Contribution au développement d'un casque " see-through " de réalité augmentée : fusion de données pour une meilleure localisation 3D

Marwene Kechiche

N° d'ordre NNT : 2020LYSEE006

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée conjointement
au sein de l'Ecole centrale de Lyon
et de l'Ecole Nationale d'Ingénieurs de Saint-Etienne

Ecole Doctorale N° 488
Sciences et génie de l'environnement

Spécialité
Informatique et Ingénierie

Soutenue publiquement le 19/11/2020, par :
Marwene KECHICHE

Contribution au développement d'un casque « see-through » de réalité augmentée : fusion de données pour une meilleure localisation 3D

Devant le jury composé de :

Querrec, Ronan	Professeur des universités	ENIB, Brest	Président
Derrode, Stéphane	Professeur des universités	École Centrale, LIRIS, Lyon	Rapporteur
Labbani-Igbida, Ouiddad	Professeur des universités	XLIM, Limoges	Rapporteuse
Gonon, Xavier	Ingénieur	Thalès	Examinateur
Toscano, Rosario	Professeur des universités	ENISE-LTDS	Directeur de thèse
Fortunier, Roland	Professeur des universités	Isae-Ensma	Co-directeur
Ivan, Ioan-Alexandru	Maître de conférences	ENISE-LTDS	Encadrant
Baert, Patrick	Ingénieur de recherche	ENISE	Invité

Liste de publications :

Avant la thèse :

- Forces characterization from trajectory equations in virtual reality simulations for industrial applications
  Date de publication Août 2016
- Serious game framework confronted with specificities of industrial training : application to steel industry
  Date de publication : Octobre 2015
- Using Haptic Forces Feedback for Immersive and Interactive Simulation in Industrial Context
  Date de publication : Août 2015

Pendant la thèse :

- Titre :A New Loose-Coupling Method for Vision-Inertial Systems Based on Retro-Correction and Inconsistency Treatment
  Date de publication : juillet 2019
- Titre : Visuo-Haptic virtual exploration of single cell morphology and mechanics based on AFM mapping in fast mode
  Date de publication : octobre 2020
- Titre : Characterization of micro/nano-rheology properties of soft and biological matter combined with a virtual reality haptic exploration
  Date de publication : juillet 2019

Remerciement

Je voudrais dans un premier temps remercier mon directeur de thèse, monsieur TOSCANO Rosario pour sa patience, ses conseils, sa disponibilité et son encadrement bienveillant.

Je remercie aussi Monsieur FORTUNIER Roland co-directeur de ma thèse pour ses conseils et son encadrement scientifique et professionnel. Un très grand merci pour ses encouragements dans un contexte particulièrement compliqué.

Je remercie encore et évidemment Monsieur IVAN Ioan-Alexandru, mon co-encadrant de thèse pour son accompagnement, encadrement, ses encouragements, conseils, et surtout sa disponibilité durant la thèse.

Monsieur BAERT Patrick, responsable de la plateforme réalité virtuelle et chef de projet REVE5D pour l'ENISE, je te remercie pour ta disponibilité, ton organisation, tes conseils, ton aide et surtout ton soutient moral dans les périodes les plus difficiles.

Je tiens à remercier l'ensemble de partenaires de ce projet REVE5D : Thalès Angénieux (Raphael GOUVERNEUR), Institut Pascal (Eric ROYER, Marc CHEVALDONNEE, Gautier CLAISSE) , la société SFI (Catherine BOUCQUET, Gérard FAYOLLE, Sylvain MONTANA), la société Nexter Systems (Paul BARA), et enfin la société Vapérail.

Je remercie aussi toute l'équipe pédagogique de l'école nationale d'ingénieurs de Saint-Étienne ainsi que les responsables administratifs.

Je dédie ce modeste travail à :

Ma mère et mon père

Ma sœur et mon frère

Ma grande famille

Mes amis

LISTE DES SYMBOLES

AF Ajustement de Faisceaux

CI Centrale Inertielle

CPU Central Process Unity

DT-SLAM Deferred Triangulation for Robust SLAM

FAIA Forward Additive Image Alignement

FC Filtre Complémentaire

FP Filtre particulaire ou Filtr à Particules

FPGA Field Programmable Gate Arrays

GPU Graphic Process Unit

IP Institut Blaise Pascal

LSD-SLAM Large-Scale Direct Monocular SLAM

MCSLAM* Multiple Constrained SLAM

MTP Motion To Photon

MTPL Motion To Photon Latency

ORB Oreinted FAST and Rotated BRIEF

ORB-SLAM ce qu'elle signifie

OST Optical See-Through

PTAM Parallel Tracking And Mapping

RA Réalité augmentée

RD-SLAM Rutgers Distributed SLAM

REVE5D Réalité Augmentée pour la Culture, les Infrastructures et l'Industrie

RK-SLAM Robust Keyframe-based Monocular SLAM for AR

RV	Réalité Virtuelle
SFM	Structure From Motion
SIFT	Scale Invariant Feature Transform
SIR	Sequential Importance Resampling
SIS	Sequential importance sampling
SLAM	Simultaneous Localization and Mapping
SURF	Speeded Up Robust Features
VST	Vidéo See-Through

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

CONTEXTE DE L'ÉTUDE

La réalité augmentée a beaucoup évolué durant ces deux dernières décennies. Cette évolution est arrivée avec celle des ressources informatiques, qui ont permis l'explosion de la puissance de calcul (processeur et cartes graphiques). Les petits supports comme les téléphones portables, les tablettes, ou les cartes personnalisées permettent l'exécution d'algorithmes qui ont une complexité conséquente dans un temps relativement court. Cette évolution a facilité l'intégration et l'exploitation de la réalité augmentée dans plusieurs secteurs et domaines.

On trouve dans la littérature plusieurs définitions de la réalité augmentée, afin de bien cerner le concept, nous commencerons par une définition étymologique, ensuite une définition scientifique, et une définition fonctionnelle.

Réalité : Caractère de ce qui existe en fait (et qui n'est pas seulement une invention, une illusion ou une apparence) que l'on résumera comme l'environnement concret et matériel de l'homme.

Augmentée : Rendre plus grand, développer, étendre, par addition d'une chose de même nature.

En combinant dans l'ordre les deux définitions de réalité et augmentée, on obtient une première interprétation qui est très proche du concept de la réalité augmentée. C'est la définition étymologique.

La première utilisation scientifique du terme réalité augmentée était en 1992 par deux ingénieurs aéronautiques. L'objectif de cette appellation désignait la superposition entre des informations informatiques de synthèse et le monde réel. Une proposition de définition a été faite par Milgram et al. [71] qui définit la réalité augmentée comme un continuum entre le monde réel et le monde virtuel (Figure. 1).

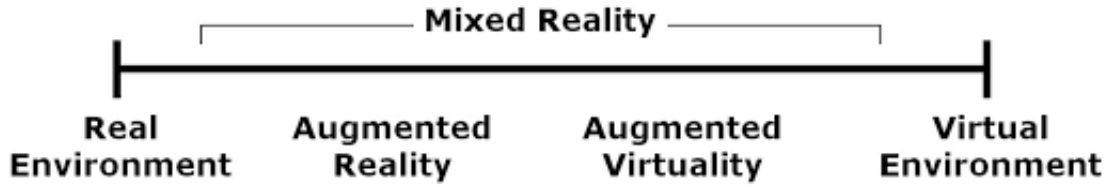


FIGURE 1 – Continuum de Milgram pour classifier la réalité augmentée [71]

Un peu plus tard Azuma [5] a complété cette définition par : Le système de réalité augmentée est un système qui complète (augmente) le monde réel par des objets virtuels (générés d'une manière synthétique). Les mondes réel et le virtuel sont dans le même espace et ils sont perçus comme un seul élément. Le système de réalité augmentée doit vérifier les trois contraintes (Figure. 2).

— Associer des objets réels avec des projections virtuelles dans l'environnement réel
— Assurer l'interaction en temps réel
— Recaler les objets réels et virtuels d'une manière précise.

FIGURE 2 – Exemple d'utilisation de la réalité augmentée pour superposer des informations virtuelles sur un monde réel

Le concept de la réalité augmentée a donc été développé initialement par les chercheurs aéronautiques. La première exploitation de ce concept fut le sujet d'une simulation avec un affichage de synthèse. Depuis sa première utilisation, la réalité augmentée est appliquée dans d'autres secteurs, tels que la culture, la santé, et l'industrie.

L'évolution des technologies a permis aussi l'évolution de dispositifs de réalité augmentée. Parmi les outils de réalité augmentée qui sont disponibles sur le marché on peut citer :

-Le système d'affichage à tête haute qui consiste à projeter les informations directement dans le monde réel. Une projection d'informations dans un cockpit d'un avion permet aux pilotes d'avoir une assistance et les mains libres. Ce système est utilisé aussi dans les voitures pour projeter les informations de navigation sur le pare-brise. Il permet aux conducteurs de conduire d'une manière sereine et voir les informations de vitesse, indication GPS, etc...

FIGURE 3 – Système de visualisation de réalité augmentée a tête haute (HUD)

- Les systèmes montés sur la tête, tels que les lunettes, les casques see-through vidéo, les casques see-through optique (Figure. 4). Le principe de fonctionnement d'un système see-through optique consiste à filmer avec une caméra l'environnement, calculer les augmentations en fonction de la

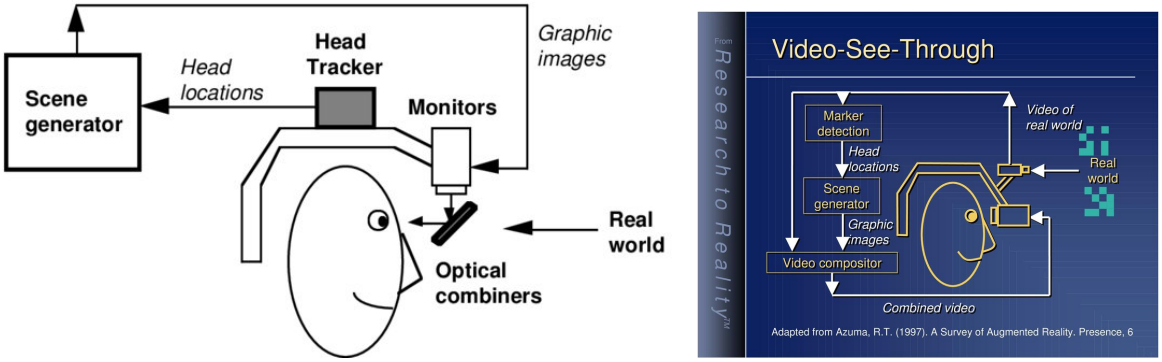


FIGURE 4 – Deux casques de réalité augmentée : (gauche) casque optique où on voit le monde réel directement à travers les vitres optiques, (droite) casque vidéo où on voit le monde réel à travers une caméra vidéo

position de la tête, et finalement afficher sur le casque les augmentations uniquement (Figure. 4 (gauche)). En revanche, le système see-through vidéo (Figure. 4 (droite)) se base sur les images d'une caméra externe qui filme l'environnement réel. L'environnement réel et les augmentations sont projetés sur le casque directement. L'opérateur n'a pas d'interaction directe via l'environnement réel, il passe systématiquement par la caméra.

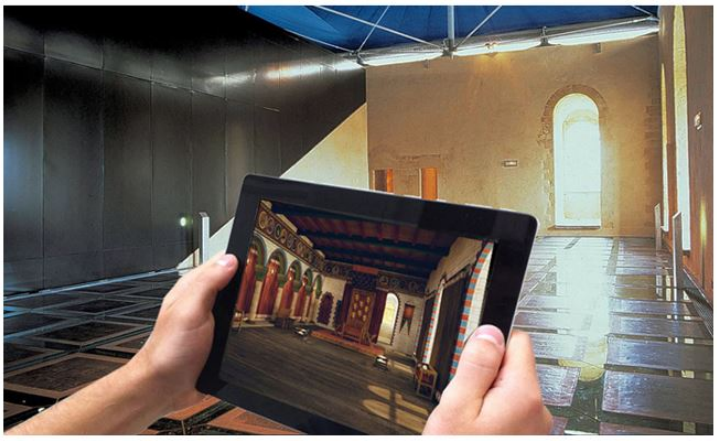


FIGURE 5 – Système de visualisation de réalité augmentée portable

- Les dispositifs portables comme les téléphones, tablettes ou ordinateurs (Figure. 5). Les dispositifs présentés précédemment permettent l'exploitation de la réalité augmentée de manières différentes. Malgré la diversité de dispositifs, le processus reste identique. Une application de réalité augmentée est structurée de la manière suivante :

— L'acquisition de données des différents capteurs pour la localisation (localisation)

— Le calcul de rendu en fonction de la localisation dans l'environnement réel (rendu)

— L'affichage de différentes augmentations (projection)

La navigation ou l'évolution dans le monde réel, consiste à pouvoir se positionner dans l'environnement réel d'une manière continue. Les algorithmes d'estimation de la position et l'orientation

permettent d'avoir ces informations. Le couple position et orientation forme une pose. Une pose comprend six degrés de libertés, les trois composantes x,y, et z de la translation et les trois angles de rotations.

Le calcul d'affichage est l'étape finale dans l'application de réalité augmentée. Après la collecte de toutes les informations nécessaires, notamment la pose. L'application change la position de la caméra virtuelle dans la scène 3D et calcule le rendu final qui représente l'image à projeter sur le périphérique d'affichage final.

Les trois étapes décrites permettent l'affichage d'augmentations 3D dans/sur un dispositif de réalité augmentée. Cependant, le temps de traitement de chaque partie implique une latence de traitement. Le système est dit latent lorsque les augmentations affichées ne sont pas cohérentes avec l'environnement réel. La non cohérence de l'affichage gêne l'utilisateur et génère des effets secondaires comme le mal de mer. Les effets de la latence ont un impact aussi sur l'acceptabilité d'outils. En d'autres termes si le dispositif de réalité augmentée est latent l'utilisateur abandonnera l'expérience rapidement.

Le projet **ARCHEOUIDE** (Figure. 6) [100] est un projet européen, achevé en 2003, qui vise à développer un système de reconstruction en réalité augmentée de ruines antiques, sur la base de la position de l'utilisateur et l'orientation de sa tête dans le site culturel. L'image de rendu est calculée en temps réel. Il intègre une base de données multimédia du matériel culturel pour l'accès en ligne aux données culturelles, visites virtuelles, et des informations de restauration. Il est basé sur des interfaces-utilisateurs multimodales et personnalise le flux d'informations au profit de son utilisateur afin de répondre aux besoins des utilisateurs professionnels et de loisirs, et pour des applications allant de la recherche archéologique, à l'éducation, l'édition multimédia, et le tourisme culturel.

FIGURE 6 – Présentation du projet ARCHEOGUIDE [100]

Le projet **CAMONDO** [35] d'application mobile réalisé en partenariat avec Les Arts Décoratifs pour le Musée Nissim de Camondo à Paris, à l'occasion de la Nuit Européenne des Musées 2013. Les Arts Décoratifs, gestionnaire du Musée Camondo, ont voulu marquer l'édition 2013 de la Nuit Européenne des Musées avec une expérience sensorielle innovante. Il a été créé à cette occasion une application mobile sur Android et iPhone qui permet aux visiteurs de vivre une expérience immersive sur leur smartphone, au Musée Nissim de Camondo. En se rendant dans le Cabinet des Porcelaines du Musée et en scannant certains ornements d'oiseaux présents sur le service de table Buffon, les visiteurs peuvent faire revivre ces oiseaux, directement sur leur smartphone.

Le projet **STN** (Figure. 7) [94] avait pour objet de développer un prototype de réalité augmentée ainsi que des contenus afin d'évaluer l'apport de cette technologie pour les visites culturelles. Ce projet a permis de valider le concept et de pointer les points critiques qui servent de point d'entrée à ce travail.

Le projet industriel **ARMAR** [41] vise à définir le design et le développement de systèmes de réalité augmentée expérimentale pour la maintenance et l'assistance. L'objectif a été d'étudier et

FIGURE 7 – Projet STN

d'évaluer la faisabilité de développement d'un prototype de système de réalité augmentée adaptatif qui permet d'augmenter la productivité de personnels de la maintenance. Ce système est valable pour la formation et sur le terrain (inSitu) (Figure. 8).

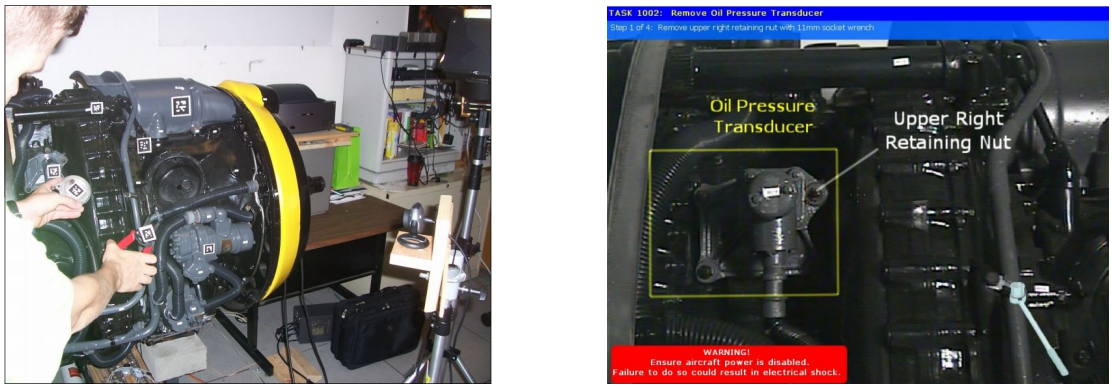


FIGURE 8 – Projet ARMAR [41]

Le projet **SIFORAS** [76] était centré sur l'utilisation de la réalité augmentée avec des dispositifs portables (tablettes) dans la réalisation de procédures de maintenance. Ce projet regroupait plusieurs industriels intéressés par la technologie. Il a permis de mettre en évidence la nécessaire légèreté de l'outil de visualisation ainsi qu'un mode d'interaction adapté au contexte d'utilisation. Les résultats suggèrent entre autre que le dispositif mobile doit peser moins de 500 grammes répartis sur l'ensemble du corps de l'utilisateur, la tablette tactile étant préconisée pour la consultation de documents textuels avec la possibilité d'annotation par stylet et les visiocasques sont à favoriser pour des utilisateurs devant garder les mains libres.

Les projets médicaux [39, 48, 98] s'intéressent à l'affichage des augmentations sur des cibles déformables. Pendant les opérations chirurgicales, la forme des tissus, organes, et les veines sont modifiables vu leurs caractéristiques élastiques. Les augmentations s'affichent en fonction du maillage déformable. Les données pré-opératoires sont utilisées pour faire les augmentations et superposer les organes, tumeurs et veines virtuels sur les organes réels (Figure. 9).

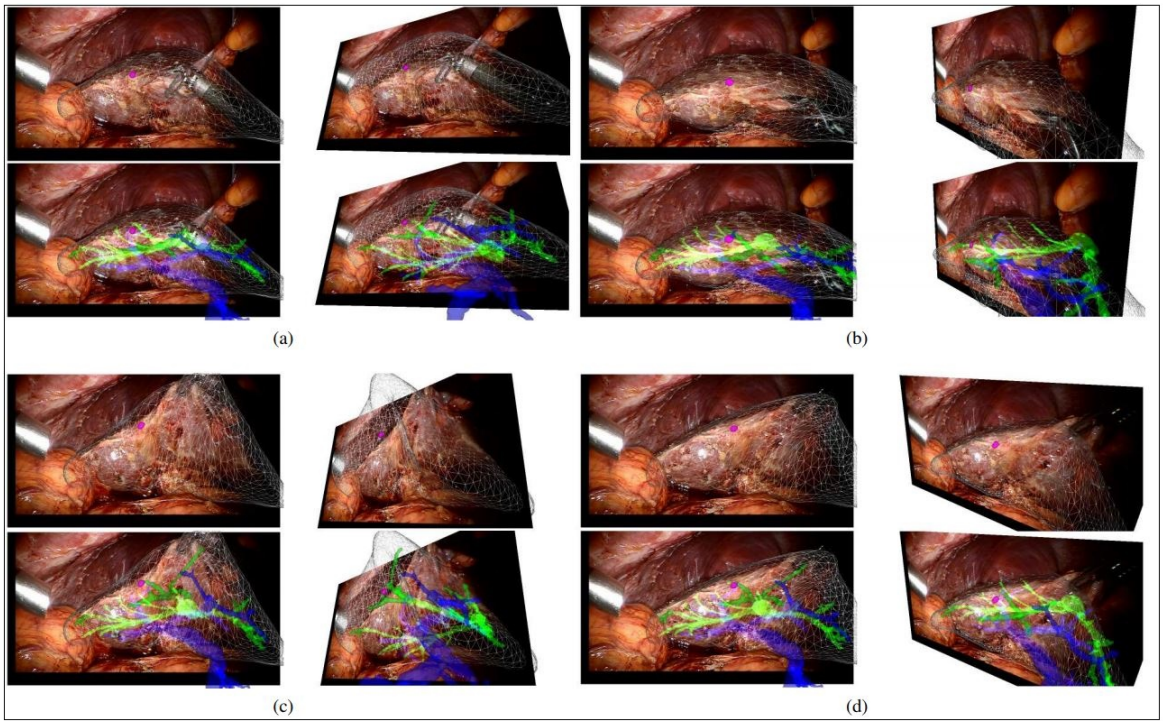


FIGURE 9 – Une séquence d'images pour présenter la superposition d'un modèle 3D sur un modèle réel déformable. Le foie est représenté en fil de fer, la tumeur en violet, le foie la veine est représentée en bleu et la veine porte en vert. [39]

Le travail de thèse, présenté dans ce document s'inscrit dans le cadre du projet **REVE5D**. Ce projet consiste à développer un casque de réalité augmentée portable sur la Tête. Le casque visé et de type 'optical see-throught' et destiné pour les applications **Culturelle et industrielles**. **REVE5D** s'agit de concevoir, créer, et tester un casque de réalité augmentée à bas coût pour le rendre accessible aux professionnels et aux particuliers (grand public). La robustesse du système et la stabilité de traitement représentent aussi des éléments importants dans le projet.

Le projet **REVE5D** est une collaboration entre différents acteurs (Figure. 10) afin de développer les trois grandes parties identifiées dans le projet.

— La partie mécanique du système

— La partie affichage, électronique, et câblage

— La partie logicielle pour la localisation et la fusion de capteurs

La thèse s'adosse au développement de cette dernière partie. Il s'agit d'exécuter les applications de réalité augmentée en temps réel avec un confort d'utilisation d'opérateur dans un intervalle de quinze minutes à une heure, durées minimales et maximales requises pour l'exploitation d'une application de réalité augmentée en évitant les effets secondaires comme les maux de tête et les vertiges ou les troubles oculaires... Deux scénarii et scènes de test on été développées en utilisant le moteur 3D Unity :

— Application pour une visite culturelle au musée

— Application d'assemblage d'un faisceau de câbles dans un véhicule spécifique.

FIGURE 10 – Les partenaires REVE5D

Les challenges du projet REVE5D sont donc basés sur la fiabilité et sur la pertinence de la solution (pendant la durée d'utilisation mentionnée). Cela nécessite une bonne localisation 3D (précise) pour pouvoir localiser l'utilisateur dans le monde réel et recaler les objets correctement dans l'environnement virtuel.

Le temps réel et l'interactivité du système ont une importance majeure aussi dans ce projet. Les enjeux à traiter dans cette thèse répondent directement à ces deux problématiques et plus précisément les points suivants :

- Assurer une localisation 3D précise dans un environnement quelconque avec le minimum de modifications sur le monde réel.
- Implémenter les algorithmes sur une carte embarquée et des composants à bas coût.
- Assurer un réponse en temps réel pour l'acceptabilité de l'application.

La structure du manuscrit est donc la suivante :

Contribution au développement d'un casque « see-through » de réalité augmentée : fusion de données pour une meilleure localisation 3D

Contexte de l'étude :
Présentation générale de la réalité augmentée ainsi que le projet REVE5D.

Chapitre 1 :
Méthodes de localisation 3D en réalité augmentée :
Ce chapitre représente un état de l'art sur les techniques de localisation et estimation d'une pose pour les applications de réalité augmentée. A l'issue de cet état de l'art, nous pouvons caractériser l'ensemble des techniques exploitables pour la résolution de nos problématiques, les limitations de chaque techniques et les possibilité d'améliorations pour converger vers une solution.

Chapitre 2 :
Choix d'un filtre pour le traitement des données :
Dans ce chapitre nous allons parler de techniques de filtrage ainsi que des outils mathématiques utilisés pour l'estimation d'une pose avec une précision acceptable. Les deux techniques majeures de filtrages sont le filtre de KALMAN et le filtrage particulaire.La rédaction est organisée de la manière suivante : une introduction des filtres, une comparaison entre les filtres, présentation des outils mathématiques, et une conclusion.

Chapitre 3 :
Implémentations d'un algorithme de rétro-correction :
Dans ce chapitre nous allons détailler les approches développées apportant des contributions sur la rétro-correction et sur les prédictions ainsi que le traitement de poses inconsistantes.

Chapitre 4 :
Évaluation de la latence du système :
Ce chapitre est dédié à la caractérisation de la latence et les tests appliqués pour réduire la latence générale du système. Cette réduction voire suppression de latence permet d'avoir un affichage concret.

Enfin, nous terminons par une **conclusion générale** et donnons quelques **perspectives**.

CHAPITRE 1

MÉTHODES DE LOCALISATION 3D

1.1 Introduction

La localisation en 3D est une problématique de recherche très répandue. Les chercheurs de plusieurs domaines s'intéressent à cette problématique dans des environnements quelconques. Les domaines d'application sont nombreux. On peut citer par exemple la robotique, l'aéronautique, la réalité virtuelle et augmentée...

Comme nous l'avons vu précédemment, les applications de réalité augmentée utilisent la pose de l'opérateur (position et orientation) pour appliquer une procédure de recalage entre le monde réel et le monde virtuel. La cohérence des affichages est donc assurée par une bonne localisation 3D : La caméra virtuelle du moteur 3D, qui calcule le rendu final utilise les poses pour afficher les projections 3D (objets virtuels). Dans la plupart des systèmes de réalité augmentée, la localisation 3D exploite un système de vision, avec une caméra externe qui filme la scène écologique en continue. Les images de la caméra sont traitées pour extraire une estimation de la position et l'orientation du système dans l'environnement réel. Cette procédure est appelée estimation de pose.

Une multitude de techniques sont proposées pour l'estimation d'une pose. On parle de localisation externe (outdoor), ou interne (indoor), en fonction du type d'environnement (intérieur ou extérieur). Nous nous sommes limités à la localisation interne. Les approches sont tout d'abord présentées selon le degré de connaissance de la scène. En effet, pour les applications de réalité augmentée, les approches de localisation interne peuvent être classées en deux catégories :

— Une connaissance a priori, même partielle, de l'environnement réel

— **Aucune connaissance de l'environnement réel**.

Ensuite, dans le contexte de notre travail (temps réel), la principale technique d'optimisation entre la précision et la rapidité du traitement est introduite : la fusion entre un algorithme de vision et une localisation basée sur une centrale inertielle (CI). Le choix de la technique d'estimation de pose est argumenté en conclusion.

1.2 Approches avec connaissance de la scène réelle

Lorsque la scène réelle est connue a priori, même partiellement, la plupart des techniques développées sont basées sur un système de vision, et l'estimation de la pose utilise des correspondances de points

2D-3D. Comme le montre la figure 1.1, le principe de la localisation par vision est de déterminer l'ensemble des transformations (rotation **R** et translation **T**) appliquées sur les images filmées par une caméra. Le modèle sténopé est celui qui est le plus utilisé pour réaliser cette correspondance.

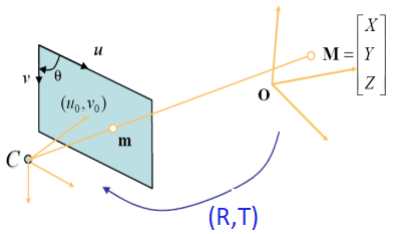


FIGURE 1.1 – Modèle sténopé

Dans la figure 1.1 les repères C et O sont respectivement les repères caméra et monde. Le point m (u, v) est la représentation dans le repère image (2D) du point M (X, Y, Z). Cette représentation dépend des paramètres intrinsèques de la caméra :

— la distance focale f ;

— les coordonnées (x_0, y_0) du point origine du repère image ;

— les densités de pixels k_u et k_v par unité de l'image dans chaque direction.

L'équation de représentation s'écrit alors :

$$\begin{cases} u = k_u \left(u_0 + f \dfrac{X}{Z} \right) \\ v = k_v \left(v_0 + f \dfrac{Y}{Z} \right) \end{cases} \tag{1.1}$$

Il existe des outils de calibration d'une caméra (gratuit ou payant) qui permettent d'estimer les paramètres intrinsèques de la caméra cité dans l'équation 1.1. Par exemple, la bibliothèque OpenCV [10] dispose d'un outil de calibration puissant qui permet d'estimer d'une manière correcte ces paramètres. Une fois que la calibration est faite, les paramètres intrinsèques de la caméra sont fixés. Cette dernière peut être utilisée pour estimer les poses.

Dans ce paragraphe nous détaillerons deux catégories de méthodes utilisant une connaissance a priori de la scène :

— les méthodes basées sur des marqueurs positionnés dans la scène réelle ;

— les méthodes basées sur un enregistrement au préalable de la scène réelle.

1.2.1 Utilisation de marqueurs

Avant de pouvoir identifier un marqueur potentiel, il est nécessaire de s'assurer que le système sera capable d'utiliser l'image obtenue pour détecter un marqueur. Il a été largement reconnu que les marqueurs dans les images en noir et blanc sont plus faciles à détecter que les marqueurs dans des images en couleur. Par conséquent, un certain nombre de méthodes ont été développées pour transformer l'image en nuances de gris et ainsi la préparer pour une utilisation ultérieure pour la détection du marqueur. Il existe plusieurs méthodes de conversions d'images en nuances de gris dans

la littérature. Citons par exemple la moyennation simple des niveaux de rouge (R), vert (V) et bleu (B) pour obtenir l'intensité I en niveau de gris :

$$I = \frac{R + V + B}{3} \tag{1.2}$$

Après la conversion de l'image en nuances de gris, l'étape suivante consiste à segmenter l'image. Cette segmentation est basée sur les techniques de seuillage.

- seuillage par histogrammes
- seuillage par minimisation de variances
- seuillage entropique
- seuillage par la méthode de pourcentage
- seuillage adaptatif local

Une fois que l'image est segmentée, la détection et reconnaissance de marqueurs devient possible.

Le principe de détection des marqueurs repose sur l'identification de pixels connectés qui forment des régions connexes. Cette opération est faite par une analyse de la totalité de l'image et ensuite l'extraction de pixels qui ont une connexion de voisinage (4-voisins ou 8-voisins). Après cette analyse, les régions détectées sont labellisées, et une procédure d'extraction de contours commence. Il existe plusieurs techniques dans la littérature qui permettent l'extraction de contours d'une image segmentée (Marr-Hilderth, Canny, Laplace...). Les arêtes sont ensuite détectées en utilisant la transformation de Hough [87], et les coins avec le détecteur de coins Harris [40]. Après l'identification des coordonnées des coins des marqueurs, la position d'une caméra par rapport à l'installation peut être déterminée.

La réalité augmentée a démarré avec l'utilisation de marqueurs artificiels. Les premiers marqueurs utilisés ont été proposés pour fonctionner avec la librairie ARToolkit (SIGRAPH 1999). [102]. Un marqueur est une image de référence, facilement détectable et identifiable. Il peut contenir une référence ou un code qui permet de retrouver le modèle 3D associé. Les marqueurs peuvent êtres classés dans deux classes :

- le marqueur modèle ("template marker") illustré dans la figure 1.2 ;
- le marqueur à codes à barres ("barcode marker") illustré dans la figure 1.3.

FIGURE 1.2 – Modèle de marqueur circulaire (gauche) et carré (droite)

Les marqueurs modèles sont des marqueurs noirs et blancs qui ont une image ou présentation à l'intérieur d'un bord noir. Typiquement, les marqueurs sont identifiés grâce à un système qui compare les images acquises avec les modèles enregistrés dans une base de données. Un coefficient de correspondance (matching) est calculé. Le modèle qui a le plus fort coefficient de correspondance avec l'image est considéré comme associé à cette image. Les limitations de cette technique sont liées aux tailles, positions, et orientations des marqueurs à identifier, qui causent parfois des erreurs de correspondances [80].

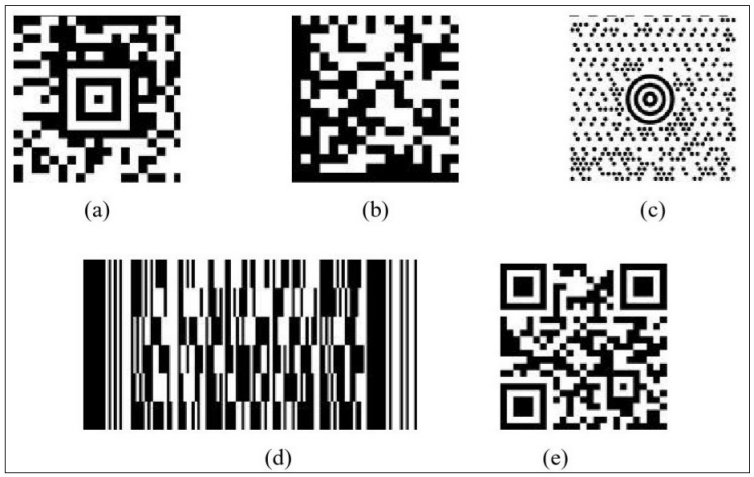


FIGURE 1.3 – Modèle de marqueur code à barres Aztec Code (a), Data Matrix (b), Maxi Code (c), PDF417 (d) and QR Code (e) [33]

Les marqueurs type code barres sont aussi des marqueurs noirs et blancs mais, pour ceux-ci, les données sont codées en ”0” ou ”1” en divisant la région du marqueur en plusieurs carrés noirs et blancs, qui sont décodés lorsqu'ils sont identifiés. L'ensemble des données du marqueur peut être présenté sous la forme d'une série de valeurs binaires, ou le nombre binaire associé. Il existe plusieurs standards de marqueurs 2D à code à barres. Les plus connus [33] sont illustrés dans la figure 1.3 :

— AZETEC code
— DATA Matrix
— MaXi code
— PDF417
— QR code

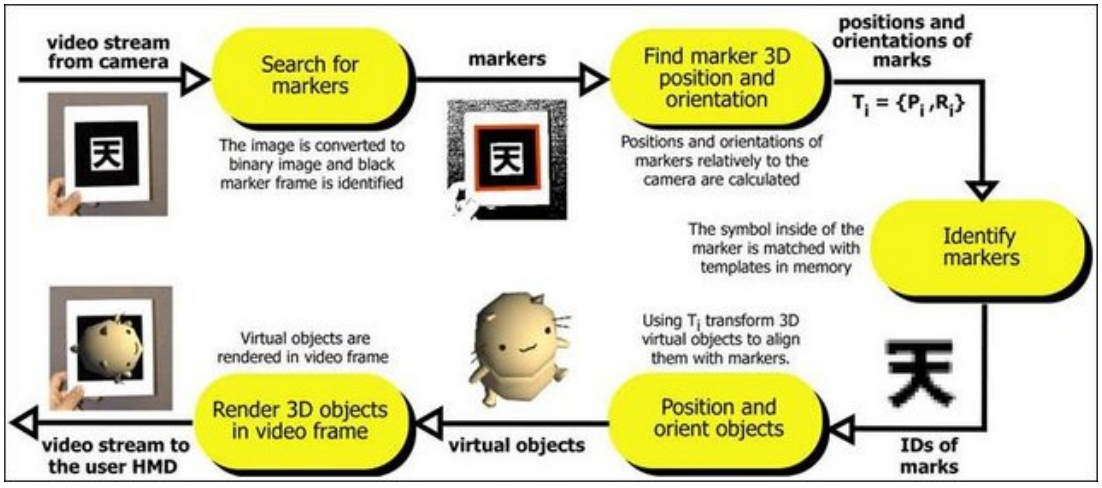


FIGURE 1.4 – Processus complet d'affichage de modèles 3D en se basant sur les marqueurs [46]

La figure 1.4 explique les étapes fondamentales décrites par les travaux de [46] pour afficher un modèle 3D en réalité augmentée. Le processus commence par une récupération du flux vidéo de la caméra. Après une étape qui englobe le passage en niveaux de gris, le seuillage, et la binarisation, l'extraction de contours permet d'identifier la position et l'orientation du marqueur. Ensuite l'identification du marqueur permet de lui associer un modèle 3D. Une fois le modèle 3D associé, une mise à jour de sa position et de son orientation lui est appliquée selon les poses du marqueur, avant un calcul du rendu et un affichage final.

1.2.2 Utilisation de modèles 3D

L'utilisation de marqueurs nécessite une étape de préparation. L'impression de marqueurs et l'installation dans l'environnement est systématique . Dans le contexte industriel et pour une optimisation de résultats, il faut que les marqueurs soient entretenus de temps en temps. Pour s'affranchir de ces limitations, il est nécessaire d'intégrer de nouveaux outils qui ne nécessitent pas une configuration, installation, et un entretien dans l'environnement. L'idée donc est d'exploiter les ressources qui sont déjà présentes dans l'environnement. Plusieurs techniques sont développées dans la littérature qui exploitent les ressources directes de la scène. Cette section détaille l'utilisation de modèles 3D pour la localisation dans des scènes. Dans cette approche les parties de l'environnement réel sont utilisées comme un marqueur de recalage des objets virtuels. Cette approche de localisation basée sur le modèle 3D peut être classée en trois sous-catégories selon la technique de traitement. La première catégorie est basée sur l'extraction et l'exploitation des arêtes. La deuxième consiste à suivre le flot optique d'une séquence d'images. Finalement, la troisième utilise les informations extraites d'une texture.

Extraction et exploitation des arêtes

L'extraction de segments ou d'arêtes est la première technique utilisée dans la réalité augmentée pour le suivi basé sur un modèle 3D [59]. Dans ce cas de figure, la pose de la caméra est estimée par une mise en correspondance entre un modèle 3D présenté sous forme fil de fer 'wireframe 3D' et les arêtes extraites depuis les images acquises. Cette mise en correspondance est faite par une projection de modèle dans une image et la minimisation de la distance entre les arêtes du modèle et les arêtes extraites de l'image. Pour établir la mise en correspondance, une bonne estimation de la pose de départ est bénéfique pour le déroulement de l'algorithme. Généralement dans cette approche l'initialisation de la pose est faite d'une manière manuelle. Une fois que la première pose est estimée, elle est utilisée pour faire une projection du modèle 3D à l'itération suivante. Les déplacements de la caméra sont relativement petits, ce qui rend la prédiction possible et affecte peu les résultats de correspondance. On peut citer deux méthodes principales :

— Les méthodes où les arêtes sont échantillonnées en points (du modèle 3D). Le segment est alors déterminé par régression via la méthode de descente de gradient [69]. Dans cette approche les points échantillonnés sont choisis pour être des points de contrôle qui sont utilisés ensuite pour l'appariement. En se basant sur l'estimation précédente, les arêtes visibles sont déterminées. Une fois que l'arrête est identifiée, les points de contrôles sont projetés uniquement sur les arêtes visibles en se basant sur l'estimation de pose précédente. Les arêtes sont extraites d'une image en utilisant les gradients sur les deux axes x et y horizontal et vertical respectivement. Ensuite une recherche dans le voisinage de cette pose est réalisée afin de trouver les points de correspondances. Finalement la pose de la caméra est déterminée en fonction de la pose calculée d'une image à une autre.

— Les méthodes de détection explicite des arêtes et les appariements avec la projection du modèle. Cette approche utilise un détecteur de lignes (la transformée de Hough). Le modèle 3D est

projeté dans une image en utilisant la pose précédente de la caméra. La projection et les arêtes extraites de l'image sont comparées pour estimer la nouvelle pose [53]

Ces méthodes sont robustes face au changement d'éclairage. Cependant, elles sont peu performantes face aux mouvements rapides de la caméra, car les images projetées vont être décalées par rapport à leur propre localisation. Les erreurs de mise en correspondance représentent une limitation considérable de cette technique. Plusieurs raisons peuvent amener à mal estimer la correspondance entre les deux images, notamment les zones d'ombrage ou un arrière plan encombré.

Suivi du flot optique

Le suivi du flux optique est une approche qui permet d'estimer les mouvements apparents des éléments. Contrairement aux méthodes précédentes, l'information spatiale n'est pas exploitée. L'information temporelle est utilisée à sa place. Les mouvements sont repérés par les mouvements de projection d'objets dans l'image elle-même. L'initialisation de cet algorithme reste toujours manuelle. Après l'initialisation, le flux optique est estimé entre images capturées aux instants t et $t+1$. L'algorithme détermine dans un premier temps les points qui sont présents dans les deux images. Une fois que les points sont détectés, les déplacements de ces points au cours du temps sont calculés en utilisant l'algorithme de Kanade-Lucas [37]. Les déplacements sont utilisés pour estimer la pose de la caméra. Dans cette méthode, les erreurs peuvent s'accumuler car il y a une série d'intégration au cours du temps.

Extraction d'information d'une texture

Comme l'indique son nom, cette technique exploite la texture d'une image, en d'autres termes, les informations visuelles. On y trouve la catégorie qui permet une définition d'un modèle représenté par une image. La position de la caméra est calculée par rapport aux transformations (positions, orientations, taille...) appliquées à ce modèle dans l'image acquise. Cette technique est connue sous le nom de **mise en correspondance d'un modèle** ou 'Template matching'. Elle se base sur les informations globales de l'image. Une deuxième catégorie appelée **technique de points d'intérêts** utilise la texture pour extraire les points d'intérêts d'une image acquise. Le tracking se fait généralement en utilisant la méthode décrite dans [37] ou d'autres variantes de cette techniques. Une amélioration peut s'effectuer en utilisant les plans du modèle 3D au lieu d'utiliser la totalité de ce dernier. La technique de points d'intérêts est connue comme une technique peu coûteuse en terme de calcul et supporte le changement d'éclairage. Cependant elle est connue également pour le cumul des erreurs et des faits des dérives considérables.

Approche hybride

Afin d'améliorer les performances de tracking, une hybridation entre les **segments** et la **texture** est parfois faite. Une intégration des mouvements calculés par une estimation basée sur la texture, dans les poses calculées par la méthode basée sur les segments, peut par exemple augmenter la précision et donner une tendance vers un tracking en temps réel [82]. Des estimateurs [3] sont rajoutés dans le processus de tracking pour améliorer la robustesse face aux zones d'ombrage et lumières spéculaires. Une autre approche consiste à combiner la méthode basée sur les segments avec l'extraction de points d'intérêts Harris [21] [81]. Les points d'intérêts sont considérés comme des projections sur le modèle 3D. La localisation consiste à optimiser une fonction de transfert qui regroupe les points d'intérêts dans deux images consécutives et leurs projections sur le modèle 3D.

1.3 Approche sans connaissance de la scène réelle : le SLAM

Dans la section précédente nous avons vu les techniques de localisation basées sur le système de vision, mais qui nécessitent une connaissance a priori de la scène ou de l'environnement de travail. Cette section détaille un outil existant pour une localisation à l'intérieur sans connaissance a priori de l'environnement. Pour la vision on trouve les méthodes SFM (Structure From Motion) [95] et la famille de la localisation et cartographie simultanées (SLAM) probabiliste [8, 14, 20, 25, 73] et par ajustement de faisceaux [91, 110, 63, 27, 109, 58]. Dans le cadre de ce travail, une méthode SLAM avait au préalable été choisie. Nous détaillons donc cette méthode de vision.

La méthode de localisation et cartographie simultanées (le SLAM) a été initialement développée pour le domaine de la robotique. Un robot peut aujourd'hui se localiser dans un environnement quelconque sans une connaissance a priori, grâce à des travaux pionniers [90]. Plusieurs algorithmes ont été développés en utilisant une variété de capteurs. Les capteurs de distance tels que sonars, télémètres laser, et radars ont été largement exploités. Un capteur ultrason a aussi été utilisé pour modéliser la navigation d'un robot [19]. Une carte est déjà pré-acquise et le robot évolue et met à jour cette carte.

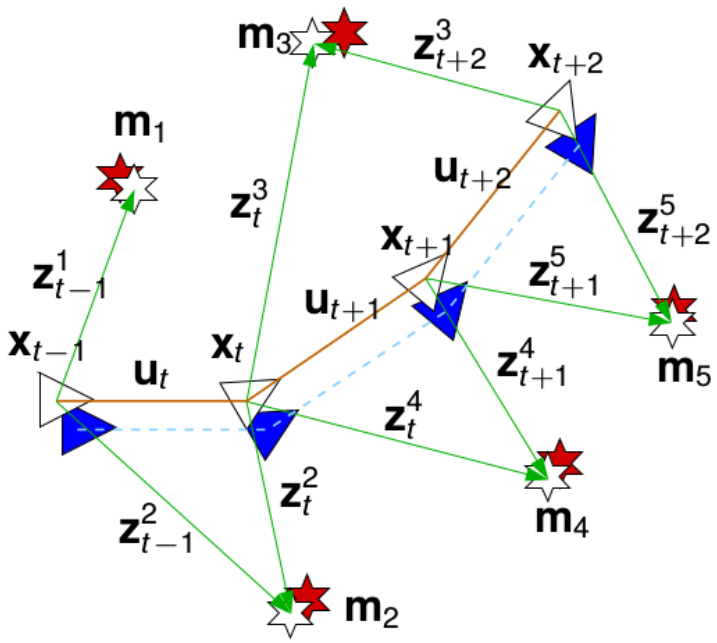


FIGURE 1.5 – Principe de fonctionnement d'un SLAM basé sur l'estimation et la correction de la pose en fonction de mesures par différents capteurs [24]

La figure 1.5 montre le principe générique d'un algorithme du SLAM. Les triangles bleus présentent les déplacements réels d'un robot dans son environnement. Les triangles vides représentent les estimations SLAM basées sur des algorithmes d'estimations. Les étoiles blanches représentent la mesure d'amers en utilisant les capteurs du robot. Les étoiles rouges pleines sont les positions réelles d'amers. X_t est l'estimation. U_t est la commande envoyée au robot. Z_t^i est la mesure de l'iemme amers.

