Skyline : un marqueur pour la réalité augmentée sur mobile en contexte urbain

Mehdi Ayadi

N° d'ordre NNT : 2019LYSE2051

THESE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de

L'UNIVERSITÉ LUMIÈRE LYON 2

École Doctorale : ED 512 Informatique et Mathématiques

Discipline : Informatique

Soutenue publiquement le 2 juillet 2019, par :

Medhi AYADI

Skyline : un marqueur pour la réalité augmentée mobile en contexte urbain.

Devant le jury composé de :

Luce MORIN, Professeure des universités, Université Rennes 2, Présidente

Anne RUAS, Directrice de Recherches, IFSTTAR, Rapporteure

Adel ALIMI, Professeur d'université, Université de Sfax, Examinateur

Patrick SAYD, Dirdecteur de Recherches, Commissariat à l'Energie Atomique (CEA), Examinateur

Serge MIGUET, Professeur des universités, Université Lumière Lyon 2, Co-Directeur de thèse

Chokri BEN AMAR, Professeur d'université, Université de Sfax, Co-Directeur de thèse

Mihaela SCUTURICI, Maitresse de conférences, Université Lumière Lyon 2, Co-Directrice de thèse

Contrat de diffusion

THÈSE EN COTUTELLE PRÉSENTÉE
POUR OBTENIR LE GRADE DE

DOCTEUR DE
L'UNIVERSITE LUMIERE LYON 2
spécialité « **Informatique** »

ET DE L'UNIVERSITE DE SFAX
spécialité « **Ingénierie des Systèmes Informatiques** »

opérée au sein des laboratoires **LIRIS** et **ReGIM**
par

Mehdi Ayadi

SKYLINE : UN MARQUEUR POUR LA RÉALITÉ AUGMENTÉE MOBILE EN CONTEXTE URBAIN

Thèse soutenue le XX/XX/2018 devant le jury composé de :

M.	XXXXXXX	XXXXXXXXXX	(Président)
M.	XXXXXXX	XXXXXXXX	(Rapporteur)
Mme	XXXXXX	XXXXXXXX	(Rapporteur)
M.	XXXXX	XXXXXX	(Examinateur)
Mme	MIHAELA SCUTURICI	MCF, Université Lyon 2	(Co-directrice)
M.	SERGE MIGUET	PR, Université Lyon 2	(Directeur)
M.	CHOKRI BEN AMAR	PR, Université de Sfax (ENIS)	(Directeur)

À mes parents ...
Je ne vous aimerai jamais assez ... !

Titre Skyline : un marqueur pour la réalité augmentée mobile en contexte urbain

Résumé L'objectif principal de cette thèse consiste à proposer une approche qui permet à un utilisateur se déplaçant en milieu urbain, de visualiser en Réalité Augmentée, à l'aide d'un smartphone, l'incidence d'un projet de construction sur le paysage urbain, une fois le bâtiment réalisé. En particulier, avec le retour actuel des projets de constructions de tours dans les grandes villes (Londres, Paris et Lyon notamment), il devient un enjeu très important pour les usagers des quartiers de pouvoir se rendre compte visuellement de l'impact d'un projet de construction sur la ligne d'horizon (Skyline) et donner un avis le plus objectif possible sur un projet de construction. Du point de vue des géographes et des usagers des quartiers, ce dispositif innovant permet de prendre le contrepied des cabinets d'architectes, qui ne proposent que des rendus très esthétiques de leurs projets de construction, avec nombre limité de vues, au détriment de visualisations plus objectives. Il ne s'agit pas nécessairement de proposer un rendu photo-réaliste, mais plutôt une esquisse la plus exacte possible de la géométrie de la scène, tout au long de la déambulation de l'usager. Des visualisations à différentes positions et orientations correspondent mieux aux usages les plus courants de la vie dans les quartiers. Nous proposons une approche dans laquelle les différentes données issues de la multitude d'instruments embarquées dans le dispositif sont fusionnées, afin d'estimer la pose de l'appareil : le compas magnétique permet d'estimer la direction d'observation ; le gyroscope et l'accéléromètre permettent d'évaluer grossièrement les paramètres de mouvement (trois degrés de liberté en translation, et trois degrés de liberté en rotation). Cette première pose estimée, associée au modèle 3D de la ville, permet, à l'aide des supports de rendu (2D et 3D) des dispositifs, de générer une image de synthèse de ce que l'utilisateur visualise théoriquement à cette position. Néanmoins, l'utilisation unique de ces instruments pour insérer un objet de synthèse dans le flux vidéo donne à l'utilisateur une impression très peu réaliste de la scène qu'il visionne : les objets de synthèse semblent « flotter » au gré des mouvements en raison des imprécisions des instruments. Pour pallier ce défaut, le skyline est extrait automatiquement des images réelles (acquises par la caméra) et virtuelles (générées). Une étape d'appariement des deux skylines permet de recaler le skyline virtuel sur le réel et ainsi permettre une incrustation en temps réel des projets de construction au sein du flux vidéo, et une amélioration selon deux critères : précision et stabilité avec une impression d'immersion bien meilleure. Plusieurs mesures de similarité sont proposées avec une approche d'optimisation basée sur une descente de gradient.

Mots-clés Skyline, réalité augmentée mobile, modèle 3D, mesures de similarité

Title A skyline based approach for mobile augmented reality

Abstract The main objective of this thesis is to propose an approach that allows a user, moving in an urban environment, to visualize in augmented reality, using a smartphone, the impact of a construction project on the urban landscape, once the building is completed. In particular, with the current attractivity for tower construction projects in major cities (London, Paris and Lyon in particular), it is becoming a very important issue for neighborhood users to be able to visualize the impact of a construction project on the skyline and to give an objective opinion on it. For geographers and neighborhood users, this innovative solution makes it possible to take the opposite view from architectural firms, which only offer very qualitative renderings of their construction projects, with a limited number of views, to the detriment of more objective visualizations. We do not necessarily propose a photo-realistic rendering, but rather a more exact geometric consistency of the scene, throughout the user's movements. Visualizations at different positions and orientations correspond better to common use cases in neighborhoods. We propose an approach in which data from smartphone's embedded instruments are merged to estimate a first user's pose : the magnetic compass estimates the viewing direction ; the GPS, gyroscope and accelerometer roughly evaluate the parameters of motion (three degrees of freedom in translation, and three degrees of freedom in rotation). This first estimated pose, associated with the 3D model of the city, allows, using the rendering supports (2D and 3D) of the devices, to generate a synthetic image of what the user theoretically visualizes at this position. Nevertheless, the unique use of these instruments to insert a synthetic object into the video stream gives the user a very unrealistic impression of the viewed scene : due to instrument's data inaccuracies, synthetic objects appear to "hover" and "float" with the user's movements. For this, the skyline is automatically extracted from real (acquired by the camera) and virtual (generated) images. A matching step between the two skylines allows to realign the virtual skyline onto the real one, allowing thus a real time insertion of the 3D object in the real-time video stream, and an improvement according to two criteria : precision and stability, giving with a much better immersion impression. In this matching step, several similarity metrics are proposed that are used in an optimization approach based on a gradient descent.

Keywords Skyline, mobile augmented reality, 3D city model, comparison metrics.

TABLE DES MATIÈRES

LISTE DES FIGURES

Liste des Algorithmes

Liste des tableaux

Remerciements

Nous y voilà maintenant, après quatre années de dur labeur, devant le fait accompli.

Je tiens tout d'abord à souligner toute la chance que j'ai eu durant toute la durée de cette thèse en co-tutelle avec mes trois directeurs de thèse pour m'avoir accueilli au sein de leurs laboratoires et m'avoir donné une chance de travailler sur un domaine qui me tient à cœur. Lors de ces quatre années difficiles mais épanouissant-es, il ont tous les trois réussi à me conseiller scientifiquement sur mon sujet de thèse en me dirigeant sans cesse et en me donnant de nombreux conseils avisés. Ils m'en ont tellement donné que je ne les remercierai jamais assez : tantôt à me former scientifiquement ; tantôt à m'aiguiller pédagogiquement dans mes activités d'enseignements ; tantôt à me transmettre leurs plaisirs à faire de la recherche ; tantôt à me lire et relire, à me corriger, à m'expliquer et surtout à me ré-expliquer ..! Et plus que tout, ils m'ont prouvé qu'il y avait, dans ce monde, de très grandes personnes. Au-delà de cet aspect professionnel, je souhaite saluer l'esprit de convivialité et de bienveillance de ces superbes personnes, qui, heureusement pour moi, ont déteint sur moi leur qualités humaines.
Mihaela, Chokri, Serge : encore une fois, je ne vous remercierai jamais assez !

Je tiens aussi à remercier les membres du Jury : Patrick SAYD et Luce MORIN qui ont accepté cette lourde tâche de rapporter mon manuscrit et d'y apporter leur connaissances et surtout leur expertise, ANNE RUAS pour avoir accepté d'être mon examinatrice, ainsi que Pr. M.Adel ALIMI, le directeur de mon cher laboratoire ReGIM et président de mon jury.

Un grand merci à toute les équipes du LIRIS et en particulier l'équipe IMAGINE pour m'avoir épaulé pendant les moments difficiles, et en particulier Pr. Laure Tougne, pour ses conseils, sa grande connaissance en informatique et pour m'avoir rattrapé au vol quand j'en avais besoin. Je tiens particulièrement à remercier Catherine Villesesche pour sa réactivité et sa rapidité remarquable à tous les besoins que j'ai pu (et tous le(s) laboratoire(s)) formulé à tout instant. Ma gratitude s'adresse aussi à Dominique Maniez pour ses conseils avisés et ses recommandations pertinentes, pédagogique, humaines et scientifique et surtout pour la relecture du mémoire.

Mes acolytes vont de même y passer : je remercie particulièrement nos deux Rémis (0) et |1|, Adel, Sarah, Evan, Xavier, Aurélie, Mathieu, Carlos, et tous ceux avec qui j'ai eu des interactions très enrichissantes. De même, je n'oublierais pas l'ensemble

des membres du laboratoire LIRIS, pour m'avoir accueilli, pour leur disponibilité et pour m'avoir fourni les moyens nécessaires pour mener à bien mon travail.

Finalement, je voudrais maintenant présenter des remerciements plus personnels. En effet, un engagement dans une thèse a forcément des répercussions sur la vie personnelle. Sans le soutien sans faille de ma famille, ce projet de thèse n'aurait pu aboutir. Je tiens donc à remercier l'Homme que je voudrais être un jour, mon modèle et que nul ne pourrait à jamais égaler : mon père, mon cher. Et à celle que j'aimerai plus que tout au monde, et bien au delà. À la femme de ma vie, à tout jamais, ma mère : celle qui m'a donné vie et fait de moi l'homme que je suis aujourd'hui. À tout leurs sacrifices, que je ne comprenais pas jusque là, mais que je réalise de plus en plus, de jour en jour. Je remercie mon frère de m'avoir toujours appuyé sans , corrigé et encouragé dans tous mes projets. Vous trois, je ne vous aimerai jamais assez.

1 Introduction Générale

Contexte de la thèse

Ces travaux de thèse se situent au croisement de plusieurs domaines que sont la Réalité Augmentée, la Réalité Virtuelle, la modélisation 3D, le génie logiciel, le traitement d'image, les sciences humaines et sociales avec une petite pincées de mathématiques. Elle s'insère dans le cadre de collaborations interdisciplinaires entre informaticiens, géographes et urbanistes. Ces collaborations ont déjà permis l'élaboration d'un projet financé par *l'Agence Nationale de la Recherche* (*ANR*) intitulé « *Skyline* » et d'un projet du Labortoire d'Excellence « *Intelligence des Mondes Urbains* » (*IMU*) intitulé *«Mesure Géomètrique du Skyline* ». Ces interactions ont permis de faire émerger le besoin en outils et techniques spécifiques, susceptibles d'avoir des retombées pour l'ensemble des communautés concernées. Cette thèse est donc une inspiration de mes encadrants au travers de ces projets et de leurs expériences.

Cette thèse est une co-tutelle entre *l'Université Lumière Lyon 2* et *l'Université de Sfax*. Au *LIRIS*, à Lyon elle s'insère dans les thématiques de l'équipe *IMAGINE*, spécialisée dans l'analyse des images et des vidéos. Au *ReGIM*, à l'École Nationale d'ingénieurs de Sfax (ENIS), elle contribue aux activités de l'équipe *Traitement des signaux et des Images*.

1.1 Constat

Les technologies numériques se sont largement développées depuis le début du millénaire. Tout d'abord, elles étaient restreintes et considérées comme un outil de travail au bureau. Puis, elles se sont démocratisées et faufilées chez les particuliers qui se les est appropriées. De plus, avec la démocratisation d'Internet, les distances se sont vues raccourcies et les échanges beaucoup plus faciles d'un bout du monde à l'autre, ce qui a permis d'*imposer* de nouveaux modes de collaborations qui sont devenus indispensables aujourd'hui. Ensuite, sont apparus les smartphones qui y ont contribué doublement, en se rendant comme outil indispensable, non pas aux professionnels et particuliers actifs uniquement, mais s'étend même jusqu'aux enfants et ce, dès le plus jeune âge. Cependant, ces technologies se sont vue restreinte au domaine de l'information et de la communication, sans pouvoir agir directement sur le monde réel. Ce n'est qu'avec l'avènement de l'**IOT** (Internet Of Things), en Français

(Internet des Objets) que ces différentes technologies ont pu agir sur le monde réel. Enfin, avec l'arrivée des modèles 3D et le développement des techniques de *CAO*, une personne est désormais capable de voir, virtuellement, à travers plusieurs types de dispositifs, un endroit dans le monde sans s'y déplacer. L'exemple le plus connu est *GoogleEarth*, qui permet de faire le tour du monde de chez soi, bien assis sur son fauteuil. Néanmoins, cet exemple relève plus du domaine de la *Réalité Virtuelle* (RV), et non pas de la *Réalité Augmentée* (RA). Nous détaillerons ces deux termes dans les chapitres qui suivent.

La différentes technologies de RV permettent de visualiser le monde réel grâce à des images de synthèse, générées à partir de modèles 3D qui leur ressemble comme deux gouttes d'eau (selon la précision). Celles de RA, de pouvoir agir sur le monde réel et de le changer (ou augmenter, ou enrichir) en y rajoutant (visuellement) des objets de synthèse en 3D, afin d'avoir un premier ressenti (idée) sur *"comment sera le futur monde réel si on y rajoutait cet objet là en particulier"*. En effet, de par son principe à rehausser notre perception du monde réel en y rajoutant des objets issus du numérique, la RA permet d'envisager une multitude de possibilités dans plusieurs domaines d'application, et ce au travers d'outils et technologies en constante évolution. Ces outils peuvent très bien constituer un outil d'assistance aux experts (médecins, maintenance, architectes, etc.), mais aussi un outil ludique, de simulation ou de divertissement pour le grand public.

Cependant les différentes applications de RA des deux dernières décennies ont été restreintes aux environnements intérieurs contrôlés, à petite ou moyenne échelle. Ceci est principalement dû aux performances (calcul et capactié mémoire) des technologies utilisées, limitant la possibilité de déploiement à grande échelle en extérieur. Néanmoins, nous trouvons dans la littérature, quelques travaux de RA en extérieur, où les systèmes proposés sont très contraignant en termes de poids, volume, etc. À titre d'exemple, dans les années 2000, le projet MARS (de Höllerer et al. (1999)) a permis de démontrer la faisabilité d'un système de RA en extérieur grâce à un système physique assez lourd, porté sur le dos, des lunettes imposantes etc. (voir figure 2.16(c) du chapitre 2).

