), p. 4036.

[229] Mengde Zhang et al. "Comparison of Kalman Filters for Inertial Integrated Navigation". In: *Sensors* 19 (Mar. 2019).

[230] 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.

[231] Zhe Zhang et al. "PIRVS: An Advanced Visual-Inertial SLAM System with Flexible Sensor Fusion and Hardware Co-Design". In: (May 2018), pp. 1–7.

[232] Zhengdong Zhang et al. "Visual-Inertial Odometry on Chip: An Algorithm-and-Hardware Co-design Approach". In: *Robotics: Science and Systems*. 2017.

[233] Zhengyou Zhang. "Microsoft kinect sensor and its effect". In: *IEEE multimedia* 19.2 (2012), pp. 4–10.

[234] Zichao Zhang and Davide Scaramuzza. *Rethinking Trajectory Evaluation for SLAM: a Probabilistic, Continuous-Time Approach*. June 2019.

[235] Feng Zheng et al. "Trifo-VIO: Robust and Efficient Stereo Visual Inertial Odometry Using Points and Lines". In: Oct. 2018, pp. 3686–3693.

[236] G. Zhou et al. "On-board inertial-assisted visual odometer on an embedded system". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 2602–2608.

[237] Guyue Zhou et al. "On-board inertial-assisted visual odometer on an embedded system". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 2602–2608.

[238] A. Z. Zhu, N. Atanasov, and K. Daniilidis. "Event-Based Visual Inertial Odometry". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5816–5824.