Aujourd'hui, compte tenu de ses performances et de sa précision, le SLAM a été adopté par les

développeurs de la réalité augmentée. Après les années 2000 et avec le développement de caméras à faible coût, les chercheurs ont commencé à traiter les problèmes de SLAM en utilisant le traitement d'images. Le SLAM basé sur le traitement d'images est désormais appelé SLAM visuel ou SLAM par vision. La caméra représente un avantage, car les images sont plus riches en terme d'informations que les autres capteurs de distance. Cependant, les algorithmes d'extraction et de mise en correspondance de primitives visuelles sont plus complexes et exigent un temps de calcul important.

1.3.1 Description générale

Les méthodes de SLAM sont basées sur une approche probabiliste (SLAM probabiliste) ou sur une optimisation (SLAM par ajustement de faisceaux). Il s'agit de trianguler un point 3D de l'environnement observé à différentes positions et à différents instants. La mise à jour de la carte qui contient les points 3D utilise différents outils statistiques tels que le Filtre de KALMAN étendu, les approches bayésiennes, le filtre particulaire... Le détail de la méthode n'est pas l'objet de ce document. Le lecteur pourra se référer pour cela à des travaux dédiés [89]. Toutefois, on peut noter que le SLAM se décompose en deux familles :

— Le SLAM **direct**, où le traitement est appliqué directement sur les pixels de l'image ou sur le gradient de pixels (dense et semi-dense) ;

— Le SLAM **indirect**, repose sur le traitement de primitives extraites de l'image source.

Le principe de base pour la famille de SLAM direct est dans l'équation suivante donnant l'évolution de l'intensité de l'image en fonction des coordonnées et du temps :

$$J(x, y, t) = I(x + u(x, y), y + v(x, y), t + 1) \tag{1.3}$$

Dans cette équation, x et y sont les cordonnées des pixels, u et v sont des fonctions de déplacement de pixels entre les deux images I et J capturées à deux instants t et t+1 respectivement. Pour résoudre l'équation 1.3 la méthode ('Forward Additive Image Alignement(FAIA) à été proposée [64]. Cette méthode consiste à regrouper toutes les transformations élémentaires de fonctions u et v pour chaque pixel dans une fonction générique à minimiser :

$$\arg\min_{p} \sum_{x,y} [I(W(x, y, p)) - J(x, y)]^2 \tag{1.4}$$

$W(x, y, p)$ est une fonction qui englobe les transformations entre les deux images I et J. Et p représente les paramètres de transformation.

Dans le SLAM indirect, l'utilisation de la méthode d'extraction de primitives consiste à alléger le traitement lourd causé par les méthodes directes qui cherchent à analyser la totalité des pixels ou dans le meilleur des cas les pixels de fort gradient. Parmi les multitudes de techniques d'extraction de primitives définies dans la littérature on trouve :

— Hessian corner detector [111] ;

— Harris [21] ;

— Laplacien de Gaussien [36] ;

— Features from Accelerated Segment Test (FAST) [99] ;

— Binary Robust Independent Elementary Features (BRIEF) [12] ;

— Speeded Up Robust Features (SURF) [6] ;

— Scale invariant Feature Transform (SIFT) [68] ;

— Oriented FAST and Rottated BRIEF (ORB) [88].

Il en ressort que les méthodes directes sont robustes dans les environnements peu texturés et supportent aussi le flou causé par les mouvements rapides de la caméra. Les méthodes directes opèrent directement sur les pixels d'images acquises, donc la phase d'extraction de primitive est supprimée et son temps de traitement aussi. En conséquence, il en résulte une augmentation de performance en terme de précision et fiabilité de traitement. Cependant, les méthodes directes ne supportent pas les grands sauts entre deux images acquises car la linéarisation est faite pour les petits mouvements.

Les méthodes basées sur une extraction de primitives sont capables d'opérer avec des sauts considérables entre deux images. Le temps de calcul dans ces méthodes présente un vrai handicap, car il est relativement important. Les environnements sans textures, ou peu texturés, posent une vraie limitation. La représentation, clairsemée uniquement, constitue un autre point de limitation.

1.3.2 Le SLAM pour la réalité augmentée

Le tableau 1.1 donne plusieurs méthodes SLAM appliquées à la réalité virtuelle, avec leurs points forts et leurs points faibles. Ces méthodes sont ensuite détaillées.

Approche	Principe	Points forts	Points faibles
PTAM	- Deux threads en parallèle localisation et cartographie - Basé sur l'extraction de primitives FAST Cartographie avec le principe de AF	- Découplage de deux threads (tracking et cartographie) -Relocalisation rapide si la nouvelle images acquise est proche des images clés enregistrées	- Environnements de petites tailles - Pas de fermeture des boucles qui peut causer des dérives - Ne supporte pas les mouvements rapides et le flou de mouvement - Opérationnel dans les environnement richement texturés
ORB-SLAM	- Trois threads en parallèle (suivie, cartographie, et optimisation) - Basé sur l'ORB pour l'extraction de primitives - Cartographie selon le principe de AF	- Traitement Rapide pour l'extraction de primitives ORB - Opérationnel pour les petits et grands environnements - Robuste face aux mouvements rapides - Initialisation automatique	- Nécessite une puissance de calcul considérable -Aucune implémentation pour les téléphones mobiles ou les cartes de puissance de calcul faible

RD-SLAM	- Deux threads principaux exécutés en parallèle (tracking et cartographie) - Basé sur le principe SIFT pour l'extraction de primitives - Optimisation de mise en correspondance avec un algorithme de KD-Tree - Ajouter des images clés	- Robuste face aux mouvements relativement rapides de la caméra - Robuste face aux changements d'une petite partie de la scène (dynamique) - Méthode 'Adaptative RANSAC' pour la mise en correspondance	- Scènes dynamiques avec un petit changement (la plupart de la scène doit rester statique) - Fonctionnel uniquement pour les petites zones (environnement limité) - Fréquence de traitement de l'ordre de 25 images/s sur un ordinateur de bureau (3.3 Ghz CPU 8 Go RAM)
DT-SLAM	- Trois threads principaux (suivie, cartographie, et AF) - Détection de primitives FAST - méthode de ('Zero Mean Sum Squared Distance) utilisé pour valider les mises en correspondance.	- Utilisation de points 2D dans la carte pour les rotations pures	- Génération de sous-cartes indépendantes et difficiles à optimiser -Dans le cas où le tracking échoue la nouvelle localisation peut s'effectuer dans une nouvelle sous carte
LSD-SLAM	- Trois threads principaux (suivie, estimation de carte de profondeur, et optimisation de la carte) - Pas d'extraction de primitives mais opère directement sur les images - Minimiser la variance-normalisé de l'erreur photométrique entre deux images $E_p(\epsilon_{ji}) = \sum_{p\in\Omega D_i} \| \frac{r_p^2(p,\epsilon_{ji})}{\sigma_{rp}^2(p,\epsilon_{ji})} \|$	- Traitement pour les environnements larges - Utilisation d'une méthode directe - Supporter les zones de texture faibles	- Précision inférieur par apport aux méthode directes - Enregistre tous les pixels de correspondances durant le tracking qui induit une augmentation d'estimation d'outliers traitement coûteux due à l'utilisation de la totalité de pixels
RK-SLAM	- Deux threads principaux (suivie et mouvement) Extraction de primitives FAST nouveau modèle de tracking de primitive utilisé (homographie)	- Temps réel - Robustesse de tracking - Supporter les mouvements rapides	- Ne supporte pas les scènes dynamiques - Limitations dans les zones non texturées - Ne supporte pas les occultations - Système fermé (pas de codes source)

MCSLAM* (IP)	- Trois threads principaux (acquisition, tracking, cartographie) - Extraction de points d'intérêts Harris - Principe d'ajustement de faisceaux pour l'optimisation	- Temps réel - Environnements petites-échelles et larges-échelles	- Limitations pour les textures faibles - Problèmes de dérives d'échelle* - Zones dynamiques partiellement supportées

Tableau 1.1 – Tableau récapitulatif des performances de chaque variante de SLAM pour la réalité augmentée

Les points principaux de la méthode PTAM [49] sont :

— Localisation et cartographie séparées dans deux threads parallèles ;
— Cartographie basée sur des images clés qui sont traitées en utilisant la technique d'ajustement de faisceaux (AF) ;
— Carte initialisée à partir d'une vision stéréoscopique (algorithme à 5 points) ;
— Nouveaux points initialisés avec une recherche épi-polaire ;
— Plusieurs points cartographiés.

En séparant la localisation et la cartographie en deux processus (threads) parallèles, la localisation est détachée de la procédure de génération de la carte. Suite à ce détachement, n'importe quelle procédure robuste de tracking peut être utilisée. Le couplage entre la localisation et la cartographie est brisé, et le partage de données n'est pas obligatoire entre les deux threads. Ceci rend les threads indépendants ; Ils peuvent êtres lancés sur des "cœurs" (processeurs) différents. Vu le découplage de la localisation et la cartographie, il n'est pas obligatoire dans cette approche d'utiliser chaque image pour cartographier la scène. Les images redondantes peuvent être directement ignorées pour augmenter les performances. En ignorant les images répétitives, le traitement redondant de filtrage de mêmes données est supprimé. Le traitement dans ce cas est concentré dans des images clés plus utiles. Les images clés sont traitées sans aucune restriction sur le temps de traitement, indispensable avant l'insertion d'une nouvelle image clé. Pour la bonne exécution de cette méthode, il faut une puissance de calcul importante, plusieurs optimisations sont généralement nécessaires pour fonctionner correctement sur des périphériques mobiles qui souffrent d'une faible puissance de calcul. Les mouvements rapides de la caméra sont aussi une autre limitation de cette méthode, car cette dernière se base sur la détection de coins. En raison du flou créé lors d'un déplacement rapide, la détection de coins devient plus difficile et impacte les performances de tracking.

Une deuxième version de cet algorithme [50] a été développée pour les téléphones mobiles. Les tests ont été faits sur un iPhone 3G. La fréquence d'acquisition de la caméra est inférieur à 15 Hz. A cause de cette fréquence basse, les mouvements rapides ne sont pas détectables. La figure 1.6 présente le mode opératoire du PTAM. Les sous-figures (a), (b), (c) représentent les différentes phases de cette approche : (a) détection des points pour une initialisation, (b) détection de points d'intérêts et identification de zone (c) insertion d'objets virtuels. Comme on peut le voir dans la figure 1.6, un flou domine sur la totalité de l'image dans la partie (c), ce qui rend le tracking impossible. Dans cette configuration des changements majeurs par rapport au SLAM implémenté pour les ordinateurs sont appliqués. L'ajout d'un nouveau thread qui s'exécute en arrière plan pour appliquer l'ajustement de Faisceaux et optimiser la carte. Compte tenu de l'architecture du téléphone iPhone 3G, le nouveau thread ajouté s'exécute lorsque le thread principal du tracking est en pause. La méthode d'initialisation aussi a changé, car on ne dispose pas d'une caméra stéréo sur l'iPhone.

Cette approche a bien démontré la faisabilité d'exécuter les algorithmes sur un téléphone portable avec une faible puissance de calcul mais les performances sont relativement faibles. Pour améliorer les performances, une acquisition de 30 images/s et un champ de vision plus large sont recommandés.

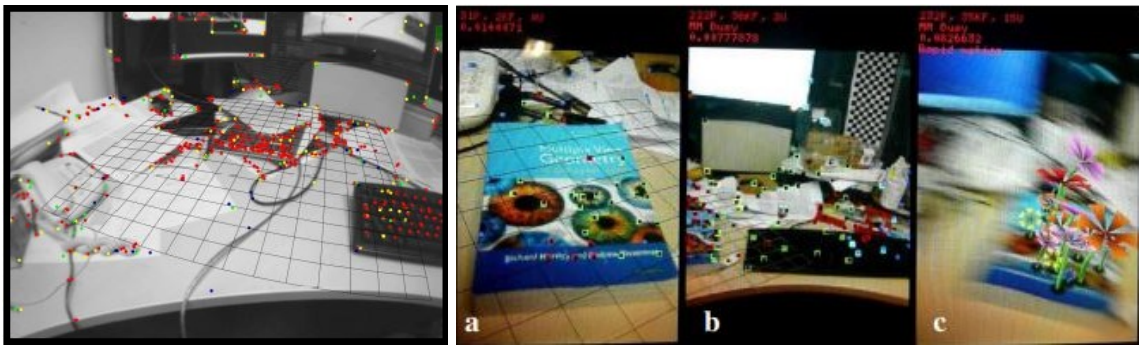


FIGURE 1.6 – Présentation PTAM : gauche (PTAM sur PC) [49] droite (PTAM [50] sur iPhone®)

RDSLAM[93] est une approche d'un SLAM monoculaire en temps réel destiné aux environnements dynamiques. Contrairement aux approches classiques du SLAM, cette approche permet à certaines parties de la scène d'être dynamiques. Le changement d'une partie de la scène et l'occultation partielle de certaines parties n'influencent pas la position d'objets augmentés. RDSLAM utilise la méthode SIFT pour l'extraction de points d'intérêts avec une représentation d'images clés. RDSLAM est robuste devant les mouvements rapides de la caméra et la relocalisation globale. Cependant, RDSLAM est fonctionnel dans des environnements relativement réduits (en terme de taille) et il n'est pas robuste devant le changement brutal de la scène. Le changement doit être progressif.

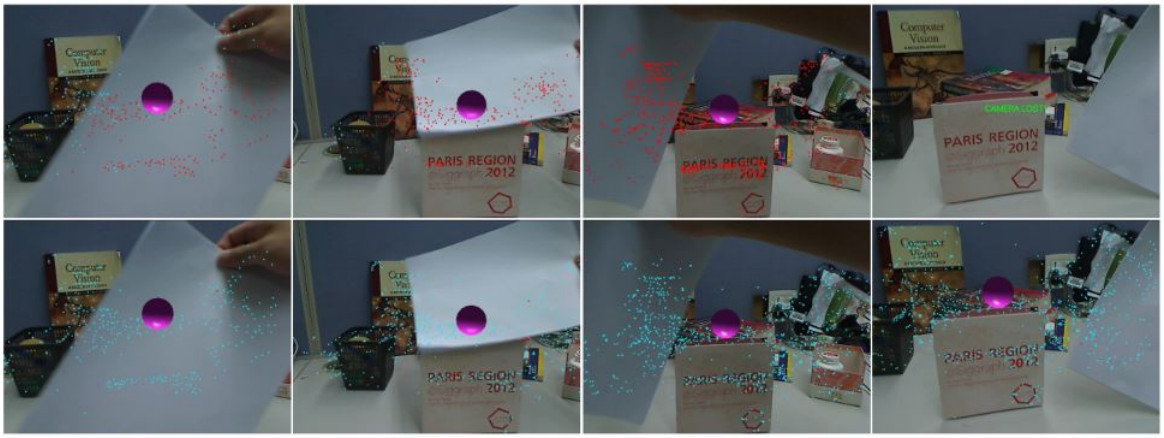


FIGURE 1.7 – Résultats de l'algorithme RDSLAM [93] sur les environnements dynamiques

La Figure 1.7 représente deux processus. Les 4 images de droites regroupent l'étape d'initialisation et les autres regroupent l'étape navigation. Dans la navigation, le SLAM a montré sa robustesse face aux occultations causées par l'ajout de la feuille blanche dans la scène et malgré l'augmentation progressive, l'objet 3D reste toujours dans sa position et affiché.

Contrairement aux approches précédentes qui opèrent uniquement dans les petits environnements et qui se basent sur l'extraction de primitives. Le LSD-SLAM[26] est une variante de SLAM qui

permet de se localiser et cartographier dans des environnements relativement importants. Celui-ci exploite les méthodes directes, et non l'extraction de primitives. Cette approche utilise aussi l'aspect stéréoscopique (deux caméras). Comme les autres SLAM, le LSD-SLAM est un algorithme qui s'actualise en temps réel. Les étapes de LSD-SLAM sont :

— La caméra est trackée par rapport à un domaine des images clés dans la carte pré-calculée. Si le mouvement de la caméra est très rapide et loin de toutes les images clés dans la carte, une nouvelle image clé est introduite dans la carte.

— La profondeur est estimée en se basant sur les images stéréoscopiques.

— Les poses des images-clés sont calculées d'une manière optimale par un alignement entre l'image et le graphe de poses de la carte.

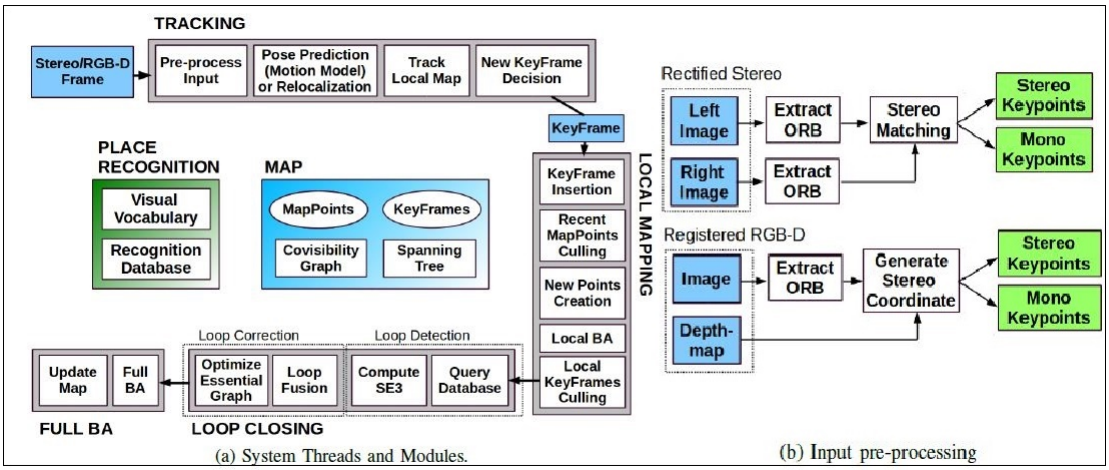


FIGURE 1.8 – Architecture détaillée ORB-SLAM2 [75]

L'ORB-SLAM est un SLAM monoculaire, qui se base sur trois threads qui s'exécutent en parallèle pour la cartographie et la localisation. Cette approche est performante pour les environnements à grande échelle. Cependant, elle est limitée à une caméra pour le tracking et elle nécessite des ressources informatiques importantes pour effectuer le calcul en temps réel. La figure 1.8 résume les trois processus principaux du système :

— Le suivi pour localiser la caméra à chaque image par mise en correspondance de primitives extraites de l'image ;

— La création d'une carte locale, et la minimisation de la projection en utilisant l'ajustement de faisceaux (AF) ;

— La gestion de la cartographie locale en appliquant le AF local ;

— La fermeture de boucles pour éliminer les dérives causées par les grandes boucles ouvertes par l'application de l'optimisation pose-graphe. Ce thread lance un quatrième thread de traitement qui applique un AF global après l'optimisation de graphe de poses, afin de calculer la structure et le mouvement optimales.

L'ORB-SLAM2 a permis l'intégration de différentes caméras (stéréo et RGB-D). Cette approche est plus généraliste que l'approche précédente qui a offert plus de stabilité à la localisation et à la cartographie. Les algorithmes testés sont exécutés sur CPU, un potentiel d'optimisation de temps de calcul est possible si l'extraction de primitives (ORB) est exécutée sur le processeur de la carte

graphique (GPU). Les avantages de cette méthode se manifestent généralement par la possibilité d'appliquer cet algorithme dans des environnements à grande échelle avec une précision plus importante que LSD-SLAM, la fermeture de boucles en temps réel, et la robustesse face aux changements d'éclairage de l'environnement. En revanche ORB-SLAM2 dans sa version actuelle n'est pas exploitable sur les périphériques de puissance de calculs modestes.

L'approche DT-SLAM[42] est basée sur trois threads principaux. Elle est expliquée par la figure 1.9, et comporte les étapes suivantes :

— tracking : qui essaye de trouver les correspondances de primitives extraites de l'image courante avec celles stockées dans la carte. Il se base sur l'hypothèse que les mouvements entre les images sont relativement faibles. La contribution de cette méthode est dans l'utilisation de primitives 3D et 3D dans un framework unifié.

— cartographie : Afin de rajouter une nouvelle image clé, l'algorithme recherche toutes les correspondances possibles depuis la carte. Si les primitives 2D observées avec un décalage large en terme de distance, l'image sera intégrée et la primitive est triangulée. Le thread de cartographie raffine la pose de la nouvelle image clé et de la primitive simultanément.

— B.A tourne toujours en arrière plan pour optimiser les images clés et la position de primitives. Ce thread traite les informations 2D et les informations 3D.

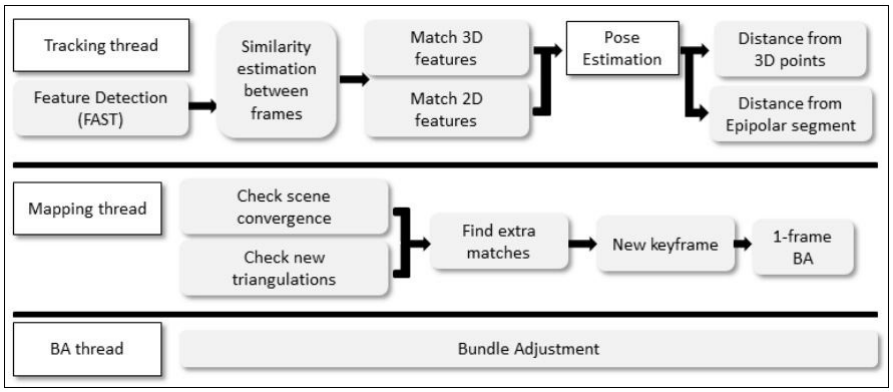


FIGURE 1.9 – Schématisation de l'architecture DT-SLAM [42]

RKSLAM [66] est une approche de SLAM développée pour les applications de réalité augmentée supporte les mouvements rapides de la caméra. Elle est identique aux autres SLAM qui se basent sur une image clé (dans l'architecture globale). La différence consiste à déplacer le thread d'optimisation de la carte locale de l'arrière plan vers l'avant. De plus et pour robustifier le tracking, les tâches du SLAM sont résolues suivant trois modèles homographiques.

— Homographie globale

— Un ensemble d'homographies locales

— Homographie planaire

MCSLAM [85] est un SLAM multi-contraint développé par l'Institut Pascal (IP). Le MCSLAM respecte l'architecture générique du SLAM. Il est utilisé généralement pour remédier au problème de dérive de facteur d'échelle lors de la reconstruction et l'estimation de trajectoire. Le SLAM contraint combine les différentes informations issues de différentes sources (capteurs) dans son processus d'estimation de la trajectoire et la modélisation de l'environnement. Il existe deux catégories de contraintes, les contraintes liées au mouvement du système et les contraintes de structure. Les

contraintes de mouvement ou de structure sont mutuellement liées et ont une influence sur la trajectoire et la structure.

Dans le travaux de [61, 56, 70, 52, 29, 57, 44, 60, 31, 67] et pour contraindre l'ajustement de faisceaux, des données de la centrale inertielle et du GPS sont utilisée pour la partie mouvement.

D'autres travaux ont opté pour le changement niveau algorithmique. Dans les travaux de [61, 56, 70, 60] on trouve plusieurs modifications algorithmiques pour ajouter des contraintes supplémentaires en minimisant l'erreur de certains critères ou en utilisant les données des capteurs pour contraindre certains critères comme l'utilisation de données inertielles pour contraindre les poses SLAM basées sur des images clés par [60].

Le MCSLAM de l'institut pascal est un SLAM contraint qui impose des contraintes sur l'estimation de trajectoire et la structure au même temps.

Contraintes de la trajectoire : Dans son implémentation et pour représenter les mouvements, Le MCSLAM [85] utilise la présentation b-spline cumulative proposé par [67]. L'évaluation de Spline à un instant t nécessite 4 points de contrôle consécutives [p0 ; p1 ; p2 ; p3] aux instants respectives [t0 ; t1 ; t2 ; t3] . ces points sont choisis suivant la relation suivante : $t1 < t < t2$. Les coefficients b-spline pour un temps normalisé u sont :

$$\{B_1(u), B_2(u), B_3(u)\} = \{(u^3 - 3u^2 + 3u + 5)/6; (-2u^3 + 3u^2 + 3u + 1)/6; u^3/6\} \tag{1.5}$$

Les positions $S(u)$, les vitesses $\dot{S}(u)$ et l'accélération $\ddot{S}(u)$ sont calculées par les formules suivantes respectives.

$$S(u) = \sum_{i=0}^{2} (p_{i+1} - p_i) B_{i+1}(u) \tag{1.6}$$

$$\dot{S}(u) = \sum_{i=0}^{2} (p_{i+1} - p_i) \dot{B}_{i+1}(u) \tag{1.7}$$

$$\ddot{S}(u) = \sum_{i=0}^{2} (p_{i+1} - p_i) \ddot{B}_{i+1}(u) \tag{1.8}$$

Avec $\dot{B}$ et $\ddot{B}$ sont les dérivées première et seconde respectivement de B par rapport u.

Une séparation de l'orientation et la position en deux différentes Splines est appliquées. La contrainte est rajoutée au niveau de « solver » pour minimiser la différence (pour les positions et les orientations) entre la pose estimée à cette image-clé et le Spline à l'instant d'acquisition de cette image-clé. L'algorithme exploite aussi les poses entre les images-clés calculées par le processus de localisation SLAM pour imposer des contraintes faibles aux Splines. Une fenêtre glissante temporelle de durée 3s est utilisée pour enregistrer les données CI et SLAM afin d'appliquer les contraintes sur les Splines. Les données sont utilisées pour minimiser les variations de la vitesse (positions et orientations) entre les différents points de contrôle de Spline.

Les contraintes de structure : Lors de l'observation d'un objet 3D, une contrainte entre l'objet lui même et ses primitives 3D est créée et ajoutée dans le processus d'optimisation. Les paramètres (position, orientation, facteur d'échelle et forme) sont initialisés d'une manière semi-automatique et optimisés en temps réel au cours d'exécution du MCSLAM. Cette initialisation consiste, à remplir approximativement un modèle par des primitives calculées via par le processus de reconstruction. Pendant l'exécution du SLAM la zone convexe peut être sélectionnée et l'algorithme interprètera les modèles 3D à l'intérieur de la zone sélectionnée.

MCSLAM intègre dynamiquement dans le processus d'optimisation les contraintes issues des objets partiellement connus de l'environnement. Afin d'assurer la convergence de MCSLAM, une étape supplémentaire pour associer entre les primitives 3D et les objets correspondants est nécessaire.

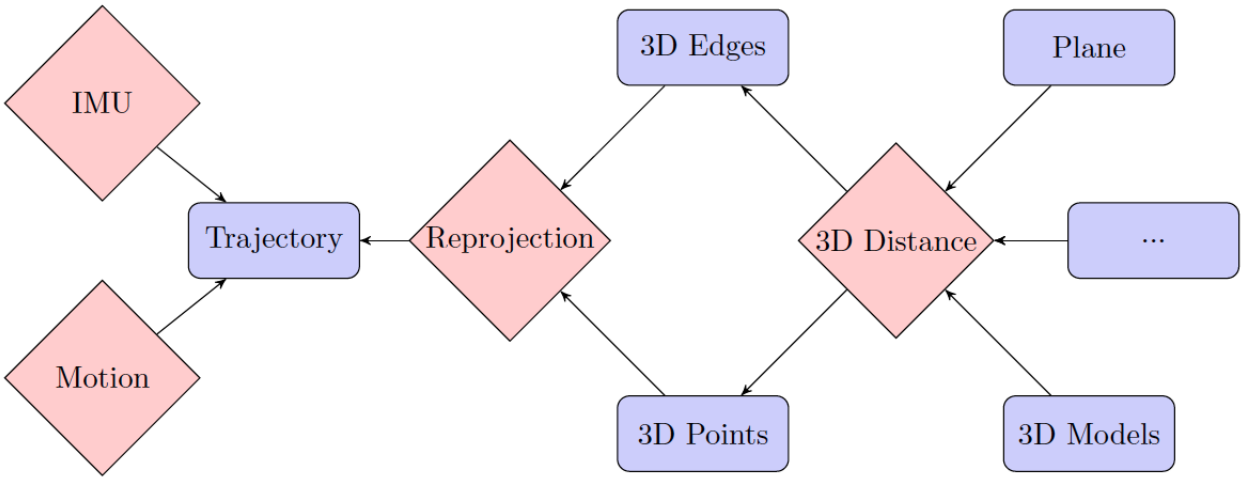


FIGURE 1.10 – Graphe de dépendances MCSLAM [85]

Une primitive 3D P est associée à un objet 3D π si la distance dp entre P et la plus proche surface d'objet π est inférieur à un seuil σ_π. Pour choisir une valeur de σ_π, invariante, par rapport le facteur d'échelle, la dimension D_π de l'objet 3D est utilisée. La valeur de σ_π est calculée de la manière suivante : $\sigma_\pi = D_\pi \text{x} \lambda$ avec λ qui est l'intensité de contraintes fixée à 0.005.

Le MCSLAM est basé sur la dépendance du graphe présenté dans la figure 1.10. Ce graphe contient une liste de contraintes, liste de paramètres et liste de dépendances. Le graphe est utilisé dans deux étapes importantes. La première étape consiste à analyser les configurations (liste de contraintes, liste de paramètres et lier à chacun d'eux) pendant la phase de la compilation pour générer un 'solver' spécifique. Le 'solver' est basé sur une fonction qui consiste à minimiser la somme de contraintes rajoutées dynamiquement présenté par l'équation 1.9.

$$\epsilon = \sum_{i=0}^{C} \sum_{k=0}^{K_i} || \frac{\rho_{i,j}}{\sigma_i} ||^2 \tag{1.9}$$

avec K_i est le nombre des observations associées à la contrainte i, $rho_{i,k}$ est l'erreur de l'observation k associée à la contrainte i, et σ_i représente l'estimation de l'erreur de mesure.

1.4 Couplage du SLAM avec une centrale inertielle

La navigation inertielle est une technique de navigation autonome. Elle est généralement basée sur les données mesurées fournies par les capteurs encastrés sur le corps rigide. Les données classiquement exploitées sont les accélérations linéaires et les vitesses angulaires. Les capteurs se regroupent dans une unité appelée "centrale inertielle". Une centrale inertielle (CI) contient des accéléromètres qui mesurent les trois accélérations et des gyroscopes qui mesurent les vitesses angulaires sur les trois axes (x, y, et z). Les données mesurées sont relatives au corps rigide sur lequel est monté la centrale inertielle.

Les accélérations et vitesses angulaires sont mesurées relativement aux mouvements du corps dans son repère :

$$\vec{A}_m = \vec{A}_r + \vec{g} + \vec{n} \tag{1.10}$$

$$\vec{\omega}_m = \vec{\omega}_r + \vec{n} \tag{1.11}$$

Dans ces équations, où $\vec{A}_m$ est la valeur de l'accélération mesurée à la sortie de capteur, $\vec{A}_r$ est la valeur de l'accélération de corps réelle, $\vec{g}$ est la force de pesanteur, $\vec{\omega}_m$ est la vitesse angulaire mesurée à la sortie du capteur gyroscopique, $\vec{\omega}_r+$ est la vitesse angulaire réelle du corps, et $\vec{n}$ est le bruit associé au capteur.

L'accéléromètre fournit des accélérations linéaires mélangées avec la pesanteur $\vec{g}$, et des vitesses angulaires. Ainsi, l'estimation de la position et des rotations se base sur une double intégration en temps de l'accélération et sur une intégration des vitesses angulaires. Les rotations peuvent avoir plusieurs présentations, les angles d'Euler, les matrices de rotation, les quaternions... La représentation de la rotation sera détaillée dans le chapitre suivant dédié au filtrage.

type de bruit	description	résultat $\int dt$ gyroscope	résultat $\iint dt^2$ (accéléromètre)
Biais	Un biais constant ϵ dans le signal de sortie	Une erreur angulaire en croissance constante $\theta(t) = \epsilon * t$	croissance quadratique s(t) = $\epsilon \frac{t^2}{2}$
Bruit blanc	bruit blanc avec une déviation σ	déviation standard $\sigma_\theta = \sigma * \sqrt{\delta t * t}$	bruit blanc de second ordre l'erreur $\sigma_s(t) = \sigma t^{\frac{3}{2}} * \sqrt{\frac{\delta t}{3}}$
effet température	biais résiduel dépent de la température	Tout biais résiduel est intégré dans l'orientation, ce qui provoque une erreur d'orientation qui croît linéairement avec le temps	Tout biais résiduel provoque une erreur de position qui augmente quadratiquement avec le temps
calibration	Les erreurs déterministes dans les facteurs d'échelle, linéarité , alignements et accélération	Dérive d'orientation proportionnelle à la vitesse et à la durée du mouvement	Dérive de position proportionnelle à la cadence au carré et à la durée de l'accélération
instabilité du biais	Fluctuations de biais, généralement modélisées comme une marche aléatoire de biais	bruit de second ordre	Une marche aléatoire de troisième ordre en position

Tableau 1.2 – Tableau récapitulatif de différents types de bruits pour les deux capteurs accéléromètre et gyroscope

Le tableau 1.2 donne les différents bruits de mesure, ainsi que leur influence potentielle sur l'orientation et sur la position. Ainsi, une centrale inertielle permet d'avoir une estimation très rapide de la position et de l'orientation, mais les intégration successives dans le temps peuvent générer des erreurs importantes. A contrario, le SLAM, traité comme un capteur qui prend en entrée une séquence d'images et fournit en sortie une pose (position et orientation), souffre d'un temps

de calcul important dans toutes les approches présentées. Sa fréquence est limitée par la fréquence d'acquisition de la caméra d'une part et par la puissance de calcul d'autre part, mais il ne présente pas de dérives cumulées et reste donc précis.

Dans cette section, nous présenterons des techniques de couplage entre ces deux capteurs, afin d'améliorer la qualité et la précision du tracking tout en gardant une rapidité de traitement. Dans la littérature, il existe deux formes de couplage qui permettent de fusionner plusieurs sources de données : le couplage fort et le couplage faible. Nous présenterons ces deux méthodes, avec les notations suivantes :

— I : représente le repère centrale inertielle.
— C : représente le repère de la caméra mono.
— W : représente le repère monde.
— OB : représente le repère objet (local) pour les augmentations.
— $\vec{A}_I$: représente le vecteur A exprimé dans le repère de la centrale inertielle.
— $\vec{A}_C$: représente le vecteur A exprimé dans le repère de la caméra.
— $\vec{\omega}_{Im}$: représente la vitesse angulaire mesurée par la centrale inertielle dans le repère de la centrale inertielle.
— $\vec{a}_{Im}$: représente l'accélération mesurée dans le repère de la centrale inertielle.
— $\vec{\omega}_{Ir}$: représente la vitesse angulaire réelle (sans bruit) de la centrale inertielle dans le repère de la centrale inertielle.
— $\vec{a}_{Ir}$: représente l'accélération réelle (sans bruit) de la centrale inertielle dans le repère de la centrale inertielle.

1.4.1 Le couplage fort

Le couplage fort est une forme de fusion de données qui consiste à fusionner les éléments dans une même structure et fournir un résultat final. En d'autres termes, il existe une interdépendance entre les différents systèmes de calcul. Dans la fusion on parle généralement de plusieurs capteurs donc, pour pouvoir se localiser correctement il faut définir les repères de localisation et les données de sortie de chaque capteur, leurs repères, et les notations de repères.

Dans la littérature, plusieurs algorithmes de fusion ont été développés. Toutes les approches gravitent autour d'une approche générique présentée dans la figure 1.11

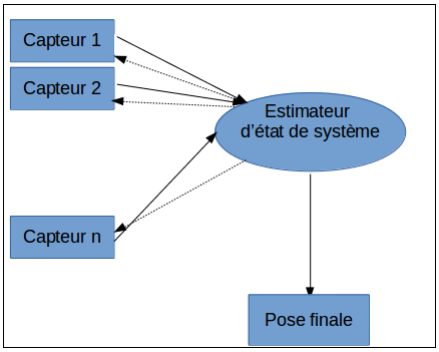


FIGURE 1.11 – Architecture générique d'un couplage fort entre n capteurs

Une approche de couplage fort entre un SLAM stéréoscopique basé sur une méthode directe et une centrale inertielle a été développée [9, 96]. Dans ces travaux, un filtre de Kalman étendu a été

exploité pour la fusion de données (inertielles et SLAM) pour l'estimation d'une pose. Une approche de couplage fort pour un SLAM mono caméra, basée sur une méthode directe, est également proposée [18]. Afin de réaliser ce type de couplage, une architecture basée sur trois threads a été proposée avec :

— Thread 1 (T1) : Thread qui s'exécute avec un fréquence correspondante à la fréquence de l'application de réalité augmentée qui calcule le rendu. Ce thread, est appelé une fois, à chaque calcul d'une nouvelle image de rendu. Il estime les mouvements de la caméra en utilisant une optimisation non linéaire et fournit une carte semi-dense.

— Thread 2 (T2) : Thread qui crée une deuxième carte semi-dense en se basant uniquement sur les images acquises et plus précisément sur les pixels de fort gradient d'une image. Ce thread est destiné uniquement pour le tracking de la caméra.

— Thread 3 (T3) : Thread qui estime une carte dense-complète de la scène. Ce thread s'exécute avec une fréquence relativement faible.

Cette approche est efficace pour une exécution en temps réel avec un système proche d'un ordinateur de bureau (3.5 GHz CPU, 8 G.O RAM). Cette méthode a été testée avec une caméra a 30 Hz et CI a 100 Hz. La précision de l'approche a été validée par une comparaison entre une trajectoire enregistrée par un système de caméras de tracking 'VICON' et le système de couplage fort. Dans cette approche les tests sont faits sur des séquences enregistrées. Les latences mesurées sont des latences de calcul algorithmique. Après l'intégration dans un système de réalité augmentée et lorsque ce système s'exécute côte à côte avec une application de réalité augmentée. Une répartition de ressources informatiques (CPU +GPU) entre les deux applications induit une chute de performances du tracking. Cette chute est justifiée par l'absence de ressources informatiques suffisantes. Cette méthode se base sur une technique de SLAM direct qui, malgré son couplage avec une centrale inertielle aura toujours le problème de méthodes directes citées dans la section précédente.

D'autres travaux proposent de coupler fortement un ensemble de quatre caméras avec une centrale inertielle [77]. Dans ces travaux une carte 'FPGA' externe est utilisée pour alléger certaines parties du SLAM. Étant donné que le flux de 4 caméras est important à traiter, les opérations de traitement d'images et l'extraction de primitives sont faites d'une manière logique sur une carte indépendante 'FPGA'. Cette carte a permis d'accélérer le temps de traitement en réduisant le temps d'utilisation de CPU pour les opérations d'extractions de primitives sur les images. Une synchronisation entre les données de la caméra et les données de la centrale inertielle est faite pour avoir un point de référence entre les événements (caméras et CI). Une implémentation 'FPGA' est faite sur la carte qui permet de transformer les images en entrée aux primitives rapidement sans consommer autant d'énergie et de temps de calcul. La figure 1.12 montre l'implémentation mathématique du filtre Sobel pour l'extraction de contours après l'application d'un noyau gaussien sur une image d'entrée. L'implémentation FPGA de ces opérations permet d'alléger le traitement sur le processeur.

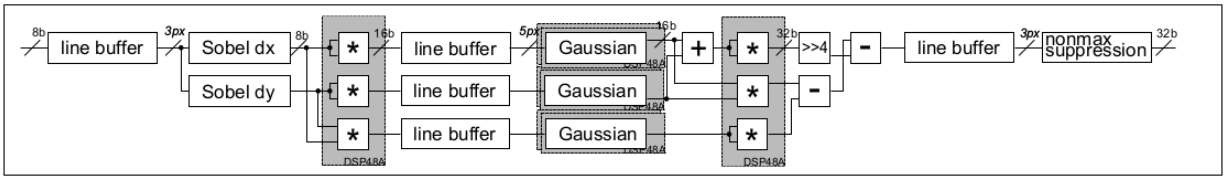


FIGURE 1.12 – Implémentation logique d'opérateur d'extraction de points d'intérêts sur un carte FPGA [77]

Après un traitement sur une carte externe, une procédure de couplage fort sur les données inertielles et les données SLAM est appliquée. Le couplage se fait suivant un modèle mathématique qui

consiste à optimiser une fonction d'énergie.

Un couplage fort entre les données inertielles et les données du SLAM a également été proposé [83]. Étant donné que le couplage fort de données monoculaires et inertielles est un système non-linéaire, l'estimation initiale joue un rôle très important dans la fiabilité de l'algorithme de détection. Une estimation de la pose de départ est faite lors de la procédure d'initialisation en se basant uniquement sur le SLAM. Entre temps les données inertielles sont acquises et le biais est estimé en fonction du décalage entre les poses SLAM et inertielles. Les données SLAM et CI sont séparées dans la phase d'initialisation. Une fois que la pose est déterminée par le SLAM, le calcul de poses CI et biais sont fait en se basant sur l'estimation SLAM et la phase d'initialisation s'achève. L'utilisation d'une fenêtre glissante permet de limiter l'augmentation de la complexité algorithmique malgré l'augmentation de graphes. Cette méthode a prouvé sa robustesse et sa précision sur différents benchmarks. Une implémentation à été également réalisée pour exécuter l'approche sur un téléphone portable (iPhone7 Plus). Elle prouve que l'algorithme est adapté pour l'utilisation avec des dispositifs mobiles. Cependant, l'utilisation de cet algorithme dans une application de réalité augmentée n'est pas prouvée encore. L'approche a des lacunes, et elle n'est pas testée avec des données acquises directement. Le test effectué sur le téléphone portable a prouvé que les dérives sont éliminées avec la technique de fermeture de boucles mais n'est pas concluant sur le temps de latence et sur l'aspect temps réel de cette approche.

Un couplage fort entre un système de vision et un système de données inertielles dans le même estimateur d'états a été développé afin de résoudre le problème de flou de mouvements qui cause des perturbations sur le système d'estimation de pose par vision [51]. Ce système est comparé à un système de filtrage statique largement utilisé (filtre de KALMAN). Dans ce travail, la fusion est faite dans les deux sens :

— Dans un estimateur de pose qui représente un filtre statique

— Dans la prédiction de données pour orienter le détecteur de segment dans la partie traitement d'images

Les données du système de vision sont exploitées aussi pour recalibrer la centrale inertielle qui dérive après un intervalle de temps. Dans cette approche, et malgré l'utilisation d'une CI qui mesure les vitesses angulaires et les accélérations linéaires, les vitesses linéaires sont fournies par la partie vision ce qui limite l'estimation de mouvements (translations), si les mouvements de la caméra sont rapides. La technique de correction de flou utilisée ici ne permet pas de corriger les segments parallèles dont la séparation est comparable à la taille de flou de mouvement local. L'architecture et la communication entre les données sont illustrées dans la figure 1.13.

1.4.2 Couplage faible

Le couplage faible repose sur le principe de fusion de différentes sources de données dans le but d'obtenir une information plus précise et plus rapide. Les différentes sources doivent fournir des données homogènes (i.e : poses SLAM dans le repère Monde (W) et poses CI dans le repère monde (W)) ou des types différents (i.e : poses SLAM dans le repère Monde (W), poses CI dans le repère CI (I)) ou des données hétérogènes (i.e : poses SLAM dans le repère monde (W), vitesse angulaires et accélération dans le repère CI (I)). Dans la littérature, la plupart des travaux ont utilisé une variante de filtre de KALMAN pour modéliser le filtre final (fusionneur).

Le couplage faible associe différentes sources des données. Les fréquences d'acquisition de différentes sources ne sont pas généralement identiques. La fréquence d'un SLAM est entre 10 et 50 Hz. La fréquence d'une CI est entre 50 et 1000 Hz. Ce gap entre les fréquence entraîne parfois une inconsistance d'estimation d'état.

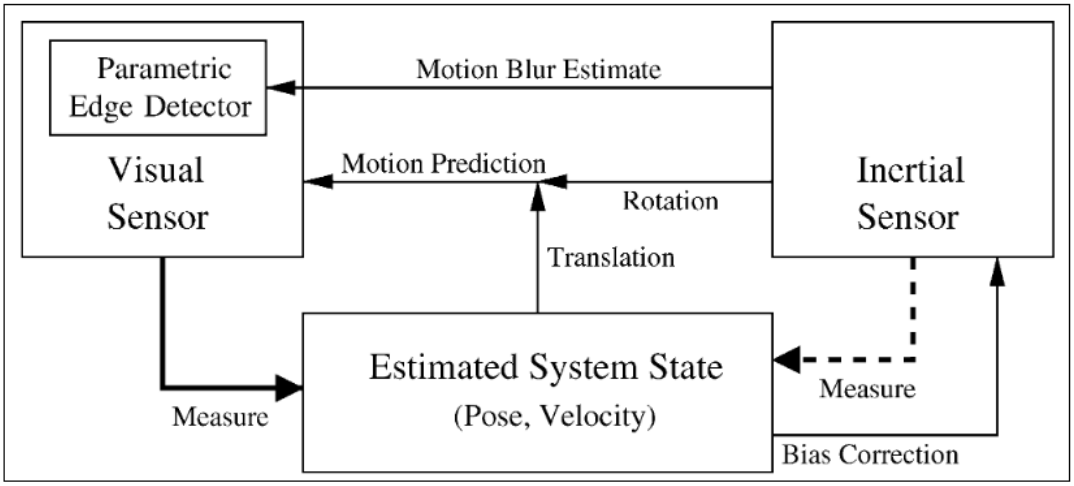


FIGURE 1.13 – Stratégie de fusion de données basée sur un couplage fort et une communication intra-modules pour l'amélioration de performances [51]

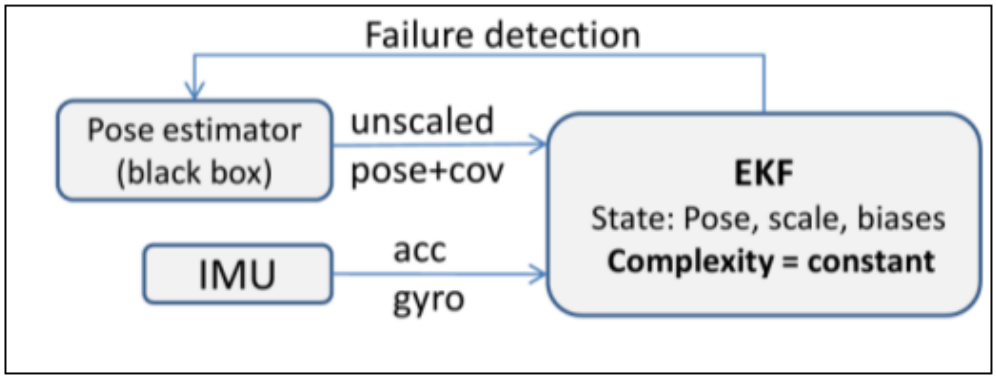


FIGURE 1.14 – Approche de couplage faible [104]

Un filtre de Kalman étendu avec un vecteur d'état de 24 éléments a été utilisé [104] avec l'équation suivante :

$$x = [p_w^{IT} \quad v_w^{IT} \quad q_w^{IT} \quad b_a^T \quad b_\omega^{IT} \quad \lambda p_I^C \quad q_I^C] \tag{1.12}$$

. L'erreur d'estimation est alors représentée par :

$$\hat{x} = [\Delta p_w^{IT} \quad \Delta v_w^{IT} \quad \delta q_w^{IT} \quad \Delta b_a^T \quad \Delta b_\omega^{IT} \quad \Delta\lambda \quad \Delta p_I^C \quad \delta q_I^C] \tag{1.13}$$

où : p_w^{IT} est la position 3D
v_w^{IT} est la vitesse de déplacement
q_w^{IT} est le quaternion de rotation b_a^T est le biais d'accéléromètre sur 3 axes
b_ω^{IT} est le biais du gyroscope sur 3 axes
λp_I^C le facteur d'échelle entre le monde réel et les estimations par vision
q_I^C est le quaternion de passage entre le repère caméra et le repère CI.