Cependant, les processeurs et les puissances de calcul n'ont cessé d'évoluer depuis permettant de faire émerger ainsi des terminaux mobiles (tablettes-PC, PDAs, smartphone, etc.) munis de capacité mémoire de puissances de calcul assez importantes. De plus, les différents capteurs qui étaient séparés en plusieurs éléments physiques (centrale inertielle, compas magnétique, GPS, accéléromètre, etc.), se sont vus miniaturisés et intégrés dans les téléphones intelligents (smartphone), les rendant, d'une part, beaucoup moins encombrant et facile d'utilisation, et d'autre part, de plus en plus précis.

Pratiquement, dans le contexte où cette thèse est proposée, toutes les marques de smartphone sont munies, d'une part, de plusieurs caméras (au moins deux) avec des résolutions très satisfaisantes permettant une acquisition précise et de haute qualité de l'environnement réel, et, d'autre part, d'une batterie de capteurs permettant l'estimation de la position (gps) et de l'attitude (comportement en translation et rotation) de l'utilisateur dans le repère du monde (accéléromètre, gyroscope, compas magnétique, etc.). Ces téléphones intelligents ont donc permis d'envisager l'exportation des applications de RA vers le monde extérieur, avec néanmoins, plusieurs limitations en termes de précision, et qui seront détaillées tout au long de cette thèse. En effet, ces différents instruments ne permettent pas d'avoir une vérité absolue de la position et du comportement de l'utilisateur, mais plutôt des **estimations** plus ou moins précises, selon les conditions dans lesquelles ce dernier se trouve : qualité des signaux gps (selon le nombre de satellite), bruits magnétique (tramway qui passe, champ magnétique, etc.), changement de luminosité, bruits dans l'image (nuages, ...), calibration des capteurs, etc.

Le contexte (ou domaine d'application) dans lequel nos travaux se situent est le contexte urbain. En effet, nous proposons, à la fin de cette thèse, une application de RA mobile en contexte urbain.

1.2 Constat

Le constat initial qui a inspiré les géographes, urbanistes et plus particulièrement mes encadrants, est : avec le retour actuel des projets de constructions de tours dans les grandes villes (Londres, Paris et Lyon notamment), il devient un enjeu très important pour les usagers des quartiers de pouvoir se rendre compte visuellement de l'impact d'un projet de construction sur la ligne d'horizon et donner un avis le plus objectif possible sur un projet de construction. Pour cela, la RA se propose comme une alternative ***grand public*** aux très couteuses études des cabinets d'architectes, qui ne proposent que des rendus très esthétiques de leurs projets de construction, avec nombre limité de vues, au détriment de visualisations plus objectives.

1.3 Problématiques scientifique

S'inscrivant dans ce registre de RA sur mobile en contexte urbain, cette thèse a pour objectif de mettre au point un système combinant des problématiques de localisation à caractère hybride (selon deux modalités : capteurs et image) en milieu extérieur, de visualisation et d'interaction temps réel, le tout sur des terminaux mo-

biles. La localisation nous permet d'estimer la position, la direction d'observation et l'orientation de l'utilisateur (au travers du dispositif) dans le repère du monde. Ces informations offrent la possibilité de rajouter les données virtuelles, préalablement choisies (un bâtiment en particulier), au monde réel. Le résultat est une sorte d'un "monde réel enrichi" (ou "monde réel augmenté"), visualisé par l'utilisateur afin d'en donner un avis objectif. La problématique principale tourne donc autour de la *"localisation"*, ou encore appelée *"pose de l'utilisateur"*. Différentes problématiques complémentaires entourent la mise en œuvre de notre approche. Ces problématiques constituent des verrous scientifiques et technologiques importants à lever, qui seront traitées séparément tout au long des chapitres 3, 4 et en début du chapitre 5, puis une fusion globale de toutes ces contributions dans la dernière partie du chapitre 5.

Néanmoins, cette problématique de localisation en milieu extérieur a été largement étudiée en littérature. Plusieurs approches existent : les approches basées vision, le approches basées capteurs et les approches hybrides combinant capteurs et vision. Ce travail s'insère donc dans cette troisième famille hybride, où nous combinons tous les instruments sensoriels du smartphone (approche basée capteurs) avec des points d'amers (d'ancrage) extraits des images issues du flux vidéo de la caméra (approche basée vision).

1.4 CONTRIBUTIONS

Dans l'approche proposée dans cette thèse, les différentes données issues de la multitude d'instruments embarqués dans le dispositif sont fusionnées, afin d'en déduire une première approximation de la pose : le compas magnétique permet d'estimer la direction d'observation ; le gyroscope et l'accéléromètre permettent d'évaluer grossièrement les paramètres de mouvement (trois degrés de liberté en translation, et trois degrés de liberté en rotation). Cette première estimation permet, à l'aide modèle 3D de la ville et des supports de rendu (2D et 3D) des dispositifs, de générer une image de synthèse de ce que l'utilisateur *voit théoriquement* à cette position. En d'autres termes, nous plaçons une caméra virtuelle (avec les mêmes caractéristiques que la réelle) dans le modèle 3D de la ville à la même position avec la même orientation que l'utilisateur réel. Néanmoins, en raison des imprécisions de ces instruments, leur utilisation unique pour insérer un objet de synthèse dans le flux vidéo donne à l'utilisateur une impression très peu réaliste de la scène qu'il visionne : les objets de synthèse semblent « flotter » au gré des mouvements.

Pour pallier ce(s) défaut(s), il nous faut, d'une part, *stabiliser* l'augmentation de la vidéo et éviter ces effets de flottement, et d'autre part, incruster l'objet de synthèse avec une meilleure *précision* et donc son emplacement exact dans la scène. Les termes

sont dits : Stabilité et Précision. Cela seront nos critères d'évaluation tout au long de cette thèse.

Pour ce faire, le skyline est extrait automatiquement des images réelles (acquise par la caméra) et virtuelles (générées à partir du modèle de la ville). Une étape d'appariement des deux skylines permet de recaler le skyline virtuel sur le réel et ainsi permettre une incrustation des projets de construction au sein du flux vidéo, et une amélioration selon deux critères : précision et stabilité avec une impression d'immersion bien meilleure. Au travers de cette étape d'appariement, plusieurs sous-problématiques sont traitées. Plusieurs mesures de similarité sont proposées avec une approche d'optimisation basée sur une descente de gradient.

Les principales contributions de la thèse s'articulent autour des points suivants :

— La proposition d'une méthode paramétrique d'extraction du skyline, permettant de fournir le skyline le mieux adapté à la grande variabilité des scènes, aux conditions d'observation, et aux préférences de l'utilisateur (plusieurs skylines peuvent exister, à différents niveaux de profondeur) ;
— La proposition d'une approche de fusion de données issues des différents capteurs embarqués dans le dispositif mobile (smartphone) pour l'estimation de la pose tout au long de sa déambulation dans la ville ;
— Proposition d'une méthode de matching entre skyline réel et virtuel, en se basant sur plusieurs mesures de similarité ;
— Proposition d'une application mobile pour la réalité augmentée basée sur une approche hybride complétant les données capteurs par des données vision.

1.5 Organisation du mémoire

Nos contributions portent sur, tout d'abord, une méthode paramétrique d'extraction du skyline, ensuite, la proposition d'une méthode de fusion de données multi-capteurs pour l'estimation de la pose, et enfin, sur une approche d'appariement d'images basée sur le skyline.

Le *chapitre 2* présente le paradigme de RA et en expose les notions nécessaires à la compréhension des chapitres suivants. La première partie commence par définir la RA tel que présentée dans la communauté. Par la suite, nous exposons les différents principes de fonctionnement d'un système de RA, en détaillant les dispositifs indispensables à la réalisation de notre système. Nous nous préoccupons aussi des différents algorithmes possibles et existant, et qui sont clairement divisés en deux grandes familles : les algorithmes en environnements connus et en environnements inconnus. Enfin, nous y détaillons les différents dispositifs et applications possibles. Cette étude

nous permet de définir les problématiques qui entourent la mise en œuvre de notre système de RA en extérieur et les verrous scientifiques et technologiques à lever afin d'y parvenir.

Dans le *chapitre 3*, nous présentons tout d'abord quelques systèmes de localisation existant dans la communauté de RA et qui sont basés sur la combinaison de plusieurs types de capteurs. Puis, nous présentons les différents capteurs de mouvement utilisés dans le cadre de cette thèse. Ensuite, Nous esquissons notre système multi-capteurs ainsi que les différentes problématiques et enjeux relevés lors de sa mise en œuvre. Enfin, nous détaillons le protocole d'évaluation utilisé et les résultats obtenus pour cette première approche basée capteurs.

Le *chapitre 4* propose, d'abord, une revue de la littérature des approches d'extraction du skyline, puis, détaille notre algorithme paramétrique d'extraction des skylines. Cet algorithme paramétrique nous permet de fournir le skyline le mieux adapté à la grande variabilité des scènes : un skyline de premier plan, un skyline intermédiaire ou un skyline d'arrière plan. De plus, nous avons la possibilité d'adapter le skyline extrait selon les conditions d'observation et les préférences de l'utilisateur. Enfin, le protocole d'évaluation est détaillé. Les résultats obtenus sont présentés et synthétisés suivis d'une discussion. L'étude présentée dans ce chapitre a fait l'objet d'une publication avec une présentation orale dans une conférence internationale (Core :B) avec comité de lecture (Ayadi et al. (2016)).

Le *chapitre 5* présente, dans sa première partie, une revue de la littérature des méthodes de recalage d'images utilisant le skyline. Puis, en deuxième partie, nous détaillons notre algorithme de recalage qui sera présenté en deux temps : d'abord, une simplification des skylines, puis, la proposition et l'utilisation de trois mesures de similarités qui sont utilisées dans l'algorithme de recherche permettant de trouver la solution optimale (descente de gradient). L'étude présentée dans ce chapitre a fait l'objet d'une publication avec une présentation orale dans une conférence internationale avec comité de lecture (Ayadi et al. (2018)).

Pour finir, nous proposons une conclusion générale et des perspectives pour résumer toutes les contributions de cette thèse, mettre en avant les avantages et les limitations de l'approche que nous proposons ainsi qu'une discussion sur les améliorations possibles.

2 La réalité augmentée : enjeux et problématique

Plan du chapitre

Dans ce chapitre, nous commencerons par une introduction où nous définirons rapidement le paradigme de réalité augmentée et ses principes de fonctionnement. Puis, nous ferons un tour d'horizon sur les différentes techniques et algorithmes existant aujourd'hui, ainsi que les différents outils (ou systèmes) de RA. Ensuite, nous présenterons quelques travaux dans les divers contextes et domaines d'applications dans lesquels la réalité augmentée contribue, et plus particulièrement le contexte urbain, où se situent les travaux de cette thèse. Enfin, nous présenterons les problématiques entourant la mise en œuvre de tels systèmes, et dans ce cadre précis, nous exposerons les objectifs fixés dans le cadre de cette thèse.

2.1 INTRODUCTION

La réalité augmentée (RA), apparue depuis bientôt une quinzaine d'années, offre de plus en plus d'applications et de dispositifs qui ne cessent de séduire de plus en plus les utilisateurs, qu'ils soient novices ou experts. En effet, ces technologies repoussent de jour en jour les limites de l'imaginable. Depuis son apparition, elle a été longtemps restreinte aux environnements intérieurs, confinés et contrôlés. Mais depuis l'apparition des systèmes mobiles de type smartphone et tablette, nous assistons à une vraie révolution en termes d'applications de réalité augmentée en milieu extérieur à grande échelle.

Avant de pouvoir aborder le sujet de cette thèse, à savoir la réalité augmentée sur mobile en contexte urbain, et l'utilisation du skyline comme marqueur naturel, il convient de rappeler ce qui est classiquement désigné sous ce terme de : ***réalité augmentée***.

2.1.1 Définitions

Il nous faut tout d'abord définir la réalité augmentée (en anglais *"Augmented Reality"*). En effet, si l'on revient à une caractérisation terminologique, la 9^{eme} édition du dictionnaire de l'*Académie française* fournit les définitions suivantes :

— Réalité : Qualité de ce qui est effectif, de ce qui existe (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 ;

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

En parcourant la littérature, nous avons constaté que diverses définitions ont été de même proposées pour ce paradigme de réalité augmentée. Ci-après, nous passons en revue quelques-unes d'entre elles, et ce par ordre chronologique.

Nous commençons par illustrer, à la figure 2.1, le *continuum* de la réalité-virtualité introduit par Milgram (1994), et qui reste jusqu'à aujourd'hui l'espace de référence cité dans presque tous les travaux en réalité augmentée. Les auteurs introduisent tout d'abord la notion de *réalité mixte*, qui regroupe, à différents niveaux d'échelle, l'environnement réel et virtuel. Dans cet espace *réel-virtuel*, où la réalité augmentée vient se placer, d'autres représentations existent : la **R**éalité **V**irtuelle (RV) ou **E**nvironnement **V**irtuel (EV), l'**E**nvironnement **R**éel (ER) et la **V**irtualité **A**ugmentée (VA). Ces différentes représentations cohabitent dans ce continuum, dont les deux extrémités correspondent à la réalité et à la virtualité pures. Les environnements réels et virtuels ne seront pas détaillés, étant à peu près clairs pour tout le monde. Cependant, nous définissions :

— **La Virtualité Augmentée** : elle consiste à intégrer des éléments réels extérieurs à la réalité virtuelle dans l'interface de simulation 3D, afin d'enrichir l'interaction de l'opérateur humain avec le monde virtuel au moyen d'outils et de tâches issues du monde réel.
— **La Réalité Augmentée** : par opposition à la virtualité augmentée, elle consiste à incruster du virtuel dans un environnement réel dans le but d'enrichir l'interaction de l'utilisateur avec le monde physique grâce à des données et services offerts par le monde numérique (l'ordinateur).

Dans la toute première revue de la littérature, Azuma (1997), la RA est aussi définie comme un *système capable de fournir un rendu **temps réel** tout en combinant les images **réelles et virtuelles**, le tout **en 3D***. Cette définition exclut donc automatiquement du champ de la réalité augmentée toutes les techniques de superposition ou de composition 2D d'images, où une ou plusieurs images synthétiques (vignettes, rendu synthétique d'un modèle 3D selon une projection particulière, etc.) sont tout simplement *superposées* ou *collées* (parfois avec une certaine transparence) par-dessus des images réelles.

Toujours dans Azuma (1997), la réalité augmentée mixant le réel et le virtuel doit répondre à certains critères :

— elle doit **combiner** le **réel** et le **virtuel**
— elle doit fournir une interaction en **temps réel**
— la composition doit s'effectuer **en 3D**

Pour J. Vallino (1998), la réalité augmentée est considérée comme un domaine émergeant dû à la difficulté à recréer complètement l'environnement qui nous entoure. Ces systèmes de réalité augmentée offrent donc à l'utilisateur une vue **augmentée** ou **rehaussée** d'une scène réelle avec des objets virtuels modélisés par un ordinateur. Cette augmentation a pour objectif d'enrichir la perception du monde réel par des informations ou objets additionnels bien déterminés.

Une autre définition nous paraît aussi très intéressante selon Fuchs et Moreau (2003), où la RA est présentée comme étant la technologie qui regroupe les techniques permettant de mixer le monde réel et le monde virtuel. Elle utilise l'intégration d'images réelles (IR) avec des entités virtuelles (EV) : images de synthèses, objets virtuels, textes, symboles, graphiques, etc. Toutefois, ces enrichissements ne se restreignent pas qu'aux augmentations visuelles, mais peuvent aussi être une **augmentation d'un ou de plusieurs *des cinq sens*** de l'être humain (la vue, le toucher, l'odorat, etc.).