Dans cette approche les inconsistance sont traitées directement par un rejet de la valeur détectée. La fréquence SLAM (PTAM) est fixée à 20 Hz et la fréquence de la CI à 75 Hz. La fréquence de mise à

jour du filtre est fixé sur la fréquence de la CI soit 75 Hz. L'intégration de la constante λ dans le filtre permet d'avoir une correction sur le facteur d'échelle causé par les SLAMs monoculaires. Le filtre a montré son efficacité à estimer le facteur d'échelle même si la valeur initiale (phase initialisation) introduite est trop décalée de la valeur réelle.

Le couplage est aussi parfois fait par un filtre de Kalman basé sur l'erreur de l'état (error state Kalman Filter) [104]. Le filtre optimise le vecteur d'état
$x = [P_{WI} \quad v_{WI} \quad q_{WI} \quad b_a \quad b_\omega]^T$, où P_{WI} est la position 3D, v_{WI} est la vitesse 3D, q_{WI} est la rotation, b_a est le biais de l'accéléromètre, et b_ω est le biais gyroscopique. Notons que le vecteur d'état est de plus faible dimension que dans le cas précédent. Il contient 16 éléments soit 10 éléments de moins par rapport la technique précédente [38]. L'objectif est d'estimer la position (3), vitesse (3), quaternion(4), biais accéléromètre (3), biais gyroscope (3). Afin de s'affranchir des limitations de la synchronisation de données, une hypothèse est fixée sur la fréquence de deux capteurs. Le SLAM et la CI tournent avec des fréquences constantes. La synchronisation des données est réalisée grâce à l'utilisation d'une fenêtre glissante permettant l'enregistrement des données passées pour une exploitation ultérieure.

Le couplage faible est une approche qui se base sur la notion d'indépendance où les capteurs ne sont pas forcément liés. Les estimations de poses sont faites en local. C'est uniquement le résultat final envoyé à un filtre final qui traitera les poses. Le coût en temps de calcul de ce filtre est très bas par rapport au couplage fort. Son architecture est extensible rapidement car les capteurs fonctionnent en mode autonome, il est très simple de rajouter un nouveau capteur ou de retirer un capteur de l'architecture. Dans la littérature, et pour le couplage faible les différentes parties sont considérées comme des boîtes noires (qui ne sont pas modifiables). Le SLAM est considéré comme une boîte noire qui prend en entrée une séquence d'images et renvoie une pose, idem pour la CI qui renvoie la pose en fonction des mouvements du corps rigide dont elle est attachée.Il est donc facile de remplacer toutes les composantes de cette architecture sans trop modifier la technique ou l'architecture de couplage. L'autonomie de chaque partie dans le système de couplage faible a un impact direct sur la synchronisation et la cohérence de données. Avec le cumul de retard pour chaque capteur, la fusion de données non synchronisées est possible. Des problèmes d'inconsistances peuvent apparaître à cause de différentes fréquences de chaque capteur du système.

Dans ce travail, nous avons donc choisi le couplage faible comme méthode de fusion pour deux raisons :

— le simplicité de modélisation et la complexité algorithmique faible de calcul

— l'extensibilité et la flexibilité de cette méthode pour rajouter/supprimer des composantes.

1.5 Conclusion

Dans ce chapitre nous avons vu les différentes techniques de localisation 3D exploitées pour la réalité augmentée. Ces techniques sont classées dans deux catégories, lorsque la scène réelle est connue a priori, et celle sans connaissance de la scène réelle. La localisation avec des connaissances a priori nécessite des installations, préparation, et rajout de contraintes dans l'environnement. L'avantage de cette famille de méthodes est la stabilité, car l'augmentation et la localisation sont attaché à un marqueur spécifique. Dans ce cas quelque soit l'évolution de la scène, la localisation demeure stable. En revanche, cette famille n'assure pas une localisation dans des environnements inconnus, l'installation de marqueurs et la configuration devient très contraignantes dans les grandes échelles. Les méthodes de localisation sans connaissance a priori, permettent d'évoluer dans un environnement quelconque sans préparation préalable. Généralement elles sont basées sur des algorithmes de vision (SLAM) et sur le principe de la navigation inertielle (CI).

La famille de SLAM offre généralement (dans les meilleures conditions) une localisation précise qui ne dérive pas beaucoup au cours du temps. Cependant elle souffre généralement d'une complexité algorithmique qui nécessite un temps de calcul important et une utilisation et occupation de ressources informatiques. La CI fonctionne en mode local, elle n'a pas besoin d'une connaissance a priori pour estimer les poses. Le calcul est très rapide et simple pour avoir une pose finale. Contrairement au SLAM qui souffre d'une complexité algorithmique importante. En revanche la CI est fiable sur des périodes relativement courtes, mais a long terme apparaîtra des dérives, mentionnées dans le tableau 1.2. Ces dérives limitent l'utilisation de la CI sur des périodes très courtes.

La fusion entre les deux méthodes permet d'avoir les avantages de deux capteurs (i.e rapidité et simplicité de la navigation inertielle et précision du SLAM). Par conséquent, deux grandes catégories de fusion de données sont développées dans la littérature. Le couplage fort consiste à fusionner toutes les informations brutes issues de tous le capteurs dans un seul filtre et fournir un résultat final, tandis que le couplage faible consiste à exécuter les différents modules dans des threads indépendants permettant finalement de récupérer les données traitées localement par chaque capteur pour une fusion finale. La majeure différence entre ces deux techniques est le temps de calcul qui est plus important pour le couplage fort, l'extensibilité et la flexibilité de l'approche qui est facile pour le couplage faible et très difficile pour le couplage fort, la facilité d'implémentation est essentiellement difficile pour le couplage fort alors qu'il est aisé pour le couplage faible.

Dans la suite de ce mémoire, nous utiliserons la variante [85] du SLAM, dont nous disposons d'une version exclusive sur une carte embarquée (UDOOx86) dans cadre du projet REVE5D. La méthode de couplage choisie est le couplage faible pour sa facilité de calcul, son implémentation, et son extensibilité. Le filtre de couplage sera présenté dans le chapitre suivant.

CHAPITRE 2

CHOIX D'UN FILTRE POUR LE TRAITEMENT DES DONNÉES

2.1 Introduction

Dans le chapitre précédent, nous avons présenté une variété de techniques de localisation pour les applications de réalité augmentée (RA). Afin d'avoir un tracking stable et en temps réel, il est préconisé d'appliquer une fusion algorithmique entre deux ou plusieurs techniques, par exemple l'approche basée sur le SLAM et celle basée sur la navigation inertielle. De ce fait, un filtre doit être développé pour assurer cette fusion. Le système choisi est un couplage faible, qui permet de fusionner n capteurs (physiques ou algorithmes) qui fonctionnent en mode autonome.

Dans ce chapitre nous présenterons différentes techniques de filtrage mentionnées dans la littérature, ainsi qu'une comparaison entre ces différentes méthodes. Ce chapitre commence par un parcours de quelques notions mathématiques utilisées par la suite, puis une présentation des différents filtres est donnée en introduisant le filtre Complémentaire, le filtre de KALMAN, et le filtre particulaire. Nous présenterons ensuite la simulation algorithmique utilisée pour générer les données synthétiques afin d'expérimenter certaines techniques et tester leurs performances. Finalement une comparaison sera effectuée entre les différent filtres en se basant sur les deux critères (précision et rapidité d'exécution). Ce chapitre permet d'identifier le type de filtre à utiliser pour la suite de la thèse avec les avantages associés. L'objectif est d'identifier le filtre adéquat pour une exécution sur les supports à faible puissance de calcul comme cartes embarquées.

2.2 Notions mathématiques

Les capteurs opèrent dans des repères différents (local ou global). Comme le montre la figure 2.1, un repère 3D est présenté par un point d'origine O et trois vecteurs unitaires le long de trois axes x, y, et z. Afin d'exprimer les différentes valeurs dans différents repères il faut savoir passer d'un repère à un autre. Admettons que (X_c, Y_c, Z_c) et (X_w, Y_w, Z_w) soient les coordonnées d'un point P respectivement dans le repère caméra et dans le repère monde. Alors on écrit le passage entre les

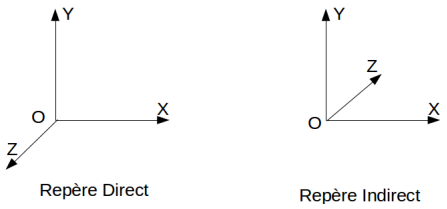


FIGURE 2.1 – Présentation de deux repères 3D direct et indirect

deux repères de la façon suivante :

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = M_c^w \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \qquad \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = M_w^c \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.1}$$

où M_c^w et M_w^c sont les matrices de passages homogènes qui permettent la transition entre les repère caméra et monde et entre les repère monde et caméra respectivement. Ces matrices s'écrivent de la façon suivante :

$$M_c^w = (M_w^c)^{-1} \quad et \quad M_w^c = \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{2.2}$$

où $R_{3\times3}$ est une matrice de rotation 3D, et le vecteur $T_{3\times1}$ est le vecteur de translation entre les origines des deux repères.

Dans la suite de ce mémoire, le repère monde R_w sera fixe, tandis que les repères de la caméra R_c et de la CI R_i seront mobiles. Pour des entités évoluant d'une manière continue dans le repère monde, fixe, il se pose le problème de déterminer leur position et leur orientation dans ce repère de référence. Le problème d'estimation de la rotation, ou bien attitude, a été soulevé et formulé par Wahba [103]. Les rotations dans l'espace peuvent être représentées par des matrices, via des angles d'Euler, ou par des quaternions, représentant une rotation autour d'un axe.

2.2.1 Matrices

Le changement d'orientation entre deux repères peut être exprimé par une simple matrice de rotation R de mobile vers fixe, cette matrice se décomposant par exemple en trois rotations successives d'angles α, β, γ, suivant les axes du repère courant x, y et z. Les matrices de rotation associées sont :

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \tag{2.3}$$

$$R_y(\beta) = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix} \tag{2.4}$$

$$R_z(\gamma) = \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

La matrice de rotation totale s'exprime alors en appliquant successivement, par multiplication, les rotations selon chaque axe, pour obtenir une matrice de rotation R qui satisfait les propriétés suivantes : $R^T = R^{-1}$, $RR^T = I$, et $\det R = 1$, où I est la matrice identité. Pour avoir plus de

détails sur les rotations matricielles on pourra consulter la référence [28]. Une des difficultés de cette méthode est que les résultats diffèrent selon l'ordre dans lequel on effectue cette multiplication. Par exemple, en appliquant les rotations successivement selon les axes x, y et z, on obtient la matrice $R = R_z(\gamma)R_y(\beta)R_x(\alpha)$ qui s'écrit de la façon suivante :

$$R = \begin{bmatrix} cos(\alpha)cos(\beta)cos(\gamma) & -cos(\alpha)cos(\beta)sin(\gamma) & cos(\alpha)sin(\beta) \\ -sin(\alpha)sin(\gamma) & -sin(\alpha)cos(\gamma) & \\ sin(\alpha)cos(\beta)cos(\gamma) & -sin(\alpha)cos(\beta)sin(\gamma) & sin(\alpha)sin(\beta) \\ +cos(\alpha)sin(\gamma) & +cos(\alpha)cos(\gamma) & \\ -sin(\beta)cos(\gamma) & sin(\beta)sin(\gamma) & cos(\beta) \end{bmatrix} \tag{2.6}$$

Lorsque les angles de rotation α, β et γ sont suffisamment petits, cette expression peut être simplifiée. En particulier, l'équation dynamique de la rotation R en utilisant les vitesses angulaires sur 3 axes $\dot{\alpha}$, $\dot{\beta}$, $\dot{\gamma}$ peut être présentée sous la forme suivante :

$$\dot{R} = \begin{bmatrix} 0 & -\dot{\gamma} & \dot{\beta} \\ \dot{\gamma} & 0 & -\dot{\alpha} \\ -\dot{\beta} & \dot{\alpha} & 0 \end{bmatrix} R \tag{2.7}$$

2.2.2 Quaternions

Une rotation peut aussi être représentée par un angle de rotation θ autour d'un axe $\vec{u}$. Plusieurs formes mathématiques sont définies dans la littérature pour formaliser cette présentation. Le modèle le plus répandu est celui du quaternion. Un quaternion unitaire est représenté par $q = [w \quad q_v^T]^T$ avec $w = cos(\frac{\theta}{2}) \in \mathbb{R}$ et $q_v = sin(\frac{\theta}{2})\vec{u} \in \mathbb{R}^3$. Ce quaternion unitaire vérifie la relation $w^2 + q_v^T q_v = 1$. Le quaternion identité, le quaternion conjugué, et la norme sont définis respectivement par :

$$q_{id} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T \tag{2.8}$$

$$\bar{q} = [w \quad -q_v^T]^T \tag{2.9}$$

$$\|q\| = \sqrt{w^2 + (q_v^T q_v)^2} \tag{2.10}$$

et le produit de deux quaternions unitaires

$$q_1 = \begin{bmatrix} w_1 & q_{v1}^T \end{bmatrix}^T \quad et \quad q_2 = \begin{bmatrix} w_2 & q_{v2}^T \end{bmatrix}^T \tag{2.11}$$

s'écrit :

$$q_1 \otimes q_2 = \begin{bmatrix} w_1 w_2 - q_{v1}^T q_{v2} & w_1 q_{v2}^T + w_2 q_{v1}^T + (q_{v1} \times q_{v2})^T \end{bmatrix}^T \tag{2.12}$$

La concaténation de plusieurs rotations q_A^B, q_B^C, q_C^D de A vers B, de B vers C, et de C vers D respectivement permet de déterminer la rotation totale q_A^D de A vers D :

$$q_A^D = q_C^D \otimes q_B^C \otimes q_A^B \tag{2.13}$$

La rotation d'un point $p(X, Y, Z)^T$ par un quaternion q s'écrit :

$$p_{rot} = q \otimes p \otimes \bar{q} \tag{2.14}$$

L'équation dynamique d'un quaternion s'écrit, en fonction de vitesses angulaires $\dot{\alpha}$, $\dot{\beta}$ et $\dot{\gamma}$ sur les 3 axes x, y et z sous l'une des deux formes suivantes :

$$\dot{q} = \frac{1}{2} q \otimes \omega \text{ avec } \omega = [0 \quad \dot{\alpha} \quad \dot{\beta} \quad \dot{\gamma}] \tag{2.15}$$

$$\dot{q} = \frac{1}{2} \Omega(\omega) q \text{ avec } \Omega(\omega) = \begin{bmatrix} 0 & -\dot{\alpha} & -\dot{\beta} & -\dot{\gamma} \\ \dot{\alpha} & 0 & \dot{\gamma} & -\dot{\beta} \\ \dot{\beta} & -\dot{\gamma} & 0 & \dot{\alpha} \\ \dot{\gamma} & \dot{\beta} & -\dot{\alpha} & 0 \end{bmatrix} \tag{2.16}$$

Dans ce travail, nous avons utilisé les quaternions. En effet, seuls quatre paramètres sont à estimer, au lieu de 9 valeurs incluant des fonctions trigonométriques. De plus, malgré leur interprétation physique peu intuitive, les quaternions sont robustes face aux problèmes de singularité causés par les matrices de rotations, et le temps de calcul est rapide.

2.3 Filtres

Dans le chapitre précédent, nous avons évoqué la fusion de capteurs avec deux techniques de couplage. Le couplage fort consiste à utiliser les différentes sources de capteurs dans la même architecture et à les fusionner dans une unité unique pour avoir une estimation finale. Le couplage faible consiste à faire fonctionner les capteurs d'une manière autonome pour que, à chaque pas de rafraîchissement, chaque capteur envoie vers une entité générique son résultat. Les entités (unités de traitement) utilisées dans les deux approches de fusion sont réalisées par des filtres. Donc le filtre dans notre cas est une unité de traitement qui prend en entrée différentes informations des différents capteurs et sort un résultat. Le choix a été fixé sur un système de couplage faible grâce à son architecture extensible et facile à implémenter (cf chapitre 1, section couplage). Les données d'entrée sont les poses d'une caméra (issue du SLAM) et celle d'une (CI).

Dans la communauté de réalité augmentée, les filtres sont largement utilisés, que ce soit pour le couplage faible ou pour le couplage fort, pour fusionner des données de vision, odométrique, ultrason, inertielle... L'idée d'exploiter différentes sources de données vient de la robotique. L'utilisation de différentes sources des données permet à un robot de corriger d'une manière automatique sa trajectoire et de garder son équilibre. Cette approche est inspirée de la navigation humaine où la localisation du corps et l'identification de l'environnement externe sont faites via le système de vision, tandis que l'équilibre est traité par le système d'oreille interne. En plus de l'utilisation du SLAM, la réalité augmentée a beaucoup utilisé la technique de fusion de données pour améliorer la qualité de sa localisation 3D. Il existe une vaste littérature portant sur ce sujet (voir par exemple [11, 4]).Dans cette section, nous détaillons les trois grandes catégories de filtrage et fournissons les principaux modèles mathématiques associés.

2.3.1 Filtre complémentaire

Les travaux de [97, 55, 32] consistent à utiliser les propriété de complémentarité entre les différentes sources de données afin d'estimer une pose 3D. Dans [97], les données d'estimation d'une orientation sont issues d'une CI. Une combinaison entre les trois capteurs disponibles sur la CI (gyroscope, magnétomètre et accéléromètre) permet d'estimer les angles de rotation. De nombreuses techniques sont décrites dans la littérature qui permettent d'estimer une rotation à partir d'une triade de capteurs.

Dans les travaux de [55], Un filtre complémentaire est associé à un filtre de KALMAN pour traiter l'ensemble des données issues de la CI et du système de vision. Une logique floue est associée

pour obtenir le gain de filtre complémentaire. L'angle de lacet fourni par le système de vision est adopté pour corriger l'angle de lacet fourni par le magnétomètre. De plus, un nouvel algorithme est proposé, qui exploite l'erreur en tant que variable d'état du filtre de KALMAN. Le résultat de ce calcul est utilisé pour compenser l'angle obtenu par le gyroscope et les données de l'accéléromètre, du magnétomètre et de la vision. Un filtre complémentaire est utilisé pour estimer l'attitude. Selon les données de retour des erreurs et de leurs dérivées, une approche par la logique floue est adoptée pour ajuster le gain du filtre complémentaire dans le but d'améliorer la robustesse et la précision de l'algorithme proposé.

Cet algorithme d'estimation a été testé sur des drones pour une application d'atterrissage automatique. Des données de simulations et des données expérimentales sont utilisées pour évaluer les performances de ce système. Les résultats présentés sont prometteurs du point de vue du temps de traitement, de la précision, et de la robustesse. Cependant, cette méthode est développée et testée pour un seul angle. L'efficacité est aussi présentée pour un algorithme de vision qui calcule un seul angle. Le modèle utilisé est linéaire, le gain est estimé à partir d'une logique floue qui se base sur une donnée 1D et sur une combinaison entre un filtre de kalman et un filtre complémentaire.

Dans les travaux de [32], les données sont issues de différents capteurs. Un capteur visuel renvoie des poses à partir d'un certain algorithme de traitement d'images, et un autre calcule les orientations et positions à partir de mesures issues d'une CI. Le filtre complémentaire dans ce cas est utilisé comme un filtre de pondération. La différence de la fréquence d'estimation de la pose mène à une utilisation d'un filtre complémentaire pour gérer la disponibilité des données. L'utilisation de ce filtre pour des sources de données hétérogènes est peu adapté dans les applications de réalité augmentée.

D'un point de vue mathématique, considérons N mesures $z_i = X + n_i$, avec $i = 1, ..., N$ où X est la pose et n_i modélise le bruit de capteur.

Des fonctions de transfert complémentaires $F_i(s)$ sont définies de telle sorte qu'elles respectent la relation de complémentarité $\sum_{i=1}^{N} F_i(s) = 1$. Dans la modélisation des filtres, il est préconisé de choisir des fonctions de transfert passe haut et passe bas. Ainsi, à partir de variables s, l'estimation de $X(s)$, notée $\hat{X}(s)$, s'obtient de la façon suivante :

$$\hat{X}(s) = \sum_{i=1}^{N} F_i(s) z_i = X(s) + \sum_{i=1}^{N} F_i(s) n_i \tag{2.17}$$

L'équation 2.17 montre la structure linéaire du filtre complémentaire. Il existe d'autres structures qui permettent l'utilisation de filtres complémentaires non linéaires. Pour le calcul de rotations, la structure du filtre complémentaire présenté dans la figure 2.2 peut être utilisée. Cette figure montre la structure et la disposition d'un estimateur non linéaire de type filtre complémentaire. Les données de l'accéléromètre f et du magnétomètre h sont traitées par un algorithme de moindres carrées récursif afin d'estimer un quaternion dans le repère global. Le quaternion est ensuite envoyé vers un estimateur pour estimer le biais et corriger la rotation totale.

Dans les travaux de [107], illustrés par la figure 2.3, une formulation d'un filtre complémentaire basé sur la multiplication de quaternions est utilisée. Une combinaison entre les données d'un accéléromètre et d'un magnétomètre permet d'estimer l'orientation du système dans un référentiel global via l'utilisation d'un filtre complémentaire et exploite également les données gyroscopiques. Le système de filtre est défini par la relation suivante, où ω_x, ω_y et ω_z sont les vitesses de variation angulaires issues des capteurs :

$$\begin{cases} \begin{bmatrix} \dot{\hat{q}}_0 \\ \dot{\hat{q}}_1 \\ \dot{\hat{q}}_2 \\ \dot{\hat{q}}_3 \end{bmatrix} = \begin{bmatrix} -(\hat{q}_1\omega_x + \hat{q}_2\omega_y + \hat{q}_3\omega_z) \\ \hat{q}_0\omega_x - \hat{q}_3\omega_y + \hat{q}_2\omega_z \\ \hat{q}_3\omega_x + \hat{q}_0\omega_y - \hat{q}_1\omega_z \\ \hat{q}_1\omega_x - \hat{q}_2\omega_y + \hat{q}_0\omega_z \end{bmatrix} \otimes T \end{cases} \tag{2.18}$$

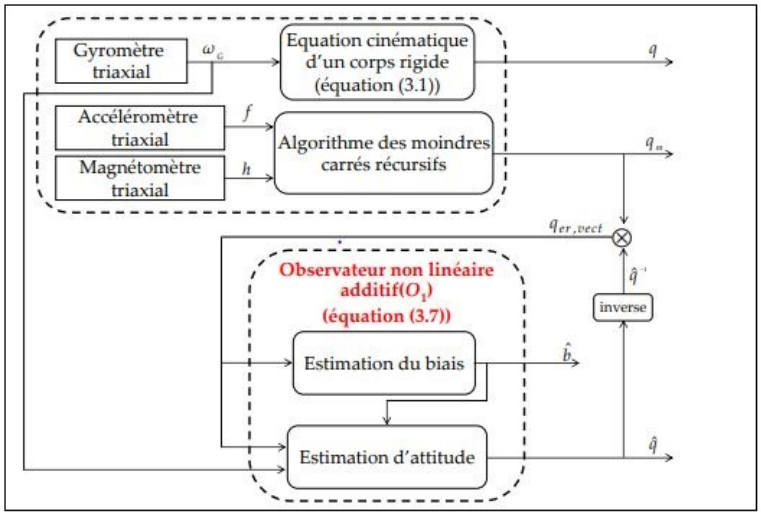


FIGURE 2.2 – Schéma bloc d'une structure non linéaire d'un filtre complémentaire pour estimer les quaternions, d'après [30]

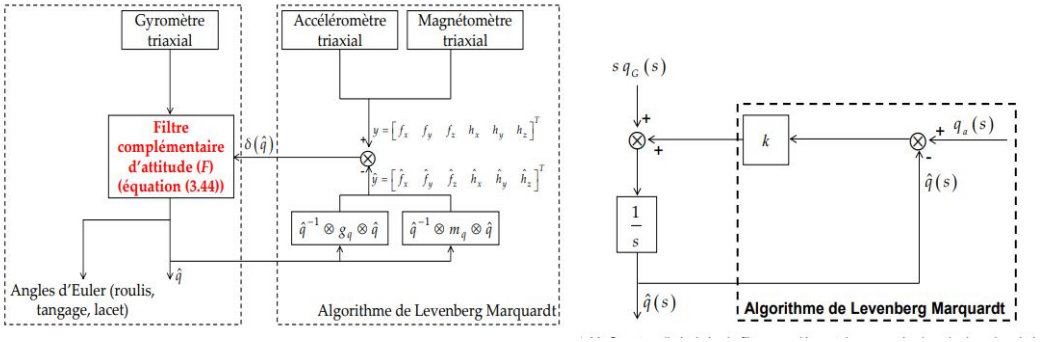


FIGURE 2.3 – Schéma bloc filtre complémentaire (gauche) représentation temporelle, (droite) représentation laplacienne [107]

La figure 2.3 montre que le filtre complémentaire fusionne les quaternions issus du bloc accéléromètre et magnétomètre et du bloc gyroscope, fusion qui après conversion peut amener vers une estimation d'angles d'Euler. Le quaternion estimé $\hat{q}$ est utilisé pour corriger les quaternions calculés en fonction des données issues des accéléromètres et des magnétomètres. Le fusionneur de type filtre complémentaire dans ce cas peut être traité comme une fonction qui prend en entrée les données gyroscopiques, et un quaternion dans un référentiel global issu d'un accéléromètre et d'un magnétomètre. Dans notre cas, et par analogie avec ce filtre complémentaire, le SLAM peut remplacer le bloc qui génère les rotations calculées à partir des données des accéléromètres et des magnétomètres. Donc le terme de correction T dans l'équation 2.18 devient le terme calculé à partir de données SLAM. L'équation présentée dans le système précédent permet ainsi la fusion entre le SLAM et le gyroscope et la détermination du biais, où :

$$\hat{s} = [0 \quad \hat{s}_x \quad \hat{s}_y \quad \hat{s}_z] = \hat{q}^{-1} \otimes s_q \otimes \hat{q} \tag{2.19}$$

L'erreur de modélisation peut être optimisée de la manière décrite dans [107] avec une méthode de régression minimisant le critère d'erreur $\epsilon(q)$ lié a $\delta(\hat{q})$:

$$\epsilon(q) = \delta(\hat{q})^T \delta(\hat{q}) \tag{2.20}$$

Le SLAM et le gyroscope peuvent être couplés avec une utilisation plus simplifiée de la notion filtre complémentaire présentée par l'équation 2.21. Le gyroscope permet de fournir des orientations précises sur un intervalle relativement court en raison des dérives qui sont causées par le bruit de mesure présenté dans le tableau 1.2. Le SLAM quant à lui, calcule des rotations précises dans un autre repère (monde) fixé au préalable ou fixé par le repère de la première image. Ce calcul est relativement important ce qui induit forcement un temps de traitement important et une latence d'estimation de l'orientation. Le filtre complémentaire correspondant utilisant des coefficients complémentaires s'écrit :

$$\hat{q} = \alpha q_g + (1 - \alpha) q_{SLAM} \tag{2.21}$$

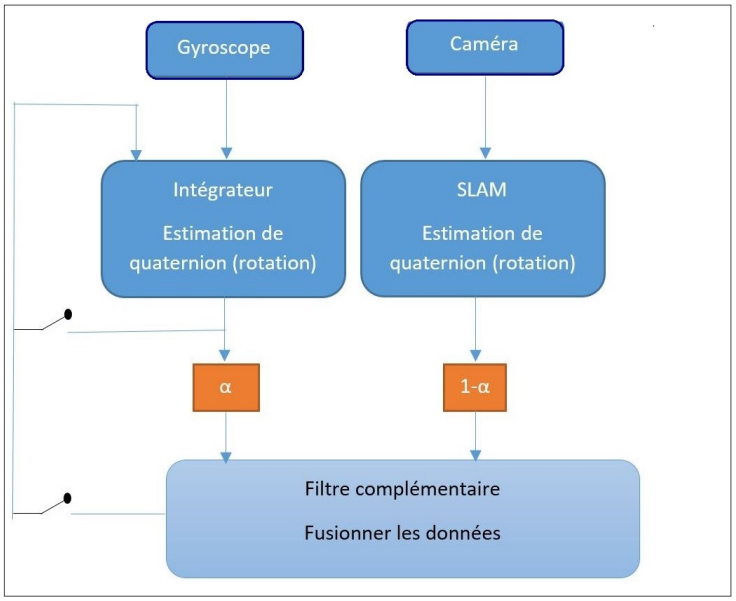


FIGURE 2.4 – Schéma bloc d'un filtre complémentaire fusionneur entre SLAM et gyroscope

Dans toutes les structurations proposées du filtre, les capteurs ont toujours la même fréquence d'acquisition. Dans le cas d'une fusion entre le SLAM et le gyroscope cette condition n'est pas validée. Afin de palier ce problème, les données gyroscopiques sont cumulées dans un quaternion et l'évaluation du filtre ce fait uniquement entre deux poses SLAM. En d'autres termes, les poses sont calculées par le gyroscope entre deux poses SLAM et ensuite, vérifiées par la pose SLAM. Étant donné que le SLAM est plus précis qu'un gyroscope sur des long intervalles de temps, l'importance est donnée au SLAM face au gyroscope (figure 2.4). La détermination du coefficient α est faite à partir de données de simulation qui seront présentées dans la section suivante. Un traitement de l'inconsistance des données, qui vérifie la cohérence de l'orientation SLAM avec celle estimée par le gyroscope, est présenté dans le chapitre suivant.

2.3.2 Filtre de KALMAN

Le filtre de KALMAN est un estimateur d'état récursif. Il est basé sur le principe de prédiction et correction utilisé dans de nombreuses applications. Plusieurs travaux sont développés autour de ce filtre en faisant varier quelques paramètres [72],[79] , [86], [106], [45], [43], [54], [74], [92], [78], [108]. Dans sa version de base, le filtre de KALMAN est développé pour les modèles dynamiques de forme linéaire, mais il est possible de l'utiliser dans des cas non linéaires avec certaines modifications structurales. Une modification consiste à linéariser le modèle à chaque pas de temps (filtre de KALMAN étendu). Le filtre de KALMAN étendu, le filtre de KALMAN non parfumé, et le filtre de KALMAN flou sont des versions avancées du filtre classique. La taille du vecteur d'état, les variables, le corps du filtre, les équations d'estimations changent d'un estimateur à un autre. Différentes représentations de rotations (quaternions, matrices...) sont utilisées dans les différents filtres de KALMAN qui définissent les systèmes de vision associés à une CI. Les matrices de rotations, les angles d'Eulers, et les quaternions sont présents, le biais des différents capteurs est parfois inclus dans le vecteur d'état. Dans certains filtre les matrices de passage entre les différents repères sont également incluses. Le choix du filtre et du type de vecteur d'état se base sur deux critères principaux :

— Type de couplage utilisé (fort ou faible)

— Type de système (linéaire ou non)

D'un point de vue mathématique, le filtre de Kalman développé par **Rudolf KALMAN** [105] repose sur l'estimation de l'état d'un système dynamique bruité. Le problème de l'estimation d'état d'un système en fonction des observations, notées z_i dans la suite, peut être divisé en trois parties :

— Le filtrage, qui consiste à estimer un état du système à l'instant k à partir des mesures précédentes $\{z_1, ..., z_k\}$;

— La prédiction, qui consiste à estimer l'état futur du système, à l'instant k, à partir de mesures précédentes $\{z_1, ..., z_l\}$ avec $l < k$;

— La correction, qui consiste à estimer un état de système à un instant k à partir d'observations sur un temps plus long $\{z_1, ..., z_l\}$ avec $l > k$.

Filtre de KALMAN linéaire

Supposons un système linéaire stationnaire discret :

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + w_k \\ y_{k+1} = Cx_k + Du_k + v_k \end{cases} \tag{2.22}$$

avec à chaque instant k :

— $x_k \in \mathbb{R}^n$ représente le vecteur d'état du système

— $u_k \in \mathbb{R}^m$ représente les données d'entrées du système

— $w_k \in \mathbb{R}^q$ représente les perturbations aléatoires appliquées au système

— $y_k \in \mathbb{R}^p$ représente le vecteur de mesures

— $v_k \in \mathbb{R}^p$ représente les perturbations aléatoires appliquées aux mesures

Les équations 2.23 à 2.27 sont la base du filtre de KALMAN linéaire. L'équation 2.23 est une estimation de l'état du système à priori en se basant sur l'état précédent et sur les entrées du vecteur contrôle. L'équation 2.24 est une estimation a priori de la covariance du système, elle présente notamment la mesure de la précision de l'état estimé. Ces deux équations représentent l'étape de prédiction du filtre. L'équation 2.25 représente le gain de KALMAN. L'équation 2.26 est l'estimation a

posteriori, elle consiste à corriger l'estimation a priori en utilisant l'innovation, qui consiste à exploiter les informations supplémentaires issues de différentes mesures. L'équation 2.27 représente la mise à jour de la matrice de covariance. Le filtre de Kalman est un outil efficace, qui permet d'obtenir une estimation optimale de l'état du système. En revanche, le filtre de KALMAN a certaines limitations. Il faut avoir une connaissance parfaite sur les matrices de covariance. Dans la plupart des cas et dans notre projet, les matrices ne sont pas parfaitement connus.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{2.23}$$
$$P_k^- = AP_{k-1}A^T + Q_{k-1} \tag{2.24}$$
$$G_k = P_k^- C^T (CP_k^- C^T + R_k)^{-1} = P_k^+ H_k^T R_k^{-1} \tag{2.25}$$
$$\hat{x}_k^+ = \hat{x}_k^- + G_k(y_k - C\hat{x}_k^-) \tag{2.26}$$
$$P_k^+ = (I - G_k C)P_k^- \tag{2.27}$$

Pour des raisons de simplifications, le filtre de KALMAN est souvent divisé en deux : une partie pour estimer la position et la vitesse de déplacement de l'objet, une partie pour estimer les rotations. Afin d'avoir une représentation linéaire pour le calcul d'orientation du filtre, les angles d'Euler sont choisis pour représenter les rotations donc le vecteur x contient trois composantes d'angles α, β, γ, et trois composantes de biais b_α, b_β, b_γ.

Algorithm 1 Filtre de KALMAN linéaire

Initialisation :
$X_0 = V_0$ et $P_0 = 10 * I_{6\times 6}$
A et B données par l'équation 2.28
$C = I_{6\times 6}$
for k =1 .. N
Prédiction :
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$
$P_k^- = AP_{k-1}A^T + Q_{k-1}$
Correction :
$P_k^- = AP_{k-1}A^T + Q_{k-1}$
$G_k = P_k^- C^T (CP_k^- C^T + R_k)^{-1} = P_k^+ H_k^T R_k^{-1}$
$\hat{x}_k^+ = \hat{x}_k^- + G_k(y_k - C\hat{x}_k^-)$
$P_k^+ = (I - G_k C)P_k^-$
fin for

L'algorithme 1 présente le déroulement des calculs du filtre de KALMAN pour le cas du modèle linéaire 2.28. Grâce à sa représentation linéaire les matrices A, B, et C du modèle sont constantes, et calculées une seule fois dans la phase d'initialisation. Les phases prédiction et correction sont faites en fonction des données d'entrées reçues par le vecteur de contrôle u_k et les mesures y_k. L'estimation de l'angle s'écrit donc de la manière suivante :

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\\ b_\alpha\\ b_\beta\\ b_\gamma\end{bmatrix}_{k+1} = \underbrace{\begin{bmatrix}1&0&0&-dt&0&0\\ 0&1&0&0&-dt&0\\ 0&0&1&0&0&-dt\\ 0&0&0&1&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1\end{bmatrix}}_{A}\begin{bmatrix}\alpha\\ \beta\\ \gamma\\ b_\alpha\\ b_\beta\\ b_\gamma\end{bmatrix}_{k} + \underbrace{\begin{vmatrix}dt&0&0\\ 0&dt&0\\ 0&0&dt\\ 0&0&0\\ 0&0&0\\ 0&0&0\end{vmatrix}}_{B}[g_x\ g_y\ g_z] + \begin{bmatrix}n_\alpha\\ n_\beta\\ n_\gamma\\ n_b\\ n_b\\ n_b\end{bmatrix} \tag{2.28}$$

Filtre de KALMAN étendu

Le filtre de KALMAN est un estimateur d'état adapté aux modèles linéaires avec un bruit gaussien. Afin d'exploiter le principe de filtre de KALMAN avec des modèles non linéaires il faut linéariser le modèle. Généralement la linéarisation passe par un schéma d'Euler d'ordre n, selon la précision recherchée. L'équation intégrée de la dynamique de quaternions 2.16 étant une forme non linéaire, son utilisation dans le filtre de KALMAN nécessite une opération de linéarisation. En utilisant un développement limité à l'ordre 1 de cette équation, on écrit la relation linéaire suivante entre deux quaternions successifs :

$$q_{k+1} = (I_{4\times 4} + \frac{\Delta t}{2}\Omega_k)q_k + w_k \tag{2.29}$$

Dans le cas non-linéaire, le passage entre deux états nécessite donc d'actualiser la matrice associée, comme dans l'équation ci-dessus Ω_k. Les équations 2.23 à 2.27 deviennent alors les équations 2.30 à 2.34, où les matrices en jeu sont actualisées à chaque pas de temps, et l'algorithmique du filtre de KALMAN étendu devient l'algorithme 2.

$$\hat{x}_k^- = A_{k-1}\hat{x}_{k-1} + B_{k-1}u_{k-1} \tag{2.30}$$
$$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1} \tag{2.31}$$
$$G_k = P_k^- C_k^T (C_k P_k^- C_k^T + R_k)^{-1} = P_k^+ H_k^T R_k^{-1} \tag{2.32}$$
$$\hat{x}_k^+ = \hat{x}_k^- + G_k(y_k - C_k\hat{x}_k^-) \tag{2.33}$$
$$P_k^+ = (I - G_k C_k)P_k^- \tag{2.34}$$

2.3.3 Filtre particulaire

Les travaux de [1, 7, 22, 23, 84] ont exploité la méthode de monte carlo séquentielle nommée aussi filtrage particulaire. Cette technique de filtrage est plus généraliste car elle supporte les modèles non linéaires et le bruit non gaussien. Par rapport au filtre de Kalman, l'exploitation de cette technique de filtrage reste peu répendue à cause de son temps de traitement. La figure 2.5 représente un modèle de markov caché. ce modèle est composé d'états cachés x_k, d' observations y_k, et de transitions $p(x_k|x_k - 1)$.

Les filtres présentés précédemment ne sont pas conçus pour les modèles non linéaires, ils sont optimaux pour les modèles linéaires et pour le bruit blanc gaussien mais ils sont limités dans les autres cas. Le filtre particulaire est une approche probabiliste qui supporte les modèles non linéaires et permet d'opérer sur plusieurs formes de bruits et non uniquement sur les bruits gaussiens. Dans

Algorithm 2 algorithme de filtre du kalman étendu

Initialisation :
$X_0 = q_0$ et $P_0 = 10 * I_{4x4}$
for k =1 .. N
Prédiction :
calcule A_{k-1}, B_{k-1} et C_{k-1}
$\hat{x}_k^- = A_{k-1}\hat{x}_{k-1} + B_{k-1}u_{k-1}$
$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1}$
Correction :
$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1}$
$G_k = P_k^- C_{k-1}^T (C_{k-1}P_k^- C_{k-1}^T + R_k)^{-1} = P_k^+ H_k^T R_k^{-1}$
$\hat{x}_k^+ = \hat{x}_k^- + G_k(y_k - C_{k-1}\hat{x}_k^-)$
$P_k^+ = (I - G_k C_{k-1})P_k^-$
end for

les travaux de [34], il est démontré que les filtres particulaires peuvent être utilisés pour la résolution de problèmes d'estimation optimaux.

Soit le système dynamique non linéaire défini par les équations :

$$\begin{cases} x_k = f_k(x_{k-1}, w_{k-1}) \\ y_k = h_k(x_k, v_k) \end{cases} \tag{2.35}$$

Le filtre particulaire ou filtre à particules consiste à approcher la densité probabiliste $p(x_k|y_{0:k})$ par des échantillons dites particules. En d'autres termes, à chaque instant $k-1$, une approximation de la distribution conditionnelle est réalisée. Une fonction de vraisemblance doit être mise en place afin d'adapter le comportement des particules à la dynamique du système. On suppose que l'évaluation de la fonction de vraisemblance est possible à chaque incrément de temps.

Le système (2.35), représente le modèle d'équations qui peut être utilisé pour caractériser les mouvements de la tête. Ce système peut être considéré comme un système discret non linéaire non gaussien. Dans ce système x_k et y_k désignent l'état du système et les sorties du système à l'instant k respectivement. Le modèle d'état peut être présenté par un modèle de Markov de premier ordre. Les observations sont indépendantes par rapport aux états. Les fonctions f et h représentent le processus déterministe et le modèle de mesures respectivement avec le bruit de process w_k et le bruit de mesures v_k. Pour compléter les spécifications du modèle une distribution d'initialisation ($k = 0$) doit être imposée, soit $p(x_0|y_0) \cong p(x_0)$.

Dans ces travaux [2], une utilisation de filtre particulaire avec le système d'échantillonnage **SIR** à été démontrée. Le système utilisé est composé d'un bloc de vision et d'une CI. Le modèle cinématique utilisé est celui proposé par [15, 16]. Le mouvement de la tête est donc estimé par un vecteur à 15 composantes $\{\theta \quad \omega \quad x \quad \dot{x} \quad \ddot{x}\}$, où θ représente les angles d'Euler en suivant la séquence **Z-Y-X**, ω est la vitesse angulaire et x, $\dot{x}$, $\ddot{x}$ sont les positions, vitesses, et accélérations linéaires du système. Les tests sont appliqués sur des données synthétiques. Les bruits sont estimés d'une manière synthétique en suivant un modèle Gaussien. Le filtre particulaire à montré une performance et une stabilité de tracking par rapport à un filtre de KALMAN étendu exécuté sur les mêmes données synthétiques.

Les travaux de [84] présentent une implémentation d'un filtre particulaire avec un mode d'échantillonnage différent et repose sur l'utilisation des quaternions pour l'estimation des rotations. Ce développement est basé sur une approche algorithmique génétique. Dans cette étude les performances de ce filtre en

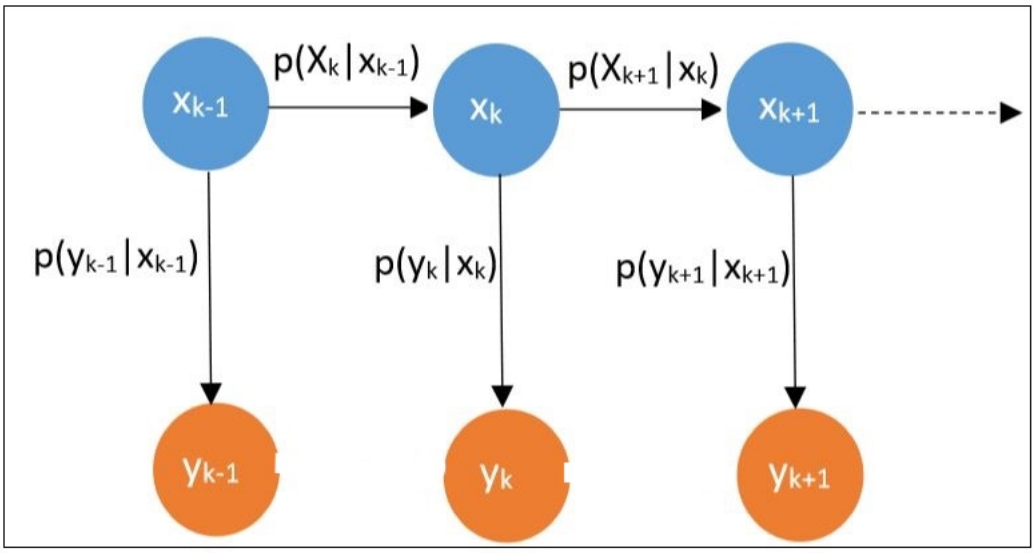


FIGURE 2.5 – Modèle de Markov caché pour un estimateur de poses

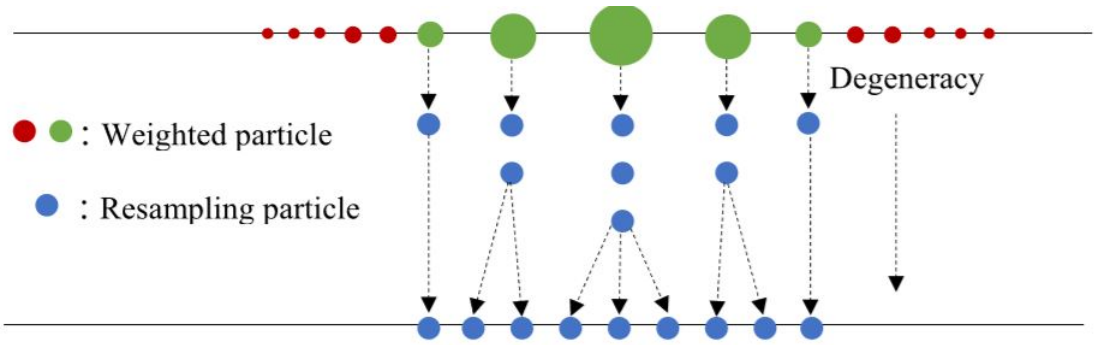


FIGURE 2.6 – Régénération de particules en fonction de leurs poids associé [84]

terme de précision sont comparées à celles d'un filtre de KALMAN étendu et d'un filtre de KALMAN non parfumé (unscented).

Le filtre particulaire comporte un paramètre important qui est le nombre de particules utilisées pour estimer une loi de distribution. Plus le nombre de particules est important plus le temps de calcul est conséquent. Le temps de calcul est un facteur à ne pas négliger dans les applications de réalité augmentée en raison de son importance pour la cohérence d'affichage. Cette cohérence se traduit généralement par une cohérence spatiale et temporelle. La technique de re-échantillonnage impacte aussi directement le temps de calcul. Plusieurs techniques ont été présentées dans la littérature, qui diffèrent par leur précision et leur temps d'exécution. parmi les techniques existantes on peut distinguer :

— systématique
— SIS
— génétique

Algorithm 3 Algorithme détaillé du filtre particulaire

Initialisation : $k = 0$
for i =1, .., N
$xp_0^{(i)} = p(x_0)$ Génération de particules
$w_0^{(i)} \propto q_0(Y_0 - h_0(xp_0^{(i)}))$
$\hat{xp_0^{(i)}} = \sum_{j=1}^{N} w_0^{(i)} \delta_{xp_0^{(i)}}$
endfor
for $k >= 1$
Prédiction :
for i=1, .., N
générer le bruit w_k
générer les particules $xp_0^{(i)} = f_k(\hat{xp_{k-1}^{(i)}}, w_k)$
endfor
Correction :
for i = 1, .., N
$w_k^{(i)} \propto q_k(Y_k - h_k(xp_k^{(i)}))$
endfor $\quad \hat{xp_k^{(i)}} = \sum_{j=1}^{N} w_k^{(i)} \delta_{xp_k^{(i)}}$
endFor

Famille	informations		
	Type de filtre	points forts	points faibles
filtre de KALMAN	FK simple	+ Matrices A et B fixes et calculées une seule fois + Meilleur estimateur d'état pour les modèles linéaires	- Opérationnel uniquement pour les modèles linéaire - Optimal uniquement pour le bruit gaussien
	FK étendu	Supporte les modèles non linéaires après une étape de linéarisation avec le développement de Taylor	
	FK non parfumé	+ supporte les modèles non linéaires après une linéarisation et les bruits non gaussiens	- Une étape de vérification supplémentaire sur n échantillons - Temps de calcul important
filtre particulaire	SIR	Supporte tous les modèles et tous les bruits + Précision d'estimation	- Temps de calcul important - Implémentation difficile - Définition d'une fonction de vraisemblance

Tableau 2.1 – Avantages et inconvénient de chaque technique de filtrage classées par catégories

Le tableau 2.1 présente les avantages et les inconvénients des différentes approches existantes pour le filtrage.

2.4 Simulation

Dans ce paragraphe nous présentons la mise en application des différentes techniques mentionnées précédemment. Ces techniques sont appliquées sur des données simulées. La simulation consiste à calculer une trajectoire quelconque et à produire ensuite les données issues de chaque capteur. Afin de produire des données réalistes plusieurs cas de figure sont envisagés : la distribution du bruit pour la CI, la fréquence d'acquisition de la CI, la fréquence de calcul des poses SLAM, le bruit SLAM, et la latence de calcul.

Les données de **la trajectoire réelle** sont simulées dans l'ordre suivant. Une fréquence fixe pour générer les données de la trajectoire. Cette fréquence est fixée à 10 kHz pour avoir une résolution supérieure à toutes les résolutions des capteurs utilisés. La trajectoire et la modification de la position sont basées sur **les accélérations linéaires** qui permettent après une double intégration de calculer les positions. Les rotations quant à elles, sont générées à partir de **vitesses angulaires** qui sont intégrées une fois au cours de temps. Une multitude de données peut être stockée en utilisant les accélérations linéaires et la vitesse de rotation (i.e la vitesse, la position, et l'orientation).

La simulation de la CI est faite à partir de la trajectoire générée, en se basant sur l'équation 1.11 qui représente le modèle d'un gyroscope. Les données de la vitesse angulaire (propre) sont issues de la trajectoire réelle. Ensuite, les données bruitées sont rajoutées en fonction de la spécification technique du gyroscope. Cette spécification est présente dans la fiche technique donnée par le constructeur. Pour les gyroscopes MEMS, les caractéristiques sont : le biais additif, le bruit aléatoire, l'évolution du biais en fonction de la température et la sensibilité. **Le biais constant ou additif** est un biais qui peut être mesuré à la sortie du capteur utilisé lorsque il est en mode repos. Ce biais est généralement constant sur des intervalles courts et peut évoluer sur une grande période. **Le bruit aléatoire** est dans la plupart de cas modélisé par une loi gaussienne. L'écart type de la loi gaussienne pour la modélisation de bruit aléatoire est fournit généralement par le constructeur. La trajectoire est supposée évoluer dans un environnement isotherme ainsi l'influence de la température interne sur les données est négligeable. La trajectoire générée doit également prendre en compte la plage de mesure de tous les gyroscope MEMS. Pour une étude plus profonde sur le comportement de différents filtres, le bruit additif et le bruit aléatoire du gyroscope sont modélisés à partir de 3 lois de probabilité :

- Loi uniforme qui permet d'avoir une distribution quasi uniforme de l'erreur entre $-\frac{\sigma}{2}$ et $\frac{\sigma}{2}$
- Loi gaussienne qui permet d'avoir une distribution gaussienne de l'erreur
- Loi gamma qui permet d'avoir une distribution gamma

La valeur de σ est calculée à partir de données enregistrées sur une CI au repos pendant une heure. C'est la moyenne de toutes les données acquise durant cette heure.

La simulation de l'accéléromètre est plus complexe. Les accélérations mesurées par l'accéléromètre sont un mélange de la projection de la gravité sur le capteur, les accélérations linéaires, un biais constant, et un bruit aléatoire. Les accélération linéaires de l'objet sont mesurées dans le repère local de l'accéléromètre. Cependant, la gravité est mesurée dans le repère global, donc on parle de la projection de la gravité sur le repère local de l'accéléromètre. Afin de calculer cette projection sur les axes de l'accéléromètre, les angles de rotation de la trajectoire sont utilisés. En utilisant les angles de rotation, les matrices sont calculées à chaque instant à l'aide de l'équation 2.6. La séquence utilisé pour calculer la matrice de rotation est **ZYX**. Le vecteur de la pesanteur $\vec{g}$ est multiplié par cette matrice. Après la détermination de la projection de la pesanteur dans le repère local de l'accéléromètre et la disponibilité des accélérations linéaires, il reste l'ajout du bruit représentant le

bruit de mesure du capteur accéléromètrique. Comme pour le gyroscope trois formes de bruit sont modélisés : gaussien, uniforme, et gamma.

La simulation du SLAM consiste a récupérer les données de la trajectoire (données propres) et à ajouter un bruit gaussien sur la sortie. Dans cette simulation on se base sur les deux hypothèses suivantes :

— Pas de dérives pour le SLAM

— Le SLAM est bien fonctionnel dans l'environnement de la trajectoire de synthèse (simulée)

En utilisant ce deux hypothèse, la simulation du SLAM sera simplifiée. Les données de la trajectoire réelle sont bruitée en utilisant les 3 types de bruits utilisés précédemment pour le gyroscope et pour l'accéléromètre.

Afin de déterminer l'écart type du bruit SLAM σ_{SLAM} deux expériences ont été réalisées. La première a consisté à faire éxécuter le SLAM sur des images fixes et calculer le biais constant. Dans cette expérience, le biais SLAM est quasiment nul. La deuxième expérience a consisté à étudier le comportement et la sortie du SLAM sur un système de type 'pendule'. Ce système oscille entre deux points A et B. Une caméra est montée sur le système et les acquisition SLAM et CI sont faites. Le bruit de sortie est estimé en fonction de la pose fournie par le SLAM et la position réelle. En répétant cette expérience le bruit dans les conditions spécifiques de l'expérience a été estimé à $\sigma_{SLAM} = 0.005\ rad.$ Les conditions de réalisation de cette expérience sont idéales pour le bon fonctionnement du SLAM. Pour avoir plus d'ouverture plusieurs SLAM sont simulés avec des bruits différents. Le σ_{SLAM} varie donc entre [0.001 - 0.1]. La fréquence du SLAM varie aussi en fonction de la vitesse de déplacement, du niveau de texture des images acquises, mise en correspondance de points, et performances du système sur lequel le SLAM est exécuté. Les fréquences du SLAM simulées sont les fréquence les plus proches de notre système : 50Hz, 25Hz, 20Hz, 15Hz, et 10Hz.

En résumé, pour cette simulation, plusieurs données sont générées :

— Données gyroscopiques : fréquence d'acquisition fixée à 1KHz, trois lois de bruits sont utilisées

— Données accéléromètriques : fréquence d'acquisition fixée à 1KHz, trois lois de bruits sont utilisées

— Données SLAM : fréquences variables, écart type variable, trois lois de bruit par fréquence et par écart type.

Les trajectoires ont été simulées sur différentes période pour observer tous les phénomènes possibles. Les accélérations linéaires sont générées d'une manière aléatoire sous forme de pics. Les rotations sont sur trois axes, représentant les mouvement possibles de la tête dans le cadre d'une visite culturelle ou dans le cadre d'utilisation professionnelle dans un environnement industriel. Les vitesses angulaires de rotations sur l'axe Y dans le repère direct (Figure. 2.1) sont plus importantes que les autres rotations et sont présentées dans le tableau (Tableau. 2.2).

Axe de rotation	Expression analytique
Axe Y	$\frac{\pi}{2} * sin(2\pi\omega t)$
Axe X	$\frac{\pi}{6} * sin(2\pi\omega t)$
Axe Z	$\frac{\pi}{12} * sin(2\pi\omega t)$

Tableau 2.2 – Les différentes expression Analytiques pour les rotations sur chaque axe pour la simulation

Type de bruit	Erreur FKE (rad)	Erreur FP (rad)
Gaussien	4	6
Uniforme	6	1.46
Gamma	6.21	2

Tableau 2.3 – Comparaison entre le filtre de kalman étendu (FKE) et le filtre particulaire en se basant sur la précision pour 1000 poses estimées

2.5 résultats

Afin d'analyser les performances de filtres et surtout le filtre de KALMAN et le filtre particulaire, la simulation théorique de données est adaptée aux conditions du projet.

Pour bien comprendre les besoins en terme de filtre les paramètres étudiées sont :

— La fréquence d'estimation de poses par l'algorithme SLAM et son impact direct sur les performances de filtre. La fréquence d'estimation de poses SLAM n'est pas fixe elle est relative aux ressources informatiques, vitesses de déplacement de la caméra, richesse de l'environnement en terme de texture.

— La distribution de bruit pour le SLAM et son impact sur les performances du filtre. Elle est due généralement aux problèmes d'acquisition de la caméra et aux erreurs d'interprétation de poses SLAM.

— La distribution de bruit et la variance de l'erreur pour la CI et leurs impacts sur le filtre. La variance de bruit de même capteur peut changer en fonction de la température ambiante, la fréquence d'acquisition, la durée d'utilisation.

Le but de faire varier les différents paramètres est d'observer l'impact direct de cette variation sur le comportement et le résultat du filtre.

Les données gyroscopiques et le SLAM sont récupérés à partir d'une trajectoire théorique de simulation. Une fois que toutes les données sont disponibles, une comparaison sur les performances du filtre complémentaire, du KALMAN étendu, et du filtre particulaire, a été réalisée.

La figure 2.7 présente les histogrammes de différentes formes de bruits utilisés pour la simulation. Le premier histogramme (haut) est généré à partir d'une loi normale avec un $\sigma = 0.1 rad/s$ pour 100 poses. Le deuxième (milieu) représente une répartition uniforme de d'erreur sur un intervalle de $[\sigma \quad \sigma]$. Tandis que la troisième (bas) représente une distribution gamma.

Le modèle de bruit calculé pour les capteurs utilisé dans le contexte du projet REVE5D est un modèle gaussien soit la première loi représentée par le graphique 2.7. La simulation d'autres bruits rentre dans l'objectif de trouver le meilleur filtre qui supporte la totalité de distributions dans le souci de rajouter un autre capteur de modèle de bruit différent. L'ajout d'autres capteurs rejoint la stratégie extensible de l'architecture développée (présentée dans le chapitre suivant).

Dans la simulation, le SLAM est aussi bruité. La nature du bruit est différente de celle du gyroscope (un biais constant et un bruit blanc). Dans notre cas on considère que le SLAM est affecté par un bruit non nul qui peut être représenté par les trois distributions gaussienne, uniforme, et gamma. La figure 2.8 montre les trois distributions et leurs histogrammes.

2.6 Conclusion et interprétations

Dans ce chapitre, nous avons mentionné les éléments mathématiques les plus utilisés pour définir les rotations et les calculer à partir de données gyroscopiques. Ensuite nous avons présenté les tra-

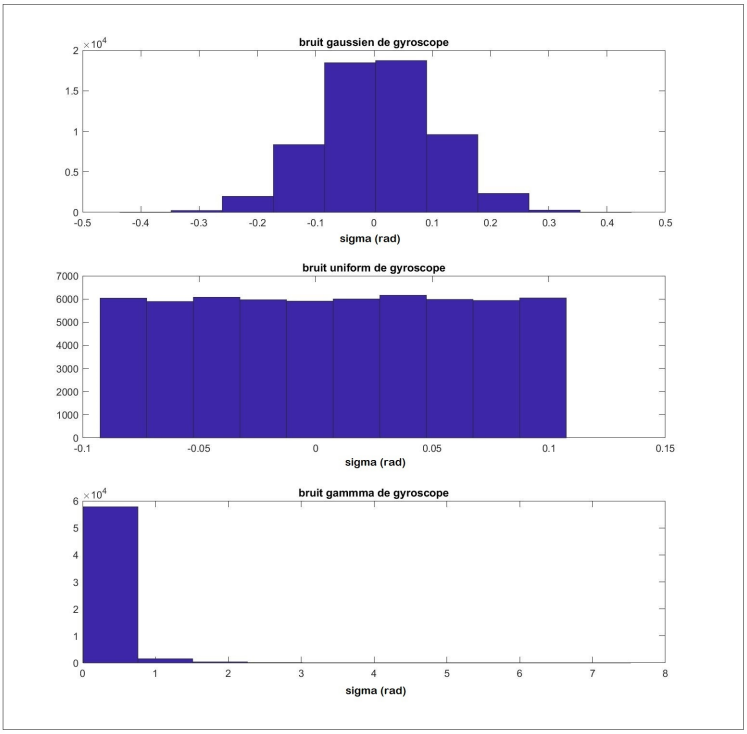


FIGURE 2.7 – Simulation de bruit gyroscopique selon 3 modèles de bruit (haut) gaussienne (milieu) uniforme (bas) gamma

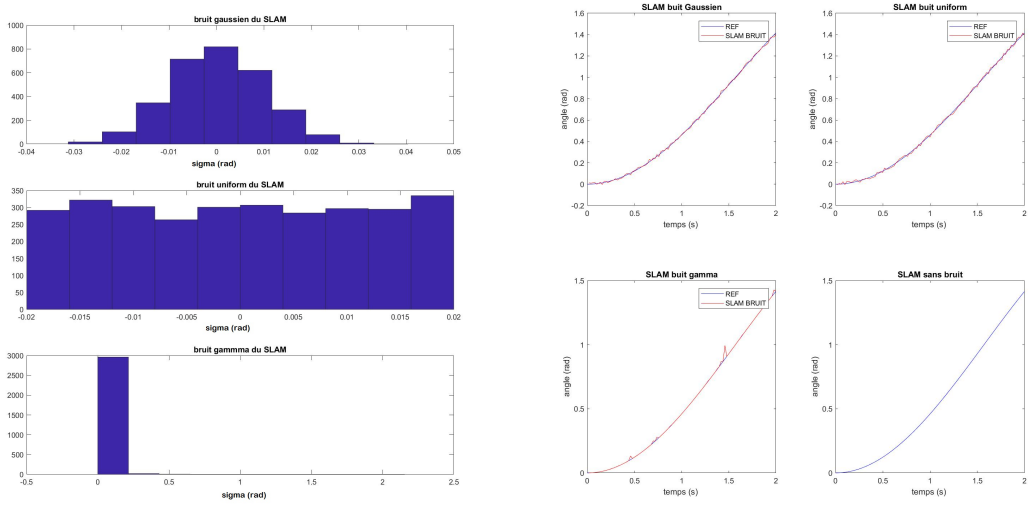


FIGURE 2.8 – Histogramme de différents bruits appliqués sur le SLAM

Type de bruit	Temps FKE (s)	Temps FP (s)
Gaussien	5	46
Uniforme	6	40
Gamma	6	42

Tableau 2.4 – Comparaison entre le filtre de kalman étendu (FKE) et le filtre particulaire en se basant sur le temps de calcul pour 1000 poses estimées

vaux de fusion et les techniques de filtrage les plus utilisées et mentionnées dans la littérature avec un développement de différents filtres (complémentaire, KALMAN et particulaire). Un simulateur théorique à été créée afin de tester les performances de chaque technique. Les performances sont le temps de traitement, la précision, et la stabilité. Les éléments de performances sont définis dans un ordre de priorité. Les tableaux 2.3 et 2.4 présentent les résultats d'une simulation faite sur une carte UDOOX86 de 4 processeurs à 1.6 GHz et 8 G.O de RAM. La variance du bruit SLAM simulé est de 0.01 rad. La variance de la CI simulée est de l'ordre de 3.5rad/h. Ces conditions rapprochent les condition expérimentales d'utilisation du casque REVE5D. L'évaluation des résultats théoriques mentionnés dans les deux tableaux 2.3 et 2.4 montre que le filtre complémentaire est très rapide en termes de temps de calcul et n'est pas performant en terme de précision et stabilité pour les fréquences faibles du SLAM. Le filtre de KALMAN quant à lui présente l'avantage d'un faible temps de traitement par rapport au filtre particulaire mais la précision et la stabilité ne sont garanties que pour les modèles linéaires et en situation de bruit gaussien. Le filtre particulaire est applicable aux systèmes (linéaires et non linéaires) et a tous types de bruits (gaussien, gamma, uniforme, etc...). L'inconvénient majeur du filtre particulaire est son temps de traitement qui dépend du nombre de particules.

Le filtre particulaire est un estimateur précis si le nombre de particules est largement suffisant pour estimer la distribution probabiliste de bruit. L'augmentation du nombre de particules entraine un temps de traitement plus conséquent mais sans entrainer nécessairement une meilleure précision. Afin de caractériser un choix entre les différents filtres disponibles, le développeur doit fixer certains éléments :

— Le temps de traitement : si l'application développée nécessite une exécution en temps réel il faut utiliser les filtres rapides

— la précision : si le système demande une précision importante et est moins exigeant sur le temps de calcul il faut choisir les filtres précis

— temps de traitement et précision : Dans ce cas il faut regarder les performances du périphérique d'exécution. Le choix d'un filtre particulaire est possible avec une précision modérée en baissant le nombre de particules.

Durant le développement de ce chapitre, nous avons constaté que les données fusionnées sont issues de différentes sources qui fonctionnent à des cadences d'échantillonnage différentes. Cette indépendance génère une incohérence temporelle entre les différentes données. La synchronisation et la mise en correspondance des données cohérentes est une étape importante pour augmenter la précision du système. Dans le chapitre suivant nous allons détailler notre contribution pour organiser les différents événements sur une horloge commune et modifier le traitement du filtre afin de prendre en considération le retard et le décalage temporel entre les différentes sources.

CHAPITRE 3

IMPLÉMENTATION D'UN ALGORITHME DE RÉTRO-CORRECTION

3.1 Introduction

La localisation 3D est une étape fondamentale dans la chaine générique d'exécution d'une application de réalité augmentée. Les techniques citées dans les chapitres précédents sont intégrées dans un module de localisation 3D qui représente le cœur d'une application de réalité augmentée. Le processus de la réalité augmentée peut être décrit par une séquence d'étapes : **Acquisition de données, Localisation dans l'espace, Rendu de la scène virtuelle, et Affichage**. Les quatre modules sont indépendants et connectés à la fois. Le module d'acquisition récupère les données brutes issues de chaque capteur d'une manière indépendante. Le module "Tracking" (localisation 3D) récupère les données brutes et estime une pose en fonction des données fournies et des contraintes imposées au sein d'un filtre ou plus généralement d'un estimateur. Ensuite, la pose estimée est envoyée vers le module de calcul de rendu. Ce module, en fonction de la pose fournie, estime la position de la caméra virtuelle dans la scène virtuelle et calcule une image de synthèse. Cette image est appelée rendu et récupérée par le système d'affichage. Le système d'affichage projette les images calculées par le module "Rendu" sur un support d'affichage.

Dans ce chapitre, nous allons définir l'architecture informatique de notre système de RA. Cette architecture supporte la récupération, l'envoi, et la réception des données entre les différentes entités. Ensuite, une estimation de temps de traitement pour certaines opérations sera réalisée. Cette estimation permet d'identifier les techniques de prédictions adéquates. Dans la troisième section, une description des méthodes de correction sera fournie. Ensuite, on propose la méthode de rétro-correction et son principe de fonctionnement. Afin d'évaluer et comparer les performances de la rétro-correction par rapport aux méthodes existantes, un jeu des tests a été développé. Les résultats et leur interprétation sont présentés à la fin de ce chapitre.

3.2 Application de RA

Le système de réalité augmentée développé dans le cadre du projet FUI **REVE5D** consiste à projeter des augmentations 3D sur un système d'affichage (type "optical see-throught") en fonction de la pose de l'opérateur dans son environnement réel. Les projections 3D sont préparées à l'avance dans une scène virtuelle. Les positions des projections sont relatives à une référence globale imposée dans l'environnement réel. Une référence globale est créée pour les deux mondes (réel et virtuel), les poses calculées dans le monde réel et dans le monde virtuel sont toujours relatives à cette référence. Lorsque l'utilisateur évolue dans le monde réel, le système, sur la base des informations des capteurs de position, déplace de manière algorithmique une caméra virtuelle. Les déplacements de la caméra virtuelle sont corrélés a ceux de l'opérateur dans son environnement. La caméra filme les éléments 3D de la scène virtuelle et renvoie sur l'écran du média de RA, des images calculées par les algorithmes de synthèse d'images. Le casque est le dispositif complet qui porte l'ensemble (matériel et logiciel) pour faire fonctionner les applications de réalité augmentée. Notons qu'une partie de calcul se fait sur un boitier déporté externe liée au casque via une liaison filaire.

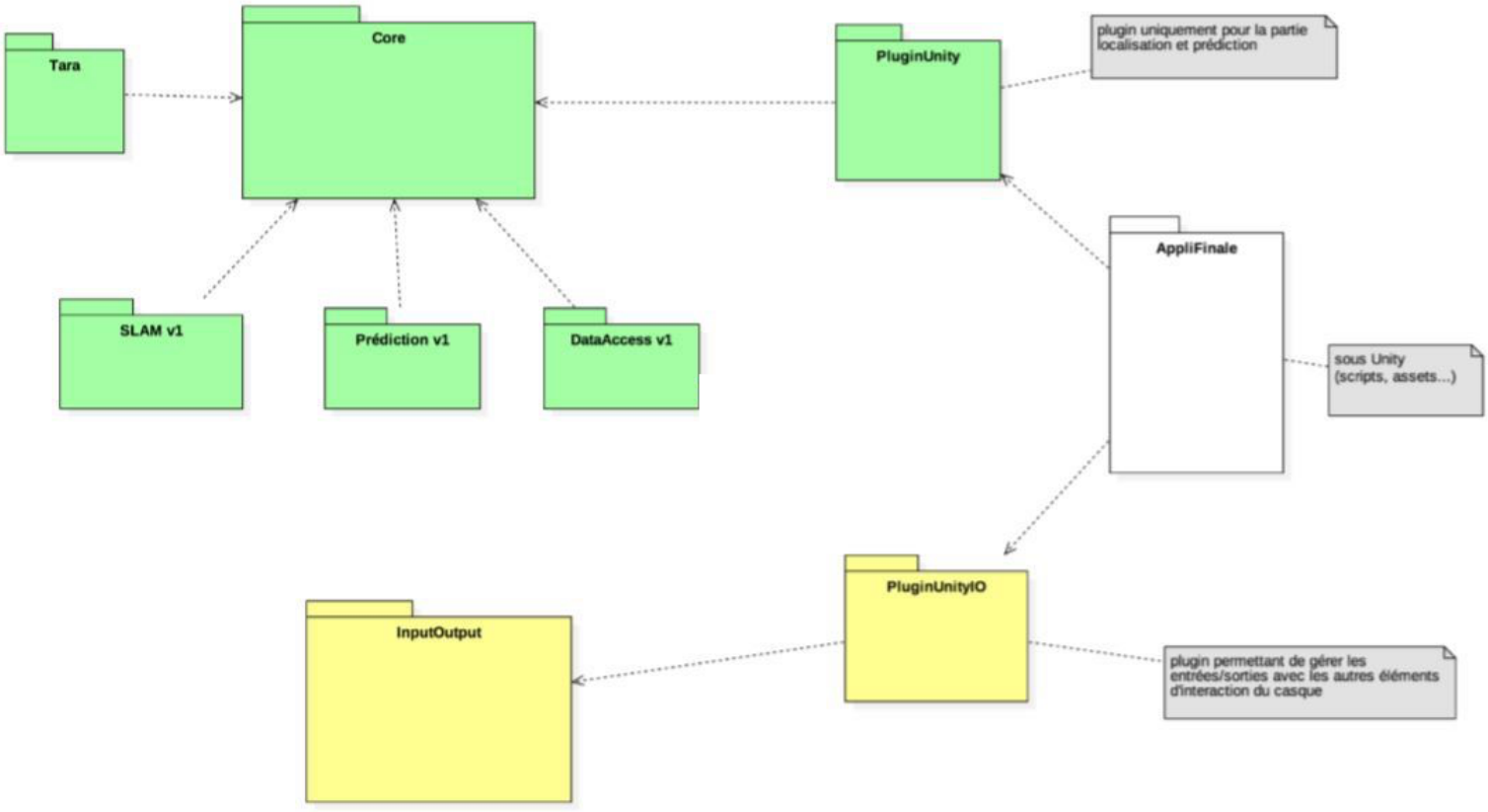


FIGURE 3.1 – Diagramme de paquetage de l'architecture REVE5D

La figure 3.1 présente le diagramme de paquetage proposé pour le système REVE5D. Ce diagramme et cette conception sont le fruit des séances de réflexions entre les différents partenaires de ce projet. Le diagramme présente trois parties :

— Core : Désigne la partie la plus importante du logiciel à développer. Elle gère l'ensemble la logique globale du système ainsi que les interactions. Elle permet notamment de rassembler les différentes briques technologiques développées autour d'elle.

— Application : Représente le contenu proposé à l'utilisateur final. elle contient notamment les interactions 3D. La communication de cette partie avec le 'Core' est assurée via le 'plugin unity' qui invoque les méthodes développées dans le 'Core'.

— Entrées/Sorties : Cette partie permet d'une part d'enregistrer les flux de données de sortie sous forme de messages ou 'Log' et d'autre part de récupérer certaines entrées comme les boutons, son, etc.... Cette partie peut être encapsulée dans un autre 'plugin Unity'.

Le module 'Core' est la composante principale dans l'architecture de notre développement. La figure 3.2 présente les détails de la communication entre 'core' et les différents modules externes.

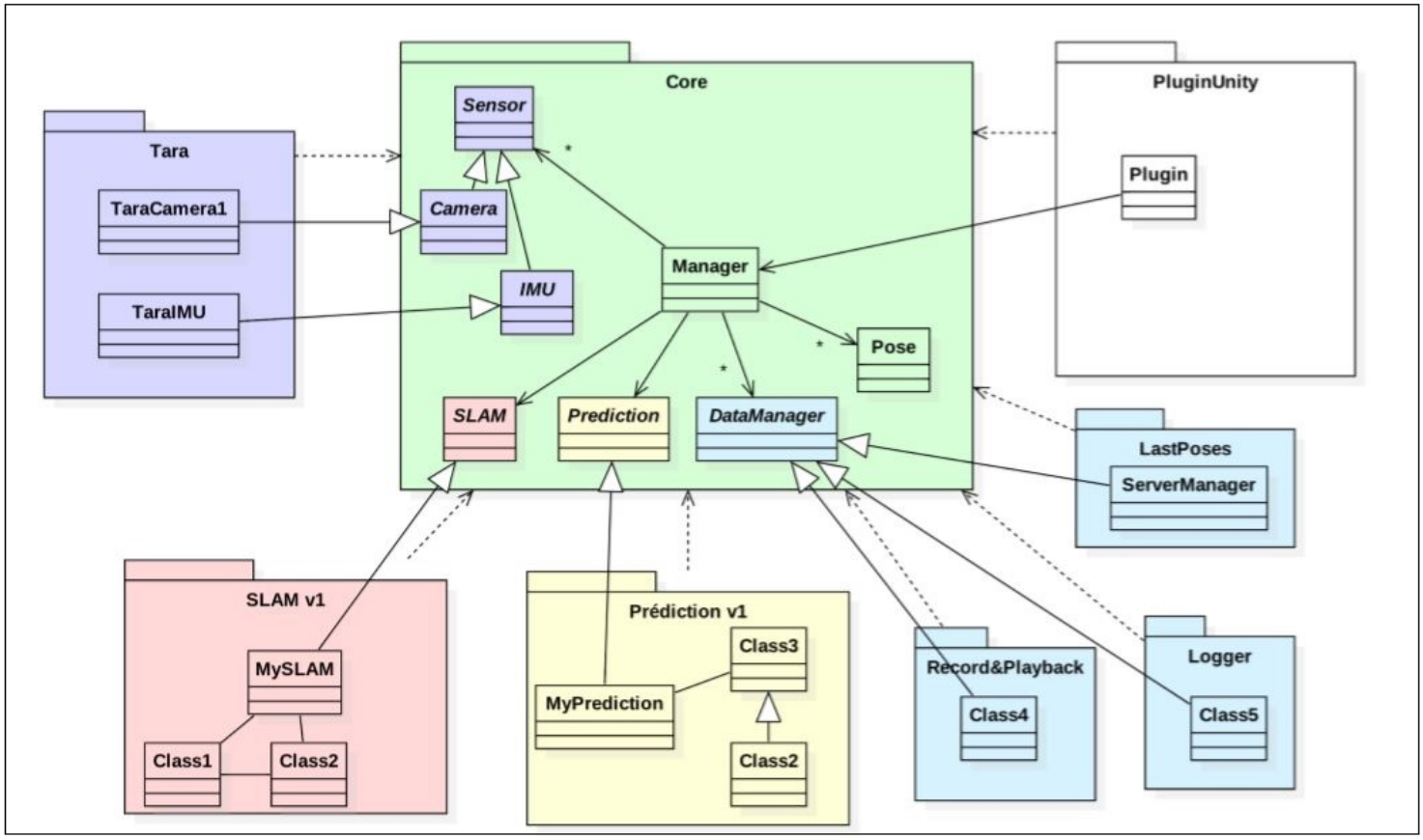


FIGURE 3.2 – Diagramme de 'packages' détaillé avec les modules internes de chacun

Le Core est composé d'un **manager** qui est l'acteur central. Il est en lien direct avec les différents modules. Le manager permet de gérer les capteurs, les abstractions (SLAM + Prédiction), le plugin Unity, et les données d'estimation algorithmiques via la classe DataManager. **Sensor** est une classe abstraite qui permet au manager de supporter plusieurs capteurs de différentes formes et natures. **Slam** est également une classe abstraite permettant au manager de supporter les différentes types

d'algorithme du SLAM. Les différentes implémentations du SLAM héritent de cette classe mère.

Prediction est une classe abstraite qui permet de gérer les outils de prédiction et optimisation dans le calcul de poses. La classe **DataManager** est une implémentation qui permet de véhiculer les données avec différents modes de gestion, selon le niveau applicatif. Les implémentation de classes abstraites représentées dans la figure 3.1 sont :

- Tara : est un module qui représente le périphérique d'acquisition utilisé pour acquérir des images et des données inertielles. Le package tara se devise en deux parties (caméra et centrale inertielle) qui héritent de la classe générique Sensor. L'utilisation de la nomenclature TARA fait référence au périphérique utilisé au cours de cette thèse.
- SLAM v1 : est une implémentation concrète d'un SLAM qui respecte la charte de la classe abstraite SLAM dans le package Core. l'architecture extensible peut supporter plusieurs types de SLAM. L'appellation SLAM v1 désigne une première version du SLAM.
- Prediction v1 : Est une implémentation concrète de la classe abstraite prédiction dans le package core, qui envoie les prédictions de poses dans le système de RA
- Plugin Unity : est la composante qui véhicule les données fournies par le SLAM et la prédiction vers l'application finale. L'appellation 'Prediction v1'suit la même règle que celle de SLAM v1.
- Le "logger" et RecordPlayback : font partie du 'Log' où plusieurs actions de suivi peuvent se faire (enregistrement de comportement, log, historique de poses).

Le choix de cette architecture permet de créer un système ouvert et extensible à toutes nouvelles fonctionnalités. La partie capteurs est également extensible par le fait que l'insertion de nouveaux capteurs est possible à la seule condition de respecter la charte de la classe mère Sensor. Le SLAM n'est pas directement intégré dans le 'Core', ce qui permet son remplacement par une mise à jour 'upgrade' ou une autre composante SLAM. De cette manière il est possible de profiter des plusieurs SLAMs open sources disponibles. La partie prédiction est également remplaçable. Le manager assure la communication, d'une manière asynchrone entre les différentes composantes principales.

3.2.1 Module de capteurs

Dans l'application de RA développée et dans le cadre d'utilisation de notre casque, le système de capteurs est conçu dans un premier temps pour assurer la localisation d'un utilisateur dans un environnement interne 'indoor localization'. Les capteurs utilisés sont les caméras et la centrale inertielle. Chaque capteur a une sortie de données libre et autonome, la fréquence d'acquisition est différente pour chacun. En effet, le système TARA utilisé est constitué de deux caméras identiques et synchronisées entre elles pour assurer l'acquisition stéréoscopique. Les images sont véhiculées comme mentionné précédemment par le Manager dans le 'Core'. Comme indiqué dans le paragraphe précédent l'ajout de capteurs est possible grâce à l'architecture développée. Cette opération nécessite uniquement l'implémentation des codes sources du capteur en question et de faire dériver son implémentation de la classe mère Sensor. Dans notre contexte les images acquises sont d'une résolution de 640 x 480 pixels et d'une fréquence de 50 images par seconde. La centrale inertielle quant à elle, fournit des données gyroscopiques qui représentent les vitesses angulaires en rad/s et des données accéléromètres qui représentent les accélérations linéaires du système fournies en fractions de l'unité d'accélération $\vec{g}$. La fréquence d'acquisition de la centrale inertielle est fixée à 954 Hz.

3.2.2 Module de localisation

Le module de localisation calcule les poses (positions et orientations) de la TARA (caméra + CI) dans le monde réel. Il est composé de deux sous-modules, **SLAM** et **Prédiction**. Le SLAM comme

présenté dans le premier chapitre, est un algorithme de vision par ordinateur qui permet d'estimer la pose en fonction des images acquises. Dans notre cas, le SLAM utilisé est celui de l'Institut Pascal (IP) adapté spécifiquement pour le projet REVE5D. Il hérite de la classe générique SLAM de l'architecture de notre système. Dans le cadre de cette thèse, nous avons exploité uniquement le SLAM développé par l'institut Pascal pour le projet REVE5D. Il est considéré comme une boîte noire inaccessible. Grâce à l'architecture ouverte, ce SLAM peut être remplacé par un autre, sous condition qu'il respecte la charte de la classe mère SLAM du 'Core' après avoir adapté le développement pour supporter cette modification. Comme indiqué dans le premier chapitre,le SLAM a un temps de traitement variable en fonction de plusieurs éléments. Ceux qui affectent directement le temps de traitement sont : la texture de l'environnement, la rapidité de mouvements, et la puissance de calcul du support. Dans notre implémentation, le SLAM fonctionne d'une manière autonome et indépendante. Il possède sa propre fréquence de réponse qui n'est pas constante.

3.2.3 Module de prédiction

Le module prédiction intègre le calcul des poses en fonction de données issues des différents capteurs. Les données sont raffinées et filtrées afin d'améliorer la précision et réduire le bruit. Ce module comporte plusieurs sous briques utilisées pour optimiser l'exécution des autres modules et améliorer les performances de l'application. Le code se décompose en deux catégories de calcul :

— **Prédiction pour le SLAM** : La prédiction pour le SLAM est une pose (généralement la rotation), envoyée au SLAM pour lui indiquer la rotation entre deux instants t_1 et t_2. t_1 représente l'instant d'acquisition de l'image précédente pour le calcul d'une pose SLAM et t_2 représente l'image acquise pour fournir une pose ultérieurement (après le traitement SLAM). La centrale inertielle fonctionne à une fréquence plus élevée que celle du SLAM, elle peut mesurer les rotations appliquées sur le système entre deux acquisitions. Par ce biais il est donc possible de fournir une prédiction de pose pour le SLAM. Cette donnée prédictive a pour objectif de réduire la région d'intérêt traitée par le SLAM, pour chercher les primitives à extraire, et déduire la pose plus rapidement. L'angle entre deux images est calculé à partir des données gyroscopiques non filtrées. L'objectif est de fournir cette estimation lors de l'acquisition d'image pour réduire le temps de traitement SLAM pour éliminer tout temps de calcul additif. Il faut aussi noter que les poses fournies par une centrale inertielle sur une période courte gardent un niveau acceptable de précision. La quantité de bruit ou les dérives qui affectent les acquisitions sont minimales.

— **Poses pour le module rendu et affichage** : Lors de la réception d'une demande, le module de prédiction fournit la dernière pose disponible sur le bus de localisation. Ensuite, une prédiction de la pose, prenant en considération le temps de traitement du rendu, sera calculée. Ce temps est fixé au lancement de l'application à une valeur V. Le temps d'affichage de $17.6ms$ est également pris en compte. La position envoyée à un instant t comporte un couple de la dernière position sur le bus et une prédiction de $V + 17.6ms$.

Dans la figure 3.3, le processus général est présenté ainsi que l'interaction entre les différents modules pour le calcul du rendu final sur le casque.

3.2.4 Module rendu et affichage

Ce module consiste à construire des images de synthèse à base des scènes 3D. Celles-ci sont conçues dans le moteur 3D dans un repère global qui, par convention, est identique à celui des modules de Tracking et de calcul de rendu. Afin de calculer une nouvelle image de synthèse, le module de rendu envoie une requête au module du Tracking (décrit dans le paragraphe suivant) pour obtenir

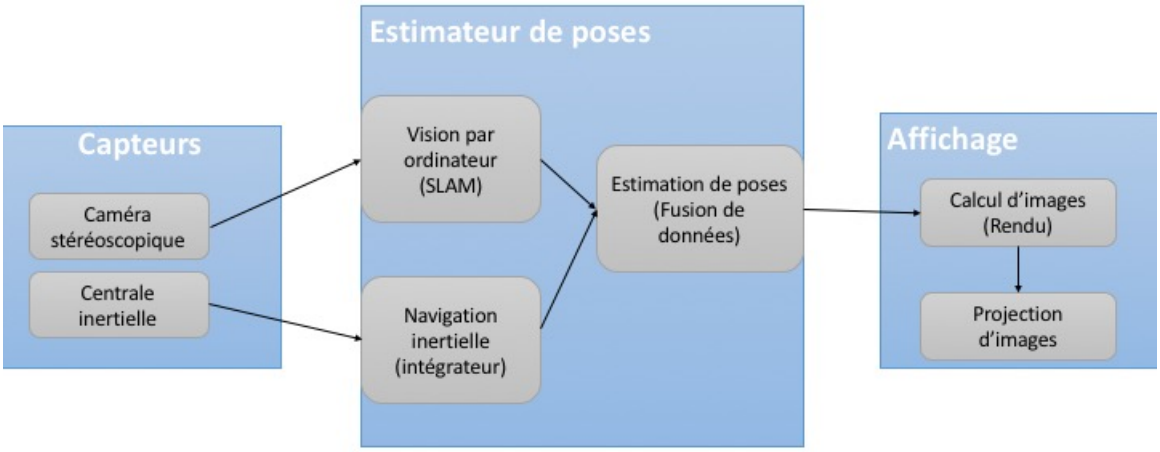


FIGURE 3.3 – Le processus générique d'une application R.A utilisant une fusion du SLAM et centrale inertielle

une pose. Cette dernière est récupérée pour modifier la position de la caméra virtuelle, qui implique le calcul de la nouvelle image pour transmission in fine vers un 'buffer' (tableau), puis projection sur l'écran du casque.

La fréquence de projection dépend du type de projecteur utilisé, fixée et fournie par le constructeur. Néanmoins, la fréquence du rendu est variable en fonction de la complexité de la scène virtuelle et de l'orientation de la caméra. Les performances du système de calcul de rendu affectent directement le temps d'obtention de l'image. Afin d'améliorer le rendement de l'application, une optimisation, détaillée dans le chapitre suivant (Caractérisation de la latence), est faite sur le temps de calcul du rendu en fixant sa fréquence.

Après la définition de l'architecture informatique et présentation de la structure générique de l'application. Nous détaillerons par suite les différentes techniques utilisées au sein de chaque module.

3.3 Prédiction

La prédiction consiste à estimer ou calculer une valeur future de l'état d'un système quelconque à un instant futur en fonction d'un historique et de l'état courant. Dans le chapitre précédent, les filtres KALMAN et particulaire ont été présentés comme des filtres de modèle prédicteur/correcteur.

La figure 3.4 présente une trajectoire avec différentes poses mesurées réellement à des instants précis. Dans la suite de ce paragraphe, cette trajectoire est utilisée comme référence et les poses (triangles remplis) sont utilisées comme poses de références pour illustrer les concepts visuellement et d'une manière schématique.

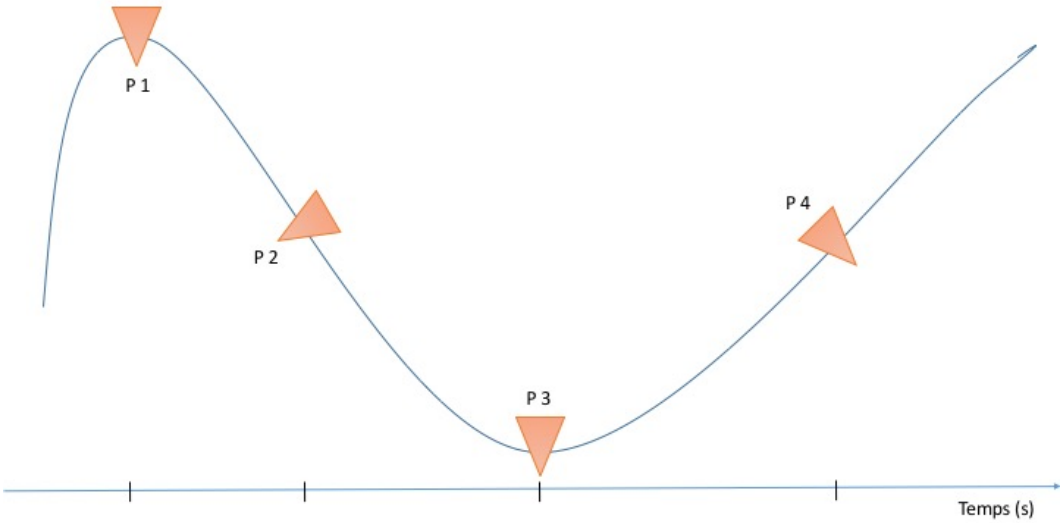


FIGURE 3.4 – Exemple d'une trajectoire avec des positions réelles à des instants différents

vitesse angulaire	intervalle	valeur prédite
45˚/s	40 ms	1.8˚
90˚/s	40 ms	3.6˚
180˚/s	40 ms	7.2˚

Tableau 3.1 – Tableau de valeurs prédites pour différentes vitesses angulaires

La valeur prédite dans le tableau 3.1 montre le décalage entre la pose au moment d'affichage qui succède l'instant de récupération de la pose de 40 ms et l'instant de demande de la pose. Pour les mouvements relativement lents, le décalage angulaire est de l'ordre de 1.8 ˚et dans les mouvements relativement rapides il est de l'ordre de 7.2 ˚. On observe donc que le décalage varie en fonction de la vitesse du mouvement de la tête. La valeur de 40 ms est choisie d'une manière stochastique dans cet exemple à titre indicatif. Les prochaines expériences et mesures faites dans ce chapitre ou dans le suivant vont faire converger ce temps vers une valeur expérimentale plus correcte.