Le principe incontournable, et sur lequel se rejoignent donc toutes ces définitions, est que la réalité augmentée nous offre la possibilité de rehausser, d'augmenter ou d'instrumenter notre perception du monde réel en lui rajoutant des entités virtuelles.

Bien évidemment, c'est l'aspect visuel qui prédomine aujourd'hui dans la plupart des applications dans l'état de l'art, pour la simple raison que c'est le sens que nous mettons le plus à contribution dans notre perception de l'environnement qui nous entoure.

Ce paradigme n'a donc cessé d'évoluer durant ces dernières années, et ses champs d'applications se sont diversifiés, faisant donc passer la réalité augmentée des environnements d'intérieur (en anglais *"indoor"*), préparés et contrôlés, vers des environnements à grande échelle en extérieur (en anglais *"outdoor"*), non contrôlables et non préparés.

En effet, de nouveaux types de dispositifs ont fait leur apparition ces dernières années, et plus particulièrement les smartphones et tablettes, qui nous sont inévitablement devenus indispensables de nos jours. Leurs capacités de calcul ont de même évolué, nous permettant de rompre les frontières entre le monde réel et le monde du numérique. Ces dispositifs nous permettent essentiellement de faire migrer la réalité augmentée traditionnelle des espaces de travail traditionnels (bureau) vers d'autres environnements extérieurs (la ville, par exemple). En effet, ils sont munis de performances de calcul qui ne cessent de croître d'une part, mais aussi de la possibilité de rester connecté au réseau Internet en continu, et ce à travers les réseaux 3G, 4G ou 4G+ (bientôt 5G). De plus, ils sont de plus en plus munis de capteurs en tout genre (GPS, caméra, accéléromètre, etc.) qui connaissent un développement en termes de précision et de miniaturisation, comme cela sera détaillé au chapitre 3.

Nous commençons donc par présenter dans ce qui suit les principes de fonctionnement de la réalité augmentée.

2.1.2 Principes de la RA

Afin de mieux comprendre les contraintes qui seront liées aux travaux présentés dans cette thèse, et plus particulièrement liées à la réalité augmentée mobile, il est tout d'abord indispensable d'en détailler les principes de fonctionnement. Pour cela, deux grandes familles d'approches en réalité augmentée mobile existent :

1. La réalité augmentée basée sur les **capteurs** (instruments)
2. La réalité augmentée basée sur la **vision** (marqueurs de tout genre)

En effet, l'objectif de tout système de RA est d'estimer la pose de l'utilisateur (i.e position et orientation) dans un repère bien particulier, et ce avec la meilleure précision possible.

En général, le repère utilisé est celui de la scène qui est filmée en temps réel, comme mentionné dans la section 2.1.1. Une fois cette pose estimée, il est possible

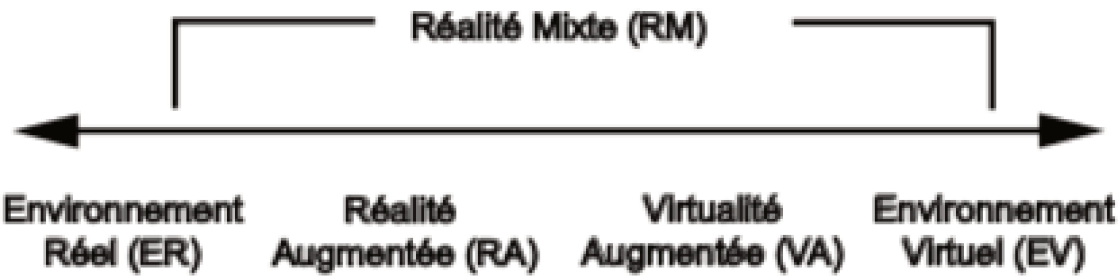


FIGURE 2.1 – *continuum Réalité Virtualité d'après Milgram (1994)*

d'augmenter la scène visualisée en ajoutant tout type d'objet de synthèse mixant ainsi le réel et le virtuel. Ce principe est assez connu dans la littérature, et, selon la famille d'approche (basée marqueur ou basée vision), il peut être décomposé en quatre étapes, comme illustré à la figure 2.2 :

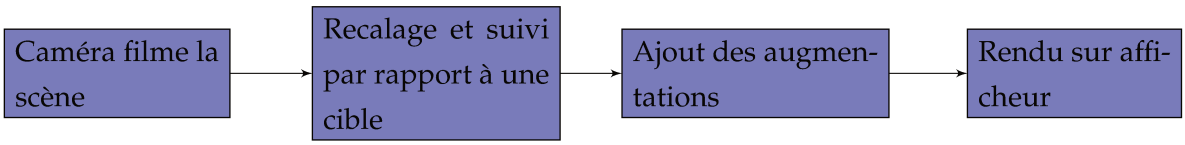


FIGURE 2.2 – *Principe global d'un système de RA*

— La caméra filme la scène ;
— L'objectif est de connaître la pose de la caméra (i.e. position, orientation). Dans chaque image filmée :
  — Un objet (marqueur, image de référence, projection de modèle 3D) est recherché ;
  — La pose de la caméra est estimée grâce aux instruments embarqués dans le dispositif (accéléromètre, gyroscope, baromètre, etc.) ;
— Connaissant la pose de la caméra, les augmentations (objets virtuels) sont ajoutées en cohérence avec le monde réel ;
— Le rendu est affiché sur le dispositif (PC, smartphone, tablette, lunettes intelligentes, etc.).

Une fois ces différents principes détaillés, nous proposons une revue des principaux systèmes de RA existant aujourd'hui, et en particulier ceux dédiés aux environnements extérieurs.
De plus, nous proposons une revue plus détaillée sur les applications en contexte urbain, ainsi que les différents travaux et projets menés, en partie 2.3.5. Cette partie servira aussi de référence à certaines discussions menées au chapitre 3.

2.2 ALGORITHMES ET OUTILS POUR LA RÉALITÉ AUGMENTÉE

Dans cette partie, nous détaillons les différents éléments pour la mise en place d'un système de RA. Cette analyse nous permet d'identifier les contraintes et les limites des systèmes actuels, et les verrous à lever pour la mise en œuvre du système proposé dans cette thèse.

En effet, les algorithmes de RA ont pour objectif de trouver la translation et la rotation entre un objet connu dans la scène et la caméra. Nous présentons en Annexe **??** le modèle de la caméra communément utilisé : le modèle *sténopé*. Ce modèle sera de même utilisé dans le cadre de nos travaux.

Nous commençons par un tour d'horizon sur les principales méthodes de suivi basées sur le concept de RA. Ces méthodes sont divisées en deux grandes catégories. Les méthodes de suivi dans un :

— environnement **connu a priori**

— environnement **inconnu**

Pour les méthodes de suivi dans un *environnement connu*, plusieurs approches existent :

— Les approches basées *marqueur*

— Les approches basées *image de référence*

— Les approches basées sur un *modèle 3D*

Nous présentons ensuite les différentes librairies (frameworks) de RA existant aujourd'hui. Enfin, nous détaillons les différents outils et dispositifs de rendu utilisés en RA.

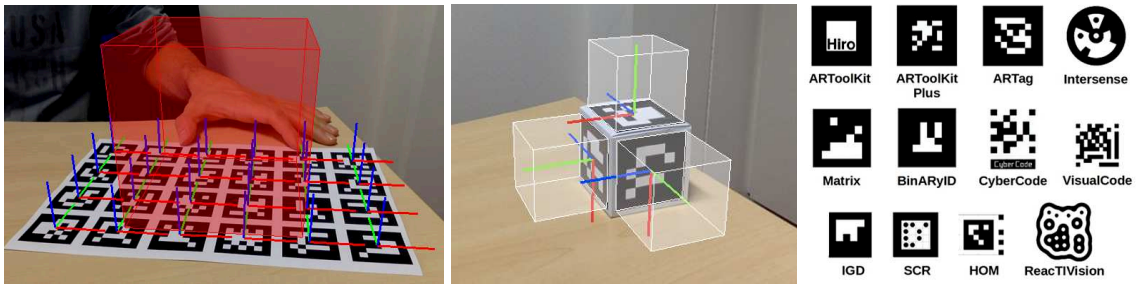
2.2.1 RA en environnement connu

Les approches de RA en environnement connu, comme leur nom l'indique, nécessitent une connaissance ***a priori*** de la scène visualisée, ou du moins d'un des objets qui la compose. L'objectif principal d'estimation de la pose se décline donc en l'identification et la recherche de cet objet en particulier. L'hypothèse qui doit toujours être vérifiée est : l'objet recherché existe dans la scène. De ce fait, il faudra donc avoir les caractéristiques de ce dernier, qui peut être : un marqueur plan (QR-Code, un motif particulier, etc.), des points d'intérêt particuliers extraits et caractérisés (SIFT, SURF, etc.), un modèle 3D de l'objet, etc.

Ces différentes méthodes se déclinent en trois approches. Dans les sous-sections 2.2.1.1, 2.2.1.2 et 2.2.1.3, nous donnons une synthèse et une discussion sur les avantages et inconvénients de chacune d'entre elles.

2.2.1.1 RA basée marqueurs

Les premiers outils de RA basés marqueurs ont été introduits à la conférence *SIGGRAPH* grâce à Kato et Billinghurst (1999), qui ont développé le framework de référence en RA (*ARToolkit*). Nous illustrons à la figure 2.3 un exemple de ces marqueurs [1]. Plusieurs contraintes doivent être respectées afin que le marqueur recherché soit convenablement reconnu et identifié dans la scène. D'une part, sa forme et sa taille doivent être connues. Puis, ces marqueurs sont souvent en noir et blanc afin de permettre leur discrimination rapide par rapport aux autres objets de la scène. Enfin, étant donné sa forme et les motifs qu'il renferme, à chaque marqueur correspond un identifiant unique permettant de le différencier d'autres marqueurs, qui pourraient éventuellement être présents en même temps dans la même scène (augmentations multiples).



(a) (b) (c)

FIGURE 2.3 – *Exemples de marqueurs pour la RA*

Ces contraintes permettent, dans la plupart des cas, de retrouver le marqueur très rapidement. Dans ce flux vidéo augmenté en temps réel, chaque image (en anglais *"frame"*) est traitée indépendamment de toutes les autres. En effet, le repérage du/des marqueur(s) permet de calculer la pose de la caméra par rapport à l'objet visualisé. La figure 2.4 illustre le processus standard des différents algorithmes de RA basée sur le marqueur.

Tout d'abord, l'image est convertie en niveaux de gris puis en une image binaire. Ce processus n'a, évidemment, aucune influence sur le marqueur présent dans la scène, qui, lui est initialement binaire. Cette binarisation se fait grâce à un seuil de binarisation qui se veut variable, selon les cas. Puis les contours du carré sont détectés en utilisant, par exemple, la transformée de *Hough* (Hough (1962)). Ensuite, les coins du motif sont de même détectés avec un détecteur de coin, par exemple celui de *Harris* (Harris et Stephens (1988)). Les coordonnées de ces coins permettent de faire ressortir le marqueur de l'image et de le reconnaître précisément parmi tous ceux qui pourraient être éventuellement présents. Cette identification s'appuie généralement

1. http ://www.labri.fr/perso/pbenard/teaching/rv/td4.html

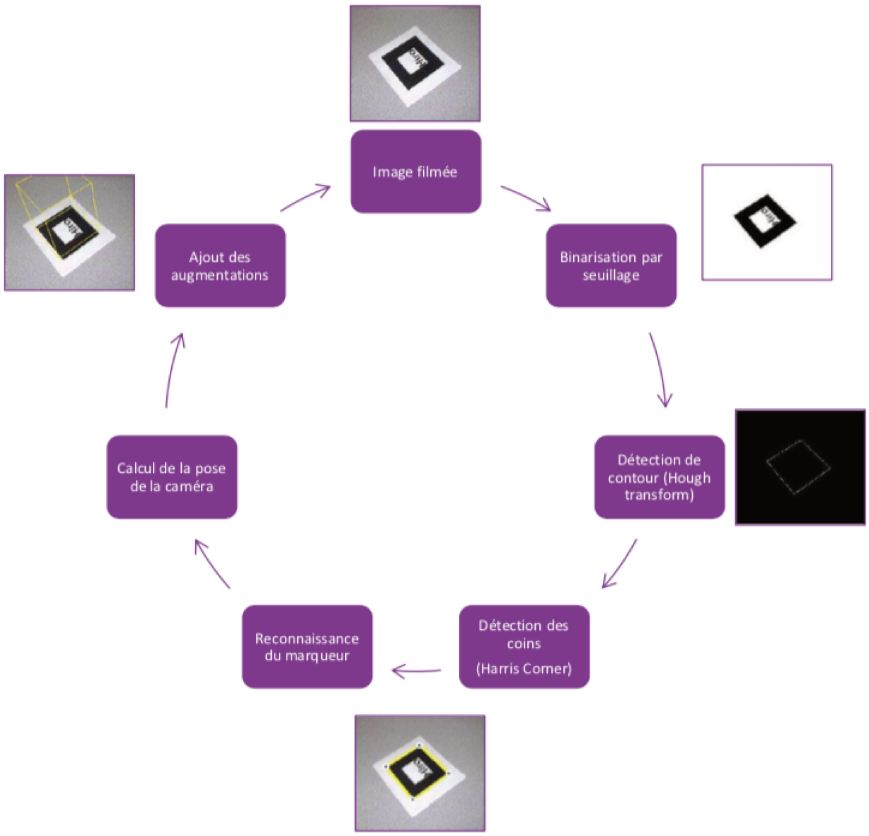


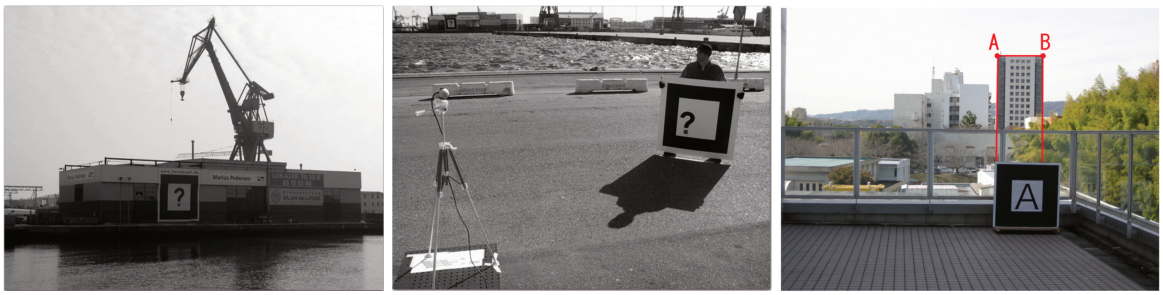
FIGURE 2.4 – *Processus de RA basée marqueurs, extrait de V. Havard (2018)*

sur la partie centrale en fond blanc comportant les paramètres d'identité du marqueur. Bien évidemment, il faudrait avoir une base de données de marqueurs déjà construits. Enfin, la pose de la caméra est estimée par rapport à ce marqueur et l'augmentation est appliquée sur l'image. La visualisation se fait alors sur le dispositif d'affichage du système.

Parmi les avantages de ce type de méthode, nous citons la *rapidité de calcul* dans la détection et l'identification des marqueurs. En effet, la simplicité des marqueurs permet d'en détecter un ou même plusieurs à la fois en même temps tout en ayant un système stable d'un point de vue détection, tournant en temps réel sur des dispositifs mobiles (smartphone).

Cependant, les performances de la détection sont fortement impactées par les changements de luminosité, les conditions d'éclairage et les occultations (en anglais *occlusions*). En effet, si une partie (même petite) ou la totalité du marqueur est occultée par un quelconque objet de la scène réelle, le processus d'augmentation échoue. Une contrainte majeure à ce type d'approche est la nécessité d'avoir un marqueur pouvant se voir de loin dans la scène réelle. Nous illustrons à la figure 2.5 une des approches que nous traiterons dans le chapitre 3, où des marqueurs ***géants*** sont utilisés pour augmenter la scène en grandeur nature (marqueur de taille 10x10 mètres).

Ces méthodes basées marqueurs sont donc prisonnières des marqueurs artificiels qui peuvent présenter une contrainte forte dans certains contextes (urbain, industriels, etc..).



(a) (b) (c)

FIGURE 2.5 – *Extrait de Rune Nielsen (2005)*
(a) Marqueur portatif, (b) Marqueur de taille 10x10 mètres fixés pour démonstration
Extrait de Yabuki et al. (2011)
(c) : Marqueur pour une augmentation en extérieur

Pour se prémunir de telles contraintes, et éviter de nous déplacer avec des marqueurs artificiels en tout lieu, il existe des marqueurs naturels (par exemple le Skyline).
Dans ce qui suit, nous présentons les approches de RA basées sur des images de référence.

2.2.1.2 RA basée image de référence

Dans certains contextes, la présence de marqueurs artificiels dans la scène réelle peut être très contraignante. Pour cela, les approches de RA basées sur des images de référence permettent de répondre à ce besoin. En effet, ces approches permettent de détecter des images ***particulières*** déjà ***préparées*** et ***choisies*** à l'avance, qui remplacent les marqueurs artificiels. Nous en donnons un exemple à la figure 2.6.

FIGURE 2.6 – *Extrait de Baggio et al. (2012) : Marqueur naturel*

Nous citons les travaux de Koch et al. (2014), où la localisation est estimée grâce aux panneaux de signalisation présents à l'intérieur des bâtiments. Ces panneaux présentent des textures et des images très particulières qui se distinguent des autres objets de la scène. En effet, de ces panneaux plusieurs points d'intérêt sont facilement et rapidement extraits permettant de caractériser cette image, et donc ce panneau. Au préalable, une base de données de tous les panneaux est construite, avec leur position dans le bâtiment. De plus, ces images sont associées à leurs caractéristiques grâce au même algorithme employé par l'utilisateur. Enfin, une mise en correspondance est établie entre l'image captée par la caméra filmant la scène réelle et la base de données d'images préalablement traitées. Cette mise en correspondance permet de déterminer l'image que l'utilisateur est en train de visualiser, et donc sa position dans le bâtiment.

En ce qui concerne l'algorithme utilisé pour la caractérisation des panneaux, ou toute autre image, il en existe une multitude qui sont assez connus dans le monde du traitement d'images. Le choix de l'algorithme est généralement fait selon le contexte de l'application. Ces algorithmes permettent, par exemple, la détection de :

— **points d'intérêt** : SIFT (Lowe (2004)), SURF (Bay et al. (2006)), FAST (Rosten et Drummond (2005))
— **coins** : Harris (Harris et Stephens (1988))
— **bords et contours** : Canny (Canny (1986)), *Roberts*, *Sobel*, *Prewitt*

Chacun de ces algorithmes a des caractéristiques particulières. Un des algorithmes les plus utilisés en littérature est *SIFT* dont les points extraits possèdent des propriétés assez intéressantes :

— **Local** : La détection de chacun des points est indépendante, et par conséquent leur détection est robuste aux occlusions.
— **Distinct** : Chacun des points extraits est caractérisé par un descripteur différent des autres.
— **Invariable** : Chaque descripteur est invariable :
  — À la translation
  — À la rotation
  — Aux changements d'échelle
— **Quantité** : Le nombre de points extraits est très important, même pour une image de petite taille, ne dépendant que de la texture de l'image.
— Efficacité : Les performances de l'algorithme sont proches du temps réel.
— Extensible : Différents descripteurs peuvent être utilisés.

Ces approches, basées sur les algorithmes de traitement d'images et l'extraction de certains points d'intérêt, présentent plusieurs avantages par rapport à celles utilisant des marqueurs simples. En effet, contrairement aux approches basées marqueurs, l'occultation d'une petite partie de l'élément de référence n'affecte pas tout

le processus de détection. Cet avantage est dû à la quantité du nombre de points détectés et leur distribution sur toute l'image. Néanmoins, plus le nombre de marqueurs à détecter augmente, plus le nombre de points d'intérêt à extraire augmente, si bien que la complexité de l'algorithme et les temps de calcul deviennent beaucoup trop importants pour des dispositifs mobiles. **Parmi ces points d'intérêt, on peut en citer quelques-uns qui nous intéressent très particulièrement dans cette thèse : le *Skyline*.**

2.2.1.3 RA basée modèle 3D

Ce type d'approche nécessite une connaissance a priori de l'objet à détecter, et plus encore, son modèle de synthèse en 3D. De plus, une phase de *préparation* est nécessaire. Tout d'abord, la phase de préparation consiste à générer des images de synthèse correspondant à des poses particulières de la caméra. En d'autres termes, ces images regardent l'objet en question sous plusieurs angles. Puis, des points d'intérêt sont extraits de ces images de référence, et leurs descripteurs, nommés (***descrip1***), sont calculés et associés à la pose correspondante. Ensuite, les mêmes descripteurs sont calculés à partir de l'image réelle, nommés (***descrip2***). Enfin, une phase d'appariement entre ***descrip2*** et l'ensemble des ***descrip1*** de la base de données permet de déterminer ceux qui se ressemblent le plus, d'où la pose de la caméra. L'augmentation peut alors se faire et s'afficher sur le dispositif.

De telles approches présentent l'avantage d'être insensibles au point de vue avec lequel l'objet est regardé (visualisé). D'une part, contrairement aux méthodes de la partie 2.2.1.2, où les descripteurs ne peuvent être calculés que sur la texture de l'objet recherché, ces méthodes présentent l'avantage de pouvoir baser leur reconnaissance sur d'autres critères comme la forme. Les travaux dans Drost et al. (2010) ou Hinterstoisser et al. (2016) en sont des exemples. D'autre part, ces méthodes présentent l'avantage d'être assez robustes aux variations de luminosité.

Cependant, ces approches nécessitent d'avoir un modèle 3D de l'objet recherché. Comme nous le verrons dans la partie 2.3, où nous ferons un tour d'horizon des domaines d'application de la RA, les modèles 3D se sont démocratisés ces derniers temps. On trouve non seulement des modèles 3D d'objets industriels, de design ou d'architecture, mais aussi des modèles 3D de villes entières, comme celle de Lyon. Les recherches les plus récentes sur le suivi d'objets basés modèles 3D le prouvent Hinterstoisser et al. (2016), Kehl et al. (2015), Radkowski (2016).

2.2.2 RA en environnement inconnu

Dans cette section, nous abordons brièvement les approches de localisation dans un environnement inconnu. Les méthodes les plus connues essayent de reconstruire à la volée et en temps réel la scène étudiée, et ce grâce à des méthodes de type *SLAM* (*Simultaneous Localization And Mapping*). Une information est alors récupérée sur cette scène, a priori inconnue. Ces informations sont alors couplées / croisées avec l'empreinte au sol de l'environnement dans lequel évolue l'utilisateur pour le localiser.

Contrairement aux méthodes des parties 2.2.1.1, 2.2.1.2 ou **??**, qui se basent sur l'image pour localiser l'utilisateur, ces méthodes se basent plutôt sur des capteurs différents. En effet, on peut citer le ***FootSLAM*** (de Angermann et Robertson (2012)) ou le ***WiSLAM*** (de Bruno et Robertson (2011)) qui reposent, respectivement, sur l'utilisation d'un capteur inertiel ou sur l'intensité des signaux reçus des bornes *Wifi* à l'intérieur du bâtiment.

Ces méthodes ne seront pas abordées dans cette thèse. Néanmoins, il nous paraissait important de les citer et d'avoir une vue d'ensemble sur ce qui se fait et comprendre les méthodes de localisation en intérieur et extérieur.

Dans la partie suivante, nous allons détailler les différents dispositifs utilisés pour faire de la RA, ainsi que les différentes librairies existantes et qui se basent sur les algorithmes cités dans les parties 2.2.1 et 2.2.2).

2.2.3 Outils de la RA

Les dispositifs de rendus en RA sont très divers. Idéalement, et pour avoir un rendu *parfait*, le dispositif de rendu qui devrait être utilisé est l'œil humain.

Pour l'instant, il n'est pas encore possible de projeter du contenu directement dans la vision d'un sujet humain. Il est donc nécessaire de passer par des dispositifs intermédiaires : portable (smartphone), tablette, lunettes intelligentes, projecteur mixant contenu réel et virtuel, etc.

Nous illustrons à la figure 2.7, un exemple d'application où les auteurs (Behzadan (2008)) proposent un outil d'assistance au travail pour la visualisation géo-référencée et dynamique des constructions. Cette étude a été possible grâce à la plateforme matérielle *ARVISCOPE* de Behzadan et Kamat (2005).

Dans ce qui suit, nous proposons un état de l'art des différents dispositifs existant en RA.

2.2.3.1 Dispositifs de rendu

Les dispositifs ou appareils de rendu sont très divers. Ils sont clairement classés en trois types. Les appareils :

(a) (b)

FIGURE 2.7 – *(a) Behzadan (2008), (b) Behzadan et Kamat (2005)*

— **Portés par l'utilisateur**
Les exemples les plus connus sont le casque de réalité virtuelle ou les lunettes intelligentes. Aujourd'hui, on peut citer, à titre d'exemple, le casque *Microsoft HoloLens* ou son concurrent *Meta*. L'avantage de ces méthodes est d'avoir les mains libres tout en visualisant la scène augmentée. Ces appareils peuvent être de type *Optical-See-Through (OST)* ou *Video-See-Through (VST)*, détaillés ci-dessous.

— **Tenus en main par l'utilisateur :**
Ce sont les appareils de type *tablette* ou *smartphone*. L'avantage de ces dispositifs est qu'ils sont très connus du grand public permettant ainsi de cibler un très large panel de personnes, tout en offrant la possibilité d'interagir avec l'application grâce à leurs écrans tactiles.

— **Par projection** :
Ces appareils sont, en revanche, très méconnus dans le monde de la RA. Leur inconvénient tient dans le fait qu'ils fournissent des rendus très peu réalistes.

Une question se pose très vite lors de l'étude d'un projet de RA :
Quel appareil ou dispositif faut-il choisir ?

En effet, pour répondre à cette question, plusieurs critères doivent être pris en considération.
Tout d'abord, le nerf de la guerre : le ***prix*** du dispositif !
Puis, selon le contexte du projet, faut-il opter pour un affichage ***OST*** ou ***VST*** ?

Ensuite, quels ***champs de vision*** offre le dispositif ?
Enfin, l'affichage est-il ***binoculaire*** ou ***monoculaire*** (cas des ***VST***) ?

Nous illustrons à la figure 2.8, un récapitulatif des différents dispositifs existants aujourd'hui sur le marché de la RA.

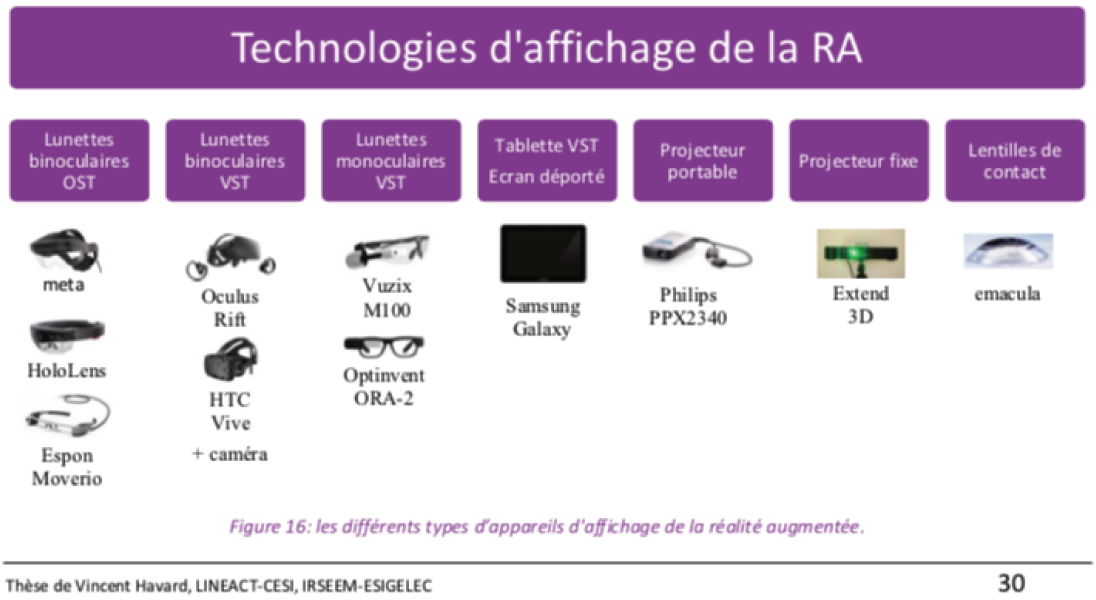


FIGURE 2.8 – *Dispositifs actuel de RA*

2.2.3.1.1 Dispositifs VST et OST

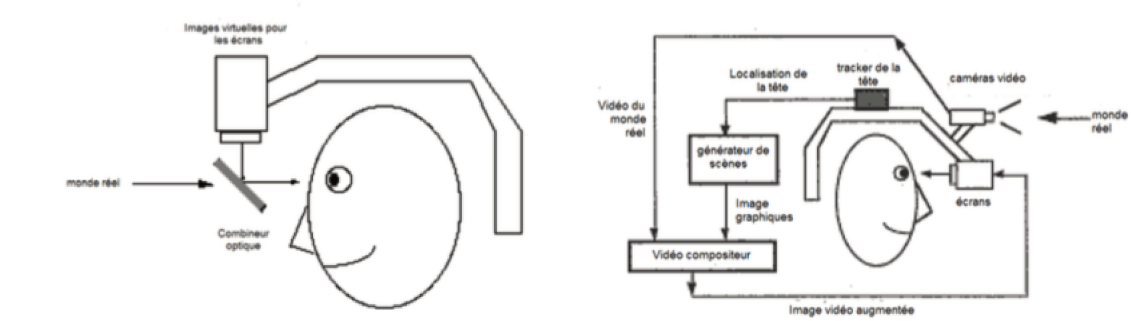
Ces dispositifs, aussi connus sous le nom *casques de Réalité Virtuelle* ou de *Réalité Augmentée* (en anglais *HMD :Head Mounted Display*), représentent les dispositifs les plus courants et les plus utilisés aujourd'hui en RA. Il en existe deux sortes : les casques de type optique (OST) et les casques de type vidéo (VST).

Nous illustrons à la figure 2.9, la classification qui a été proposée par Azuma (1997).

En effet, selon l'appareil choisi (OST ou VST), la RA sera visionnée différemment par l'utilisateur.

Les casques VST proposent une augmentation *synchrone* au sein d'un écran. En effet, l'avantage de ces méthodes est qu'il n'y a pas de décalage entre le rendu du monde réel, capté par une caméra, et les éléments virtuels rajoutés. Néanmoins, l'utilisateur ne voit le monde réel qu'à travers la caméra qui lui en livre les images. Sa vision réelle est donc altérée, ce qui le rend assujetti aux caractéristiques de la caméra (son champs de vision, espace de couleurs, etc.).