La figure 3.5 représente le concept de la prédiction. A un instant présent, on utilise les données passées pour estimer une pose future. La figure montre des poses réelles dans le présent et passé avec des poses estimées par le calculateur. Les poses déjà estimées étaient prédites dans un premier temps et après l'avancement chronologique de la trajectoire la pose prédite devient une pose estimée. Les poses "prédites" deviennent "estimées"

En pratique les mouvements de la tête ont généralement une amplitude, autour d'un axe vertical, comprise entre 60 et 120 $^{\circ}/s$. Pendant le calcul et l'estimation de la pose, la tête continue a faire des mouvements à des vitesses différentes. La prédiction a pour but de prendre en considération ce temps de calcul et fournir une pose la plus précise possible. Dans ce chapitre et pour des raisons de simplification, on suppose que les mouvements sont statiques sur l'intervalle de prédiction. En d'autres termes l'évolution de la vitesse angulaire est linéaire ou statique. On suppose dans un premier temps que les intervalles de calcul de rendu sont identiques, de valeur 20 ms et que la fréquence

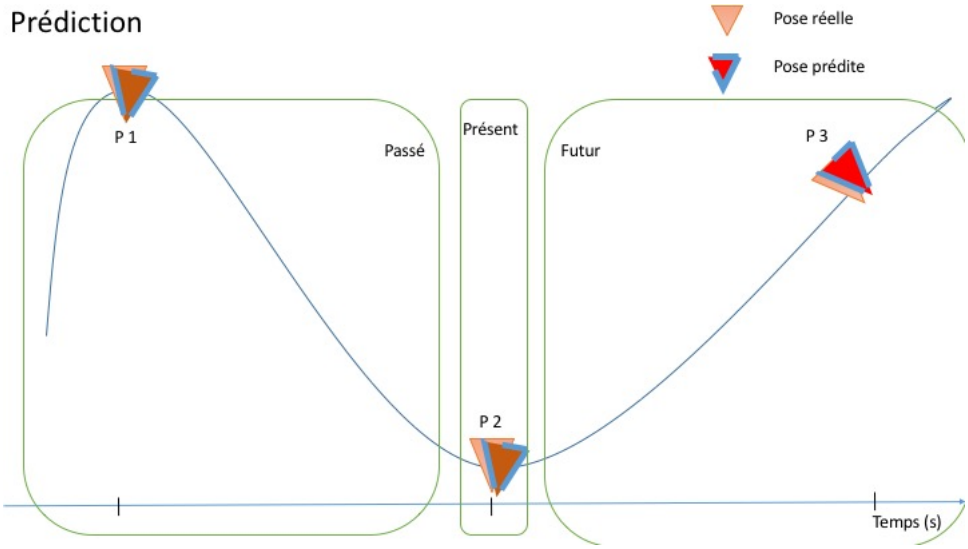


FIGURE 3.5 – Présentation du concept de prédiction avec les 3 phases temporelles : passé, présent et futur

d'affichage est de 50Hz. Les détails de changement de ces deux paramètres seront développés dans le chapitre suivant. Dans cette section, une comparaison entre les différentes techniques de prédiction est faite dans un premier temps. Elle est effectuée sur des données synthétiques de deux mouvements, sinusoïdale et arbitraire quelconque. En se basant sur les analyses et les performances de chaque test, le choix de la technique adoptée pour la suite sera statué.

3.3.1 Techniques de prédiction

La fréquence d'acquisition élevée de la centrale inertielle permet d'avoir une estimation de pose supplémentaire entre deux images acquises à une fréquence faible du SLAM. Ces poses intermédiaires peuvent être utilisées par le SLAM comme une prédiction de la pose à venir. La nouvelle pose (SLAM) estimée correspond à la transformation entre l'image à (t+i) et l'image acquise à l'instant précédent (t). La rotation calculée entre ces deux instants représente bien la prédiction de résultat qui va être fournie plus tard par l'algorithme du SLAM. C'est la prédiction adoptée pour améliorer le comportement du SLAM. Cette partie se décompose en deux sous parties. La première est l'estimation d'une pose entre deux images utilisées par le SLAM qui est développée en cours de cette thèse. La deuxième partie concerne la réduction de la région d'intérêt pour le calcul SLAM sur la base de la pose estimée. La réduction de région d'intérêt est gérée par notre partenaire de projet et notre fournisseur d'algorithme SLAM l'IP.

La deuxième famille (de prédiction) présente la prédiction de données futures pour le module de rendu et affichage.

L'**extrapolation de données** consiste à étendre les données sur des instants futurs en suivant l'historique de distribution. Plusieurs schémas d'extrapolation sont disponibles. Dans ce travail on s'intéresse à l'extrapolation linéaire et spline. Ces deux méthodes sont testés en utilisant la fonction d'extrapolation fournis par le logiciel MATLAB, avec des paramètres spécifiques à configurer. Le **filtre de KALMAN** fait partie des outils de prédiction présentés précédemment. En utilisant son équation d'état, les données futures peuvent être prédites à des instants bien précis. Le **filtre particulaire**, comme le filtre de KALMAN, permet d'estimer des données futures en fonction de

l'état à un instant k, en utilisant la fonction heuristique (SIR) qui définit le comportement du système.

3.3.2 Comparaison

Après la présentation des différentes techniques de prédiction à tester. une sélection de la technique la plus performante doit être effectuée. Cela permettra d'améliorer la précision des résultats finaux qui se basent sur cette prédiction. Les jeux de comparaison sont générées d'une manière synthétique en simulant une trajectoire quelconque. Ils incluent la vitesse angulaire, l'accélération, la vitesse, la position, et la rotation. Les données d'entrées et le bruit sont donc identiques. Afin de simuler correctement les capteurs de la CI, la vitesse angulaire et l'accélération sont soumises à des bruits additifs gaussiens. Le bruit est rajouté en fonction de l'étude faite dans le chapitre précédent, portant sur le comportement d'une centrale inertielle. Les tests sont effectués sur les données de la manière suivante :

— **prédiction de données brutes** : Cette phase consiste à prédire les données gyroscopiques et les données accéléromètriques en utilisant une fonction de prédiction

— **prédiction de rotations** : Cette prédiction consiste à estimer une rotation future en utilisant l'historique des rotations et de données brutes de la CI au même instant. Les rotations qui sont utilisées, elles mêmes sont estimées à partir des données brutes de la CI.

Dans la suite, la prédiction des données sera réalisée à l'aide d'une technique d'extrapolation (linéaire et spline). Pour l'appliquer, il faut exploiter l'historique des données à prédire, ce qui nécessite de fixer sa taille de manière optimale pour assurer une bonne estimation. Si on se base sur un historique très réduit on risque de fausser les prédictions lointaines. A l'inverse un historique très développé risque d'augmenter le temps de calcul et de lisser trop fortement les données. Le choix s'est porté sur une stratégie à base d'un historique variable en fonction de nombre de pas de prédiction. La taille de l'historique est fixée à deux fois le nombre de pas de prédiction.

Nombre de pas	Erreur de l'extrapolation linéaire (rad/s)	Erreur de l'extrapolation de type Spline (rad/s)
1 pas	0.0195	0.0195
10 pas	0.1190	1,328
20 pas	0.2310	8,96087
50 pas	0.5648	12,47
100 pas	1.1130	25,17
500 pas	5.0702	100

Tableau 3.2 – Tableau comparatif des erreurs de prédiction sur les données gyroscopiques par rapport une trajectoire de référence

Le tableau 3.2 présente la comparaison entre l'extrapolation linéaire et l'extrapolation de type spline en fonction du nombre de pas de prédiction. Le pas de prédiction est un paramètre qui détermine l'instant de cette prédiction. Par exemple en supposant que la fréquence d'acquisition de la centrale inertielle soit de 1 Khz, le pas de prédiction représente $1/1000 = 0.001s$. Pour 500 pas, on aura une prédiction de $0,5s$. On définit une bonne prédiction comme étant la plus fidèle à la valeur réelle même avec un nombre de pas élevé. Dans notre étude l'intervalle de confiance est de 95 %.

Les données du tableau 3.3 représentent la moyenne de la prédiction sur les données en utilisant différents modèles de bruits et en générant plusieurs trajectoires de formes sinusoïdales et quelconques. L'analyse de ces données montre que l'extrapolation linéaire est plus robuste que l'extrapolation de type spline en se basant sur la stratégie d'un historique équivalent à 2 fois le nombre de pas. Dans ce cas et pour le reste de la thèse la prédiction de données brutes utilisée est basée sur

l'extrapolation linéaire. Les comparaisons sont effectuées entre les données bruitées et les données prédites.

Nombre de pas	Erreur de prédiction par filtre de Kalman	Erreur de prédiction par filtre particulaire
1 pas	0.08	0.163
50 pas	0.034	0.5
500 pas	0.8	1.2

Tableau 3.3 – Tableau comparatif de données prédites selon différentes méthodes et erreurs de prédiction des rotations par rapport une trajectoire de référence

L'approche par extrapolation de données permet d'avoir des données futures en se basant sur un historique. Ces données sont injectées dans les filtres afin d'estimer des poses futures. En utilisant les filtres, deux cas de figures sont possibles.

— Premier cas : La prédiction de pose se base sur les données futures qui sont estimées à partir d'une extrapolation linéaire.

— Second cas : La prédiction de pose se base sur la réitération du filtre en utilisant les mêmes données à l'instant t. On se base sur l'hypothèse que l'évolution de données est quasiment nulle ou linéaire sur l'intervalle de prédiction.

Afin de déterminer la méthode de prédiction à utiliser, une évaluation des performances entre ces deux approches doit être réalisée. Le test est effectué avec des données gyroscopiques et des données SLAM toutes deux "synthétiques". Le test est réalisé avec 3 fréquences SLAM différentes : 1 kHz proche de celle de la centrale inertielle, 100 Hz et une fréquence relativement faible représentative du cas où le SLAM serait exécuté sur une plateforme peu performante.

3.4 Estimation de poses : Apporche classique vs contribution

Une fois les données acquises et récupérées dans le système, une procédure de calcul commence. Deux algorithmes fonctionnent en parallèle et exécutent des tâches différentes. Le SLAM, considéré comme une boîte noire, récupère des images et estime la pose de la caméra en fonction d'images déjà récupérées. La Centrale Inertielle (CI) estime l'orientation en utilisant le gyroscope. Dans la suite de ce mémoire, pour des raisons de simplifications et d'efficacité, les positions utilisées sont calculées uniquement par le SLAM et le traitement de fusion est utilisé uniquement pour la prédiction. Aucun lissage et aucune correction n'a été appliquée. Le changement de position, dans le cas des applications de réalité augmentée est moins conséquent que le changement de rotation. L'estimation de l'orientation résulte d'une fusion de deux sources de données issues du SLAM et CI. Dans la suite de ce paragraphe, nous décrirons, l'approche classique utilisée pour corriger les poses de la centrale inertielle, puis notre contribution et l'origine de la méthode de rétro-correction[47].

3.4.1 Approche classique

L'approche classique de la fusion de données dans les filtres mentionnés précédemment consiste à calculer la nouvelle pose corrigée en fonction de la dernière pose SLAM disponible et la dernière pose de la CI. Comme évoqué dans les chapitres précédents, les threads sont lancés d'une manière autonomes et leur comportement est totalement indépendant. Le SLAM calcule des poses bien plus lentement par rapport l'estimation des poses utilisant la CI. A un instant t la caméra et la centrale

inertielle reçoivent une image, une vitesse angulaire et accélération respectivement. Le SLAM traitera les données de l'image reçue pour fournir une pose après un certain temps de traitement appelé T_s. D'après les expériences réalisées, le temps T_s est variable en fonction du support d'exécution et de l'environnement. Les mesures montrent que ce temps peut varier entre $[20ms, 100ms]$. Plus de détails sur le temps de traitement SLAM et la latence de capteurs sont fournis dans le chapitre suivant.

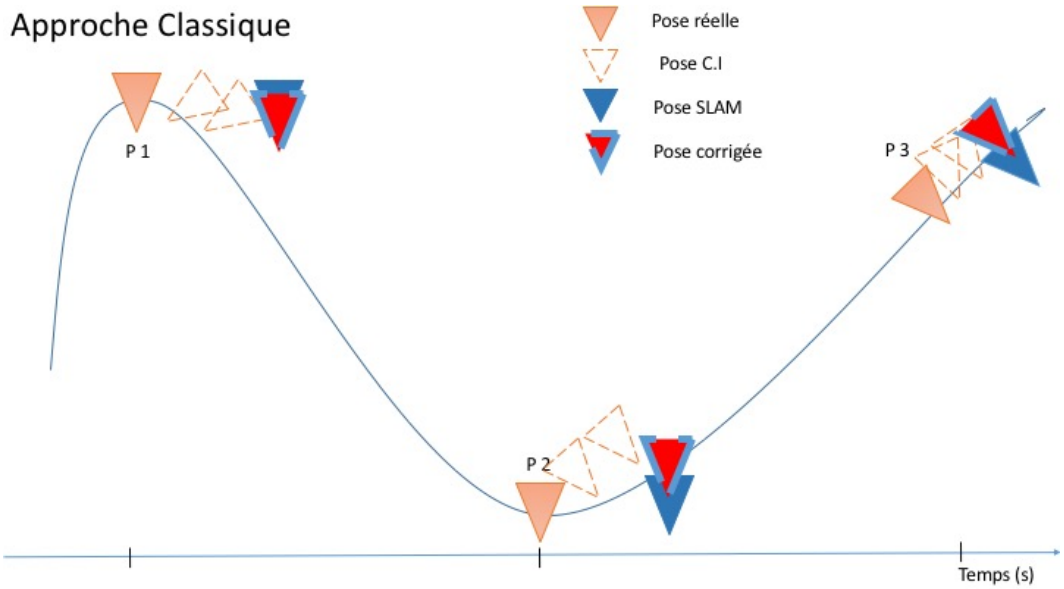


FIGURE 3.6 – Poses estimées en utilisant l'approche classique sur la trajectoire de référence

La figure 3.6 illustre le principe de fusion et correction des poses en se basant sur l'approche classique. Les décalages entre les poses réelles et les poses SLAM sont importants. Ils sont dus au temps de traitement de données pour estimer la pose SLAM. Les triangles rouge avec un contour bleu sont les corrections des poses CI par les poses SLAM. Les poses corrigées sont décalées par rapport aux poses de références car les corrections ne prennent pas en considération le temps de traitement SLAM. L'approche de fusion classique est basée sur les poses reçues par le SLAM et les poses calculées à partir de la CI. L'objectif de cette fusion est généralement d'estimer/corriger les dérives de la centrale inertielle. Les filtres tels qu'ils sont implémentés, ne prennent pas en considération les décalages temporels causés par le temps de traitement. Les fusions sont faites entre les données disponibles sans aucune information sur la date de production de cette information.

La figure 3.7 montre la propagation chronologique de la correction dans l'approche classique. La première ligne de temps représente les acquisitions d'images via la caméra. La seconde ligne, l'avancement chronologique d'estimation de poses SLAM. Le temps de traitement SLAM se manifeste sur cette ligne dans le décalage entre l'acquisition de l'image et le renvoi de la pose correspondante à cette image. La troisième ligne, représente l'acquisition et l'estimation de poses en utilisant la CI. Le temps de traitement et de calcul de poses pour la centrale inertielle est considéré comme négligeable. Entre la réception de l'image et le calcul de la pose SLAM correspondante à cette image la CI aura estimé plusieurs poses.

Cette approche classique permet de corriger les poses estimées par la CI à partir des poses SLAM. Dans les techniques de couplage fort toutes les données sont synchronisées et intégrées dans la même architecture. Contrairement au couplage faible, la correction se fait entre la dernière pose SLAM et la dernière pose CI. Les données prélevées dans le tableau 3.4 montrent d'une manière

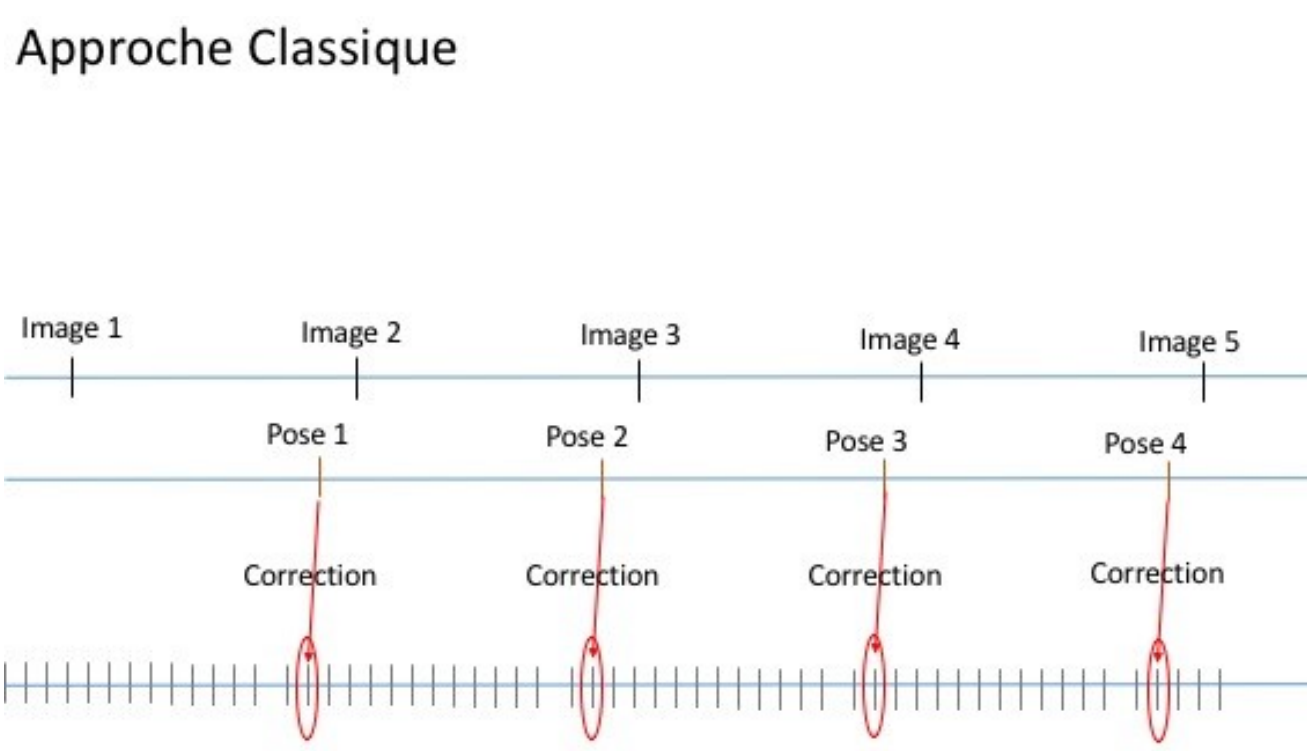


FIGURE 3.7 – Représentation chronologique du processus de correction classique basé sur les données de vision pour corriger les dérives de la CI en passant par un filtre spécifique

théorique le décalage de correction de poses SLAM pour une architecture classique. Les données du tableau 3.4 montrent que la correction SLAM retarde les poses IMU car elles sont corrigées pour des estimations antérieures dues au temps de traitement SLAM. Dans le cas de la mesure 3, le SLAM a pris plus de temps pour estimer la nouvelle pose qui est en réalité de 4.05˚. La pose SLAM estimée a cet instant est celle estimée auparavant dans l'instant précédent soit 1.32˚. Cette pose SLAM sera fusionnée avec la pose de la CI dans le filtre de traitement et elle impactera forcement l'estimation. Si la fréquence SLAM est élevée, l'influence de ce retard sera mineur. Cependant pour les fréquences faibles, l'influence est majeure et peut atteindre des dizaines de degrés dans certain cas. La contribution de cette thèse dans le projet REVE5D consiste à prendre en compte le temps de traitement SLAM afin d'obtenir une prédiction aussi précise que possible, d'où l'implémentation de notre approche de rétro-correction.

Mesure	Temps (ms)	SLAM (˚)	CI brute(˚)	Approche classique (˚)	Position réelle (˚)
1	0	0	0	0	0
2	30	0	1.39	1.33	1.35
3	60	1.32	2.8	2.2	2.7
4	90	1.32	4.2	3.0	4.05
5	120	3.01	5.65	3.5	5.4
6	150	4.00	6.98	4.6	6.86

Tableau 3.4 – tableau synthétique d'une situation de correction en utilisant la méthode classique

3.4.2 Rétro-correction (approche proposée)

Le terme de rétro-correction est utilisé pour désigner le retour en arrière et la correction des données passées. Plus précisément, avec une pose (SLAM) estimée au présent, on corrige une pose CI estimée au passée. Cette partie représente la contribution de cette thèse pour l'alignement temporel des événements.

Afin de bien comprendre le déroulement, supposons qu'à un instant t la caméra a récupéré une image I_t. Cette image est relative à la pose réelle de l'objet PR_t à l'instant t. La CI a également récupéré aussi des données d'accéléromètre et de gyroscope au même instant t. Le temps d'acquisition de données est supposé négligeable pour la caméra et la centrale inertielle. L'image I_t correspondant à la pose réelle PR_t est envoyée au module SLAM pour traitement. Après un certain temps de traitement T_s, le SLAM renvoie à l'instant $t+T_s$ son estimation de la pose réelle PR_t . Étant donné que la fréquence d'acquisition de la centrale inertielle est élevée par rapport à celle de la caméra, les données acquises entre t et Ts estiment les pose de l'objet durant le traitement SLAM. À l'instant $t+T_s$, l'approche classique corrige la pose estimée par la CI correspondant à la pose réelle à l'instant $t+T_s$ par une pose SLAM, estimée à l'instant $t+T_s$, pour une pose réelle PR_t. Le **traitement de la rétro-correction** consiste à créer deux tableaux circulaires. Le premier garde en mémoire les poses SLAM et le deuxième enregistre les poses de la CI. Les données enregistrées dans les tableaux sont la pose et la date de référence correspondante à l'acquisition de l'image et l'acquisition de données inertielles pour le SLAM et la centrale inertielle respectivement. Afin de dater les événements, une horloge principale liée à la fréquence du processeur est initialisée. Cette horloge est partagée, date tous les événements dont les principaux sont : l'acquisition de données et l'estimation de poses.

L'acquisition d'images et de données inertielles s'effectue avec des fréquences différentes. Dans notre traitement on s'autorise à un décalage de 1 à 2 ms entre la données image et les données de CI. Un décalage de 2 ms signifie que si l'objet se déplace à une vitesse angulaire de 180˚/s, il aura pivoté de 0.36 ˚. Cette valeur est acceptable car elle reste comprise dans les limites de la précision souhaitée. Sachant q'un déplacement ordinaire de la tête est entre 60 ˚/s et 120 ˚/s.

Le traitement algorithmique consiste donc à chercher la pose de la CI la plus proche en terme de date de la pose SLAM, ce qui est le cas si la différence de dates entre la pose SLAM et la pose CI est inférieure à 2 ms. A cette condition la pose sera retenue sauf si une autre pose dont la différence est inférieure à 2ms et inférieure à la différence entre les poses SLAM et CI. En d'autres termes, s'il n'existe pas d'autre pose CI plus proche que celle retenue. La recherche et la comparaison de dates de poses ainsi que la sélection est illustré par **l'algorithme 4**. La rétro-correction est appliquée après la récupération de la nouvelle pose SLAM. Le tableau circulaire SLAM est aussi utilisé pour une prédiction de poses qui sera détaillée par la suite.

La structure du tableau circulaire est FIFO (First In First Out), qui désigne premier entré, premier sorti. Cette structure permet d'éjecter les poses anciennes et de récupérer les nouvelles poses. À l'issue de ce premier traitement, une deuxième procédure consiste à corriger la pose de la CI correspondante. L'algorithme cherche la dernière pose de la CI correspondante à la date de sortie de la pose SLAM, calcule l'écart entre la pose de la CI à la date d'origine et à la date de sortie de la pose SLAM, et applique finalement la correction. Cette dernière consiste à mettre à jour la dernière pose CI. Cette dernière sera utilisée par la suite pour estimer les nouvelles poses CI. Avec ce type de traitement, les dérives de la centrale inertielle sont éliminées grâce aux estimations précises du SLAM.

La procédure de correction entraîne dans certains cas et sous certaines conditions une mauvaise estimation de pose. Dans ce cas la proposition est incohérente par rapport à l'historique, les données, et le mouvement ou la trajectoire. Ce cas de figure est défini sous le nom de poses inconsistantes.

Les poses **inconsistantes** sont des poses aberrantes, causées généralement par un dysfonctionnement de capteur ou par un traitement appliqué en utilisant des données aberrantes. Les travaux

de [104] portent sur la détection d'inconsistances lors de la fusion entre un SLAM et une CI en mode couplage faible. La détection se base sur des discontinuités dans la courbe de la trajectoire. Cette approche, détecte les inconsistances et les rejettent directement sans aucun traitement spécifique,

Algorithm 4 Algorithme de mise en correspondance des données SLAM et des données inertielles

X_{SLAM} = identity ;
CB_SLAM : tableau circulaire SLAM
CB_IMU : tabeleau circulaire Centralle inertielle (CI)
X_{SLAM} = CB_SLAM(0)
for (i<taille(CB_IMU))
{
if (abs(CB_SLAM(0).date - CB_IMU(i))) < 0.00001)
return CB_IMU(i)
}

L'algorithme 4 présente la mise en correspondance de poses SLAM et CI datées par notre horloge.

Mesure	Temps (ms)	SLAM (°)	CI brute(°)	Rétro-correction (°)	Position réelle (°)
1	0	0	0	0	0
2	30	0	1.39	1.33	1.35
3	60	1.32	2.8	2.56	2.7
4	90	1.32	4.2	3.95	4.05
5	120	3.01	5.65	5.2	5.4
6	150	4.00	6.98	6.68	6.86

Tableau 3.5 – Tableau synthétique d'une situation de correction en utilisant la méthode de rétro-correction

Le tableau 3.5 montre la souplesse de l'algorithme de rétro-correction. Il combine la rapidité de la centrale inertielle et la fiabilité du SLAM en gérant le décalage temporel de l'estimation. Pour la 4eme mesure où le SLAM a mis plus de temps pour estimer une pose, la rétro-correction a bien géré ce retard et la pose fournie est très proche de la pose réelle.

La figure 3.9 présente le principe générique de la méthode de rétro-correction. Une horloge générique contrôle et date toutes les acquisitions de tous le capteurs (dans notre cas caméra et CI). Une fois les données datées, elles sont envoyées pour traitement et extraction d'informations (poses). Les images sont envoyées au module vision (dans notre cas le SLAM) et les données inertielles sont envoyées au module de navigation inertielle. Les deux blocs opèrent indépendamment. Les poses calculées par le SLAM sont envoyées avec une date $t + T_s$. Le temps de calcul d'une pose inertielle est tellement faible, on le considère comme négligeable. Dans ce cas, la pose de sortie est datée à l'instant même d'acquisition t. Les deux données sont envoyées vers le module de rétro-correction qui fait un ajustement chronologique des poses et met en correspondance les poses SLAM et les poses CI. Afin de réaliser cette procédure un tableau circulaire est mis en place. Les données SLAM sont enregistrées dans une structure de données qui garde en mémoire la date, la pose, et les données de la CI. Une fois que les poses sont mises en correspondance, l'opération de filtrage est lancée. Les données CI et SLAM sont envoyées au filtre pour obtenir une pose finale.

La figure 3.10 présente la première étape dans le processus prédictif avec rétro-correction. En effet, la première étape consiste à recevoir une demande de la part de l'application. Cette demande

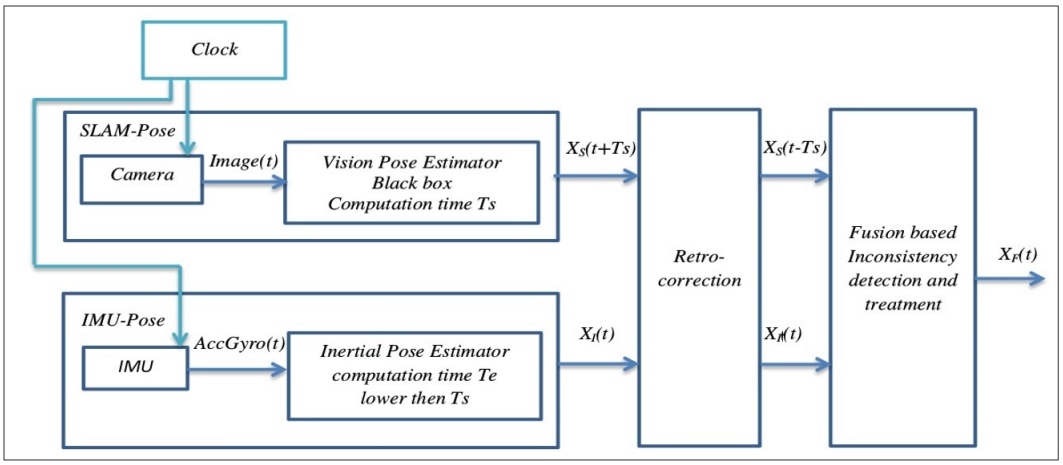


FIGURE 3.8 – Architecture de module de tracking avec l'estampillage de chaque événement [47]

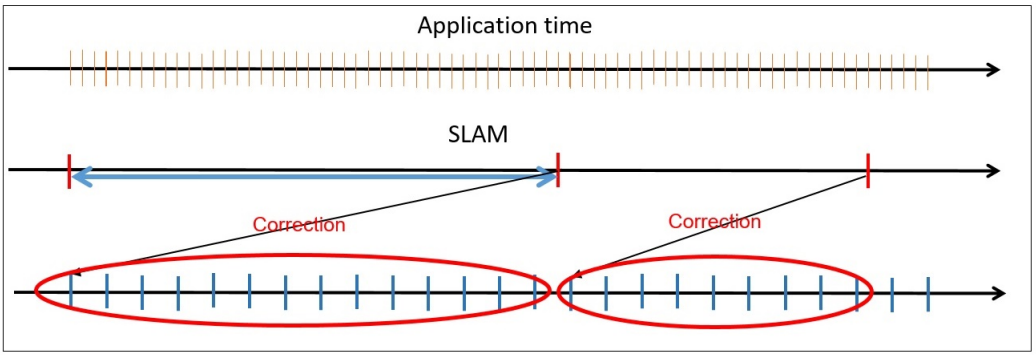


FIGURE 3.9 – Principe de la rétro-correction

intervient à un certain instant t, et l'image de cette demande va être affichée à un instant $t + T_r$ avec T_r est le temps de calcul du rendu. Sur le module de couplage faible les données affichées sont des données prédites sur l'intervalle T_r. La prédiction de pose est envoyée au module d'affichage (application). Cette pose est basée sur la prédiction de données gyroscopique en fonction de son historique.

La figure 3.11 illustre la deuxième étape de la prédiction de données qui est celle de l'acquisition des données brutes réelles de la centrale inertielle. Dans cette figure, le système d'affichage, après le temps de traitement, affiche l'image "Rendu 1" relative à la pose prédite à l'instant $t + T_s$.

La figure 3.12 présente le schéma prédictif avec deux types de données. La partie bleue, avant la deuxième demande de l'application représente les acquisitions réelles. La deuxième partie jaune, représente les données prédites pour le rendu prochain après un certain temps de traitement noté précédemment T_r. Pour la deuxième demande, il s'agit du même traitement que l'étape 1, avec des données mesurées avant la demande.

La figure 3.13 montre le principe générique avec la présence de toutes les mesures et toutes les opérations. Après l'arrivée de la donnée SLAM une correction est envoyée en retour pour corriger la pose à l'instant d'acquisition d'image. Le cercle rouge représente l'opération de mise à jour de toutes les poses calculées après la date d'origine. Les poses mises à jour sont entourées par le cercle rouge.

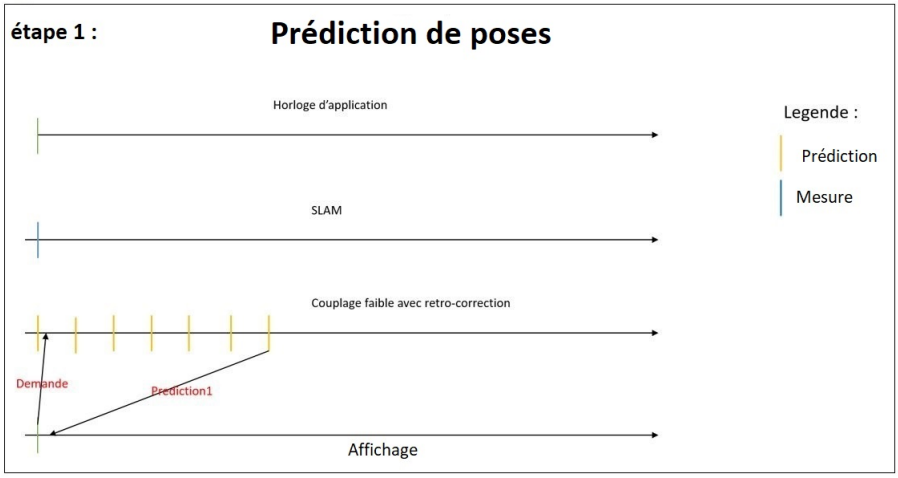


FIGURE 3.10 – Étape 1 : prédiction de données après la réception d'une demande de pose de la part de l'application de réalité augmentée

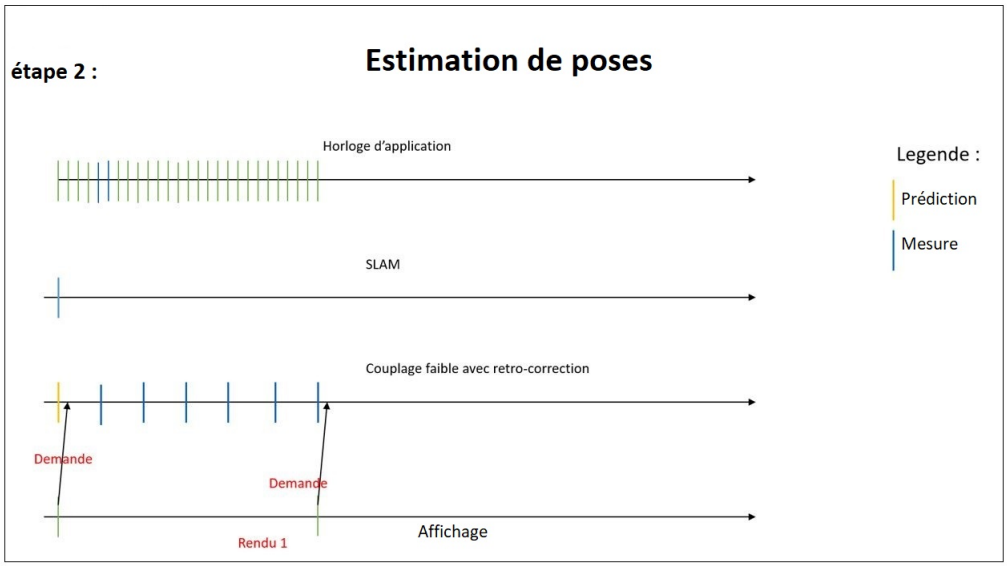


FIGURE 3.11 – Étape 2 : Acquisition de données inertielles et affichage de l'image 1 correspondante à la pose obtenue par le module de couplage après une prédiction et renvoi d'une nouvelle demande

La prédiction suivante exploite les corrections appliquées par la rétro-correction et par lissage du filtre.

L'architecture définie par la figure 3.2, permet l'exécution de différentes étapes présentées précédemment en mode parallèle. L'horloge générique permet de référencer toutes les actions et ordonner tous les événements. La fusion et le traitement s'appliquent sur une durée de quelques microseconde. Le mode asynchrone d'envoi de données est avantageux car il n'entraine aucune latence, les données sont envoyées suivant un mode de fonctionnement interne indépendant de chaque capteur intelligent

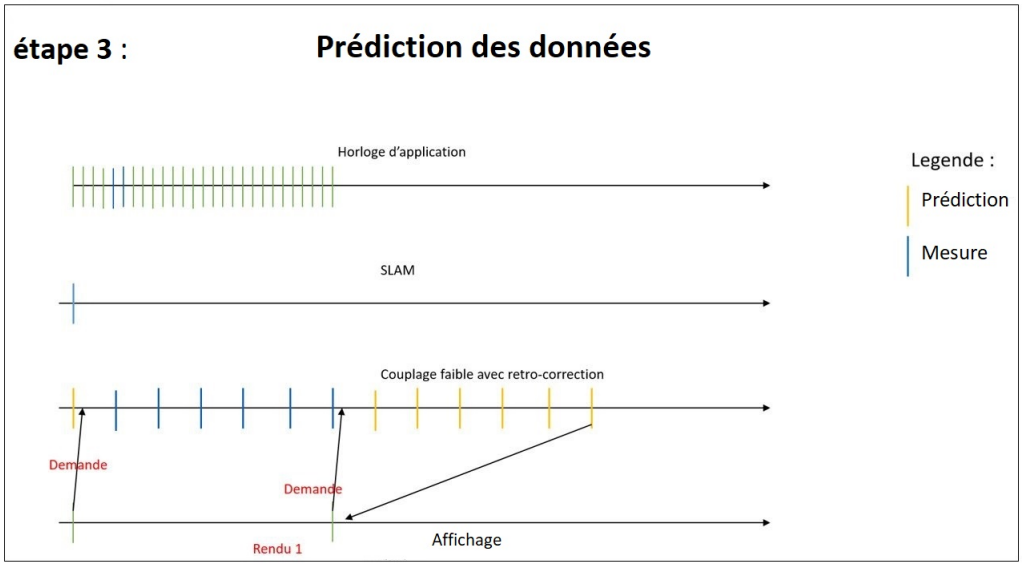


FIGURE 3.12 – Étape 3 : re-application de l'étape 1 en utilisant les données mesurées jusqu'à l'instant de la demande (t_d) et les données prédites pour $t_d + T_r$

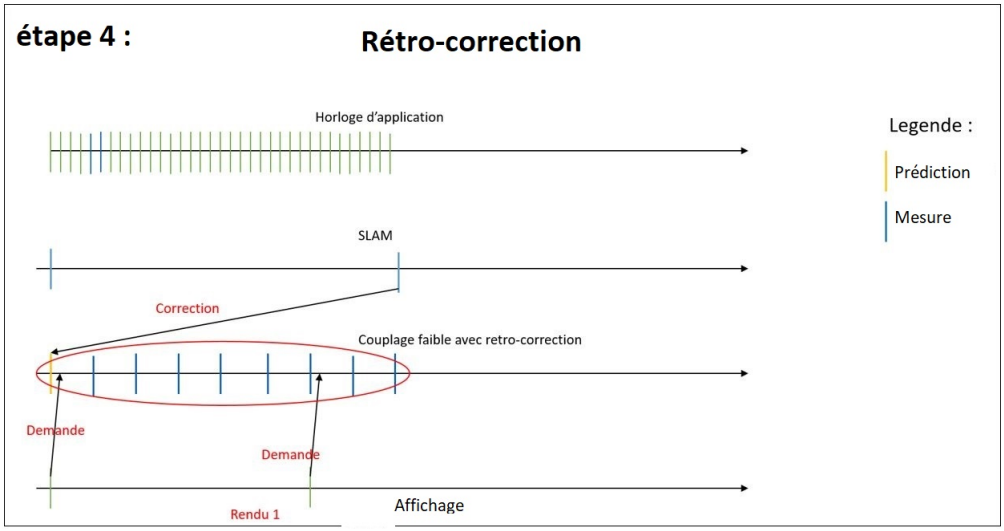


FIGURE 3.13 – étape 4 : la réception d'une donnée SLAM et l'application de principe de rétro-correction pour corriger toutes les poses précédentes et l'utilisation pour une futur prédiction

(Smart Sensor).

3.4.3 Traitement des inconsistances

Les poses inconsistantes sont des poses qui représentent une discontinuité dans la courbe d'estimation. En d'autres termes, ces poses représentent des valeurs aberrantes issues de l'algorithme ou

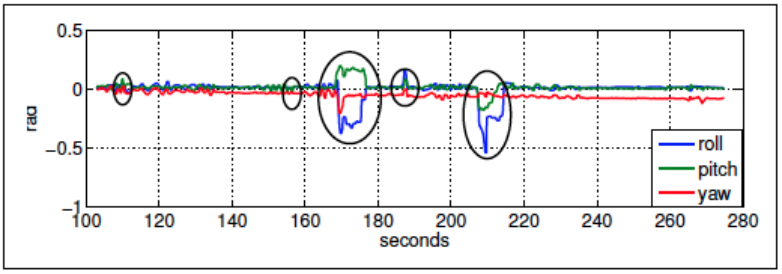


FIGURE 3.14 – Traitement d'inconsistances [104]

du traitement. L'analyse du comportement des filtres de fusion à pu mettre en évidence l'apparition d'inconsistances [62]

Pendant l'estimation d'une pose, plusieurs cas de figures sont possibles.

— Utilisation d'une donnée SLAM aberrante suite à une mauvaise estimation SLAM.

— Acquisition de données CI aberrantes (moins probable)

— Erreur d'estimation suite a une mauvaise fusion ou divergence algorithmique

— Absence de données ou mauvaise prédiction

Les différents points cités ci-dessus peuvent causer une mauvaise estimation malgré toutes les optimisations mises en place. Afin de traiter les inconsistances, un contrôle supplémentaire doit être réalisé sur les données estimées. Dans les travaux de [104], les inconsistances sont détectées et rejetées directement sans traitement. La détection d'inconsistance consiste à analyser la continuité de la courbe d'estimation et dans le cas où une valeur est jugé aberrante par rapport l'historique, elle sera directement rejetée et remplacée par la dernière estimation correcte. La détection de valeurs aberrantes selon [104] se fait en analysant la continuité et la fluidité de poses. Si une pose est aberrante elle est considérée comme pose inconstante. Dans notre approche, le traitement d'inconsistance se décompose en 3 parties :

— L'inconsistance de la prédiction de données

— L'inconsistance de l'estimation de poses

— L'inconsistance de la correction de pose

Un exemple de détection d'inconsistances est mentionné dans la figure 3.14. **L'inconsistance de la prédiction de données** est produite par le module de prédiction qui prédit des données pour le rendu. La cause de cette inconsistance peut être associée à une mauvaise estimation de pose, et l'utilisation de cette pose dans l'algorithme de prédiction avec une mauvaise prédiction de données. Généralement les données prédites sont utilisées pour un traitement sur un intervalle de 17 à 20 ms (plus de détails dans le chapitre suivant). Le décalage entre la pose de référence et la pose prédite pour un intervalle de 20 ms ne doit pas excéder quelques degrés. Supposons qu'un objet se déplace avec une vitesse de $10^{\circ}/s$ pour une période de 0.02 s l'objet se déplace donc de 0.2°donc si la valeur V à l'instant t est $V = 10^{\circ}$ à t+0.02 $V = 10.2^{\circ}$ soit 2% de la valeur initiale. Si la vitesse angulaire est de 90 $^{\circ}/s$, la position sera de 11.8° soit 20% de la valeur initiale. Dans un cas extrême et si le déplacement est rapide et la vitesse angulaire de $180^{\circ}/s$ la position sera de 13.6° soit une évolution de 36% de la position précédente. Le seuil de détection est donc fixé à 80% dans le troisième cas, la pose sera traitée comme une pose inconsistante. Une fois l'inconsistance détectée, l'algorithme compare les deux dernières poses envoyées au module de calcul du rendu, puis calcule l'intervalle de temps entre ces deux poses et estime la pose prédite en fonction de ce décalage. Si la pose estimée satisfait

la contrainte de 20%, elle sera envoyée, sinon elle sera rejetée et la pose envoyée est la dernière pose estimée par la centrale inertielle. La décision de prendre en compte la dernière pose estimée par la CI est basée sur deux arguments : La rapidité de traitement de la centrale inertielle. Dans le pire des cas, la pose utilisée sera retardée de 1 ms voire 2 ms si on rajoute le temps de traitement. La centrale inertielle utilise aussi l'historique mis à jour par la méthode de rétro-correction. La pose de la CI n'est pas complètement impactées par les dérives.

L'inconsistance d'estimation de poses Idem pour l'estimation de pose, le principe est le même. Un ratio d'estimation de 20% est accepté sans rejet. L'estimation d'une nouvelle pose est réalisée en fonction de l'historique des poses. **L'inconsistance de la correction de poses** La correction de poses est effectuée au sein du filtre (lors de la fusion). L'objectif est d'effectuer un contrôle qualitatif sur les données entrant dans le filtre et s'effectue de la manière suivante : Si la pose SLAM ou CI est aberrante ou décalée par rapport aux poses précédentes elle ne sera pas prise en compte. Elle sera exclue et remplacée par la dernière pose valable dans l'historique de poses SLAM ou CI respectivement.

3.5 Résultats et interprétations

Afin d'évaluer notre contribution, nous avons utilisé deux sources différentes des données. La première génère des données synthétiques et la deuxième est basée sur des données expérimentales. Les données expérimentales sont obtenues suite aux différentes expériences en variant différents paramètres (vitesse et méthode de fusion). Les tests sont appliqués sur le filtre de KALMAN et le filtre particulaire. Les critères d'évaluation sont :

- La précision du Tracking : Les poses estimées sont proches voire identiques aux poses réelles de la trajectoire réelle.
- La robustesse et la continuité : Ce facteur d'évaluation dépend du nombre de poses inconsistantes. Plus le nombre des poses inconsistantes est faible, mieux c'est.
- La fluidité du Tracking : Ce critère est lié au temps de traitement de toute la chaine algorithmique qui doit être les plus court possible.

Données de simulation

Les données de simulation sont générées d'une manière identique à celle présentée dans le chapitre précédent (Filtrage). La génération de données synthétiques commence par la génération d'une trajectoire quelconque. Afin de faciliter le traitement, une contrainte est imposée sur la variation de la trajectoire pour qu'elle ne soit pas trop ondoyante. Le générateur de trajectoire dans le mode de simulation tourne à une fréquence de 10 kHz. Ensuite, un générateur de données inertielles qui se base sur les données de la trajectoire et simule le capteur inertiel sera défini. Ce générateur utilise les données de la trajectoire pour estimer la vitesse angulaire et l'accélération instantanée. Le bruit de chaque capteur (accéléromètre et gyroscope) est rajouté dans une deuxième passe suivant un modèle gaussien pré-calculé en utilisant les acquisitions réelles de la centrale inertielle à simuler (cf chapitre Filtrage). Une étape supplémentaire est rajoutée pour la simulation des données de l'accéléromètre en additionnant la force de gravité selon l'orientation de l'objet dans le repère terrestre. Les données de la centrale inertielle (accélération et vitesse angulaire) sont générées à une fréquence de 1 kHz. Contrairement à la centrale qui passe par des étapes de calcul intermédiaires pour estimer la vitesse angulaire et l'accélération, la simulation du SLAM exploite directement les données de la trajectoire. Un bruit additionnel basé sur le modèle SLAM (cf chapitre précédent) se rajoute pour bien rapprocher le comportement réel d'un SLAM. Les données générées sont la trajectoire, les poses SLAM, les vitesses angulaires, les accélérations, les vitesses(pour déduire l'accélération), et les modèles des bruits. Afin de se rapprocher au mieux le comportement des capteurs, le décalage de temps de traitement est intégrée. Le processus de traitement d'images pour estimer une pose SLAM entraine une

certaine latence. En d'autres termes les poses SLAM sont toujours décalées par rapport la pose de référence d'un certain temps de traitement. Ce temps est variable en fonction de l'image acquise et de la vitesse de mouvement. Pour simuler les temps de traitement SLAM, une valeur est tirée aléatoirement dans l'intervalle [0,02s - 0,1s]. Cet intervalle correspond à une fréquence d'estimation de SLAM entre 50Hz et 10Hz.