Les casques OST, quant à eux, permettent à l'utilisateur de visualiser le monde réel de ses propres yeux tout en ayant l'information en RA. Une étape de calibration



(a) Casque à vision directe (*optical see-through*) (b) Casque à vision indirecte (*video see-through*)

(a) (b)

FIGURE 2.9 – *Classification des casque de RA et RV selon Azuma (1997) : (a) Casque à vision directe (OST), (b) Casque à vision indirecte (VST)*

est nécessaire afin d'ajuster le casque selon les personnes qui le portent et recaler comme il faut les éléments virtuels et réels, selon la position du casque.

2.2.3.1.2 Dispositifs tenus en main

Les dispositifs tenus en main offrent à l'utilisateur la possibilité de visualiser le monde réel et les objets virtuels rajoutés sans avoir à porter de casque ou de lunettes spéciales. Dans le monde de la RA, nous en trouvons une grande variété, dont le plus basique est bien évidemment le smartphone. L'un des premiers dispositifs de cette famille est celui présenté par Rekimoto et Nagao (1995), illustré à la figure 2.10, nommé *magnifying glass*. C'est une sorte d'écran équipé d'une petite caméra arrière qui fournit les images de la scène réelle qu'on peut rehausser au besoin.

FIG. 1.11: Magnifying glass approach : Premier *Hand-held* [Rekimoto et Nagao, 1995]

FIGURE 2.10 – *magnifying glass de Rekimoto et Nagao (1995)*

De nos jours, on trouve une multitude d'autres dispositifs comme interface de visualisation. À titre d'exemple, on peut citer, les tablettes, les PDA ou encore les smartphones, sur lesquels portent nos travaux.

FIGURE 2.11 – *Dispositifs de visualisation tenus en mains : (a) Tablette, (b) PDA, (c) Smartphone*

2.2.3.2 Les librairies existantes

Comme vu dans la partie 2.2, la RA est basée sur des méthodes de traitement d'images et des méthodes de rendu 3D.

Dans les parties 2.2.1 et 2.2.2, nous avons détaillé les différentes approches pour développer des applications de RA. En effet, ces **algorithmes**, bien connus en littérature, sont déjà **implémentés** et proposés dans certaines **librairies** (open source ou pas). De plus, afin d'avoir un rendu synthétique et un objet virtuel à incruster, il nous faut un **moteur de rendu 3D**.

Dans ce qui suit, nous détaillons ces deux aspects.

2.2.3.2.1 Moteurs de rendu 3D

Les moteurs de rendu existant aujourd'hui reposent, dans le cas général, sur ***OpenGL***. On peut citer les plus importants : ***Unity 3D***, ***Unreal Engine***, ***Ogre 3D***, ***Belnder***. Quant aux plateformes mobiles, elles reposent sur l'*API* ***OpenGL-ES*** (**ES** pour *"**Embedded System**"* en anglais). ***Unity 3D*** est le moteur qui possède la communauté la plus active et les principaux outils de RV et RA sont compatibles avec ce moteur.
De plus, l'aspect technique qui nous a intéressé dans *Unity* en début de cette thèse, est la possibilité, à partir d'un projet Unity, d'exporter et générer des projets (applications) sur les différentes plateformes mobiles : ***iOS***,*Android* et *Windows Phone*. L'inconvénient majeur de cette fonctionnalité est de fournir des projets très compliqués et très étendus, où une contribution, aussi petite soit-elle, demande une chaîne de compréhension qui nécessite des développements très lourds. Néanmoins, l'avantage de Unity est sa compatibilité avec certains framework, comme ***OpenCV***.

Finalement, nous avons opté pour des développements dans des couches inférieures, en travaillant directement avec ***OpenGL-ES***, et ***OpenCV***, ce qui sera discuté dans la partie 3.4.4 du chapitre 3.

2.2.3.2.2 Librairies de RA

Afin d'éviter de réécrire certains algorithmes, il peut être intéressant d'utiliser des briques déjà prêtes rendues disponibles à travers des librairies (en anglais *"Framework"*). Il en existe une liste, non exhaustive, d'environ soixante-dix librairies [1]. On peut, à titre d'exemple, citer *Metaio*, un framework qui a été racheté par *Apple* en 2015 pour donner naissance au framework de référence aujourd'hui : *ARKit*. Plusieurs autres librairies peuvent de même être citées : *Vuforia* et *Wikitude*, mais elles ne sont malheureusement pas des bibliothèques *open source*. En effet, lorsque l'on souhaite faire des recherches dans le domaine de la vision et plus particulièrement dans les algorithmes liés à la réalité augmentée, il est important d'avoir la possibilité de modifier les algorithmes au besoin.

Cependant, il est à noter que certaines de ces librairies proposent le suivi d'un modèle numérique 3D, mais présentent néanmoins des limites et des contraintes. En effet, il faut tout d'abord commencer par filmer l'objet à suivre (en anglais *"Tracking"*) selon différents points de vue pour qu'il soit reconstruit et puisse être reconnu ensuite.

Le choix du framework dépend donc de plusieurs critères :

— les type de méthode de suivis supportés (basées marqueur, template, modèle 3D ou SLAM) ;
— l'intégration du framework dans les trois principaux moteurs de rendu 3D ;
— le support de tablette *Android* ou *iOS* ;
— la présence d'une communauté de développeurs ;
— la gratuité de l'outil ;
— le fait que le logiciel soit libre.

Ainsi, afin de faire preuve d'un concept ou pour le développement d'un prototype basique, il est possible d'utiliser Vuforia ou Wikitude. En revanche, pour des besoins plus spécifiques, ce qui est notre cas dans cette thèse, il est nécessaire de passer par un développement plus poussé tout en mélangeant du rendu virtuel en 3D en temps réel et des algorithmes de traitement d'images. Nous utilisons dans le cadre de nos travaux en traitement d'image la bibliothèque OpenCV qui est plutôt considérée comme une bibliothèque de développement (en anglais *SDK : Software Development Kit*) plutôt qu'une librairie. Nous l'avons retenue car celle-ci contient tous les outils permettant de faire du traitement d'image et de la réalité augmentée avec l'avantage d'être open source, nous permettant donc de mettre les algorithmes qui y sont proposés en adéquation avec les besoins que nous avons.

1. http ://socialcompare.com/en/comparison/augmented-reality-sdks, 2018

2.3 APPLICATIONS

La Réalité Augmentée (RA) a pour but d'augmenter et d'enrichir une scène réelle et donc notre *réelle* perception de ce monde. Cette augmentation se fait par le rajout d'éléments, a priori inexistant, ou du moins uniquement virtuellement, les rendant visible à l'œil humain. Différentes applications potentielles et actuelles pour la RA en temps réel existent. Nous faisons dans ce qui suit un tour d'horizon des différents contextes d'application qui existent aujourd'hui.

2.3.1 Médecine

La RA dans le domaine médical permet, à titre d'exemple, à un médecin de pouvoir visualiser des données 3D particulières par-dessus le corps du patient (images ultra-sonores, tomographie 3-D, images à résonance magnétique etc.).

On peut citer la première opération chirurgicale au monde en RA. En effet, une opération de l'épaule, unique au monde, a été menée en décembre 2017 à l'hôpital *Avicenne*, en *Seine-Saint-Denis* en *France*. Pour la première fois, un casque de RA projetant un hologramme a été utilisé, et ce afin de projeter le squelette du patient en temps réel, comme s'il le voyait à travers sa peau. Dans ce contexte, ce casque de RA a permis de remplacer trois écrans dans le bloc opératoire. Nous illustrons à la figure 2.12 une des images de l'opération qui a été diffusée en direct en streaming sur la page *Youtube* officielle des *Hôpitaux de Paris*.

FIGURE 2.12 – *Première opération chirurgicale au monde en RA*

Plus généralement, plusieurs travaux en littérature convergent vers la même problématique : la fusion automatique d'images 3-D et 2-D. Cette problématique nécessite une étape de recalage. Feldmar et al. (1997) et Roth et al. (1999) traitent le problème de recalage d'une image tomographique ou à résonance magnétique dans une séquence d'images à rayons X.

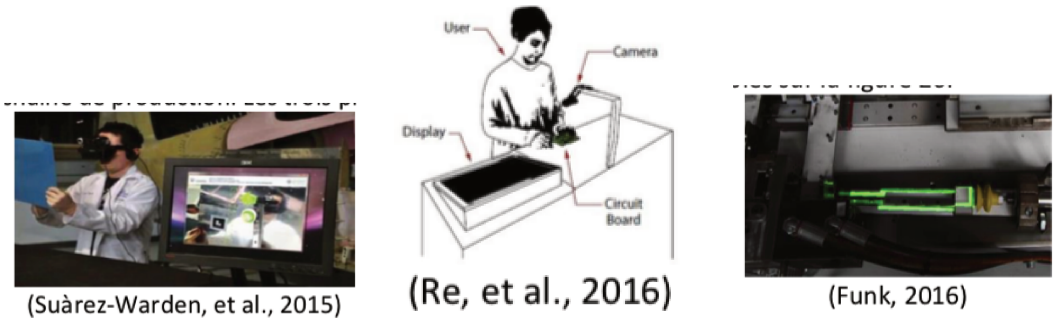
Kerrien et al. (1999) traite du recalage de reconstructions 3-D de la vascularisation cérébrale dans des images d'angiographie numérique soustraite. Ainsi, en neuro-radiologie interventionnelle par exemple, ces travaux permettent au radiologue de savoir en temps réel où se trouve son cathéter dans le corps du patient.

2.3.2 Industrie

Dans le domaine de l'assistance en milieu industriel, la RA a principalement deux grands axes de contribution : la maintenance et l'assemblage. Nous en donnerons un exemple pour chacun.

Pour les travaux de maintance, dans Martínez et al. (2014), les auteurs se servent de la réalité augmentée pour guider un opérateur dans les tâches à effectuer. Grâce à un écran géant déporté, l'opérateur se repère dans l'espace dans lequel il évolue. La tâche à effectuer étant de manipuler une grue pour changer un collimateur du *CERN*.

En termes de travaux d'assistance à l'assemblage, la contrainte de précision est plus forte. En effet, le recalage virtuel-réel doit être le plus précis possible afin que chaque élément soit placé précisément à son emplacement sur le système à assembler. Les travaux de Re et al. (2016) utilisent la RA pour faire de l'assemblage de circuits imprimés sur un écran déporté. En effet, l'utilisation d'un écran déporté (VST) permet d'avoir un rendu beaucoup plus précis que celui d'un OST, tout en laissant l'opérateur libre de ses mouvements pour travailler. Dans Suárez-Warden et al. (2015) la RA est utilsée pour placer des rivets sur une aile d'avion. Funk (2016) propose un système de RA permettant de faciliter l'assemblage de pièces industrielles sur une chaîne de production. Nous illustrons, à la figure 2.13 ces trois principaux types d'appareils.



(a) (b) (c)

FIGURE 2.13 – *RA en indsutrie : (a) Extrait de Suárez-Warden et al. (2015), (b) Extrait de Re et al. (2016), (c) Extrait de Funk (2016)*

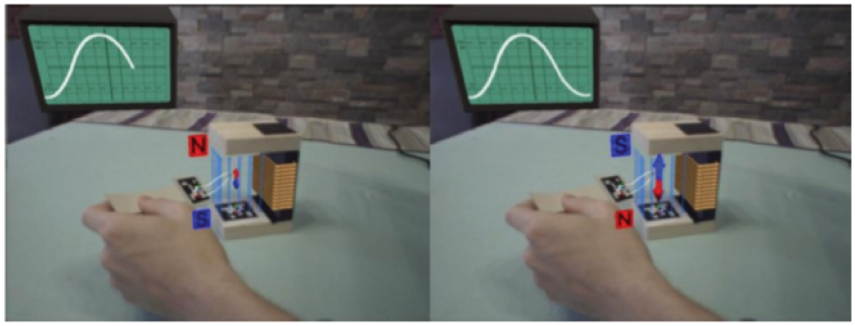
Dans sa thèse, V. Havard (2018) propose un nouveau modèle de données facilitant la création de contenu en RA. Le champ d'application est dédié aux opérations

sur les systèmes industriels, permettant ainsi à un opérateur de comprendre plus rapidement les tâches à faire qu'avec un guide papier. Cela permet de même d'éviter certaines erreurs. Ces travaux ont également permis de mettre en place un environnement expérimental basé sur une chaîne de production didactique et une application créée grâce à ce modèle.

2.3.3 Formation

Dans Martínez et al. (2014), les auteurs utilisent la RA pour projeter, sur un écran déporté, plusieurs jeux où l'on teste les compétences en mathématiques des étudiants à travers des jeux de mémoire. L'application consiste à demander les résultats des opérations, ou de se souvenir des nombres premiers. En effet, l'application consiste en dix grands marqueurs physiques posés sur le sol correspondant à des nombres de 0 à 9. L'objectif est de masquer les chiffres inclus dans la réponse à la question qui a été fournie par l'application. Cette application a été conçue pour être utilisée lors de la journée des mathématiques à Heureka, le centre scientifique finlandais.

Dans sa thèse, Cieutat (2013) propose d'utiliser RA pour l'éducation, en rendant visible des phénomènes physiques (électromagnétisme) à travers des jeux sérieux pour apprendre l'électromagnétisme, ou encore pour la formation à la soudure (voir figure 2.14).



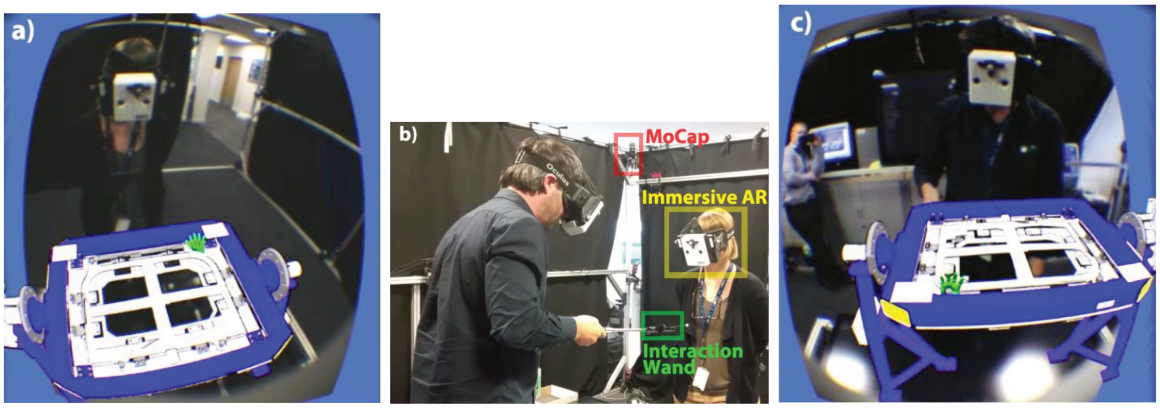
(a) et (b) : Variation du champ magnétique

Figures 4.6 : Un autre exercice réussi

(a)

FIGURE 2.14 – *Extrait de Cieutat (2013)*

González-Franco et al. (2016) propose d'utiliser la RA pour la formation des professionnels (voir figure 2.15). L'idée est d'apprendre à maintenir la porte d'un avion. L'avantage de ces formations en RA est qu'il n'est pas nécessaire de posséder le système réel pour se former.



(a) (b) (c)

FIGURE 2.15 – *Extrait de González-Franco et al. (2016)*
(a) vue du formateur pour l'opération d'assemblage virtuel
(b) vue du laboratoire avec les caméras de motion capture et les deux participants équipés : le formateur tient le bâton d'interaction
(c) vue de l'apprenant qui observe l'opération. Le bâton d'interaction est représenté par l'icône verte.

2.3.4 Navigation

L'une des premières applications de RA pour la navigation est *MARS* dans Höllerer et al. (1999). L'objectif était de démontrer la faisabilité d'un système de RA en milieu extérieur. Ce système fournit des informations sur l'environnement tout au long de la déambulation de l'utilisateur. Cette machine, considérée comme *une machine de guide personnalisée* et composée d'un ordinateur portable, sous forme de sac, porté sur le dos et d'un écran tactile pour la visualisation et l'interaction, augmente la scène avec les informations correspondant à la vue courante. La pose de l'utilisateur est fournie par GPS différentiel (DGPS) et un capteur inertiel.

Les travaux de Höllerer et al. (1999), pionnier dans ce domaine, ont permis l'élaboration du projet du projet *BARS(Battlefield Augmented Reality System)*, de Livingston et al. (2011). Ce projet, réalisé dans un contexte militaire, avait pour but de proposer aux militaires un système d'aide à la navigation et d'augmentation de sorte à pouvoir connaître, en plein champs de bataille, la position des tireurs embusqués (voir figure 2.16). Les informations de pose sont de même récupérées grâce à un GPS couplé avec une centrale inertielle. Le dispositif utilisé était composé d'une sorte de casque semi transparent, une souris tenue à la main ainsi qu'une commande vocale pour l'interaction.

Aujourd'hui, on trouve d'autres applications dans la navigation ayant des impacts sur plusieurs autres domaines, en l'occurrence la sécurité routière. En effet, on trouve aujourd'hui des applications de RA pour les casques de moto indiquant à tout

(a) Le prototype de Livingston et al. (2011) testé par un militaire (b) BARS par Yohan et al. (2000) (c) MARS par Höllerer et al. (1999)

FIGURE 2.16 – *Exemples de système de RA pour la navigation*

instant et en temps réel au conducteur, sa vitesse, sa trajectoire ainsi que celle de la route devant lui.

2.3.5 Contexte urbain

Le scénario type dans ce contexte est la simulation d'un nouveau bâtiment dans la ville. En effet, avant l'étape de construction, il est nécessaire d'impliquer le citoyen dans l'étape de décision et de procéder à une consultation large. Pour cela, la RA est considérée comme l'outil permettant de voir, sous la forme d'un rendu synthétique en 3D, le futur bâtiment présenté dans son contexte final.
Plusieurs travaux sont proposés dans ce contexte. Nous en faisons un tour d'horizon, un à un, en détaillant les approches proposées et les algorithmes utilisés.

2.3.5.1 Le projet ARTHUR

On peut citer le projet *ARTHUR* de Broll et al. (2004), qui consiste en une table ronde (physiquement) augmentée par les futurs bâtiments, et ce pour venir approuver ou discuter les décisions complexes, de conception ou de planification pour les architectes. L'approche vise à intégrer le résultat des systèmes de *CAO* de façon transparente dans l'environnement de RA collaboratif. L'approche est complétée par des mécanismes d'interaction pédagogique qui peuvent être facilement imaginables pour différents scénarios d'application. Néanmoins, cette approche reste toujours limitée et testée dans un environnement intérieur et la conception est toujours effectuée par les personnes présentes aux réunions de conception et d'examen de projets. Nous illustrons à la figure 2.17, l'exemple de cette étude.

FIGURE 2.17 – *projet ARTHUR de Broll et al. (2004)*

2.3.5.2 The Harbour Game

Dans Rune Nielsen (2005), les auteurs présentent un jeu collaboratif dédié à la RA pour la planification urbaine. Cette approche étant assez basique, nous ne n'y attarderons pas trop. C'est une approche de RA basée marqueur, qui est détaillée dans la partie 2.2.1.1.
Nous présentons à la figure 2.18 un exemple des résultats. En (a), le modèle à l'échelle réalisé par la municipalité pour discuter des plans et des projets futurs avec le public, et en (b), l'augmentation de la vidéo, grâce à un marqueur, en plaçant le stade au centre-ville.

(a) (b)

FIGURE 2.18 – *Harbour Game*
(a) le modèle à l'échelle réalisé par la municipalité pour discuter des plans et des projets futurs avec l'audience
(b) Augmentation de la vidéo, grâce à un marqueur, en plaçant le stade au centre ville

2.3.5.3 Projet Digitalo

Ce cas concerne des expérimentations effectuées grâce au projet **Digitalo** (*"Digital House"*) à *Espoo* en *Finlande*.

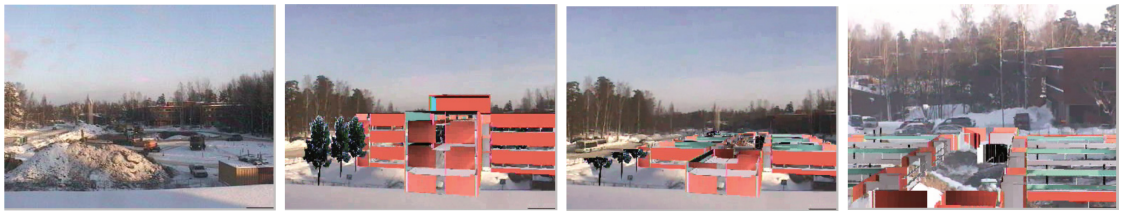
Dans Woodward et al. (2007) et Honkamaa (2007), plusieurs approches sont présentées, et plusieurs prototypes sont testés. Tout d'abord en 2003 avec le prototype ***ARWebCam***, puis en 2004 avec le prototype ***ARMobile*** implémenté sur des PC miniatures (***Sony Vaio***), ensuite en 2007 avec le prototype ***AROnSite***.
Dans ces différentes approches, les auteurs proposent différentes méthodes.

2.3.5.3.1 ARWebCam

ARWebCam est basé sur une acquisition par une webcam. Les vidéos sont accessibles depuis *Internet*, où n'importe qui peut visualiser le chantier de construction avec le modèle virtuel de Digitalo superposé à la vue réelle acquise par la webcam. Ces expérimentations ont servi à la présentation aux utilisateurs des bâtiments et autres groupes d'intérêt, ainsi qu'au suivi et à la comparaison avec les plans originaux. L'algorithme commence par capturer le flux vidéo grâce à la caméra *PTZ*, en enregistrant l'image et les valeurs PTZ de cette dernière. Le serveur augmente ensuite le flux vidéo par le modèle virtuel *Digitalo* en utilisant les paramètres de la caméra récupérés, tout en mettant à jour l'image du bâtiment virtuel pour qu'elle corresponde à la vue de la caméra réelle sur le chantier.

Nous illustrons à la figure 2.19 les résultats de cette approche, où :

— (a) : Image acquise par la webcam
— (b) : Image augmentée avec le bâtiment ***Digitalo***
— (c) : Vue sans les toits du bâtiment
— (d) : Vue en hauteur sur le bâtiment incrusté



(a) (b) (c) (d)

FIGURE 2.19 – *Extrait de Woodward et al. (2007) - ARWebCam*

2.3.5.3.2 AROnSite

AROnSite est basé sur la détermination de l'emplacement prévu du bâtiment en plaçant le modèle 3D dans *Google Earth* et en sauvegardant l'information dans un fichier KML séparé, tandis que l'emplacement de l'utilisateur est déterminé par GPS, relié sans fil à l'appareil portatif *Sony Vaio*. Les six degrés de liberté sont **initialisés manuellement** par l'utilisateur au lancement de l'application, et ce en plaçant le bâtiment au bon endroit dans l'image, grâce à une interaction avec l'application. Le bâtiment est ensuite maintenu à cette place à l'aide d'un suivi des caractéristiques dans l'image (approche basée vision uniquement). Ainsi, la taille, la *perspective* et l'*orientation* du bâtiment sont mises à jour dynamiquement pendant que l'utilisateur se déplace.

Nous illustrons dans la figure 2.20, un exemple des résultats obtenus par cette approche, où :

— (a) : Le bâtiment est placé manuellement dans *GooglEarth*.
— (b) : L'utilisateur place le bâtiment manuellement à l'emplacement où il doit être
— (c) : L'image augmentée.

(a) (b) (c)

FIGURE 2.20 – *Extrait de Woodward et al. (2007) - AROnSite*

2.3.5.4 City 3D-AR

Dans cette approche, Cirulis et Brigmanis (2013), les auteurs avaient pour tâche principale de trouver un moyen permettant aux experts en architecture et urbanisme de pouvoir se déplacer dans la ville et projeter des bâtiments virtuels en 3D en même temps que la ville réelle. La RA a été utilisée pour répondre à ce besoin. L'approche utilisée dans cette étude est la fusion de plusieurs capteurs (gyroscope, compas magnétique, GPS) afin d'estimer la pose de l'utilisateur et augmenter la scène en utilisant le modèle 3D d'une base de données préalablement préparée. Nous illustrons, à la figure 2.21, le schéma général de cette approche.

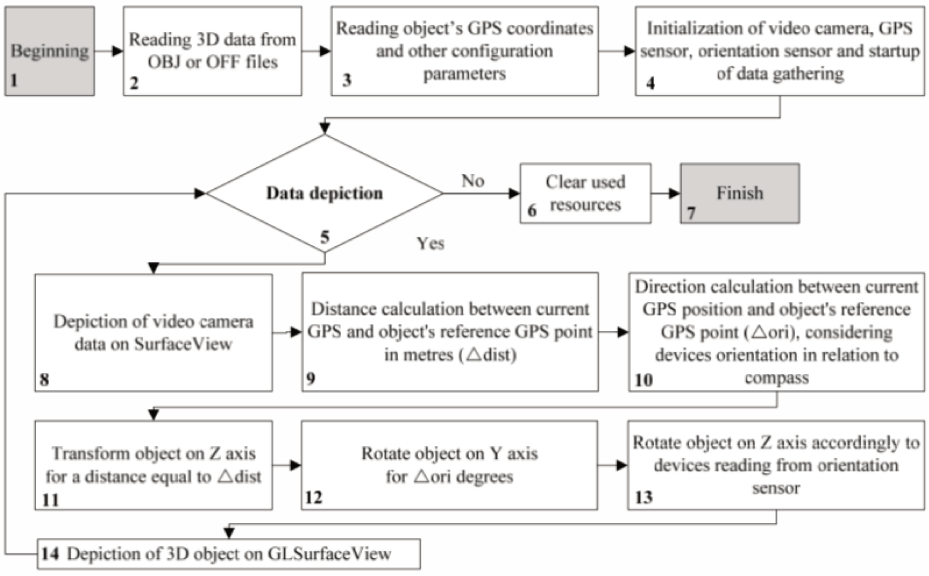
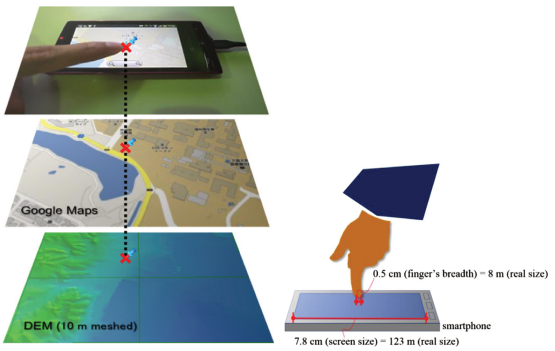


FIGURE 2.21 – *Approche proposée par Cirulis et Brigmanis (2013)*

2.3.5.5 3DMAP-AR and SOAR

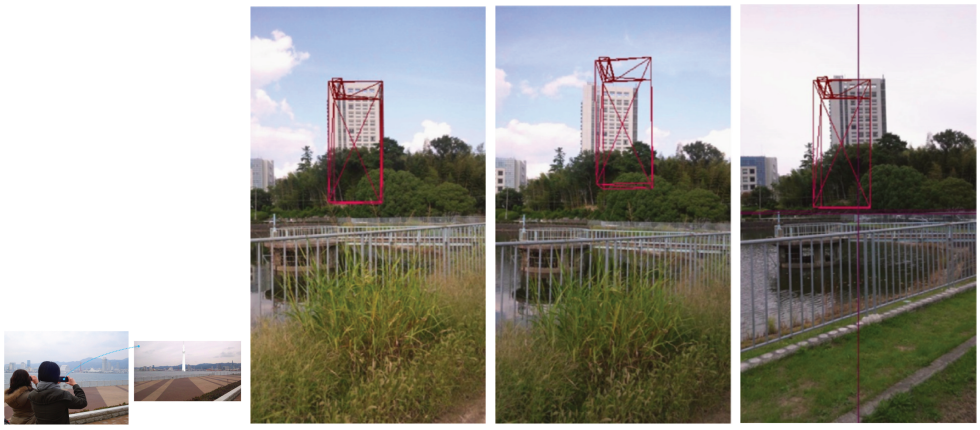
Dans Fukuda et al. (2014), les auteurs présentent un prototype d'application de RA similaire à celui de la partie 2.3.5.4. Dans cette approche, la pose est estimée de deux manières :

Pour le premier prototype, *3DMAP-AR*, la position de l'utilisateur doit être saisie par l'utilisateur grâce à une carte qui lui est affichée. L'erreur dans la saisie de cette position est ensuite approximée selon la taille et l'échelle de la carte lors de la saisie. La figure **??** illustre ce principe. La largeur des doigts est approximée et prise en compte dans l'estimation de la position. Les auteurs approximent cette erreur à moins de 8 m en termes de latitude et longitude. L'orientation est calculée en fusionnant l'ensemble des capteurs du smartphone, comme ce qui sera détaillé dans la partie 3.4.3 du chapitre 3.



(a) (b)

FIGURE 2.22 – *Carozza2014 Result de Carozza et al. (2014)*



(a) (b) (c) (d)

FIGURE 2.23 – *Extrait de Fukuda et al. (2014)*
(a) Résultat d'une image augmentée, (b) Résultat d'un recalage de l'expérience 1, (c) Résultat d'un recalage de l'expérience 2, (d) Résultat d'un recalage de l'expérience 9

Pour le deuxième prototype, *SOAR*, neuf expérimentations ont été conduites : pour chacune d'entre elles, l'un des six degrés de liberté est dynamiquement estimé grâce à l'un des instruments. Le reste des paramètres est saisi manuellement par l'utilisateur comme une variable de référence à 100% précise. La toute dernière expérimentation est semblable à ce qui est présenté dans les travaux de Cirulis et Brigmanis (2013) en partie 2.3.5.4.

Nous présentons à la figure 2.23 le résultat d'une image augmentée grâce à cette approche, ainsi que ceux des différentes expérimentations discutées ci-dessus.

2.3.5.6 Carozza et al.

Dans Carozza et al. (2014), les auteurs présentent un système de RA basé vision sans marqueur dans un environnement inconnu. De plus, le système ne repose sur aucun capteur de position (GPS ou autre). L'approche proposée est en deux phases. Une phase hors ligne, où l'environnement dans lequel évolue l'utilisateur est reconstruit obtenant ainsi un maillage 3D dense, et ce en prenant en considération les occlusions. Dans cette même phase, des points caractéristiques sont alors extraits d'images d'entraînement, associées à leurs coordonnées 3D. Dans la phase en ligne, les points caractéristiques extraits en temps réels (*SURF*) sont mis en correspondance avec la base de données de points, créée dans la phase hors ligne. Cette correspondance permet de déterminer la pose de l'utilisateur.