3.5.1 Données de Simulation

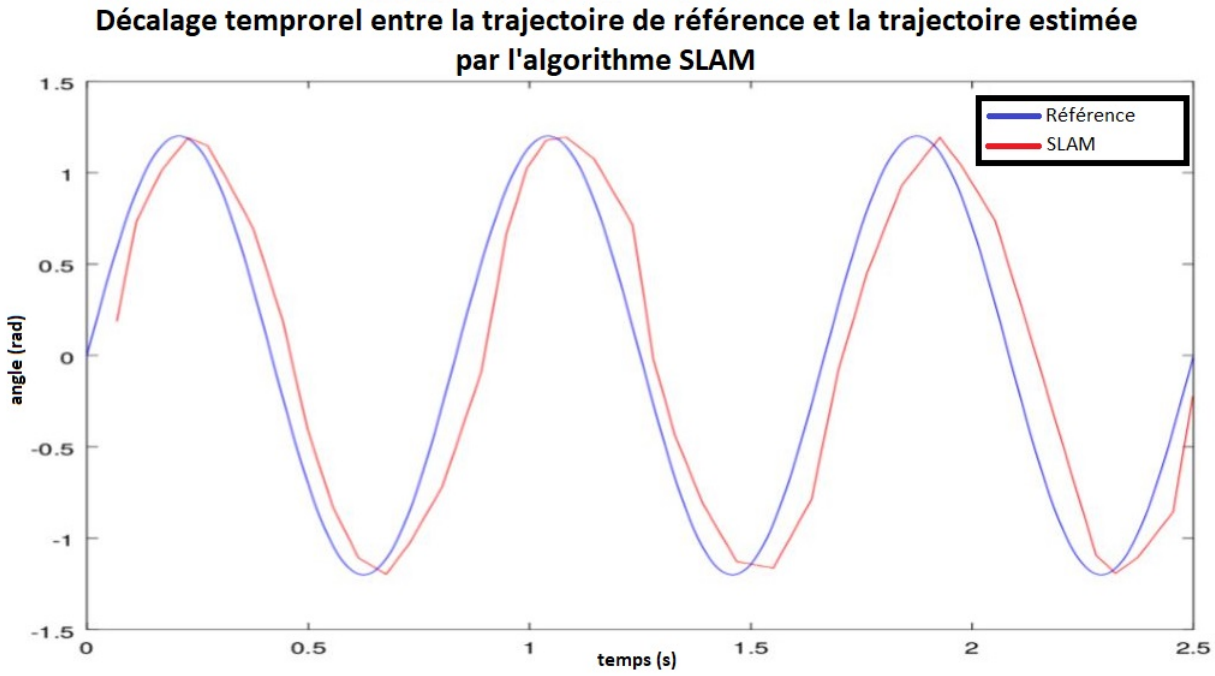


FIGURE 3.15 – Décalage temporel entre la trajectoire réelle et la trajectoire estimée par le SLAM

La figure 3.15 est composée de deux courbes (SLAM et trajectoire de référence), issues de données de simulation. La trajectoire de référence est une courbe sinusoïdale d'amplitude maximale 1.2 rad. La courbe SLAM est une simulation de l'algorithme SLAM et de son comportement. La superposition de deux courbes montre le décalage temporel entre la trajectoire réelle (courbe bleue) et la trajectoire estimée par un SLAM (courbe rouge). Ce décalage temporel par rapport la trajectoire de référence est du à l'addition du temps de traitement de l'ensemble des opérations élémentaires pour estimer une pose SLAM. Les opérations élémentaires commencent par l'acquisition de l'image à traiter jusqu'au transfert final de la pose sur le bus de poses. La courbe SLAM n'est pas parfaitement sinusoïdale car le bruit d'estimation s'additionne.

Afin d'évaluer les performances de la rétro-correction, une expérience est réalisée en se basant sur des données synthétiques. Le traitement est effectué sur les données avant d'appliquer la fusion dans le filtre particulaire. L'objectif est de comparer la fusion de données rétro-corrigées et les données utilisées directement sans aucun traitement préalable.

La figure 3.20 illustre les performances de la méthode de rétro-correction. Cette dernière est appliquée sur les poses estimées à partir de données gyroscopiques utilisées au sein d'un filtre particulaire et corrigées par des données SLAM. La fréquence d'acquisition de données gyroscopique

est de 1kHz et celle du SLAM est alentour de 10 Hz. La courbe en bleu affiche les performances du filtre particulaire sans la rétro-correction. La courbe rouge représente les performances du filtre particulaire avec l'algorithme de rétro-correction.

A l'issue de cette expérience, une conclusion primaire peut être tiré en se basant sur les données synthétiques. La rétro-correction apporte une amélioration d'estimation de poses. cette amélioration est de l'ordre de 20 %.

Données expérimentales

L'expérience est réalisée dans un environnement expérimental qui permet de répéter la même expérience avec les mêmes conditions externes y compris l'éclairage. L'ensemble du matériel utilisé pour réaliser les expérimentation est :

— Une carte Arduino pour contrôler un servo-moteur

— La carte UdooX86

— Une caméra et centrale inertielle de type TARA

La carte Arduino est programmée pour produire un mouvement automatique répétitif ou produire un mouvement manuel via un potentiomètre. Dans cette expérience nous avons intégré l'ensemble des algorithmes et des fonctionnalités cité auparavant. Le fusionneur de données utilisé est le filtre de KALMAN avec une prédiction de données. L'algorithme de traitement d'inconstance est utilisé après la fusion.

La figure 3.16 illustre le dispositif expérimental utilisé pour la réalisation de toutes les expériences. Cette étape à été réalisée avant l'installation finale de l'ensemble des capteurs sur le casque et la réalisation de tests sur le terrain. Plusieurs approches sont testées et évaluées en utilisant ce dispositif. Le servo-moteur est fixé sur un support solidaire d'une table fixe.

La figure 3.17 présente les courbes issues de l'ensemble des expériences de l'étude expérimentale. Les deux courbes sont les poses calculées pour la trajectoire du dispositif. Étant donné que le système est encastré sur une table fixe pour la répétition de l'expérience, la trajectoire de la caméra est circulaire (demi-cercle). La courbe bleue constitue la courbe de référence appliquée au servomoteur, elle correspond à l'évolution de la trajectoire réelle. La courbe rouge est la courbe d'estimation en se basant sur le couplage faible du SLAM et la centrale inertielle. Aucun autre traitement n'été rajouté pour cette phase. Sur la courbe rouge de la figure 3.17, l'estimation démarre toujours en retard par rapport à la trajectoire réelle. Une difficulté d'estimation pour les points spéciaux sur la courbe où la trajectoire change de pente instantanément (points anguleux). Sur le front montant de la courbe, on remarque deux phases : la première phase où l'estimation est en retard par rapport la trajectoire réelle. Cela est dû à la prédiction, visible sur les parties de la courbe entre 0 et $\pi/2$. Une fois que l'algorithme a pris les paramètres bien réglés, il devient en avance par rapport la courbe. La partie de transition aussi est relativement étalée sur le temps. Cette courbe prouve l'effet de retard d'images et d'estimateurs. La courbe rouge est presque droite par rapport à la courbe bleue sur les deux fronts (montant et descendant). C'est la concentration de toutes les estimation sur un petit intervalle. Vu que les données ne sont pas référencées, l'empilement des poses se fait d'une manière aléatoire d'où l'intérêt de la correction par rétro-correction.

La figure 3.18 présente le même cas que la figure précédente, le seul changement concerne les algorithmes d'estimation. Nous avons introduit la rétro-correction afin d'aligner toutes les références. La synchronisation de données a bien amélioré la courbe. On constate que la courbe rouge (estimation) est quasiment alignée sur la courbe bleue (référence). Sur le front montant, on constate une parfaitement superposition des courbes. Cependant sur le front descendant, la courbe est légèrement retardée par rapport la courbe de référence. Sur les pointes, de la courbe on constate qu'il ya une difficulté d'estimation due au changement rapide de la trajectoire. Le phénomène de retard est résolu avec la rétro-correction mais un nouveau phénomène est apparu. C'est la formation d'inconsistances.

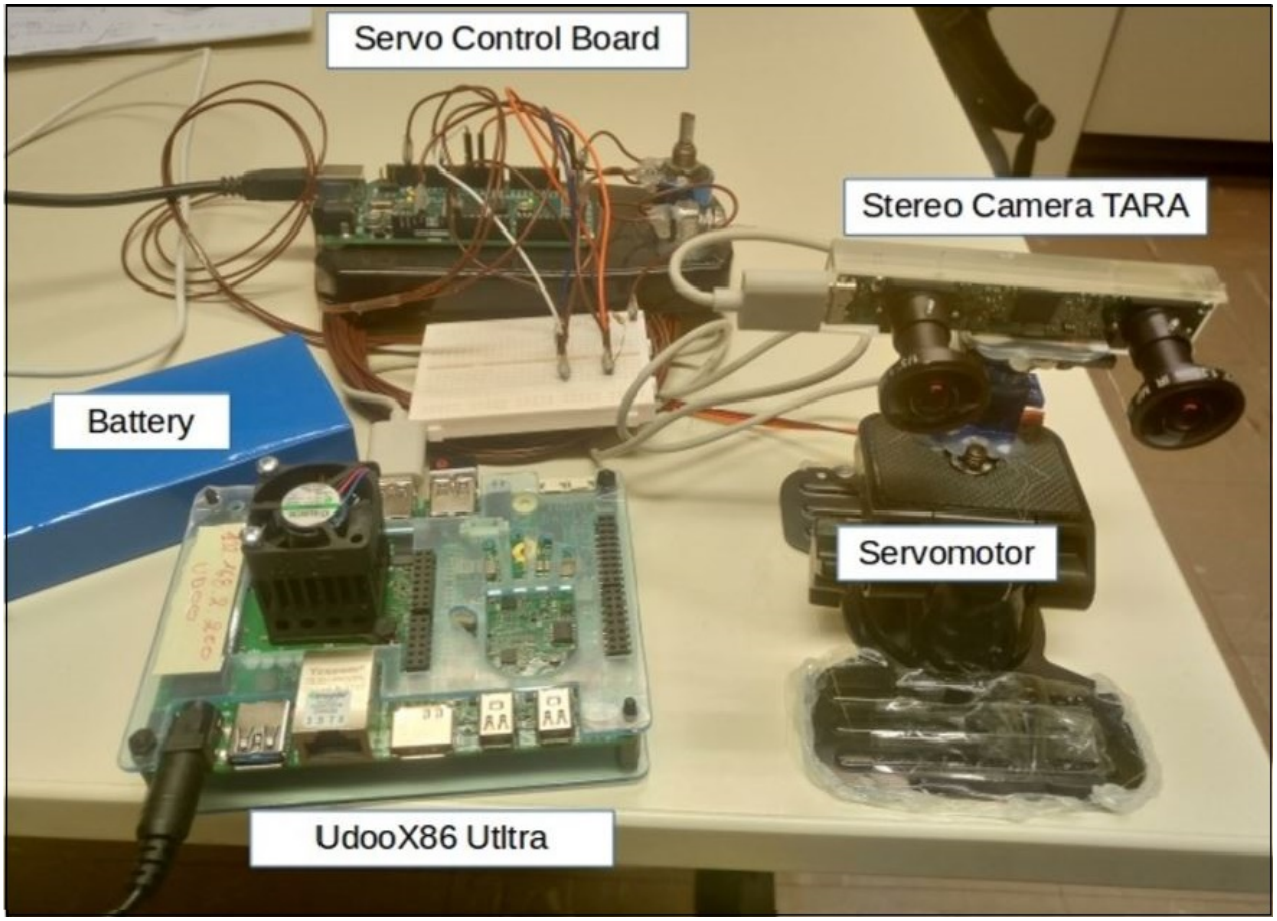


FIGURE 3.16 – Matériel utilisée pour réaliser l'étude expérimentale

Les petits pics qui apparaissent sur la courbe rouge au niveau de l'estimation d'angle $2\pi/3$ sont des poses inconsistantes. Le traitement d'inconsistance est nécessaire pour garder l'aspect fluide de l'algorithme.

La figure 3.19 illustre l'ensemble des traitements algorithmiques. La totalité des traitements à été utiliser pour l'estimation dans un contexte de couplage faible, i.e., la rétro-correction et le traitement des inconsistances. Cela donne une trajectoire lisse et continue dans le temps.

La courbe de la figure 3.20, représente la moyenne de toutes les expérimentations réalisées auparavant. L'expérience avec les mouvements automatisés est répétée 100 fois. Cependant celle avec les mouvements manuels est reproduite uniquement 5 fois. Les fréquence SLAM $<$ 20 Hz sont produites sur la carte UDOOX86. Les fréquences ¿20 Hz sont produite sur un ordinateur de bureau plus puissant (processeur et mémoire) La courbe orange (estimation sans l'utilisation de la rétro-correction) affiche une moyenne d'estimation d'erreur d'estimation de 35 $^\circ$ pour 100 poses soit 0.35°/pose . En revanche, la courbe grise (utilisation de la rétro-correction) affiche une moyenne de 20 $^\circ$ pour 100 poses soit 0.2 $^\circ$/pose.

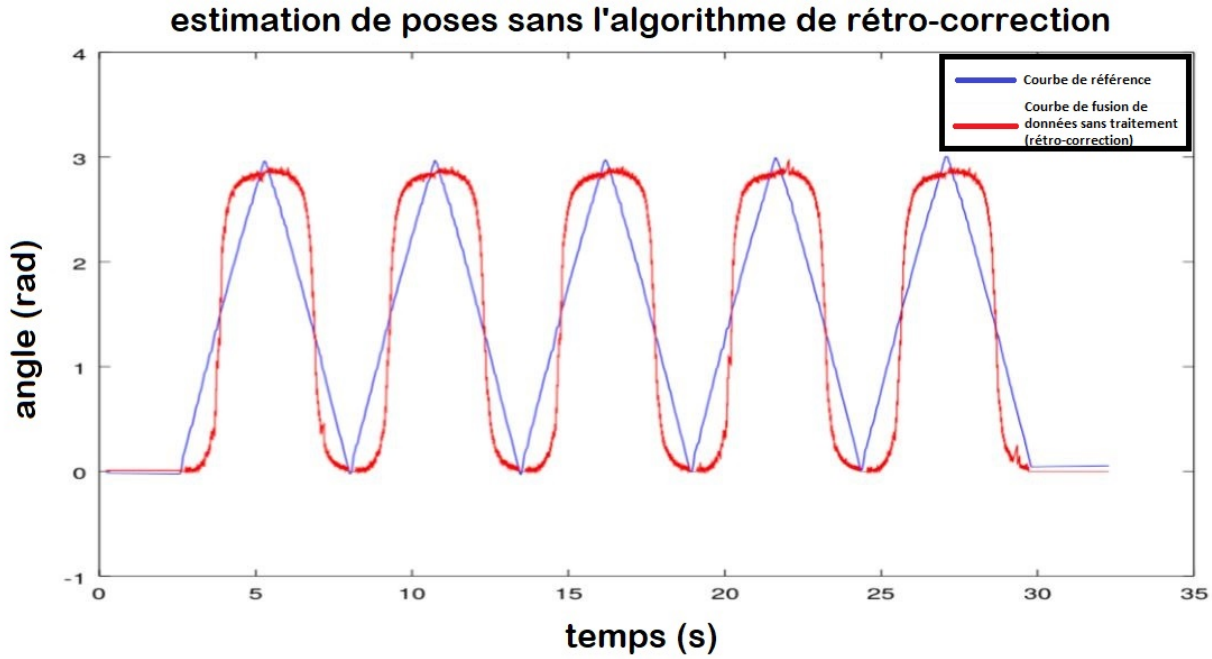


FIGURE 3.17 – Estimation de poses suivant un couplage faible et sans utilisation de la retro-correction

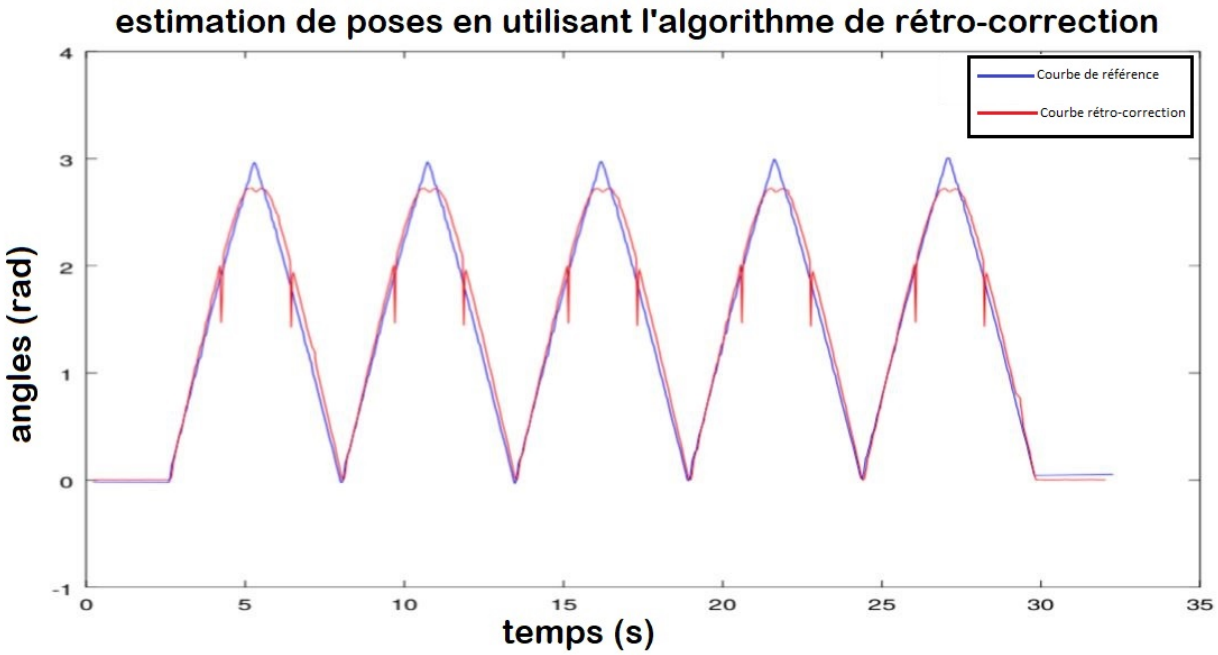


FIGURE 3.18 – Estimation de poses en utilisant le couplage faible et la rétro-correction

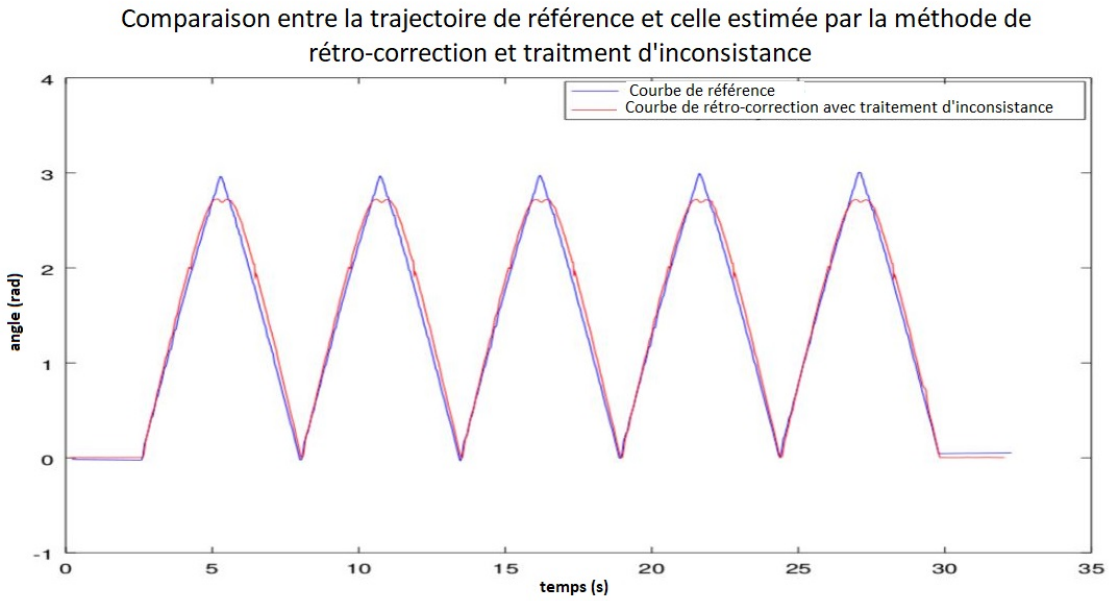


FIGURE 3.19 – Estimation de poses en utilisant le couplage faible et la rétro-correction ,avec traitement de poses inconsistantes.

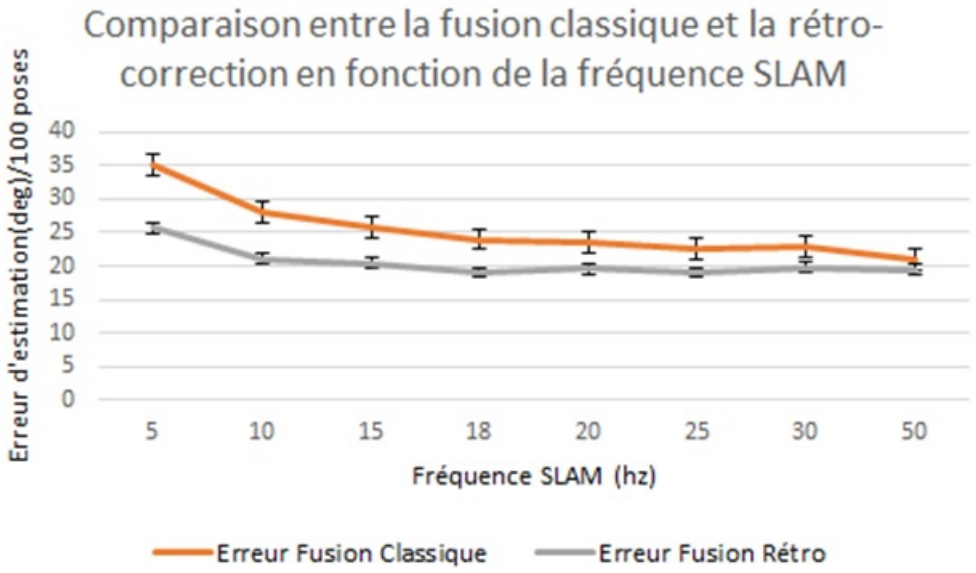


FIGURE 3.20 – Résultat d'utilisation de la rétro-correction sur les données simulées avec un filtre particulaire

3.6 Conclusion

Dans ce chapitre, nous avons présenté l'architecture informatique générique développée au sein de ce projet, afin de supporter les différentes parties d'une application de réalité augmentée. L'architec-

ture proposée a un aspect **générique** et **extensible**. La méthode de couplage **faible** permet l'ajout de différentes sources de données, et l'architecture développée permet de véhiculer les différentes données asynchrones. Ce principe supporte le principe de capteurs intelligents fonctionnant d'une manière autonome avec des fréquences différentes. La communication avec Unity (moteur de jeu choisi par convention entre les différents membres du projet) est générée via un 'Plugin' qui peut être extensible pour obtenir plusieurs fonctionnalités supplémentaires. Grâce à l'architecture mise en place, il est possible pour le module de rendu d'envoyer une demande de pose au module d'estimation de poses. Le 'Plugin' gère les transmissions de données et les poses disponibles. Étant donné qu'elle soit affichée après le temps de traitement nécessaire pour rendre une image de synthèse, la pose fournie par le plugin doit être une pose "prédite".

Dans ce chapitre, des techniques de prédiction pour prédire les données gyroscopiques et les données de rotations ont été étudiées. La prédiction de données est basée sur une extrapolation linéaire qui a montré ses performances face aux autres techniques (cubique et spline). Pour prédire les poses, le filtre de KALMAN est exploité et la technique de filtrage avec des données prédites est sélectionnée face à la prédiction de rotations avec le filtre de KALMAN lui même. Après la fixation de différents éléments qui servent à estimer une pose.Dans un premier temps les tests ont été appliqués sur des trajectoires générées de manière synthétique. Après cette étape, nous avons exploité des données expérimentales dans le but de valider les traitement proposés. La simulation théorique de différentes trajectoires à prouvé l'importance de la rétro-correction pour les fréquences SLAM faibles. Les performances de la rétro-correction sont identique à celles de la correction classique pour les fréquences élevées. Cependant on constate une amélioration de performance et l'erreur d'estimation chute de 0.045 rad à 0.03 rad soit une augmentation de 20% des performances d'estimation sur les faibles fréquences SLAM.

La rétro-correction implémentée dans ce chapitre permet de renforcer le Tracking d'une manière directe en ajustant l'historique de poses. Cet historique est utilisé pour calculer les poses présentes et les poses futures (prédictions). Cette méthode à montré son efficacité et son importance pour avoir d'une part des données précises et d'autre part des données lissées qui ne gênent pas l'expérience de l'utilisateur avec les dispositifs de réalité augmentée. Le traitement des inconsistances permet aussi d'améliorer la fluidité du 'tracking' et évite les "sauts indésirables".

Une analyse théorique est faite dans un premier temps pour valider le principe de la rétro-correction sur des données de simulation. Une fois le principe validé, une observation est réalisé sur deux systèmes différents. Le premier système est un pendule oscillant, et le deuxième est un moteur rotatif (présenté dans la section données Expérimentales).

Dans les sections précédentes, nous avons fixé le choix technologique pour l'algorithme de localisation par vision. Nous avons aussi déterminé le type de filtre adéquat avec le système utilisé. Nous avons développé la méthode de retro-correction qui permet de gagner en terme de performance et précision. Il est temps de faire les expérimentations sur le système complet. L'objectif du chapitre suivant est de montrer le bon fonctionnement des différents traitements proposés avec un minimum de latence possible malgré les contraintes imposées par le projet REVE5D.

CHAPITRE 4

ÉVALUATION DE LA LATENCE DU SYSTÈME

4.1 Introduction

Les chapitres précédents portent sur les techniques et les méthodes implémentées et étudiées dans la littérature pour améliorer la localisation 3D. Une technique de modification d'historique à été implémentée pour agir sur les poses de la CI (calculée par la navigation inertielle) en utilisant les poses SLAM. Cette méthode est nommée rétro-correction car elle prend en considération le décalage temporel causé par l'hétérogénéité des fréquences d'acquisition. Dans le chapitre précédent, nous avons caractérisé le temps de traitement de chaque module et avons proposé des algorithmes de corrections (rétro-correction et prédiction) en fonction du temps de traitement de chaque module.
Les contraintes relatives au projet REVE5D : Le contexte de développement de notre système de réalité augmentée soumis à plusieurs contraintes, généralement économiques. Deux contraintes majeurs sont imposées :

- Contrainte économique qui consiste à commercialiser le produit à un coût faible. Par conséquent il faut se limiter en terme de coût de fabrication.
- Contrainte de poids : il faut que le système soit portable et que le poids général de casque ne dépasse pas le 500 grammes avec un boitier téléporté plus lourd.

En respectant les deux contraintes mentionnées, l'ensemble de consortium à définit depuis le lancement du projet l'ensemble de matériel et les marques à utiliser pendant le projet. Le matériel utilisé est :

- UDOO X86 est une carte embarquée
- TARA avec des objectifs de type 'fish eye' pour la localisation SLAM + prédiction
- projecteurs OLED à une fréquence de rafraichissement maximale de 60 Hz.

L'ensemble matériel décrit précédemment, nous donne et dans le meilleurs de cas une fréquence de mise à jours de rendu correspondant à la fréquence de projection (projecteur OLED) soit 60 Hz. On se basant sur cette configuration et dans le meilleurs de cas la latence est de l'ordre de 16,7 ms.
Dans le présent chapitre, nous proposons dans un premier temps une définition précise de la latence, et de son origine. Ensuite, nous effectuons une étude critique sur les

différentes origines de la latence. Après cela, nous abordons notre système de calcul basé sur la rétro-correction et ses apports en termes de performances, les algorithmes proposés ainsi que leurs traitements qui permettent malgré les restrictions imposées par le projet de garder les plus faibles valeurs de latence soit une latence aux alentours de 18 ms dans le pire des cas. Enfin, nous terminons ce chapitre par une conclusion basée sur nos expériences.

4.2 Origine de la latence

Selon le dictionnaire de l'académie française la définition physiologique de la latence est : "Intervalle de temps entre une excitation et la réaction de l'organisme." En se basant sur cette définition la latence représente le temps écoulé entre l'excitation et la réponse. Une latence résulte donc de causalité de deux phénomènes (action-réaction).

La définition technique de la latence introduite par la communauté de la réalité virtuelle [65, 17] se base sur le même principe. Le temps de latence est celui qui s'écoule entre le mouvement et l'affichage associé sur l'écran (en anglais, "Motion To Photon Latency (MTPL)". Par analogie, l'excitation est donc le mouvement et la réaction représente la mise à jour de l'affichage relative à ce mouvement. Cette latence est caractérisée par et pour les pipelines des applications de réalité virtuelle et de réalité augmentée.

4.2.1 Description de la latence MTPL

La figure 4.1 est la présentation d'un processus de déroulement d'une application de réalité virtuelle. Ce processus est lancé par un mouvement détecté par un capteur de mouvements. Les données de poses sont ensuite traitées par le module du rendu pour être affiché sur un périphérique dédié. Le décalage entre l'action et la réaction affichée sur le casque est la latence du système ou plus précisément le MTPL.

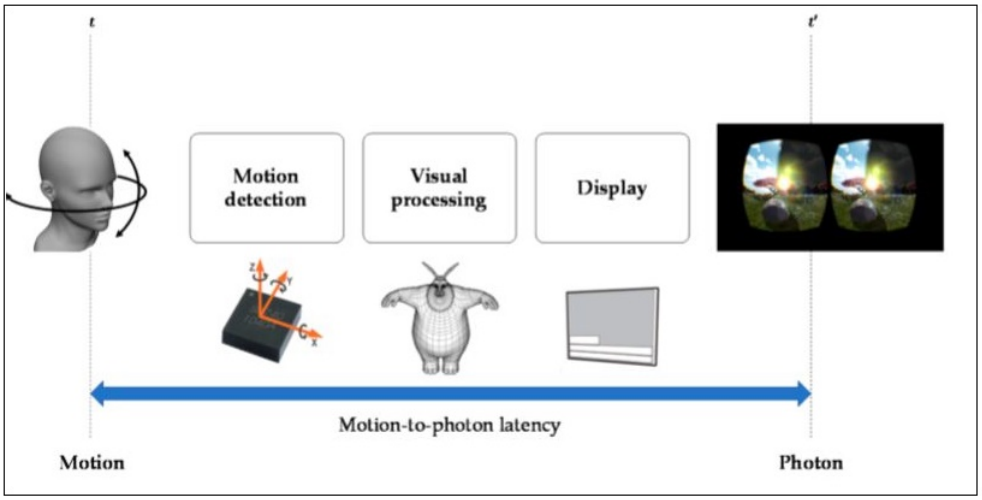


FIGURE 4.1 – Pipeline d'une application de réalité virtuelle [17]

Plusieurs études de latence ont été faites sur les systèmes de réalité virtuelle et augmentée. La

latence acceptée est celle au-delà de laquelle le système est considéré comme latent. Les applications de réalité virtuelle et celles de réalité augmentée ont des latences acceptées qui diffèrent de quelques milliseconde. Il existe une multitude de valeurs proposées dans la littérature pour définir ce seuil de latence. Les valeurs souvent retenues sont :

— $20ms$, qui est la latence maximale acceptée et inaperçue pour les systèmes de réalité virtuelle selon [13]

— $15ms$, qui est la latence maximale acceptée et considérée comme inaperçue par les opérateurs humains sur les casques de type see-throught pour les applications de réalité augmentée.

La figure 4.2 montre le processus d'une application de réalité augmentée basée sur une acquisition de données issues de plusieurs capteurs (dans le mode générique). Les données sont injectées dans le "Tracker" en mode asynchrone, qui interagit avec le module de calcul de rendu 3D, dans un cycle de demande (rendu 3d) / envoi de pose (tracker). Une fois l'image calculée, elle est envoyée au module d'affichage, et in fine sur le périphérique de visualisation.

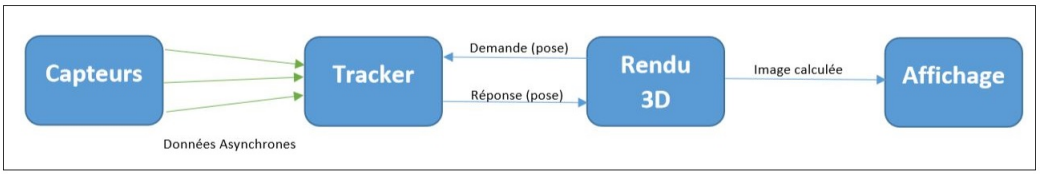


FIGURE 4.2 – Processus d'une application de réalité augmentée avec un couplage faible et un envoi asynchrone de données

Pour bien comprendre la latence d'un système complet de réalité augmentée, nous allons décortiquer les systèmes qui ont une architecture proche de notre système. Le mode de fonctionnement de tous les systèmes étudiés est asynchrone. Les capteurs opèrent en mode autonome et la fréquence d'acquisition de données leur est propre. Une fois que la donnée est acquise et traitée localement, elle est envoyée vers le "Tracker". Dans le cadre de ce travail, l'ensemble des capteurs est composé d'une caméra monoculaire ou binoculaire et d'une centrale inertielle.

4.2.2 Latence due aux capteurs

L'acquisition des données par une caméra passe par un temps d'exposition ; l'image fournie est celle de l'environnement avant le temps d'exposition. La date de l'image fournie par le capteur est celle de l'image de l'environnement réel avant le temps d'exposition et transmission. Soit τ_c le temps d'exposition, de codage et de transmission de la caméra. L'image acquise à un instant t est la représentation de l'environnement réel à l'instant $t - \tau_c$.

La figure 4.3 illustre le processus simplifié du fonctionnement du SLAM. Le SLAM récupère une image de la caméra. Après traitement, il génère une pose puis traite une nouvelle image pour l'estimation suivante. C'est un traitement répétitif en boucle ouverte. La latence totale du processus est la somme de la latence de la caméra τ_c, du temps de traitement, et du temps de propagation du résultat, généralement négligeable par rapport aux deux autres.

Comme déjà évoqué auparavant, la fréquence de la centrale inertielle est de l'ordre de $1000Hz$. Cela signifie que dans le pire des cas, la valeur envoyée au Tracker par la centrale inertielle, à un instant t, est une présentation de l'environnement à l'instant $t - \tau_I$, où $\tau_I = 1ms$.

La figure 4.4 illustre l'estimation d'une pose inertielle. Le temps de traitement et de calcul est mesuré d'une manière algorithmique en calculant la différence entre la date de démarrage de la

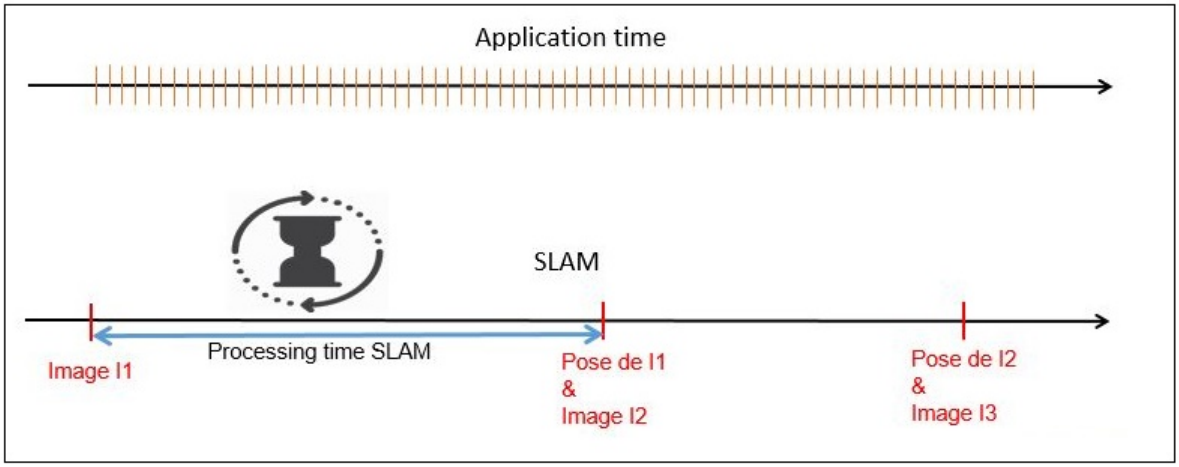


FIGURE 4.3 – estimation d'une pose en utilisant l'algorithme SLAM et la représentation du temps de traitement pour fournir des poses

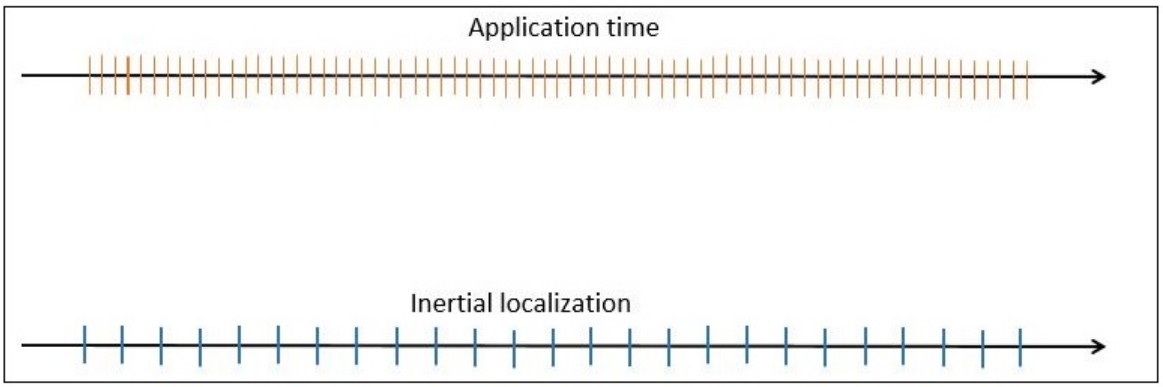


FIGURE 4.4 – Acquisition et traitement de données inertielles

procédure et la fin. Ce temps n'est que de quelques microseconde sur un ordinateur de configuration bureautique ordinaire, et de quelques microseconde sur la carte UDOO, de performance plus modérée. Le temps de traitement de la centrale inertielle sera donc considéré comme négligeable.

4.2.3 Latence due au Tracker, au Rendu et à l'Affichage

L'estimateur de poses ('Tracker') reçoit les poses SLAM et inertielles pour calculer les poses relatives. Dans ce module, la latence varie selon la méthode de fusion utilisée (filtre et technique de couplage). Les différentes techniques de fusion opèrent un traitement des données par des approches différentes. Pour chacune, les algorithmes agissent directement sur le temps de latence. Le couplage fort est généralement source d'une latence additive. Dans le cas du couplage faible, la latence peut être estimée selon les stratégies adoptées. Nous noterons τ_t la latence associée au traitement du Traker, et τ_T la latence globale associée aux capteurs et au Tracker.

La latence additive est $\tau_T = \max(\tau_c, \tau_I) + \tau_t$. Comme mentionné auparavant, cette latence est due

à la synchronisation et au traitement de données dans le filtre de fusion. C'est la latence produite généralement dans les architectures de couplage fort. La latence de couplage faible est définie autrement. Comme les capteurs opèrent en mode autonome, ils ont la capacité de fournir des poses avec des fréquences différentes. dans ce cas la latence s'écrit $\tau_T = \min(\tau_c, \tau_I) + \tau_t$.

Avec des capteurs fonctionnant en mode autonome, cas du couplage faible, la latence est donc inférieure à celle du couplage fort. Néanmoins, les temps de traitement dans le module de fusion τ_t diffèrent. Dans le premier cas, le module de fusion prend les deux entrées et applique directement la fusion. Dans le second cas, plusieurs méthodes d'optimisation de calcul sont appliquées mais elles prennent plus de temps. Cette latence est appelée "latence de mouvement vers tracking" ou 'Motion-To-Trackin Latency' (MTTL). La rapidité de traitement a un impact sur la précision, et vice versa. Il faut noter que le système de couplage fort est plus précis globalement qu'un système de couplage faible. Avec des traitements supplémentaires, la précision de systèmes de couplage faible peut atteindre celle du couplage fort.

Une fois que la pose est estimée, elle sera passée au module de rendu 3D pour générer les images à afficher. Après la réception d'une pose, la latence du module d'affichage est liée au temps de calcul d'image en fonction de la pose de la caméra. Ce temps de traitement n'est pas fixe, il est influencé par deux paramètres : la complexité de la scène et l'orientation de la caméra dans cette scène. Le nombre de polygones, l'éclairage, les animations et d'autres paramètres sont des facteurs d'augmentation de latence du calcul de rendu. La puissance du support de calcul est également un facteur déterminant dans l'estimation de cette latence. Pour les scènes simples (moins de 10000 polygones avec des sources d'éclairages modestes et d'animations simples), le taux de rafraichissement sur un ordinateur de bureau simple peut atteindre $100Hz$, ce qui induit un temps de latence de $10ms$, appelée latence de mouvement vers le rendu 'Motion-To-Rendering Latency' (MTRL). Elle est notée τ_R

L'affichage est enfin un périphérique de visualisation qui a une fréquence de mise à jour propre. De manière générale, les afficheurs ont une fréquence de $50Hz$ ou $60Hz$. Leur latence est fixe, de $20ms$ pour les afficheurs de $50Hz$ et de $17ms$ pour ceux de $60Hz$.

4.2.4 Bilan

notation	description	mesure
τ_c	latence d'exposition et de transmission de données images	temps de décalage entre l'image réelle produite dans l'environnement et la donnée fournie par le capteur
τ_I	latence de la centrale inertielle	temps de décalage entre la vitesse et accélération réelles produites dans l'environnement et la donnée fournie par le capteur inertiel
τ_T	latence d'algorithme de fusion de données (MTTL)	temps de traitement nécessaire pour produire une pose à partir des données
τ_R	latence de calcul d'images à partir de la position de la caméra (MTRL)	Mesure le temps de traitement nécessaire pour calculer une image de synthèse à partir d'une pose fournie
τ_A	latence de système d'affichage (MTPL)	latence d'affichage d'une image sur un dispositif d'affichage

Tableau 4.1 – Tableau récapitulatif de latences introduites par notre système

Le Tableau 4.1 présente les latences de différents blocs qui contribuent à la latence totale de

système. La performance finale dépend de différentes stratégies algorithmiques utilisées pour calculer et afficher les images de synthèse. Dans les approches naïves, la latence est cumulative d'un bloc à un autre. Supposons qu'on dispose d'un système de réalité augmentée basée sur :

— une caméra avec une fréquence d'acquisition à $50Hz$;

— une centrale inertielle avec une fréquence d'acquisition de $1kHz$;

— un algorithme de fusion de données basé sur une approche de couplage fort, où une image et la donnée inertielle correspondante sont fusionnées avec un temps de traitement après la réception de données de $10ms$;

— un calcul de rendu qui se base sur les poses fournies par le Tracker, avec un temps de traitement de $20ms$;

— un affichage sur un projecteur OLED de fréquence $60Hz$

Le Tableau 4.2 présente les différentes valeurs associées aux différents blocs. Les valeurs ne sont pas fixes mais elles changent en fonction de différents paramètres. Dans le cas de notre hypothèse, la centrale inertielle est opérationnelle avec une fréquence plus élevée que la caméra d'une part et, d'autres part l'algorithme de fusion se base sur les deux données (image et inertielles) pour estimer une pose. Ainsi, la latence de la centrale inertielle n'est pas cumulée avec les autres latences. On remarque aussi que la plupart des latences introduites dans le tableau dépassent largement la latence maximale pour les système de réalité augmentée optical see-throught. Dans ce cas, l'optimisation de la latence consiste à optimiser les différents modules, la communication entre les modules, et les modèles algorithmiques suivant le schéma présenté par les travaux : [101].

libelle	temps de latence (couplage fort) ms	temps de latence (couplage faible) ms
τ_c	20	20
τ_{IMU}	1	1
$\tau_{Tracker}$	30	11
τ_{Render}	20	20
$\tau_{Affichage}$	17	17
τ_{total}	67	48

Tableau 4.2 – Estimation de la latence pour chaque module

4.3 Optimisation algorithmique

Dans la section précédente, nous avons présenté les différentes sources possibles de latence. En détaillant l'architecture générale, on constate que l'optimisation de la latence peut se faire au niveau du Tracker, du rendu et de l'affichage. Après avoir décrit cette optimisation possible, le schéma général de notre application de réalité augmentée est donné.

4.3.1 Optimisation du Tracker, du Rendu et de l'Affichage

Le Module de Tracking dans notre système se compose de deux sous modules. Comme précisé dans le chapitre précédent, le couplage entre la centrale inertielle (CI) et le SLAM est faible. Les deux sources fonctionnent d'une manière indépendante. Le temps de traitement de cette partie de couplage est calculé d'une manière algorithmique en mesurant la durée totale de toutes les sous opérations incluses dans le fusionneur (module de couplage faible) avec la rétro-correction et la prédiction. Le temps de traitement varie en fonction de la date de la dernière pose SLAM disponible, il a une valeur

mesurée proche de 5ms en moyenne pour la carte UDOO, et de moins de 1ms en moyenne pour l'ordinateur, à la condition évidente que le SLAM fournisse une information fiable.

Le calcul du rendu final se base sur les différentes poses fournies par le module 'Tracker'. En utilisant cette pose, le moteur du rendu 3D (dans notre cas Unity) effectue une mise à jour sur la position de la caméra et renvoie ensuite l'image (de synthèse) vue par la caméra virtuelle. Le calcul du rendu est composé de deux catégories : La première est intrinsèque et dépend uniquement de la scène interne, et la deuxième est extrinsèque et consiste à coder l'image pour l'envoyer vers l'affichage. Dans la suite de ce paragraphe, on ne s'intéressera qu'à la première catégorie.