Cette approche est totalement basée sur la vision, et nécessite donc des performances de calcul et une reconstruction de la scène la plus fidèle possible.
Nous illustrons à la figure 2.24 quelques résultats de cet algorithme, où :

— (a) : Image à traiter ;
— (b) : Image virtuelle obtenue à partie de la pose estimée par les instruments ;
— (c) : Texture à superposer sur l'image à traiter ;
— (d) : Image augmentée.

(a) (b) (c) (d)

FIGURE 2.24 – *Carozza2014 Result de Carozza et al. (2014)*

2.4 PROBLÉMATIQUES ET OBJECTIFS DE LA THÈSE

L'objectif principal des applications de RA est l'incrustation en temps réel, stable et précise d'objets virtuels dans le rendu augmenté restitué à l'utilisateur. Différentes contraintes doivent être prises en compte pour arriver à une certaine cohérence dans le rendu final. Ces problématiques tournent essentiellement autour du recalage des objets virtuels avec les objets réels de la scène.

D'autre part, le contexte **extérieur** en RA a fait évoluer ces problématiques en terme d'estimation de la pose. En effet, la mise en œuvre d'un tel système requiert l'implémetation de trois aspects complémentaires : **la localisation**, **la visualisation** et l'**interaction**. Ces différentes problématiques sont bien détaillées dans une revue de l'état de l'art, par Zendjebil et al. (2007).

Dans ce qui suit nous commencerons par détailler les problématiques relatives à cette thèse et les objectifs que nous nous sommes fixés.

2.4.1 Problématiques

L'idée générale de la RA en contexte urbain est de faire participer les citoyens dans les projets d'urbanisme. En les intégrant et en les impliquant dans le processus de conception, on leur donne le pouvoir de décision sur les changements qui pourraient survenir sur leurs propres environnements. Depuis les années 1970, cette forme de planification a été apprivoisée en institutionnalisant la participation active des citoyens par le biais, par exemple, d'auditions publiques. Les mécanismes de participation existants sont supposés garantir une grande variété et une approche de planification multidisciplinaire. Mais bien souvent, les audiences ont lieu dans la

dernière partie du processus, transformant les participants en **observateurs** au lieu de **contributeurs**. Leur influence est basée sur les **objections** aux propositions données à la place d'en faire eux-mêmes, et permettre ainsi un dialogue constructif dès les premières étapes de la planification. La participation active de toutes les parties intéressées exige de nouvelles approches puisque les compétences et les connaissances essentielles à une participation active aux processus de planification urbaine ne sont pas nécessairement présentes dans le public.

C'est pour cela, que la RA s'est vue confier cette tâche. En effet, elle a longtemps été considérée comme un système de simulation dans lequel un objet en 3D est inclus dans l'environnement réel. Les problématiques qui y sont automatiquement associées tournent principalement autour de la **localisation**.

En effet, un système de RA, alignant réel et virtuel, s'évaluerait selon deux critères : la *précision* et la *stabilité*.
Pour cela, il faut que la pose de la caméra virtuelle soit, à chaque instant (à chaque image), précise à 100%. Cependant, le contexte dans lequel nous nous situons est assez complexe.
Tout d'abord, les mouvements de l'opérateur sont irréguliers, des fois rapides et des fois plus lents, des mouvements brusques etc.
Puis, l'environnement présente des bruits de tout type : bruits magnétiques influençant donc certaines mesures utilisées pour l'estimation de la pose (compas magnétique) ; variation de luminosité influençant donc les algorithmes basés vision ; les changements d'échelles qui doivent être pris en considération lors du développement de l'application ; etc.
Ensuite, afin de recaler un modèle réel à un modèle virtuel, il faudrait déjà disposer du modèle 3D de l'objet. Ceci n'est plus vraiment une contrainte de nos jours, vu la révolution du numérique permettant, grâce à des approches de type SLAM, de pouvoir créer un modèle 3D d'un objet en quelques minutes.
Enfin, les différents capteurs permettant d'estimer cette localisation sont eux-mêmes imprécis. Les données qu'ils nous délivrent sont entachées d'erreurs.

Pour cela, nous proposons un système hybride combinant capteurs et images. Le système commence par faire une fusion des données de ces différents capteurs, nous permettant de localiser l'utilisateur, et en d'autres termes, de trouver sa pose (position et orientation). Cette estimation sera corrigée par une approche basée vision, utilisant des points caractéristiques très particuliers : le skyline.

2.4.2 Objectifs de la thèse

La question à laquelle nous voulons répondre à travers cette thèse est :
Est-ce que le skyline peut jouer le rôle d'un marqueur pour la RA mobile en contexte urbain ?
Pour cela, nous devons trouver le moyen de faire du skyline un correcteur de pose.

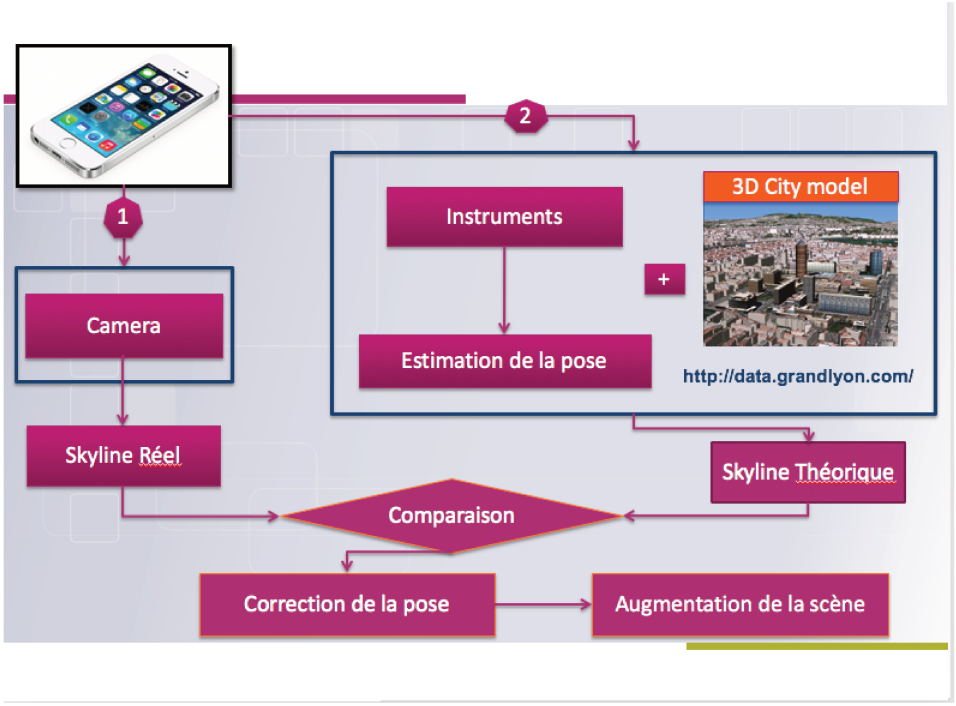
En effet, l'objectif de cette thèse se traduit par la mise au point d'une nouvelle approche en RA sur mobile. Cette approche, qu'on qualifie d'*hybride*, permet d'utiliser l'**image** pour corriger la pose estimée par des **capteurs**, et ce dans un environnement extérieur (la ville).

Le développement de cette méthode soulève plusieurs problématiques à savoir :

— La combinaison des différentes données hétérogènes issues des différents capteurs doit être faite de sorte à pouvoir estimer une pose à tout instant de la vie de l'application de RA ;
— Chaque capteur (accéléromètre, gyroscope, baromètre, etc.), y compris la caméra, et y compris les données 3D de la ville, ont des repères propres. Le repère de la caméra n'est pas celui des objets 3D, qui n'est pas celui de l'accéléromètre, et qui n'est pas non plus celui de l'image projetée (avec une adaptation des différentes unités de mesure à chaque fois) ;
— Une méthode d'extraction du skyline est à développer, permettant d'extraire ce marqueur naturel à partie des images réelles et virtuelles
— Une méthode pour l'appariement 3D/2D est à développer, et ce passant par une méthode de *matching* de skylines

L'approche que nous proposons sera évaluée, en chapitre 5, et ce selon deux aspects : la précision et la stabilité.

Nous illustrons, par soucis de clarté, le processus global et complet de ce que nous proposons tout au long de la suite (voir figure **??**).



Conclusion

Dans le présent chapitre, nous venons de proposer un tour d'horizon autour de la RA, de ses applications, des différentes méthodes et algorithmes existants ainsi que des problématiques y étant associées.
En effet, après avoir défini la réalité augmentée (RA) et ses principes de fonctionnement, nous avons présenté les différentes approches et algorithmes existant aujourd'hui. Puis, nous avons introduit les différentes technologies et les différents outils d'un système de RA. Ensuite, nous avons présenté une revue sur les divers domaines d'application possibles de la RA qui deviennent aujourd'hui de plus en plus répandus. Pour cela, nous avons proposé une étude des principaux contextes d'application, que ce soit en tant que système d'assistance médical ou bien en tant que système de visualisation de la ville. Cet intérêt ne cesse de grandir essentiellement dû aux développements des technologies telles que la téléphonie qui commence à démocratiser le concept de la RA en proposant des applications pour le grand public.

À partir de cette étude, nous avons pu cerner les différentes problématiques qui entourent la conception et la mise en œuvre d'un système de RA. Comme nous l'avons vu, ces différentes problématiques tournent essentiellement sur l'estimation de la pose, ou aussi appelée localisation.

Afin de répondre à cette problématique d'estimation de pose, cette thèse propose tout d'abord un processus dans lequel la pose est estimée en se basant sur la fusion des données de plusieurs capteurs, à savoir : l'accéléromètre et le gyroscope pour le calcul de deux degrés de liberté en rotation, le compas magnétique pour en calculer le

dernier et le GPS et le baromètre pour déterminer les trois derniers degrés de liberté en translation. Ce processus achevé, nous proposons une technique d'extraction et de matching (appariement) de skyline pour corriger cette pose estimée.

Dans le chapitre qui va suivre, nous allons présenter dans un premier temps un tour d'horizon sur les méthodes existantes en fusion de données capteurs. Nous présenterons ensuite les différents instruments sur lesquels nous basons nos travaux. Dans un second temps, nous allons présenter l'approche que nous proposons dans le cadre de nos travaux.

3 Réalité augmentée multi-capteurs

Plan du chapitre

Une bonne estimation de la pose est un enjeu crucial pour les applications de RA. En effet, l'estimation de la pose peut se faire selon deux modalités : une estimation **basée sur la vision** et une estimation **basée sur les capteurs**. L'estimation se basant sur les capteurs combine plusieurs types d'instruments afin de trouver les **six degrés de liberté** du dispositif dans le repère de la scène visualisée. L'estimation basée vision passe par une étape d'**appariement entre un monde réel et virtuel**, en général des appariements 3D/2D. Bien évidemment, pour la réalité augmentée en milieu intérieur, où les conditions d'acquisition (luminosité, d'occultation, etc.) sont contrôlées, les approches basées vision sont privilégiées. Il existe aussi des méthodes basées capteurs en intérieur qui donnent de bons résultats comparables à ceux basés vision (exemple : *iBeacon*). En revanche, dès que nous passons en milieu extérieur, ces conditions sont moins contrôlables et leur influence est grande sur les performances des systèmes basés vision proposés.

Dans cette thèse, nous nous plaçons dans un environnement extérieur urbain non contrôlé : la ville. Nous proposons un système multi-capteurs où nous combinons toutes les données issues des instruments embarqués dans le smartphone pour mettre sur pied une vraie centrale inertielle. Cette dernière nous permet d'estimer la pose de l'utilisateur dans le repère du monde. Suite à cela, nous combinons le résultat obtenu avec l'image acquise par le smartphone pour pallier les imprécisions des mesures et gagner en stabilité et précision.

Dans le présent chapitre, nous dressons un bref état de l'art sur les systèmes actuels de localisation multi-capteurs sur plateformes mobiles. Nous présenterons ensuite notre approche qui se décline en plusieurs parties.
Tout d'abord, nous partirons d'une revue de la littérature des méthodes de fusion de données multi-capteurs, afin d'avoir une idée d'ensemble des méthodes qui traitent de ce problème. Puis, nous présenterons les différents capteurs disponibles dans le dispositif de tests que nous choisissons (le smartphone) ainsi que les différentes caractéristiques de chacun d'entre eux. Ensuite, nous détaillerons l'approche que nous proposons dans le cadre de ces travaux. Enfin, nous présenterons nos résultats et les expérimentations que nous avons menées.

3.1 INTRODUCTION

L'idée de fusionner les données de plusieurs capteurs n'est pas nouvelle. Ces méthodes existent depuis les années 90, et sont employées dans plusieurs domaines : la navigation des robots autonomes en environnement extérieur inconnu ou dans l'estimation de la pose d'un utilisateur dans des environnements confinés. Toutes ces méthodes d'estimation de pose, ou de localisation, tournent autour de deux grandes approches : les approches de **localisation basées vision** et les approches de **localisation basées capteurs**. D'autres approches sont dites par ***suppléance de données***. En d'autres termes, c'est une combinaison d'une **approche basée vision** et d'une autre basée **capteurs**, traitée par exemple en Aron et al. (2007) ou en Zendjebil (2010). Ces différentes stratégies n'ont cependant été testées que dans des environnements confinés et contrôlés à petite ou moyenne échelles. Nous proposons ci-après une revue de la littérature pour ces différentes méthodes.

3.2 REVUE DE LA LITTÉRATURE : FUSION DES DONNÉES MULTI-CAPTEURS

Plusieurs travaux existent dans la littérature traitant de la problématique de fusion des données. En effet, le cheminement commun à tous ces travaux, y compris les nôtres, est qu'une première approximation de la pose de l'utilisateur est fournie par les différents capteurs de localisation. Ensuite, cette pose sera ajustée et affinée en se basant sur des algorithmes basés vision.

Dans ce qui suit, nous proposons une revue de ces principales méthodes, comme les travaux de You et al. (1999), Hol et al. (2006), Bleser (2009), G. Reitmayr (2006) et Zendjebil (2010).
D'autres méthodes, Broll et al. (2004), Rune Nielsen (2005), Woodward et al. (2007), Honkamaa (2007), Cirulis et Brigmanis (2013), Fukuda et al. (2014) et Carozza et al. (2014) ont déjà été traitées dans les parties 2.3.5.1 à **??** du chapitre 2.

3.2.1 You et al.

Les travaux de You et al. (1999) sont comparables aux nôtres selon deux modalités : l'environnement de test étant un environnement extérieur urbain, et l'estimation de la pose se faisant, d'abord, en utilisant uniquement les données inertielles sans correction visuelle, puis, en utilisant un algorithme basé vision. Néanmoins, les capteurs inertiels utilisés pour l'estimation de la pose sont un compas magnétique associé à trois gyroscopes.

En effet, dans ces travaux, l'approche se base sur un capteur hybride. La figure 3.1 présente le schéma général du système proposé.

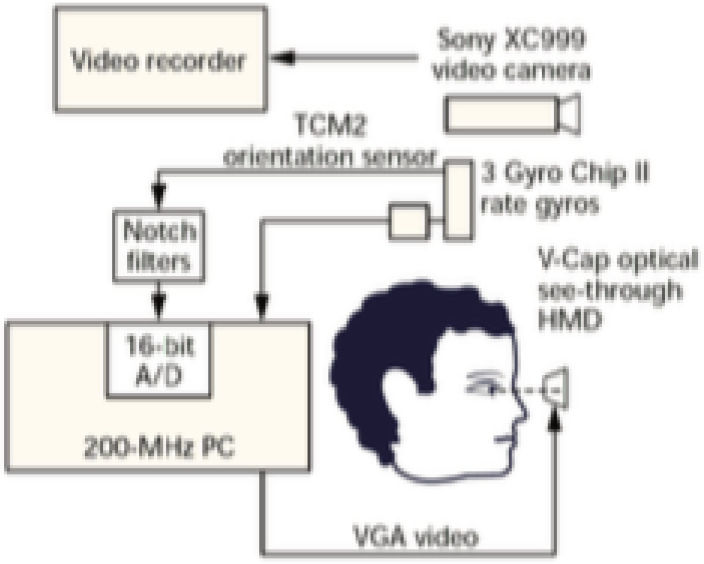


FIGURE 3.1 – *Système multi-capteurs proposé par You et al. (1999)*

Tout d'abord, une fusion des données des trois gyroscopes et du compas magnétique permettent d'estimer une orientation particulière. Due aux imprécisions des instruments, cette estimation est entachée d'erreurs.