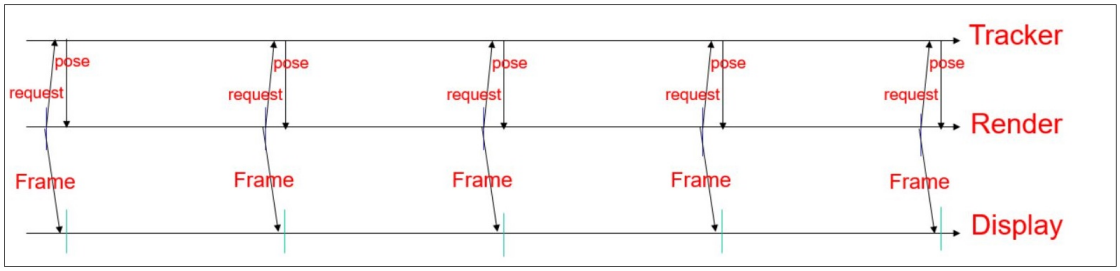


FIGURE 4.5 – Processus de calcul de rendu chronologique avec les demandes, réponses, et envoi vers le module d'affichage

La figure 4.5 montre le déroulement chronologique d'un processus de calcul de rendu. Dans un premier temps, le module de rendu envoi une demande au module de Tracking pour avoir une pose. Le module du Tracking répond à cette demande pour offrir la dernière pose estimée (méthode d'estimation détaillée dans le chapitre précédent). Une fois que la pose est reçue par le module du rendu, la mise à jour de la caméra virtuelle est effectuée et le calcul de l'image finale est réalisé. Le temps de calcul de cette image dépend de la scène et du nombre de polygones associés. Après le calcul, l'image sera envoyée directement vers le module d'affichage. Le schéma présenté par la figure 4.5 décrit le cas favorable où le calcul de rendu et l'affichage sont en phase. En d'autres termes, l'image fournie par le module de rendu est directement affichée. Cette configuration peut être obtenue dans le cas où la scène est relativement simple et le temps de calcul d'une image n'excède pas celui d'affichage. La meilleure solution pour optimiser le calcul du module de rendu est de fixer la fréquence de génération d'images (calcul de rendu final). Cela permet d'avoir un intervalle de prédiction fixe permettant de gérer le déphasage avec le module d'affichage. Cette opération est possible lorsque le temps de calcul d'une image est inférieur au temps fixé. Dans le cas contraire, le calcul continuera et dépassera le seuil fixé afin de fournir une image.

Un test à été réalisé en fixant la fréquence de rendu. L'expérience consiste à faire tourner un modèle 3D (cube) autour de son axe Oy, d'une position angulaire en degrés $P1 = (0, 90, 0)$ à une position $P2 = (0, -90, 0)$. La vitesse de rotation, de 60 degrés par seconde, est fixée en cohérence à la fréquence d'affichage de $60Hz$. Une caméra rapide (1000 image/s) filme l'écran du casque REV5D sur lequel la simulation est projetée. Le résultat attendu de cette expérience est d'avoir le cube dans une position à chaque pas d'affichage (16.66 ms). Les résultats obtenus sur la caméra montrent que le cube change de position chaque 16.66 ms dans la majorité des cas. On constate quelques cas où le cube a été affiché partiellement dans les deux positions. La figure 4.6 illustre les résultats observés sur la sortie de la caméra où l'affichage se réalise complètement au cours d'une période d'affichage (gauche) ou il s'étale sur deux périodes (droite).

La projection sur les systèmes de réalité augmentée type 'Optical see-throught' est basée généralement

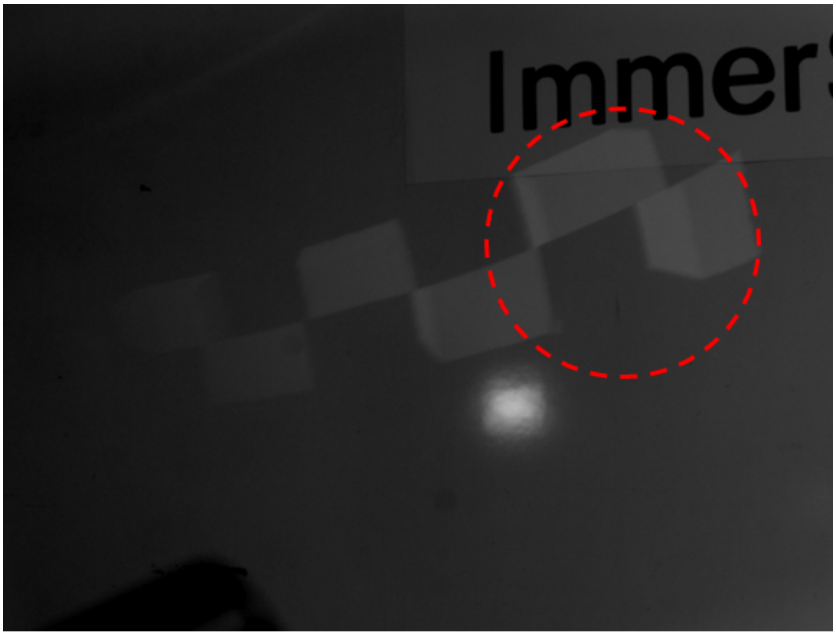


FIGURE 4.6 – Expérience réalisée sur le système d'affichage du casque REVE5D prototype1

sur les modes de projections OLED ou LCD. Dans le cadre de notre projet, le système de projection utilisée est OLED. La fréquence de mise à jours est fixée à 60 Hz pour une résolution de 1920x1080.

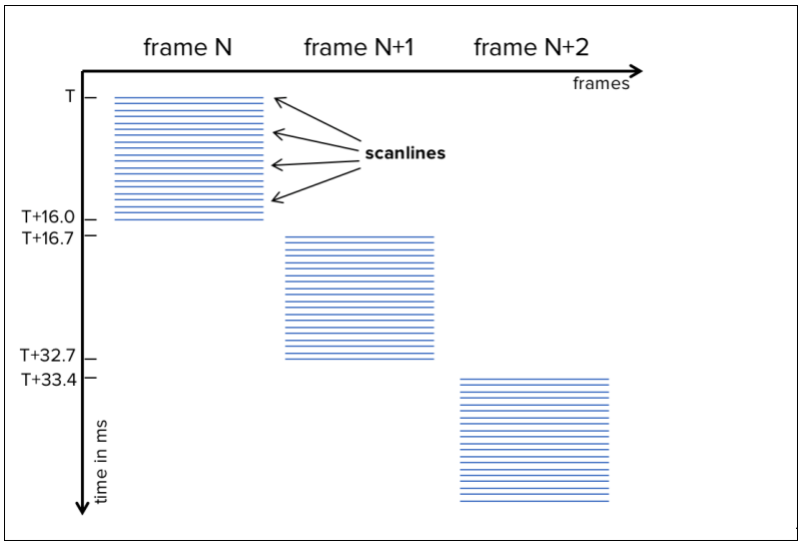


FIGURE 4.7 – Séquence d'affichage sur un projecteur OLED d'une fréquence de 60 Hz [101]

La figure 4.7 illustre le principe de projection OLED avec la représentation temporelle d'affichage d'images. La projection s'effectue d'une manière horizontale, ligne par ligne. Selon la figure, la projection s'effectue sur un intervalle de 16.0 ms. La projection d'une nouvelle image N+1 commence

après 16,7 ms afin de garder la fréquence d'affichage constante à 60 Hz. Après 16,7 ms, le système de projection commence à remplacer les lignes horizontalement d'une manière séquentielle de haut vers le bas.

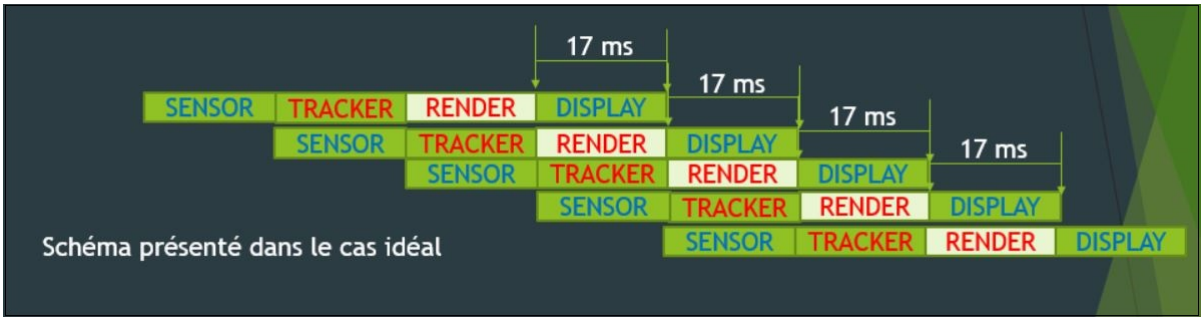


FIGURE 4.8 – Affichage en fonction de l'image calculée (le cas idéal)

FIGURE 4.9 – Affichage en fonction de l'image calculée avec un décalage 'déphasage'

La figure 4.8 présente l'agencement du pipeline pour gérer l'affichage. Les blocs sont mis en cascade pour assurer un bon phasage entre le calcul de l'image et son affichage. La figure 4.9 montre un dépassement du temps de calcul par le module RENDER ce qui induit un rendu d'image tardif. Etant donné que la fréquence d'affichage est fixe, le module d'affichage re-affiche donc la dernière image disponible. Ce qui est présenté par le bloc "Display" est en chevauchement avec le module RENDER cf figure 4.9. Dans ce cas, on a un retard d'affichage de 17ms.

4.3.2 Bilan

L'interaction entre les différents modules est toujours présentée d'une manière séquentielle. Dans la littérature, on trouve quasiment le même processus avec des légères modifications. Dans la suite, nous étudierons la possibilité d'optimiser le schéma générique afin de réduire le temps de latence affiché dans le tableau 4.2. La latence moyenne dans le bon fonctionnement de tous les modules et les sous modules est estimée à 47 ms. Cette valeur additive est réductible en agissant sur le temps de calcul du rendu par une synchronisation de l'affichage sur ce calcul.

Afin d'appliquer ce schéma générique, les ressources de poses sont partagées dans des structures de données accessibles par tous les modules et les sous modules. Le processus commence par l'acquisition de données, transmises vers les estimateurs locaux correspondants (SLAM pour la caméra et navigation inertielle pour la CI). Sur un Bus partagé, les deux algorithmes écrivent leurs résultats. Deux tableaux circulaires dédiés enregistrent respectivement les résultats SLAM et ceux de la CI. Le module de fusion traite les nouvelles données d'une manière indépendante. A chaque nouvelle pose

SLAM disponible, le module de fusion corrige l'historique de ce capteur et calcule une nouvelle pose en fonction des données SLAM et des données inertielles (voir chapitre précédent).

Avec l'utilisation de ressources partagées, la latence est mesurée avec cette formule min(latence (CI), latence (SLAM)). En d'autres termes, la latence du mouvement jusqu'au tracking est celle de la centrale inertielle qui est le périphérique d'acquisition le plus rapide dans le système de capteurs.

Le module de tracking demande des poses systématiquement au module du Tracking. Dans le schéma classique, cette pose est estimée pour l'instant t de la demande. Dans notre schéma optimisé, une prédiction est faite sur la durée de calcul d'image de synthèse (rendu). Pour simplifier la tâche, l'intervalle de prédiction est fixé à 16,7 ms, ce qui donne une fréquence d'affichage de 60Hz, choisie pour phaser l'affichage et le rendu afin de conserver constante la latence de transmission entre le rendu et l'affichage.

Le module de rendu envoi une demande de pose au 'Tracker' qui recherche la pose la plus proche disponible sur le bus de poses SLAM et les poses CI entre la date SLAM et la date de la demande de pose. Une prédiction de poses s'étale sur un intervalle de 16,7 ms pour couvrir le temps de calcul du rendu. Ensuite, toutes les poses sont entrées dans le fusionneur (voir chapitre précédent) pour fournir une pose d'affichage. Cette opération dure 3 ms sur les périphériques lent (carte Udoo X86). Dans ce cas, la perte de 3 ms de traitement entrainant un gain de 16 ms, le gain final est de 13 ms.

4.4 Mesure de la latence du système

Dans le paragraphe précédent, nous avons décrit théoriquement un processus d'optimisation et de réduction de la latence. L'objectif de cette section est de mesurer la latence totale du système (MTPL) pour voir les améliorations. Afin d'effectuer cette mesure totale, de nombreuse contraintes sont imposées sur le système. La latence totale est représentée par la réaction du système après le mouvement (MTPL). Le matériel utilisé pour réaliser cette expérience est :

— Caméra Rapide
— Centrale Inertielle
— Carte Arduino
— Oscilloscope à deux entrées
— Trépied
— Support pour le casque

La caméra rapide Phantom VEO-E est une caméra industrielle avec une fréquence d'acquisition qui peut atteindre 7500 images/s. Dans le cas de notre étude et pour des raisons techniques (quantité de lumière absorbé par l'objectif), nous nous limitons à une acquisition de 1000 images/s. Cette acquisition nous permet d'avoir une image chaque 1 ms ce qui est suffisant pour étudier la latence d'un système de réalité augmentée et plus précisément pour notre système (REVE5D). La caméra est placée derrière le casque afin de filmer les projections sur l'écran, pour représenter l'oeil de l'utilisateur final de ce casque. Étant en présence d'un système (OST), la caméra filme également le monde réel.

4.4.1 Expérience réalisée

La figure 4.10 montre la configuration du système utilisé pour mesurer la latence. Une caméra rapide encastrée sur un support permet d'effectuer des translations vers l'avant et l'arrière. Ce

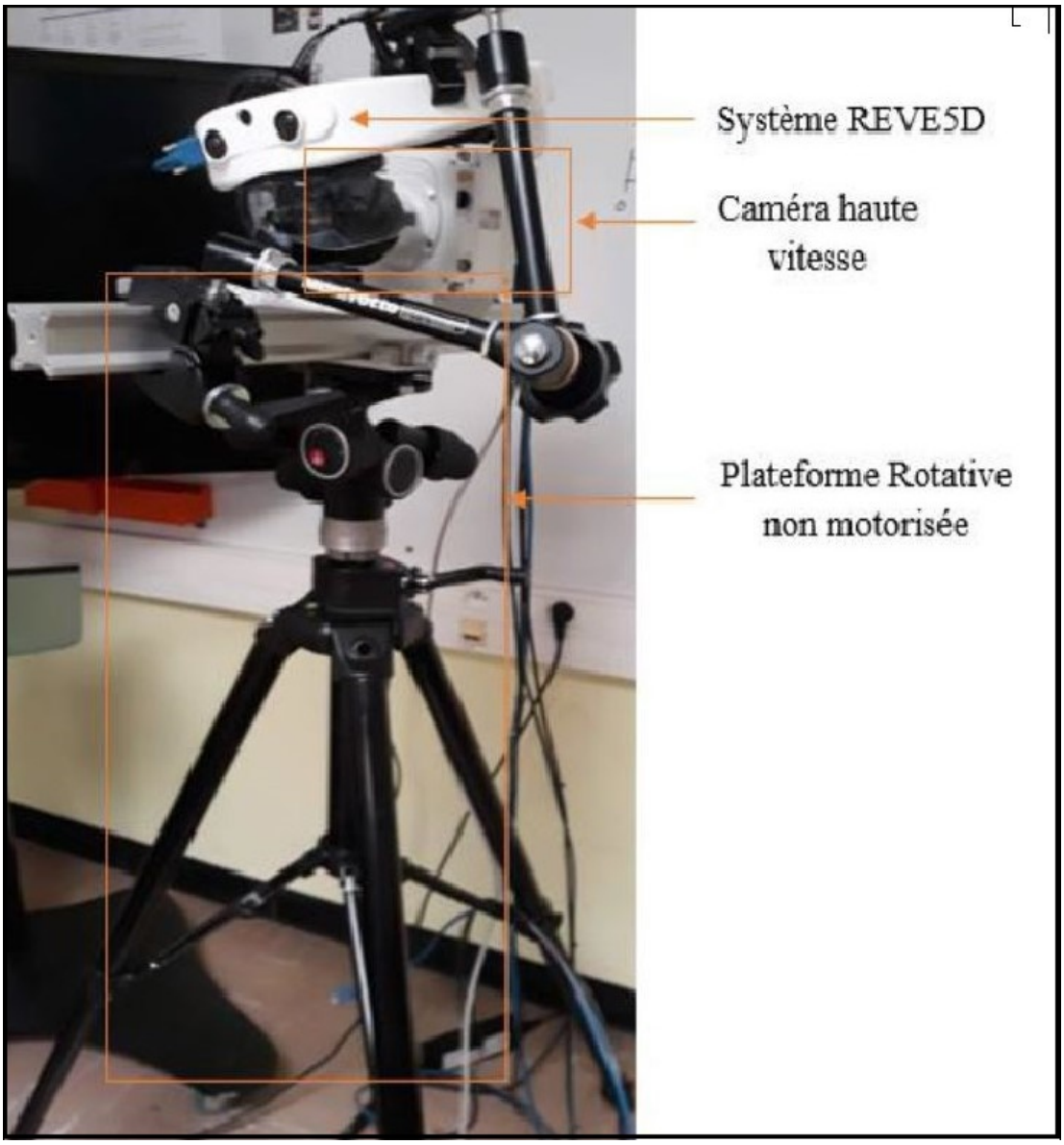


FIGURE 4.10 – Dispositif utilisé pour mesurer la latence

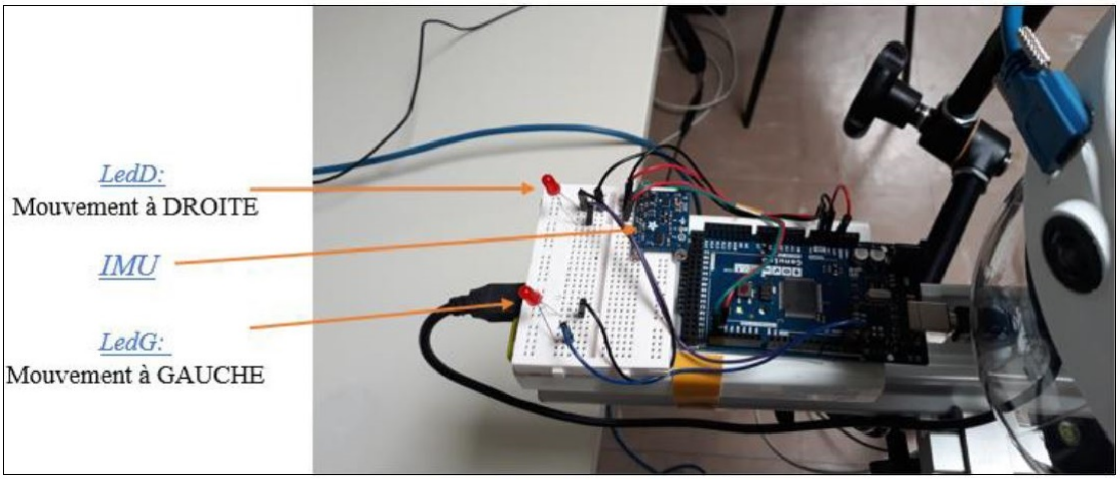


FIGURE 4.11 – Système de détection de mouvements

support est fixé sur un axe rotatif non motorisé. Le casque (REVE5D) et la caméra sont fixés sur le même support ; ils sont considérés comme un seul corps.

cas1

Dans cette expérience, le protocole propose d'effectuer un mouvement rotatif de l'ensemble (casque + caméra) afin d'interpréter directement la vidéo de sortie. Le démarrage et l'arrêt du mouvement sont détectés par une observation sur la vidéo. En réalisant cette expérience, avec différentes vitesses de rotations, il est délicat d'identifier la date de démarrage d'un mouvement avec certitude. Les mouvements de l'objet virtuel sont néanmoins identifiables sur le casque. La vidéo issue de cette expérience montre qu'il est difficile d'identifier le démarrage du mouvement réel.

cas2

Etant donné que l'identification précise de la séquence de démarrage est quasiment impossible pour toutes les vitesses, il est nécessaire d'ajouter un détecteur qui identifiera avec précision l'instant de démarrage. A cette fin une centrale inertielle avec une fréquence d'acquisition de 1khz est mise en place. Elle est installée sur le support rotatif. Une diode placée en face de la caméra signalera (par allumage) le démarrage d'un mouvement. elle reste allumée pendant 10 ms (10 images). L'imprécision due à la latence de l'allumage de la diode, sur la latence totale du système, est quantifié à 1 ms maximum.

La figure 4.11 montre le système composé de deux diodes pour détecter les mouvements (gauche + droite). La carte Arduino est utilisée pour interpréter les données IMU et allumer les LEDs. Le système de détection de mouvements est placé en face de la caméra rapide afin de voir l'état de deux diodes.

La figure 4.12 montre le système de détection complet avec le détecteur de mouvement placé devant la caméra qui facilite l'identification de la date de démarrage d'un mouvement physique.

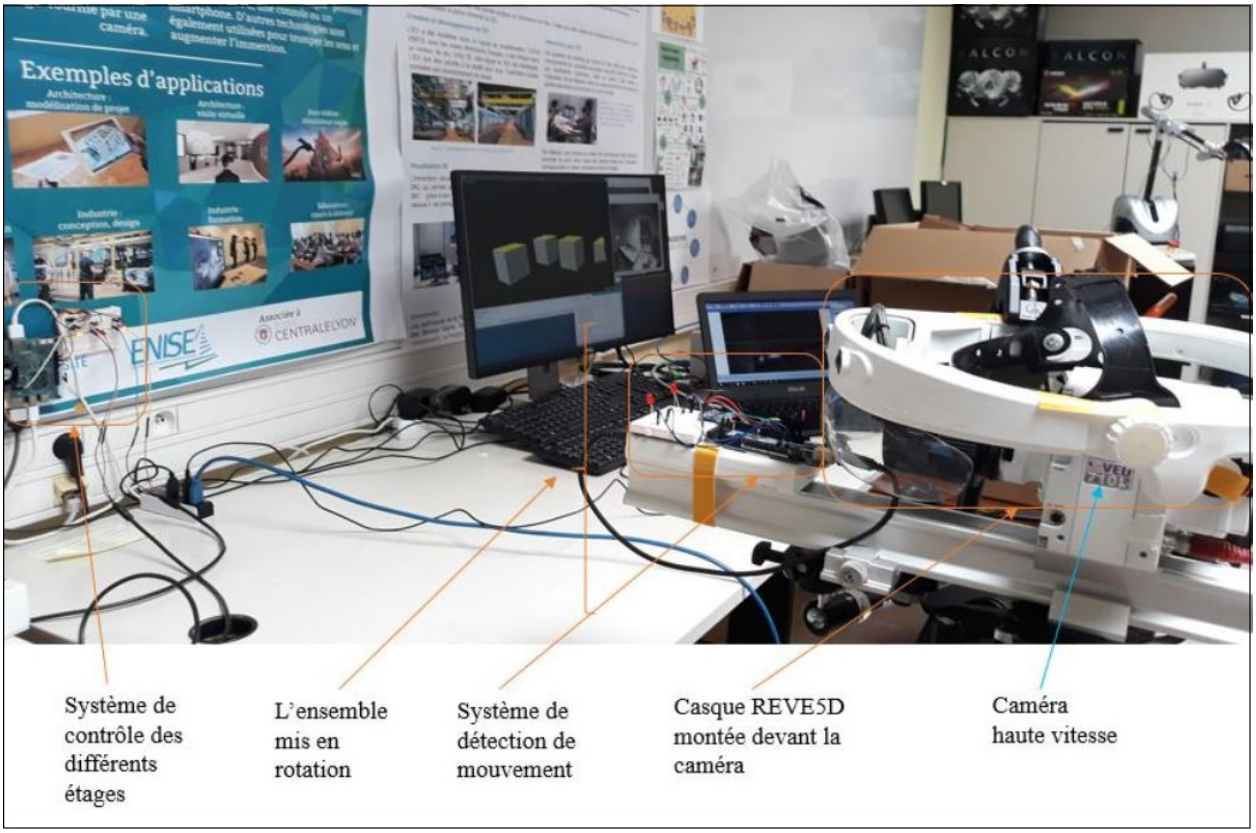


FIGURE 4.12 – Dispositif de mesure complet avec la mesure de tous les signaux de sortie

Configuration de l'environnement

L'expérience est réalisée dans un espace sans fenêtres où l'éclairage et la puissance d'éclairage sont toujours fixes et identiques d'une expérience à une autre. La carte est optimisée, elle permet d'avoir une localisation robuste. La scène de réalité augmentée utilisée est simple, basée sur des cubes placés dans des endroits de la salle riches en textures (pour une localisation SLAM facilitée). Le mouvement de la caméra est manuel. Une optimisation de la répétabilité de l'expérience doit passer par l'automatisation de la partie "rotation". Un plateau rotatif avec des moteurs non vibrants serait nécessaire dans ce but. Pour avoir plus d'information sur le comportement de la la navigation inertielle et du SLAM, nous avons connecté la carte UdooX86 à un Oscilloscope. Une mesure complémentaire est effectuée sur le temps de traitement du module de tracking.

La figure 4.13 montre la liaison faite entre la carte UDOO X86 et les diodes afin de récupérer les signaux de chaque module. A chaque demande de pose la carte envoie un signal appelé RENDER HIGH qui présente la date d'envoi de demande pour calculer une nouvelle image. Le signal SLAM HIGH représente la notification SLAM après l'estimation (disponible) d'une pose. Le IMU HIGH est le signal qui notifie la disponibilité d'une pose IMU calculée à partir de l'estimation Inertielle.

Afin de bien comprendre la latence et ses origines et bien identifier les paramètres qui contribuent directement et indirectement à la latence du système, on a fait varier plusieurs paramètres qui sont les suivants :

— **La vitesse de rotation** : Une estimation de 3 niveaux de rotations. Le premier niveau consiste

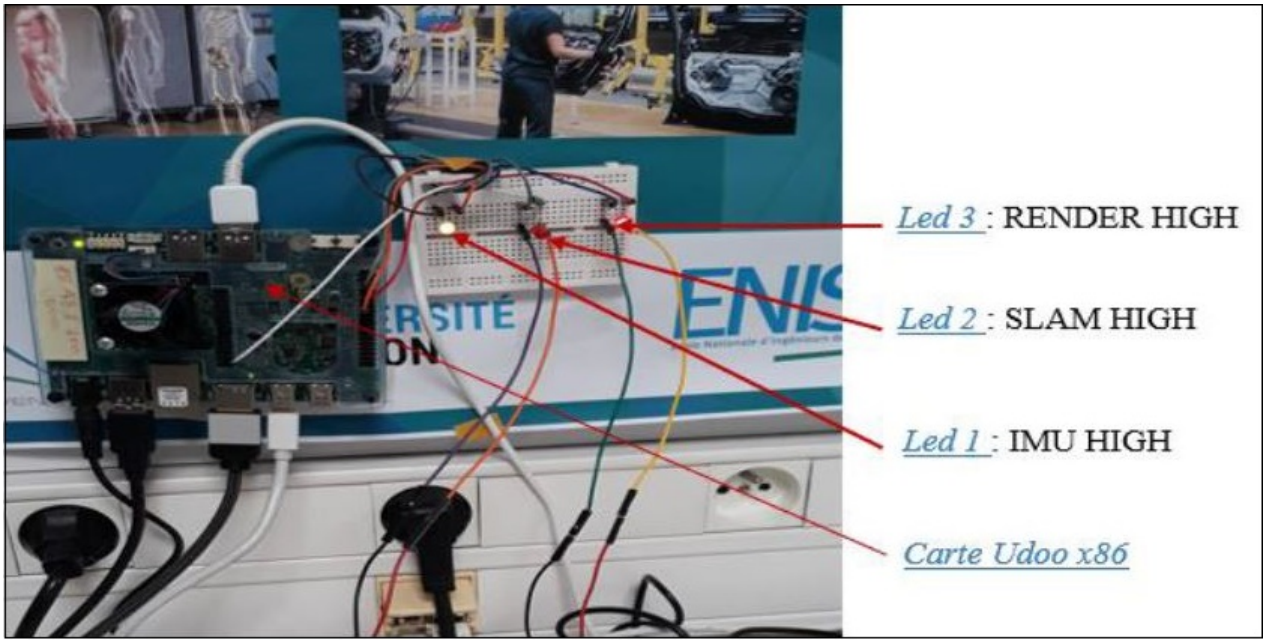


FIGURE 4.13 – Système d'envoi de signaux RENDER, SLAM, IMU pour mesurer les performances de chaque module

a produire une rotation relativement faibles par rapport un mouvement simple de la tête soit une vitesse de rotation ¡ 60 deg /s. Un deuxième niveau qui représente une vitesse de rotation moyenne qui rapproche le mouvement d'une tête dans les cas d'une observation où l'opérateur ne produit pas des mouvements brusques soit une vitesse de rotation comprise entre 60 deg /s et 120 deg /s. Finalement, la troisième vitesse représente un mouvement rapide qui simule un mouvement brutal soit une vitesse de rotation ¿ 120deg /s.

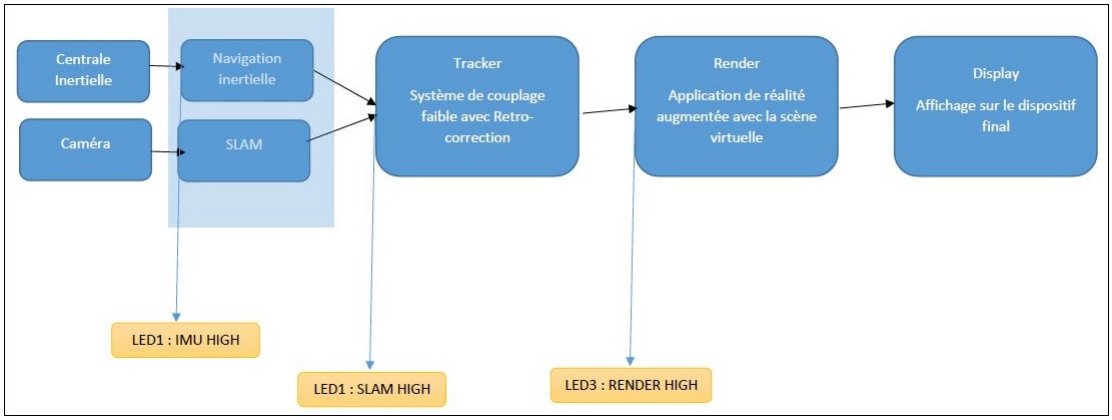
— **La fréquence de rendu (logiciel Unity)** : La fréquence minimale acceptée pour les systèmes de réalité augmentée see-through est fixé à 50 Hz. Les fréquences fixées pour les tests sont (100, 75, 60, 50, et 25)Hz.

— **L'algorithme de couplage (avec prédiction et sans prédiction)** : La contribution de la thèse. La prédiction est la partie où on envoi une future pose. Cette dernière est envoyée au module de rendu. Après le traitement nécessaire par le module de rendu, la pose future devient une pose présente ou une pose dans le passé. Si le temps de traitement dépasse le temps de la prédiction la pose devient une pose passée. Dans ce cas la prédiction consiste à baisser l'imprécision de la pose ou de décalage temporel.

4.4.2 résultats et discussion

La figure 4.14 montre le retour visuel obtenu sur l'image de la caméra rapide après la réalisation de l'expérience. Sur l'image, la diode allumée est bien identifiée. Les objets 3D sont visibles et l'analyse de leurs comportements est possible.

La figure 4.15 illustre les résultats obtenus suite à plusieurs itérations de mesures. Les courbes représentent la moyenne des latences mesurées pour les différents types de rotations : rapide, moyenne, et lente. Ce résultat est estimé par rapport à une fréquence de rendu. Une courbe intérmédiaire est



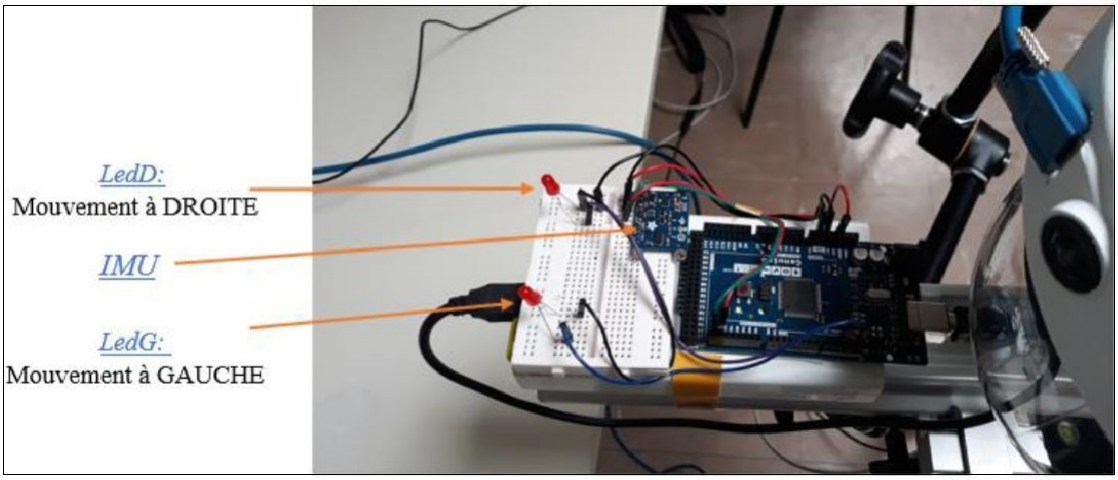


FIGURE 4.14 – Image acquise par la caméra rapide

introduite pour calculer la moyenne de toutes les latences par fréquence (courbe jaune).

Sur le graphique illustré par la figure 4.15, la courbe grise représente la latence d'une rotation rapide, qui est comprise entre 190 est 170 ms. C'est une latence, perçue par l'utilisateur car supérieure à la valeur maximale admise de 20ms. Les courbes de rotations moyennes et lentes sont quasiment confondues. La latence moyenne est comprise entre 65 et 40 ms. En analysant ce graphe, la latence minimale est plus importante que la latence autorisée pour une application de RA.

La figure 4.16 présente le graphique de mesure de latence comme la figure précédente. Cependant l'algorithme est renforcé par une prédiction. La mesure est faite comme la précédente sur une séquence de démarrage où le système change d'état (statique et dynamique).

La figure 4.16 montre que l'utilisation de l'algorithme de prédiction pour les séquences de démarrage dégrade les performances et augmente la latence du système. Les valeurs mesurées de latence pour une séquence de démarrage sans utilisation de l'algorithme de prédiction sont comprises dans l'intervalle [190 ms ,40 ms]. Avec l'algorithme de prédiction, les valeurs de la latence sont comprises entre 250 ms et 70 ms. On peut déduire dans un premier temps que la prédiction n'a pas d'apport pour le démarrage d'un système. Il faut noter qu'avec l'utilisation de la prédiction, les latences de rotations lentes et moyennes sont quasiment identiques. On peut voir qu'à partir d'une fréquence de 50 Hz le

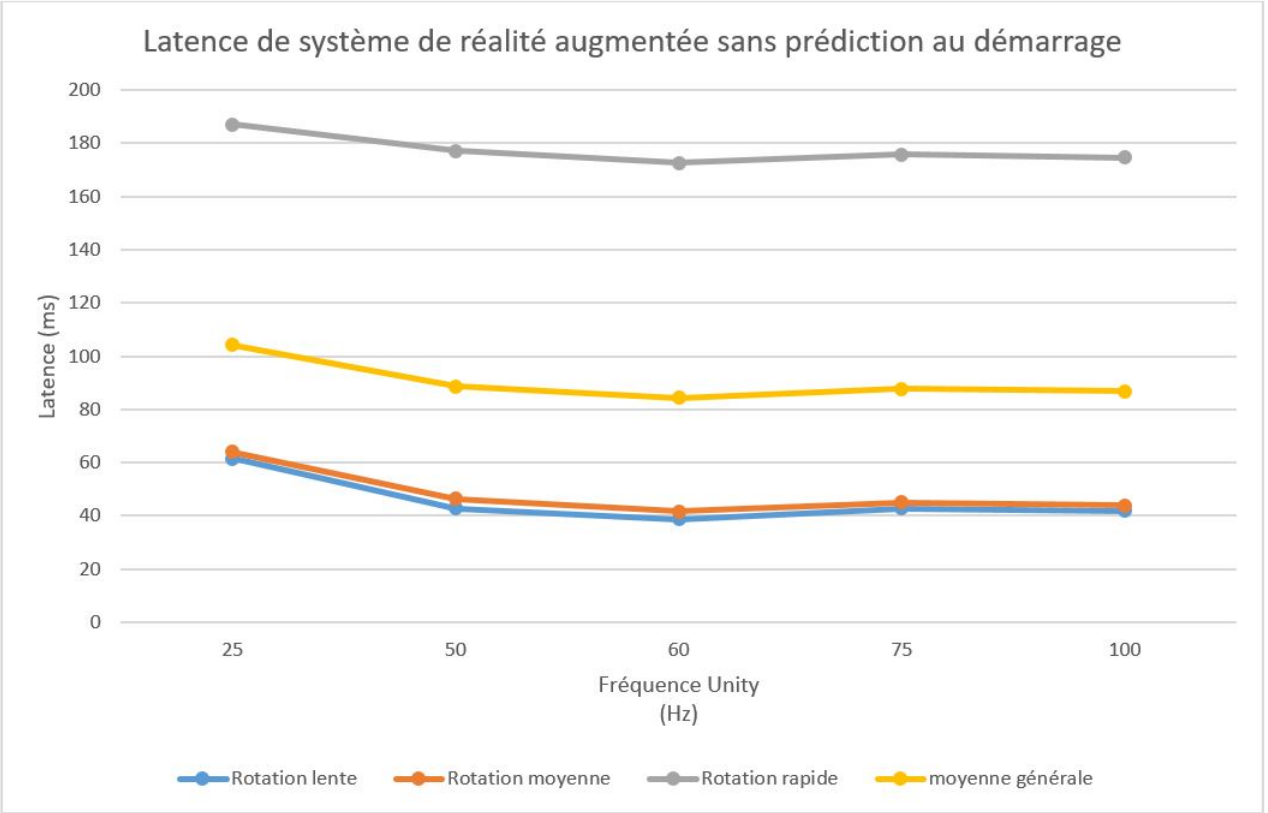


FIGURE 4.15 – Résultat de mesure de latence de notre système de réalité augmentée sans l'utilisation de la prédiction.

système tend vers un régime quasi stable.

Les courbes représentées par la figure 4.17, illustrent les différentes latences calculées en fonction des vitesses de rotations et des fréquences de calcul de rendu. Dans cette configuration l'estimation de la pose est basée uniquement sur la rétro-correction et pas d'utilisation de la prédiction. Cette latence est mesurée pendant un mouvement continue de la caméra. La latence de la rotation rapide est trop importante pour un système de réalité augmentée. Les rotations moyennes et lentes ont une latence moins importante mais qui reste importante par rapport au seuil autorisé pour une application de réalité augmentée.

Les courbes présentées dans la figure 4.18 représentent les résultats de mesure de la latence pour laquelle l'algorithme de prédiction est utilisé pendant les mouvements. En exploitant la prédiction, la latence sur les rotations lentes et rapides est moins importante que la latence mesurée sans l'utilisation de la prédiction. Les courbes bleue et orange se rapprochent à partir de la fréquence 75 Hz. La valeur minimale de latence pour toutes les courbes est de 60 Hz. Cette fréquence représente la fréquence d'affichage.

La figure 4.19 présente les courbes de latences mesurées pour une désactivation de la prédiction pour la séquence de démarrage et son activation après le reste du mouvement. L'expérience se décompose en deux parties : La détection de la phase de démarrage et la phase d'arrêt pour désactiver la prédiction et la partie de mouvement dynamique et continu.

Les courbes de la figure 4.19 montrent que les rotations moyennes et lentes et à partir de la

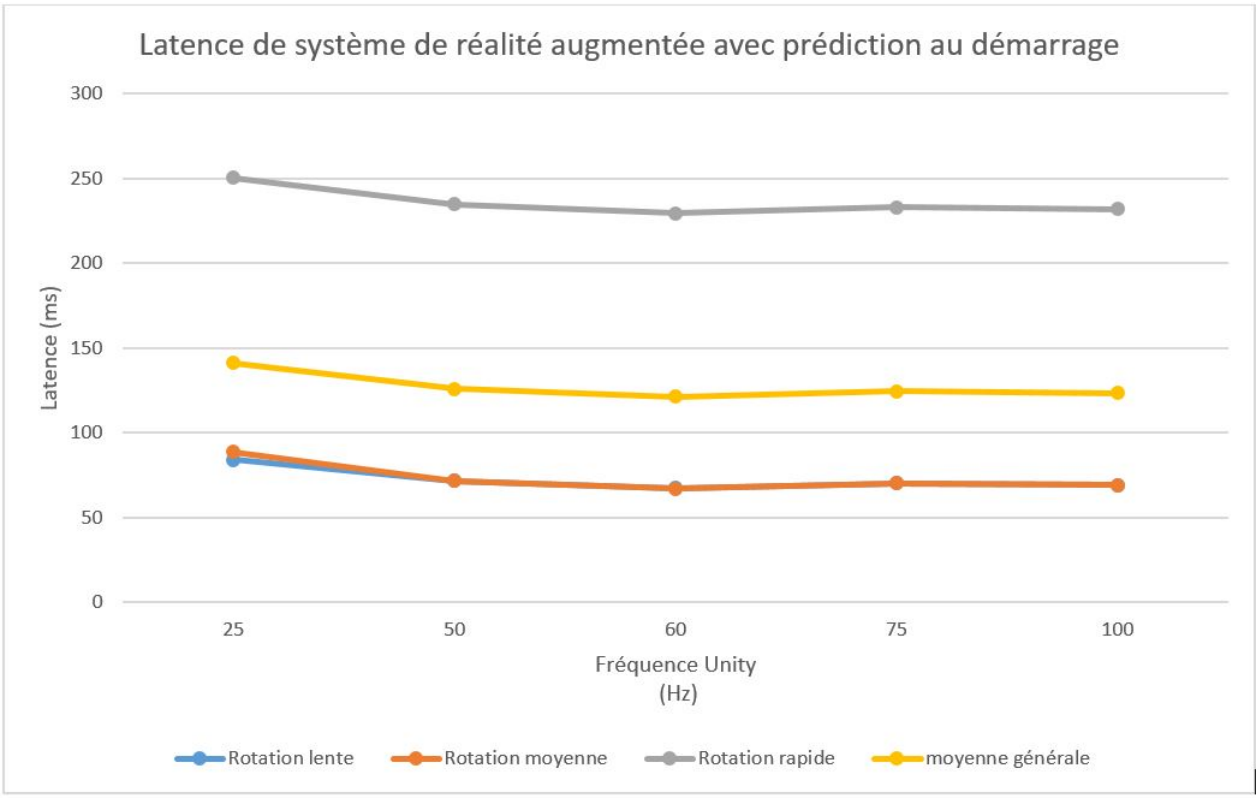


FIGURE 4.16 – Résultat de mesure de latence de notre système de réalité augmentée avec la mise en place d'une prédiction au démarrage

fréquence de rendu 50 Hz ne dépassent pas les 25 ms. La rotation lente représentée par la courbe bleue descend en dessous de la valeur de 20 ms pour la fréquence 60 Hz. Cette valeur est acceptable pour les systèmes de réalité augmentée. Pour la même fréquence, les rotations moyennes (courbe orange) atteint son minimum aussi. La latence de rotations rapides a nettement baissé par rapport aux autres mesures mais elle reste trop importante pour une application de réalité augmentée.

La figure 4.20 présente les courbes de la moyenne de toutes les latences pour tous les algorithmes utilisés en fonction de la fréquence du rendu utilisée. En analysant toutes les courbes, on observe que la latence pour toutes les fréquences inférieures à 50 Hz sont plus importantes que les autres latence pour les fréquence supérieures à 50 Hz. La valeur minimale sur toutes les moyennes est centrée sur la fréquence de 60 Hz. On peut déduire que la fréquence optimale pour une latence minimale est celle d'affichage. Cette fréquence est raisonnable. Si on augmente la fréquence de rendu, on augmente les demandes de ressources et on risque de demander des ressources informatiques qu'on ne dispose pas.

Les rotations rapides, qui sont présentées sur les figures de 15 à 20 par une courbes grise et pour tous les algorithmes utilisés dépassent toujours le seuil de latence autorisé par une application de réalité augmentée. Notre système à cet instant ne permet pas de gérer les rotations rapides.

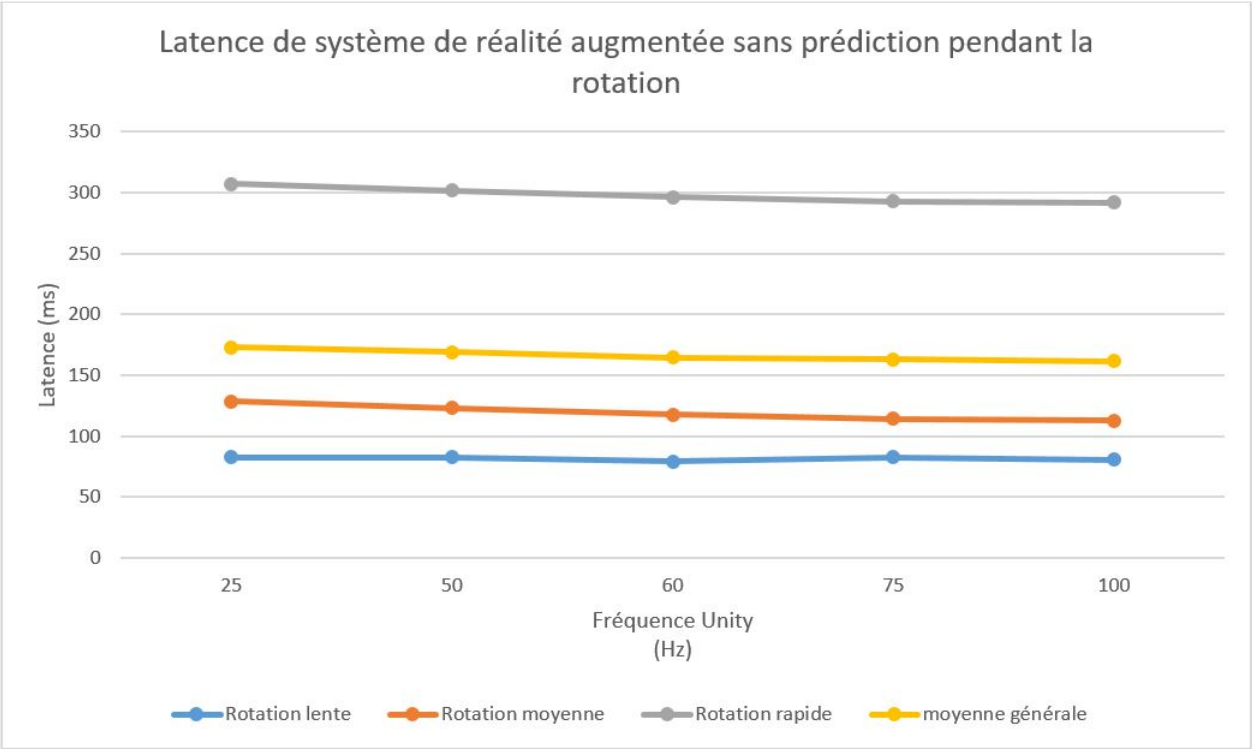


FIGURE 4.17 – Résultat de mesure de latence de notre système de réalité augmentée sans la mise en place d'une prédiction au démarrage et pendant la rotation du système

4.5 Conclusion

La latence d'un système de réalité augmentée est basée sur le temps de décalage entre les mouvements (ou les actions physiques) et les réponses électroniques sur le système. Cette latence est dûe à plusieurs facteurs comme la vitesse d'acquisition des capteurs, le temps de traitement et l'affichage. Dans ce chapitre nous avons présenté une étude sur les latences d'un système de réalité virtuelle de type OST. Une architecture de couplage faible aide à réduire le temps de la latence mais elle ne le supprime pas complètement. L'utilisation de la prédiction sur des données futures permet de réduire la latence pendant un mouvement contenu, mais elle augmente les latences de démarrage et d'arrêt du fait d'une prédiction de données qui se base sur un historique, sans tenir compte d'un changement brutal du démarrage de la séquence de mouvement. L'idéal est de démarrer la séquence de mouvement sans prédiction pour ensuite l'activer.

Cette expérience a permis de détecter les différentes latences d'une manière approximative. Afin de déterminer les facteurs vraiment influents et l'ordre de grandeur, il faut améliorer le système de mesure de la latence. Une étude conceptuelle d'un système de mesure de la latence est faite afin d'améliorer le système (pour des futurs mesures). En guise de perspectives, l'amélioration de la précision de la mesure de la latence passe par l'utilisation d'un système rotatif contrôlé par un moteur. D'autre part, l'utilisation d'un arbre téléporté permettrait d'enlever toutes les vibrations résiduelles et les harmoniques. De plus, l'utilisation d'encodeurs permettrait de récupérer avec précision le démarrage d'un mouvement. Ces encodeurs permettrait également de mesurer les positions réelles

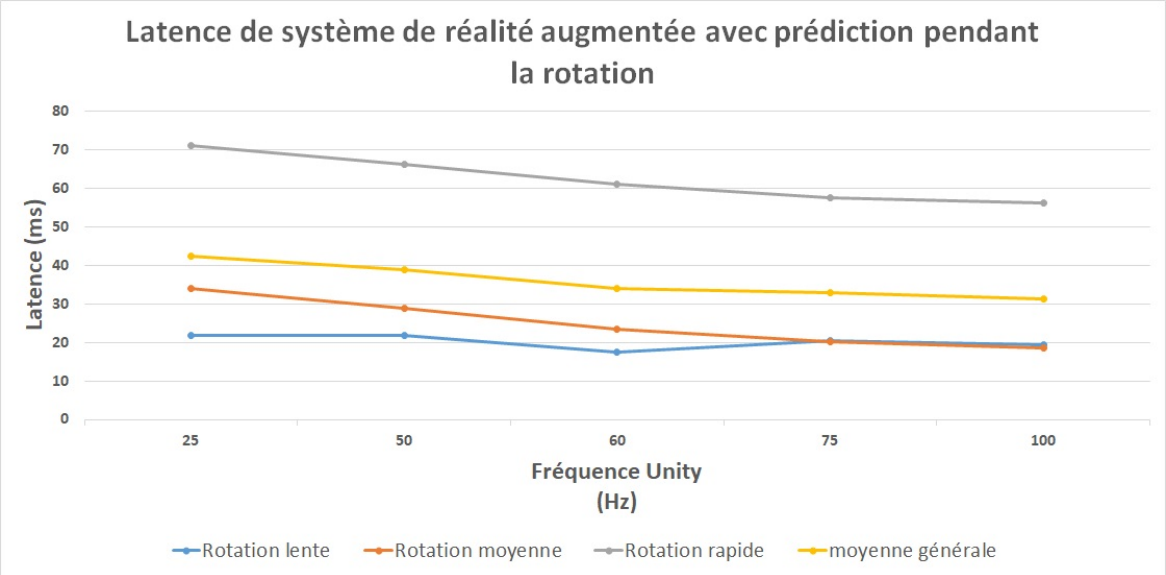


FIGURE 4.18 – Résultat de mesure de latence de notre système de réalité augmentée avec la mise en place d'une prédiction pendant la rotation du système

à tout instant. Ce qui permettrait également de comparer les résultats fournis par notre système de tracking. Ces améliorations devraient permettre de distinguer la différence entre la latence due à l'affichage ou une mauvaise estimation du système de tracking.

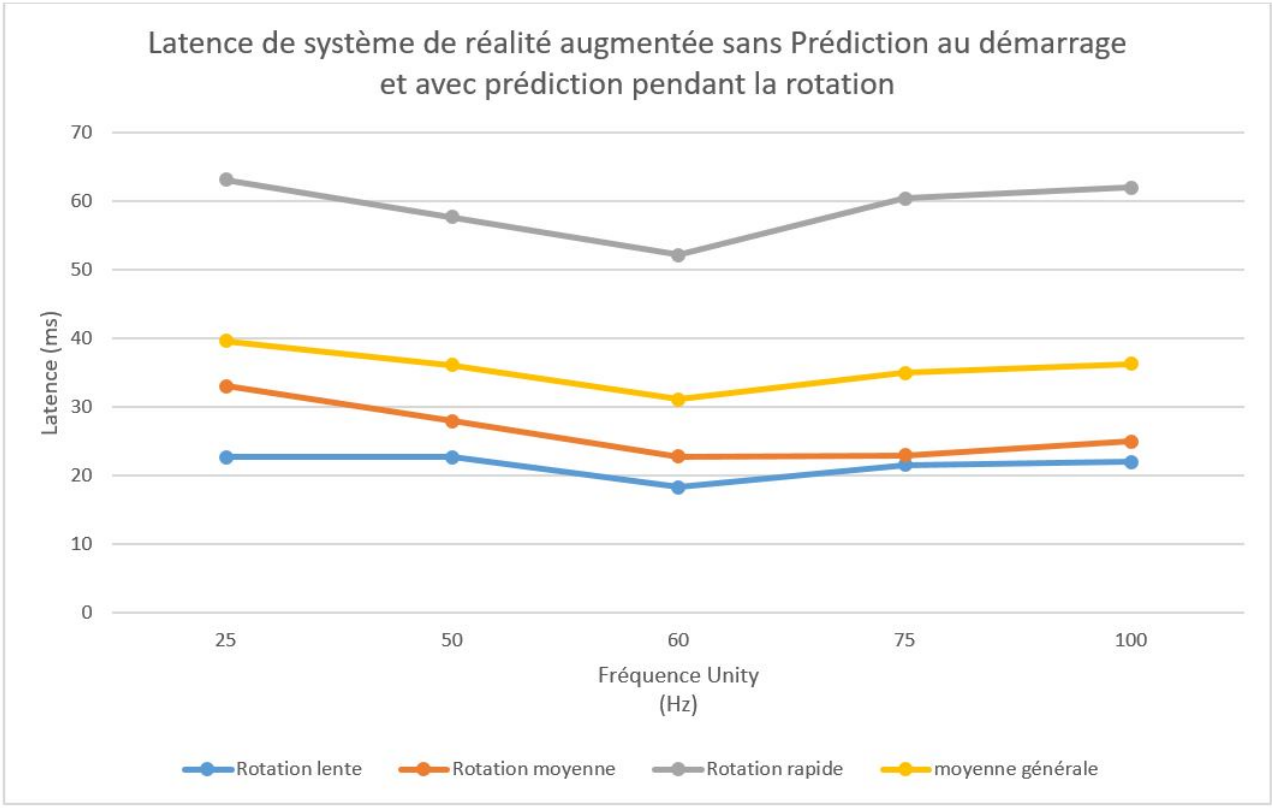


FIGURE 4.19 – Résultat de mesure de latence de notre système de réalité augmentée avec la mise en place d'une prédiction pendant la rotation du système et désactivation de la prédiction au démarrage

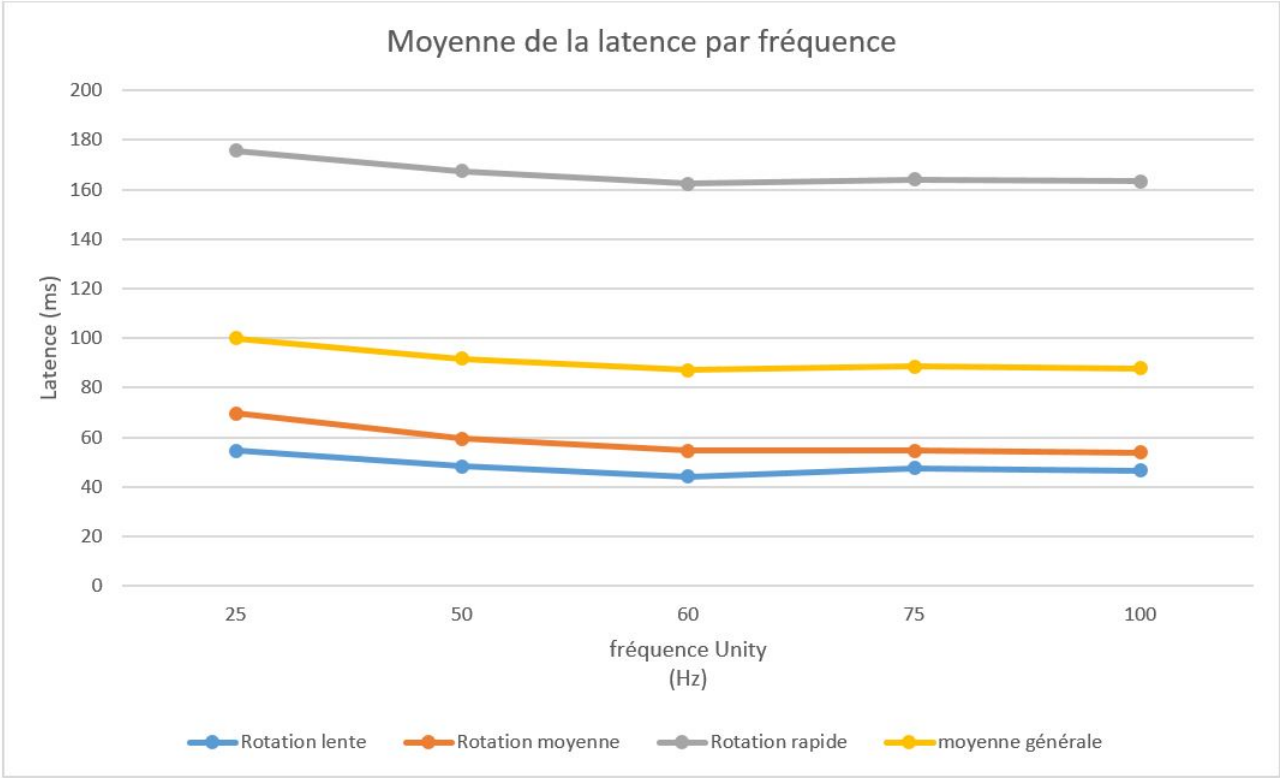


FIGURE 4.20 – Résultat de mesure de latence de notre système de réalité augmentée avec la mise en place d'une prédiction pendant la rotation du système et désactivation de la prédiction au démarrage

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Les travaux présentés dans ce document se sont inscrits dans un projet plus global (REV5D), dont le but était de développer un casque "see-trough" de réalité virtuelle. Ces travaux concernent la réalisation d'une brique logicielle, sur un système embarqué, dans laquelle la localisation 3D est améliorée en précision et en rapidité. Cette brique a aujourd'hui été incluse dans un casque prototype utilisable dans différents environnements : industriels, culturels... Plusieurs tests ont été faits avec un musée partenaire.

Le casque développé comporte une partie logicielle et une autre matérielle, elle même composée de capteurs, de cartes électroniques, et de projecteurs qui forment la partie affichage. Une carte électronique UdooX86 a été utilisée pour le développement des logiciels issus de nos travaux. Le choix de ce matériel fait suite à une analyse de l'existant sur le marché avec les partenaires du projet REVE5D : la carte 'Udoo X86 advanced' présente un bon rapport qualité/prix et de bonnes performances, considérant les autres cartes disponibles. Dans les travaux présentés dans ce document, les tests ont été faits sur un ordinateur portable assez puissant, avant d'être transférés sur la carte UdooX86. Les techniques présentées ont permis de réduire considérablement les différences de temps de calcul entre ces deux plateformes, au point de les rendre très difficiles à observer.

Le système de réalité augmentée choisi pour ce projet est de type Optical See Through. Comme dans toutes les applications de réalité augmentée, ce système consiste à faire coexister le monde réel et le monde imaginaire. Toutefois, La difficulté majeure liée aux casques OST réside dans le fait que l'opérateur observe le monde réel avec ses propres yeux, pendant que le monde imaginaire est projeté sur le casque suite à des interprétations de localisation faites en fonction des mouvements de l'opérateur dans le monde réel. La contrainte supplémentaire et lourde s'ajoute. Pour obtenir cette contrainte est la synchronisation totale entre l'environnement réel et les projections. Pour avoir une cohérence d'affichage, pour ne pas perdre l'intérêt de la réalité augmentée, et pour améliorer l'acceptabilité du système. Dans les systèmes où le monde réel est filmé par une caméra puis projeté devant l'opérateur, une marge de manœuvre est possible. L'opérateur voit ce qu'on projette ; il est possible de retarder un affichage ou de faire des sauts d'interprétations sans que l'opérateur ne s'en rende compte. L'essentiel dans notre application est de garder une fluidité et une certaine synergie par rapport aux déplacements de la tête.

D'un point de vue logiciel, le système d'exploitation Linux KUbuntu, utilisé sur la carte Udoox86,

a été légèrement modifié par un des partenaires du projet REVE5D. La communication des algorithmes avec les capteurs du casque a été améliorée et synchronisée au démarrage. Ainsi, au démarrage, les paramètres de calibration de la caméra pour le SLAM et pour la centrale inertielle sont lancés simultanément. La brique de localisation 3D a été développée en C++, puis inscrite dans une bibliothèque Linux partagée. Le logiciel de développement de contenus 3D utilisé est Unity. Pour que Unity puisse appeler la bibliothèque de localisation 3D pour estimer des poses, un point d'origine et un repère ont été définis. Les scènes et les modèles 3D qui peuplent la scène sont positionnés dans ce contexte commun.

Le développement technologique de la brique de localisation 3D repose sur deux composants et deux systèmes de navigation : le SLAM et la centrale inertielle. Dans la littérature, le couplage entre une centrale inertielle et un algorithme de vision existe depuis relativement longtemps. Un couplage consiste à faire la fusion de données issues de différentes sources à l'intérieur d'un même module pour fournir un résultat final. Cette fusion se base sur la synchronisation et traitement de données brutes à l'intérieur du filtre. Un couplage faible consiste en revanche à faire fonctionner toutes les parties d'une manière indépendante et à récupérer les poses de chaque capteur pour en faire la fusion et fournir les résultats. La différence entre les deux techniques de couplage se manifeste généralement dans l'implémentation et dans l'extensibilité. L'implémentation d'un couplage fort est complexe, et son extensibilité aussi. L'implémentation du couplage faible est plus simple, et l'ajout d'autres capteurs reste possible. Dans ce travail, nous avons utilisé le SLAM comme une brique non modifiable, ce qui nous a interdit d'utiliser le couplage fort. De plus, il était important de laisser la possibilité d'étendre le développement pour supporter d'autres capteurs.

Le cœur du système de fusion par couplage faible repose sur un filtre de KALMAN étendu, dans lequel des outils de gestion sont développés pour améliorer le rendement du système. Le système de fusion générale est appelé 'Tracker'. Ce 'Tracker' inclut un module de prédiction, un module d'estampillage pour dater les événements, un module de rétro-correction, et un module de traitement d'inconsistances. Le SLAM et la centrale inertielle sont les deux capteurs de ce module, qui estime in fine des poses. Une horloge logique a été créée pour l'estampillage de tous les événements depuis le lancement de l'application jusqu'à l'arrêt. L'acquisition d'images, l'acquisition de données inertielles, l'estimation d'une pose SLAM, et l'estimation d'une pose inertielle sont toutes datées par cette horloge. En utilisant l'estampillage et le système de datage, l'organisation de différentes données d'une manière chronologique est possible. Dans l'organisation chronologique des poses, la date affectée représente celle d'acquisition de données, et pas celle d'obtention de la pose. La pose est fournie selon une estimation d'un état du système à l'instant d'acquisition. En utilisant cette organisation, le module de rétro-correction peut opérer d'une manière performante en corrigeant la pose de référence estimée pour le même instant et le même état de système à cet instant bien déterminé. Cette correction évite les dérives de la centrale inertielle et rattrape aussi le temps de traitement important SLAM.

La prédiction dans le 'Tracker' consiste à estimer une pose future en fonction d'un historique de poses. Cette prédiction est utilisée par le module d'estimation de rendu qui est connecté au module 'Tracker'. Afin d'avoir une prédiction valide il faut connaitre le pas de prédiction qui représente le temps de traitement pour calculer une image de synthèse. Dans le cadre de ce travail, le pas de la prédiction a été fixé à 0,017s. Le rendu dans Unity a aussi été fixé a 0,017s. Cette valeur a été définie après les tests de performances présentés dans le chapitre sur l'évaluation de la latence.

Un algorithme de traitement de poses inconsistantes est enfin utilisé pour pouvoir gérer des poses aberrantes. Ces poses sont détectées par une analyse sur la fluidité d'évolution de poses au cours de temps. Une trajectoire est estimée par rapport a un historique, et un seuil d'écart à cette trajectoire est défini pour considérer une pose aberrante ou non. Ce traitement permet d'avoir une navigation fluide et continue au cours de temps. Tout cet enchainement est fait pour les données asynchrones issues de plusieurs capteurs. L'objectif est d'améliorer l'estimation de pose pour une

meilleure précision. La précision était un des objectifs initiaux de ce travail, mais la performance pour l'acceptabilité était aussi un objectif important.

La performance du système a été analysée en évaluant sa latence. Il a été observé que la prédiction n'est pas toujours la bonne solution pour améliorer les performances. En effet, cette technique est performante pendant les mouvements réguliers, mais peut dégrader le système lors d'un changement brutal d'état (démarrage après une pause, arrêt après un mouvement...). Il a également été montré que le phasage entre le module de calcul d'images (Rendu) et le module d'affichage est important pour avoir une latence minimale. L'architecture choisie est un facteur important pour la minimisation de la latence. Le choix d'un mode de fonction autonome asynchrone pour les différents capteurs, permet d'avoir une souplesse de traitement de données et d'estimation de poses. Dans le cas le plus favorable, la fréquence minimale d'estimation de poses peut monter au maximum de toutes les fréquences d'estimations de poses issues de différents capteurs. Ainsi, si le SLAM tourne avec une fréquence de 15 Hz, et la centrale inertielle avec une fréquence de 945 Hz, la fréquence d'estimation de pose dans le cas le plus favorable atteint 945 Hz.

Durant cette thèse plusieurs **perspectives** et améliorations potentielles ont été identifiées :

— Le bon fonctionnement des différentes briques technologiques est assurée par la bonne calibration du matériel. La caméra et la centrale inertielle sont calibrées pour une meilleure estimation. Au cours du développement, il est arrivé de re-calibrer la caméra à plusieurs à plusieurs reprises, ainsi que dans une moindre mesure la centrale inertielle. Une calibration ou re-calibration automatique de la caméra et de la centrale inertielle serait intéressante à développer.

— La prédiction de poses pour le module d'affichage est faite sur la base d'un pas fixe de prédiction. Le pas fixe est performant pour les scènes simples et non animées. Pour des scène dynamiques et animées, il faudra prévoir un système d'analyse et de détermination du pas de prédiction.

— Le calcul et l'estimation de positions au cours de nos travaux sont basés sur le SLAM uniquement. La prédiction de pose se base sur l'historique et sur des données de l'accéléromètre couvrant les intervalles qui ne dépassent pas 20 ms. Une meilleure exploitation des données de l'accéléromètre est possible pour pouvoir améliorer l'estimation de positions. La détection de pas et d'autres solutions peuvent être exploitées dans ce cas.

— La brique de localisation 3D a été développée et testée avec une architecture de couplage faible, et l'utilisation d'une horloge logicielle pour synchroniser entre les différents événements. Une analyse approfondie de la qualité d'estampillage d'événements a été réalisée. Une concentration de données sur des intervalles de temps irréguliers a pu être observée sur un oscilloscope. Plusieurs questions se posent par rapport aux données acquises, leurs natures, et le chemin parcouru avant leur passage par l'horloge. La concentration de données sur des intervalles montre un certain décalage dans la transmission de données entre le capteur et l'ordinateur.

— La brique de localisation 3D a été testée par notre système de mesure de latence basé sur une caméra rapide qui filme les projections sur l'écran du casque et l'environnement réel. Pour une analyse plus avancée, un autre système pourrait être utilisé. Ce système est basé sur des moteurs sans vibrations et des encodeurs à 8000 fentes avec une résolution de 0.001 degré. Cet ensemble est automatisé pour planifier des trajectoires et mesurer les positions réelles sur cette trajectoire. Des capteurs photo-diode pourraient être utilisés pour récupérer les mouvements de pixels et donc estimer la réaction sur l'écran du casque.

— Malgré les améliorations faites au niveau filtrage et au niveau prédiction, le système ne donne pas satisfaction lorsqu'il est utilisé dans une plus large échelle. Le SLAM se perd systématiquement et la navigation inertielle perd aussi ses caractéristiques. La précision du GPS avec le nouveau système de localisation européen GALILEO est de l'ordre de quelques centimètres pour les

applications publiques. Il pourrait alors être utile de combiner ce GPS avec le système de capteurs.

— Pendant notre travail, un module de réalité augmentée 'ARfoundation' a été mis à disposition de la communauté de développement Unity. Ce module n'est pas utilisable sur Linux, mais seulement sur Android et iOs. Il serait intéressante de le comparer avec les différents plugins et notre solution.

— La version du SLAM utilisée dans notre application était fournie par un partenaire. Il existe plusieurs SLAM gratuits et libres. Il serait peut-être intéressant de comparer les performances du système développé avec différents SLAMs. L'architecture conçu permet d'inclure des SLAM rapidement et facilement.

— Les téléphones portables présentent aujourd'hui jusqu'à 12 Go de mémoire RAM et 256 Go de stockage, avec 8 processeurs de 2.5 GHz. Ces performances peuvent supporter un SLAM et une centrale inertielle. Notre développement pourrait donc être porté sur un smartphone.

— La localisation 3D est une problématique pour toutes les applications 3D. Pendant ce travail, une collaboration a été initiée avec Grenoble INP pour développer une application haptique qui simule les données analysées par un microscope à force atomique. Ces travaux sont une première application du couplage de la réalité virtuelle avec un microscope à force atomique pour les nano-échelles, et pourraient être continués.

BIBLIOGRAPHIE

[1] Ababsa, F.-E. and Mallem, M. (2008). Particle filter-based camera localisation using square fiducials for augmented reality applications. International Journal of Signal and Imaging Systems Engineering, 1(3-4) :223–230.

[2] Ababsa, F.-E., Mallem, M., and Roussel, D. (2004). Comparison between particle filter approach and kalman filter-based technique for head tracking in augmented reality systems. In IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, volume 1, pages 1021–1026 Vol.1.

[3] Arcones, M. A. and Giné, E. (1992). On the bootstrap of m-estimators and other statistical functionals. Exploring the limits of bootstrap, pages 13–47.

[4] Asadi, E. and Bottasso, C. (2014). Tightly-coupled stereo vision-aided inertial navigation using feature-based motion sensors. Advanced Robotics, 28(11) :717–729.

[5] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. (2001). Recent advances in augmented reality. IEEE Computer Graphics and Applications, 21(6) :34–47.

[6] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf : Speeded up robust features. In European conference on computer vision, pages 404–417. Springer.

[7] Bergman, N. and Doucet, A. (2000). Markov chain monte carlo data association for target tracking. In 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100), volume 2, pages II705–II708 vol.2.

[8] Betge-Brezetz, S., Hebert, P., Chatila, R., and Devy, M. (1996). Uncertain map making in natural environments. In Proceedings of IEEE International Conference on Robotics and Automation, volume 2, pages 1048–1053. IEEE.

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

[10] Bradski, G. and Kaehler, A. (2000). Opencv. Dr. Dobb's journal of software tools, 3.

[11] Burusa, A. K. (2017). Visual-inertial odometry for autonomous ground vehicles. Master's thesis, KTH, School of Computer Science and Communication (CSC).

[12] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief : Binary robust independent elementary features. In European conference on computer vision, pages 778–792. Springer.

[13] Carmack, J. Latency mitigation strategies. https ://danluu.com/latency-mitigation/.

[14] Castellanos, J. A. and Tardos, J. D. (2012). Mobile robot localization and map building : A multisensor fusion approach. Springer Science & Business Media.

[15] Chai, L., Hoff, W. A., and Vincent, T. (2002). Three-dimensional motion and structure estimation using inertial sensors and computer vision for augmented reality. Presence : Teleoperators and Virtual Environments, 11(5) :474–492.

[16] Chai, L., Nguyen, K., Hoff, B., and Vincent, T. (2000). An adaptive estimator for registration in augmented reality.

[17] Choi, S.-W., Lee, S., Seo, M.-W., and Kang, S.-J. (2018). Time sequential motion-to-photon latency measurement system for virtual reality head-mounted displays. Electronics, 7(9) :171.

[18] Concha, A., Loianno, G., Kumar, V., and Civera, J. (2016). Visual-inertial direct slam. In 2016 IEEE international conference on robotics and automation (ICRA), pages 1331–1338. IEEE.

[19] Crowley, J. L. (1989). World modeling and position estimation for a mobile robot using ultrasonic ranging. In ICRA, volume 89, pages 674–680.

[20] Davison, A. J. and Murray, D. W. (1998). Mobile robot localisation using active vision. In European conference on computer vision, pages 809–825. Springer.

[21] Derpanis, K. G. (2004). The harris corner detector. York University, pages 1–2.

[22] Doucet, A., De Freitas, N., and Gordon, N. (2001a). An introduction to sequential monte carlo methods. In Sequential Monte Carlo methods in practice, pages 3–14. Springer.

[23] Doucet, A., Gordon, N., and Krishnamurthy, V. (2001b). Particle filters for state estimation of jump markov linear systems. Signal Processing, IEEE Transactions on, 49 :613 – 624.

[24] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping : part i. IEEE robotics & automation magazine, 13(2) :99–110.

[25] Eade, E. and Drummond, T. (2007). Monocular slam as a graph of coalesced observations. In 2007 IEEE 11th International Conference on Computer Vision, pages 1–8. IEEE.

[26] Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam : Large-scale direct monocular slam. In European conference on computer vision, pages 834–849. Springer.

[27] Engels, C., Stewénius, H., and Nistér, D. (2006). Bundle adjustment rules. Photogrammetric computer vision, 2(2006).

[28] Farrell, J. and Barth, M. (1999). The global positioning system and inertial navigation, volume 61. Mcgraw-hill New York, NY, USA :.

[29] Fleps, M., Mair, E., Ruepp, O., Suppa, M., and Burschka, D. (2011). Optimization based imu camera calibration. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3297–3304. IEEE.

[30] Fourati, H., Manamanni, N., and Handrich, Y. (2012). Fusion de données, estimation de la posture et navigation à l'estime : Application au Bio-logging.

[31] Furgale, P., Barfoot, T. D., and Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In 2012 IEEE International Conference on Robotics and Automation, pages 2088–2095. IEEE.

[32] Gallagher, A., Matsuoka, Y., and Wei-Tech Ang (2004). An efficient real-time human posture tracking algorithm using low-cost inertial and magnetic sensors. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), volume 3, pages 2967–2972 vol.3.

[33] Gao, J. Z., Prakash, L., and Jagatesan, R. (2007). Understanding 2d-barcode technology and applications in m-commerce-design and implementation of a 2d barcode processing solution. In 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), volume 2, pages 49–56. IEEE.

[34] Gordon, N., Ristic, B., and Arulampalam, S. (2004). Beyond the kalman filter : Particle filters for tracking applications. Artech House, London, 830(5) :1–4.

[35] Guazzaroni, G. and Pillai, A. S. (2019). Virtual and Augmented Reality in Education, Art, and Museums. IGI Global.

[36] Gunn, S. R. (1999). On the discrete representation of the laplacian of gaussian. Pattern Recognition, 32(8) :1463–1472.

[37] Hager, G. D. and Belhumeur, P. N. (1998). Efficient region tracking with parametric models of geometry and illumination. IEEE transactions on pattern analysis and machine intelligence, 20(10) :1025–1039.

[38] HaoChih, L. and Francois, D. Loosely coupled stereo inertial odometry on low-cost system.

[39] Haouchine, N., Roy, F., Untereiner, L., and Cotin, S. (2016). Using Contours as Boundary Conditions for Elastic Registration during Minimally Invasive Hepatic Surgery. In International Conference on Intelligent Robots and Systems, Daejeon, South Korea.

[40] Harris, C. G., Stephens, M., et al. (1988). A combined corner and edge detector. In Alvey vision conference, volume 15, pages 10–5244. Citeseer.

[41] Henderson, S. and Feiner, S. (2011). Exploring the benefits of augmented reality documentation for maintenance and repair. IEEE Transactions on Visualization and Computer Graphics, 17(10) :1355–1368.

[42] Herrera, C. D., Kim, K., Kannala, J., Pulli, K., and Heikkilä, J. (2014). Dt-slam : Deferred triangulation for robust slam. In 2014 2nd International Conference on 3D Vision, volume 1, pages 609–616. IEEE.

[43] Hesch, J. A., Kottas, D. G., Bowman, S. L., and Roumeliotis, S. I. (2013). Consistency analysis and improvement of vision-aided inertial navigation. IEEE Transactions on Robotics, 30(1) :158–176.

[44] Indelman, V., Williams, S., Kaess, M., and Dellaert, F. (2013). Information fusion in navigation systems via factor graph based incremental smoothing. Robotics and Autonomous Systems, 61(8) :721–738.

[45] Kaminer, I., Pascoal, A., and Kang, W. (1999). Integrated vision/inertial navigation system design using nonlinear filtering. In Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251), volume 3, pages 1910–1914. IEEE.

[46] Kato, H. and Billinghurst, M. (1999). Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99), pages 85–94. IEEE.

[47] Kechiche, M., Ivan, I.-A., Baert, P., Fortunier, R., and Toscano, R. (2019). A new loose-coupling method for vision-inertial systems based on retro-correction and inconsistency treatment. In International Conference on Augmented Reality, Virtual Reality and Computer Graphics, pages 111–125. Springer.

[48] Kim, S., Hong, J., Joung, S., Yamada, A., Matsumoto, N., Kim, S. I., Kim, Y. S., and Hashizume, M. (2011). Dual surgical navigation using augmented and virtual environment techniques. International Journal of Optomechatronics, 5(2) :155–169.

[49] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In 2007 6th IEEE and ACM international symposium on mixed and augmented reality, pages 225–234. IEEE.

[50] Klein, G. and Murray, D. (2009). Parallel tracking and mapping on a camera phone. In 2009 8th IEEE International Symposium on Mixed and Augmented Reality, pages 83–86. IEEE.

[51] Klein, G. S. and Drummond, T. (2002). Tightly integrated sensor fusion for robust visual tracking. In BMVC, pages 1–10. Citeseer.

[52] Kneip, L., Chli, M., and Siegwart, R. Y. (2011). Robust real-time visual odometry with a single camera and an imu. In Proceedings of the British Machine Vision Conference 2011. British Machine Vision Association.

[53] Koller, D., Daniilidis, K., and Nagel, H.-H. (1993). Model-based object tracking in monocular image sequences of road traffic scenes. International Journal of Computer 11263on, 10(3) :257–281.

[54] Kottas, D. G. and Roumeliotis, S. I. (2013). Efficient and consistent vision-aided inertial navigation using line observations. In 2013 IEEE International Conference on Robotics and Automation, pages 1540–1547. IEEE.

[55] Kottath, R., Narkhede, P., Kumar, V., Karar, V., and Poddar, S. (2017). Multiple model adaptive complementary filter for attitude estimation. Aerospace Science and Technology, 69 :574 – 581.

[56] Kume, H., Taketomi, T., Sato, T., and Yokoya, N. (2010). Extrinsic camera parameter estimation using video images and gps considering gps positioning accuracy. In 2010 20th International Conference on Pattern Recognition, pages 3923–3926. IEEE.

[57] Larnaout, D., Gay-Bellile, V., Bourgeois, S., and Dhome, M. (2013). Vehicle 6-dof localization based on slam constrained by gps and digital elevation model information. In 2013 IEEE International Conference on Image Processing, pages 2504–2508. IEEE.

[58] Lee, G. H., Fraundorfer, F., and Pollefeys, M. (2011). Mav visual slam with plane constraint. In 2011 IEEE International Conference on Robotics and Automation, pages 3139–3144. IEEE.

[59] Lepetit, V., Fua, P., et al. (2005). Monocular model-based 3d tracking of rigid objects : A survey. Foundations and Trends® in Computer Graphics and Vision, 1(1) :1–89.

[60] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 34(3) :314–334.

[61] Lhuillier, M. (2012). Incremental fusion of structure-from-motion and gps using constrained bundle adjustments. IEEE transactions on pattern analysis and machine intelligence, 34(12) :2489–2495.

[62] Li, M. and Mourikis, A. I. (2012). Improving the accuracy of ekf-based visual-inertial odometry. In 2012 IEEE International Conference on Robotics and Automation, pages 828–835. IEEE.

[63] Li, R., Di, K., Wang, J., Agarwal, S., Matthies, L., Howard, A., and Willson, R. (2005). Incremental bundle adjustment techniques using networked overhead and ground imagery for long-range autonomous mars rover localization. In Proc. of The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space–ISAIRAS, ESA SP-603. Citeseer.

[64] Lin, H.-T., Lin, C.-J., and Weng, R. C. (2007). A note on platt's probabilistic outputs for support vector machines. Machine learning, 68(3) :267–276.

[65] Lincoln, P., Blate, A., Singh, M., Whitted, T., State, A., Lastra, A., and Fuchs, H. (2016). From motion to photons in 80 microseconds : Towards minimal latency for virtual and augmented reality. IEEE transactions on visualization and computer graphics, 22(4) :1367–1376.

[66] Liu, H., Zhang, G., and Bao, H. (2016). Robust keyframe-based monocular slam for augmented reality. In 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pages 1–10. IEEE.

[67] Lovegrove, S., Patron-Perez, A., and Sibley, G. (2013). Spline fusion : A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In BMVC, volume 2, page 8.

[68] Lowe, G. (2004). Sift-the scale invariant feature transform. Int. J, 2 :91–110.

[69] Marchand, E., Bouthemy, P., and Chaumette, F. (2001). A 2d–3d model-based approach to real-time visual tracking. Image and Vision Computing, 19(13) :941–955.

[70] Michot, J., Bartoli, A., and Gaspard, F. (2010). Bi-objective bundle adjustment with application to multi-sensor slam. 3DPVT'10, 3025.

[71] Milgram, P., Takemura, H., Utsumi, A., and Kishino, F. (1995). Augmented reality : a class of displays on the reality-virtuality continuum. In Das, H., editor, Telemanipulator and Telepresence Technologies, volume 2351, pages 282 – 292. International Society for Optics and Photonics, SPIE.

[72] Mirzaei, F. M. and Roumeliotis, S. I. (2008). A kalman filter-based algorithm for imu-camera calibration : Observability analysis and performance evaluation. IEEE transactions on robotics, 24(5) :1143–1156.

[73] Montiel, J. M., Civera, J., and Davison, A. J. (2006). Unified inverse depth parametrization for monocular slam. Robotics : Science and Systems.

[74] Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. In Proceedings 2007 IEEE International Conference on Robotics and Automation, pages 3565–3572. IEEE.

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

[76] Nexter Training Regienov, Deltacad, D. C. D. A. V. S. N. S. E.-C. I. R. C. l. S. E. Simulation pour la formation et l'assistance.

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

[78] Pang, M. (2018). Multi-state constraint Kalman filter based visual inertial navigation system for monocular, stereo and RGB-D cameras. PhD thesis.

[79] Parnian, N. and Golnaraghi, F. (2010). Integration of a multi-camera vision system and strapdown inertial navigation system (sdins) with a modified kalman filter. Sensors, 10(6) :5378–5394.

[80] Piekarski, W. and Thomas, B. (2002). Arquake : the outdoor augmented reality gaming system. Communications of the ACM, 45(1) :36–38.

[81] Pressigout, M. and Marchand, E. (2006a). Hybrid tracking algorithms for planar and non-planar structures subject to illumination changes. In 2006 IEEE/ACM International Symposium on Mixed and Augmented Reality, pages 52–55. IEEE.

[82] Pressigout, M. and Marchand, E. (2006b). Real-time 3d model-based tracking : Combining edge and texture information. In Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pages 2726–2731. IEEE.

[83] Qin, T., Li, P., and Shen, S. (2018). Vins-mono : A robust and versatile monocular visual-inertial state estimator. IEEE Transactions on Robotics, 34(4) :1004–1020.

[84] Qiu, Z. and Qian, H. (2018). Adaptive genetic particle filter and its application to attitude estimation system. Digital Signal Processing, 81 :163–172.

[85] Ramadasan, D., Chateau, T., and Chevaldonné, M. (2015). Dcslam : A dynamically constrained real-time slam. In 2015 IEEE International Conference on Image Processing (ICIP), pages 1130–1134. IEEE.

[86] Randeniya, D. I., Sarkar, S., and Gunaratne, M. (2010). Vision–imu integration using a slow-frame-rate monocular vision system in an actual roadway setting. IEEE Transactions on Intelligent Transportation Systems, 11(2) :256–266.

[87] Reid, J. and Searcy, S. (1986). Detecting crop rows using the hough transform. American Society of Agricultural Engineers. Microfiche collection (USA).

[88] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb : An efficient alternative to sift or surf. In 2011 International conference on computer vision, pages 2564–2571. Ieee.

[89] Salas-Moreno, R. F. (2014). Dense Semantic SLAM. PhD thesis, Imperial College London.

[90] Smith, R. C. and Cheeseman, P. (1986). On the representation and estimation of spatial uncertainty. The international journal of Robotics Research, 5(4) :56–68.

[91] Steedly, D. and Essa, I. (2001). Propagation of innovative information in non-linear least-squares structure from motion. In Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, volume 2, pages 223–229. IEEE.

[92] Sun, K., Mohta, K., Pfrommer, B., Watterson, M., Liu, S., Mulgaonkar, Y., Taylor, C. J., and Kumar, V. (2018). Robust stereo visual inertial odometry for fast autonomous flight. IEEE Robotics and Automation Letters, 3(2) :965–972.

[93] Tan, W., Liu, H., Dong, Z., Zhang, G., and Bao, H. (2013). Robust monocular slam in dynamic environments. In 2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pages 209–218. IEEE.

[94] Thales Angénieux, Studio Bouquet et SFI, E. I. C. Site théâtral numérisé.

[95] Ullman, S. (1979). The interpretation of structure from motion. Proceedings of the Royal Society of London. Series B. Biological Sciences, 203(1153) :405–426.

[96] Usenko, V., Engel, J., Stückler, J., and Cremers, D. (2016). Direct visual-inertial odometry with stereo cameras. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1885–1892.

[97] Vasconcelos, J. F., Silvestre, C., Oliveira, P., Batista, P., and Cardeira, B. (2009). Discrete time-varying attitude complementary filter. In 2009 American Control Conference, pages 4056–4061.

[98] Vávra, P., Roman, J., Zonča, P., Ihnát, P., Němec, M., Kumar, J., Habib, N., and El-Gendi, A. (2017). Recent development of augmented reality in surgery : a review. Journal of healthcare engineering, 2017.

[99] Viswanathan, D. G. (2009). Features from accelerated segment test (fast). In Proceedings of the 10th workshop on Image Analysis for Multimedia Interactive Services, London, UK, pages 6–8.

[100] Vlahakis, V., Karigiannis, J., Tsotros, M., Gounaris, M., Almeida, L., Stricker, D., Gleue, T., Christou, I., and Ioannidis, N. (2001). Archeoguide : first results of an augmented reality, mobile computing system in cultural heritage sites. pages 131–140.

[101] Wagner, D. Motion to photon latency in mobile ar and vr.

[102] Wagner, D. and Schmalstieg, D. (2003). Artoolkit on the pocketpc platform. In 2003 IEEE International Augmented Reality Toolkit Workshop, pages 14–15. IEEE.

[103] Wahba, G. (1965). A least squares estimate of satellite attitude. SIAM review, 7(3) :409–409.

[104] Weiss, S. and Siegwart, R. (2011). Real-time metric state estimation for modular vision-inertial systems. In 2011 IEEE international conference on robotics and automation, pages 4531–4537. IEEE.

[105] Welch, G., Bishop, G., et al. (1995). An introduction to the kalman filter.

[106] Wu, A. D., Johnson, E. N., and Proctor, A. A. (2005). Vision-aided inertial navigation for flight control. Journal of Aerospace Computing, Information, and Communication, 2(9) :348–360.

[107] Wu, J., Zhou, Z., Fourati, H., Li, R., and Liu, M. (2019). Generalized Linear Quaternion Complementary Filter for Attitude Estimation from Multi-Sensor Observations : An Optimization Approach. IEEE Transactions on Automation Science and Engineering, pages 1–14.

[108] Wu, K., Zhang, T., Su, D., Huang, S., and Dissanayake, G. (2017). An invariant-ekf vins algorithm for improving consistency. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1578–1585. IEEE.

[109] Zhang, G., Qin, X., Hua, W., Wong, T.-T., Heng, P.-A., and Bao, H. (2007). Robust metric reconstruction from challenging video sequences. In 2007 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. IEEE.

[110] Zhang, Z. and Shan, Y. (2003). Incremental motion estimation through modified bundle adjustment. In Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429), volume 2, pages II–343. IEEE.

[111] Zheng, Z., Wang, H., and Teoh, E. K. (1999). Analysis of gray level corner detection. Pattern Recognition Letters, 20(2) :149–162.

École Nationale d'Ingénieurs de Saint-Etienne
Ecole Centrale de Lyon

N° d'ordre NNT : 2020LYSEE006

Marwene KECHICHE

Contribution to the development of a see-through augmented reality headset: data fusion for better 3D localization

Speciality : computer science and Engineering

Abstract :

Augmented reality applications require 3D localization in order to overlay or augment the real world with virtual projections.
In the context of this thesis a large-scale localization is exploited.
This localization is based on a family of computer vision algorithms called Simultaneous Localization and Mapping (SLAM). The computer vision algorithm is known for its accuracy and its minimal drifts. However, it suffers from a high processing time and an important need in terms of computing resources which makes the algorithm unusable in real time with embedded peripherals with low computing power. Inertial navigation (based on an inertial unit) allows to perform a 3D localization in real time. On the other hand, it suffers from an important drift in a narrow time interval. In this thesis, a weak coupling between the two navigation techniques is made in order to benefit from the performances of both algorithms. A feedback correction method is implemented to correct hardware synchronization problems between the two algorithms and the two pose estimators. This method is reinforced by an adaptive prediction that predicts future exposures for more accuracy and to minimize processing latency. Finally, performance tests were performed to evaluate the performance of all the techniques developed and to respect the latencies allowed for the targeted augmented reality applications. The evaluated and tested methods are used to operate a new augmented reality headset developed by partners within the REVE5D project. This augmented reality headset is intended for industrial and cultural use.

École Nationale d'Ingénieurs de Saint-Etienne
Ecole Centrale de Lyon

N° d'ordre NNT : 2020LYSEE006

Marwene KECHICHE

Contribution au développement d'un casque « see-through » de réalité augmentée : fusion de données pour une meilleure localisation 3D

Spécialité : informatique et Ingénierie

Résumé :

Les applications de réalité augmentée nécessitent une localisation 3D afin de superposer ou augmenter le monde réel avec des projections virtuelles.
Dans le cadre de cette thèse une localisation grande échelle est exploitée.
Cette localisation est basée sur une famille d'algorithmes de vision par ordinateur nommée « Localisation et cartographie simultanées (SLAM)». L'algorithme de vision par ordinateur est connu pour sa précision et ses dérives minimes. Cependant il souffre d'un temps de traitement important et une nécessité importante en terme de ressources informatiques ce qui rend l'algorithme inexploitable en temps réel avec des périphériques embarqués a faible puissance de calcul. La navigation inertielle (basé sur une centrale inertielle) permet d'effectuer une localisation 3D en temps réel. En revanche elle souffre d'une dérive importante dans un intervalle de temps étroit. Dans cette thèse, un couplage faible entre les deux techniques de navigation est fait afin de profiter des performances des deux algorithmes. Une méthode de rétro-correction est implémentée afin de corriger les problèmes de synchronisation matérielle entre les deux algorithmes et les deux estimateurs de poses. Cette méthode est renforcée par une prédiction adaptative qui prédit les poses futures pour plus de précisions et pour minimiser la latence de traitement. Finalement, des tests sur les performances ont été réalisés pour évaluer les performances de l'ensemble des techniques développées et respecter les latences autorisées pour les applications de réalité augmentée visées. Les méthodes évaluées et testées sont utilisées pour faire fonctionner un nouveau casque de réalité augmentée développé par des partenaires au sein du projet REVE5D. Ce casque de réalité augmentée est destiné à une utilisation industrielle et culturelle.