Puis, de cette estimation, le mouvement en 2D des points de l'image est approximé. Au travers de ces mesures, une approximation permet de dire dans quel sens un point x de l'image a évolué : vers la droite / gauche, vers le bas / haut, en diagonale, etc. Cette information permet de limiter l'espace de recherche dans lequel X sera recherché, ce qui permet de créer des zones de recherche.

Ensuite, le suivi visuel entre en action : il permet de rechercher dans ces zones les points qui avaient été détectés dans l'image précédente et qu'il faut retrouver. Les auteurs proposent de baser cette recherche locale sur une méthode de flot optique, en passant par une minimisation en moindres carrés afin de sélectionner la meilleure estimation du mouvement pour chacun des points d'intérêt.

Enfin, ce mouvement en 2D est converti en une orientation 3D permettant de corriger les erreurs gyroscopiques. Cette correction, appelée *"dérive"*, n'est autre que la différence entre la vitesse obtenue par les données gyroscopiques et la vitesse calculée à partir des points suivis dans l'image. Nous illustrons à la figure 3.2 les résultats obtenus pour cette approche.

Le système de You et al. (1999) diffère de celui qui est présenté dans cette thèse

FIGURE 3.2 – *Extrait de You et al. (1999)*

selon deux modalités. En effet, nous proposons d'extraire les points du skyline à la volée. Dans leurs travaux, You et al. (1999), proposent de sélectionner l'ensemble de points à suivre. Grâce aux différentes orientations estimées par le système (trois gyroscopes, compas magnétique), l'image réelle est augmentée par la re-projection des points précédemment sélectionnés. La figure 3.2 illusre cette re-projection. Les points de couleur bleue sont les points re-projetés selon les données inertielles uniquement, et les points de couleur rouge illustrant les corrections basées vision. Le protocole d'évaluation proposé, que nous utiliserons aussi en chapitre 5, se base sur le calcul de l'erreur moyenne entre la position 2D des points projetés (bleu ou rouge) et leurs positions réelles dans l'image. Cette évaluation n'est possible que dans le cas où les points à suivre sont très éloignés de la position de l'utilisateur. En effet, un changement dans la position de prise de point de vue (la position de l'utilisateur dans le repère du monde) peut être négligé et approximé par une légère rotation de la caméra. En d'autres termes, si l'utilisateur est à une position x et qu'il se déplace de 10 mètres vers la droite, cela peut être approximé par le fait que l'utilisateur peut rester à sa position x en appliquant une légère rotation de la caméra vers la droite. L'approche hybride (combinaison capteurs et vision : points rouges) donne de meilleurs résultats (de l'ordre de 4.27 pixels) par rapport à l'approche basée capteurs (points bleus).

3.2.2 Bleser et al.

Dans sa thèse, Bleser (2009) propose un système de localisation hybride basé sur une caméra et une centrale inertielle. Bien que les tests de ces travaux aient été réalisés dans un envrionnement à petite échelle (contrôlé en intérieur), ce travail méritait d'être cité, vu le contexte de fusion des données de capteurs avec une correspondance 2D/3D de points dans l'image.
Les auteurs présentent un système de RA pour des environnements connus et inconnus. Les stratégies utilisées sont : la fusion visuelle et inertielle des capteurs, le

découplage de l'estimation de la pose et de la structure et l'unification de la vision par ordinateur et des techniques de filtrage récursif.

En effet, comme pour l'ensemble des méthodes citées auparavant, la pose est calculée en fusionnant des correspondances 2D/3D avec les données fournies par la centrale inertielle. L'avantage de cette méthode est l'utilisation d'un filtre de *Kalman* étendu dans la boucle, comme ce qui est proposé dans le schéma général, extrait de Bleser (2009) à la figure 3.3.

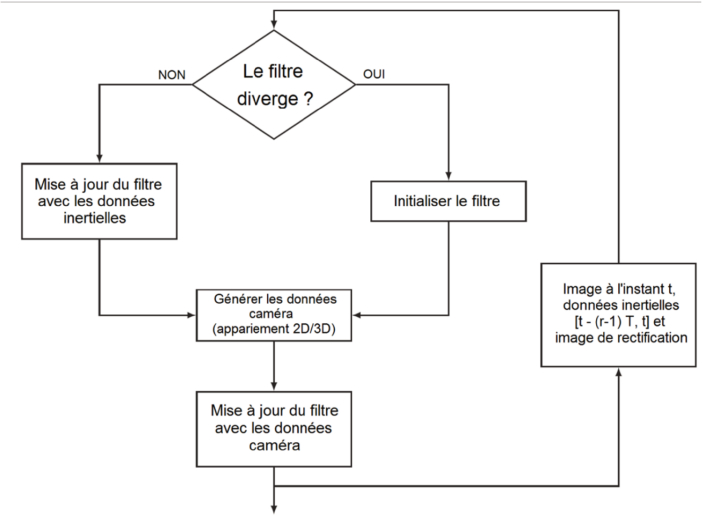


FIGURE 3.3 – *Schéma général de l'approche de Bleser (2009)*

Ce schéma explique bien les différents cas qui se présentent. À chaque instant t, le capteur hybride reçoit en entrée les données de la caméra (image) munies des données inertielles acquises sur un intervalle T. Avant de mettre à jour le filtre, un test de convergence est effectué. Ce test est basé, ou bien sur l'initialisation, ou alors sur le processus de la centrale inertielle. Pour ce faire, le système utilise les critères suivants :

— L'écart entre la prédiction et les données réelles suit une loi normale centrée ;
— La norme de Frobenius de la matrice de covariance doit être dans un intervalle défini ;
— La norme des quaternions est égale à 1.

Ces différents critères déterminent donc la divergence du filtre. Deux cas se présentent alors :

— **Si oui :** Le système procède à une initialisation du filtre faisant correspondre des points 2D/3D selon trois modes :
  — Méthode manuelle : en fixant une pose pour le capteur hybride
  — Méthode semi-automatique : en mixant une pose définie manuellement et l'orientation récupérée de la centrale inertielle

— Méthode automatique : en recalant (avec un histogramme de couleur) l'image courante sur des images de références dont les poses sont connues.
— **Si non :** Le filtre combine la pose précédente avec les données inertielles actuelle pour en prédire une nouvelle et générer donc des appariements 2D/3D. Les résultats sont injectés dans le filtre pour être mis à jour.

Comme ce qui est détaillé dans la partie **??** du chapitre 5, le critère de performance utilisé dans cette méthode est la ***distance euclidienne*** entre la position ***prédite des points caractéristiques*** et leurs ***positions réelles***. Les auteurs ont mené différentes expérimentations à différents niveaux d'échelles.

À petite échelle, le modèle d'accélération présente une erreur moyenne de l'ordre de 0.93 pixels avec un écart type égal à 0.62 pixels au lieu de 3.83 pixels (σ = 4.03 pixels) pour le modèle gyroscopique et 3.82 pixels (σ = 4.02 pixels) pour le modèle de gravité.

À grande échelle, comme ce qui est proposé dans cette thèse, les auteurs présentent une erreur moyenne qui ne dépasse pas 1.42 pixels avec un écart type de l'ordre de 1.46 pixels. Les auteurs jugent les résultats obtenus robustes, et ce en environnement à petite et grande échelles et dans différentes conditions de luminosité et de mouvements. Cependant, si l'appariement 2D/3D n'est pas réussi en raison des occultations, le système ne reste fonctionnel que seulement pour un court instant.

3.2.3 Reitmayr et al.

Il nous a paru important de citer les travaux de G. Reitmayr (2006), pionniers dans la RA hybride. En effet, ces recherches se rapprochent des nôtres, avec néanmoins plusieurs différences. Nous commençons par illustrer à la figure 3.4, le système qui y est proposé, combinant vision et mesures gyroscopiques.

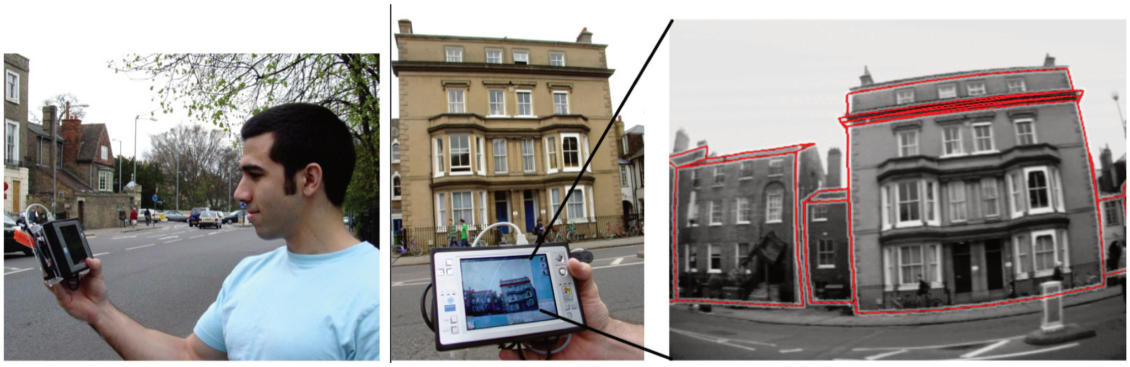


FIGURE 3.4 – *Système proposé par G. Reitmayr (2006)*

Dans cette approche, la méthode repose sur l'utilisation de plusieurs capteurs, avec des mesures gyroscopiques, de gravité et de champs magnétique. De plus, le système utilise une méthode basée vision.

En effet, le système basé vision permet d'obtenir une localisation précise, et ce grâce à une méthode de détection de contours, détaillée dans leurs travaux en Klein et Drummond (2003). Les meures récupérées des autres instruments permettent d'éviter l'effet de dérive. Et similairement aux travaux de Bleser (2009), la fusion des données repose sur l'utilisation d'un filtre de ***Kalman** étendu (EKF)* avec l'hypothèse que la **vitesse du dispositif est considérée constante**. Cette hypothèse permet de récupérer, à partir du vecteur d'état, les paramètres de pose et de vitesse. Ainsi, les données inertielles sont traitées afin d'estimer les paramètres de rotation. Ces paramètres permettent de proposer la nouvelle pose de l'utilisateur, et ainsi de faire le suivi en temps réel.

Deux cas se présentent, selon l'état du suivi basé vision :

1. Si le suivi échoue : il faudra réinitialiser le filtre de *Kalman*, en faisant un appariement entre l'image actuelle et un ensemble d'images de référence dont les poses connues a priori. Cet appariement peut :
   (a) Réussir : Une mise à jour de la pose est effectuée en calculant le mouvement permettant de passer de l'image courante à l'image de référence trouvée ;
   (b) Échouer : la pose n'est pas fusionnée et le filtre est réinitialisé ;
2. Sinon, le suivi restera toujours basé sur la vision, et le modèle de vitesse est réinitialisé à chaque fois.

Les auteurs présentent les limites de la méthode, où à la fin de la séquence, si le suivi échoue (cas *1.2* ci-dessus), le système diverge et il est difficile d'avoir une bonne ré-initialisation du filtre. Il faudrait donc une initialisation de tout le système, ce qui doit se faire manuellement par l'utilisateur. En effet, l'utilisateur est contraint de recaler manuellement, en alignant la vue actuelle à une vue prédéfinie qui lui est proposée, comme cela est également proposé dans les travaux de la partie 2.3.5.3.2 du chapitre 2. La robustesse de cette méthode à l'occultation dépend essentiellement de la vue postérieure à celle où l'occultation a été appliquée, qui doit ressembler à une des vues de référence afin que le système soit réinitialisé.

Le système a été testé en utilisant des modèles 3D constitués de surfaces planaires texturées. L'erreur de position présente un écart type de $(\sigma_x, \sigma_y, \sigma_z) = (0.0979m, 0.1577m, 0.1463m)$.

Une amélioration de cette méthode a été proposée par les mêmes auteurs en G et T. W (2007). Cette proposition vient pallier les limitations de la phase d'initialisation de la version antérieure de l'approche, et afin d'éviter la procédure manuelle précédemment proposée. En effet, dans cette phase d'initialisation, nommée *phase de récupération* dans l'article, l'image courante doit être appariée avec l'ensemble des

images de référence, ce qui est très fastidieux. De plus, il est quasiment impossible d'avoir une image de référence bien distribuée dans tout l'environnement de test. C'est pour cela, qu'un GPS sera couplé au système afin d'avoir une position 2D initialisant le suivi. Dans cette initialisation, la méthode propose de décrire une zone de recherche sous forme d'ellipse dont le centre est cette position GPS. Dans cette ellipse, plusieurs positions aberrantes sont écartées vu leurs intersections avec les bâtiments. Pour chacun des points restants, un rendu virtuel à partir du modèle 3D est réalisé et l'image résultante est comparée à l'image courante grâce à l'algorithme de détection de contour. L'image avec la meilleure comparaison (appariement) est retenue, et sa position utilisée pour l'initialisation de l'algorithme.

Ces travaux sont intéressants mais présentent néanmoins quelques inconvénients. La différence entre ce que nous proposons avec cette méthode, ou encore celle de Bleser (2009), réside dans le fait que nous n'utilisons pas d'images de référence pour réinitialiser le système. La localisation hybride que nous proposons utilise toujours les données capteurs comme pose initiale, même si elle zn présente une qui est assez loin de la pose réelle. Une telle procédure d'initialisation est trop lourde en termes de temps de calcul pour être testée sur des plateformes mobiles. De plus, dans G et T. W (2007), la translation est négligée au sein du processus de fusion. Ceci n'est pas très problématique si la zone de test est petites ou si la distance entre les différents tests est négligeable, car ce mouvement en translation peut être approximé à une rotation pure.

3.2.4 Zendjebil et al.

Dans sa thèse, Zendjebil (2010) propose une méthode de localisation hybride pour la RA en milieu extérieur. Tout d'abord, une approche de localisation basée sur une **fusion de données multi-capteurs**, puis une alternative par **suppléance de données**.

En effet, l'idée poursuivie est de commencer d'abord par fusionner toutes les données fournies par les capteurs, afin d'estimer la pose de l'utilisateur. L'approche par suppléance se traduit par le fait que le système privilégie, selon la précision des données, un des traitements parmi ceux proposés dans le système. En d'autres termes, si la vision est opérationnelle et capable de fournir des mesures précises, le système accorde sa confiance et devient donc basé uniquement vision. Pour cela, le système est muni de critères et conditions permettant de détecter les défaillances du système de vision. Sinon, le système bascule vers la deuxième solution, à savoir l'utilisation des capteurs pour l'augmentation de la scène.
Le système est doté d'une caméra qui sera le capteur principal, d'une centrale inertielle pour l'estimation des orientations et d'un GPS pour l'estimation de la position.

La caméra et la centrale inertielle sont rigidement liées et le GPS porté par l'utilisateur. Cette approche est détaillée à la figure 3.5.

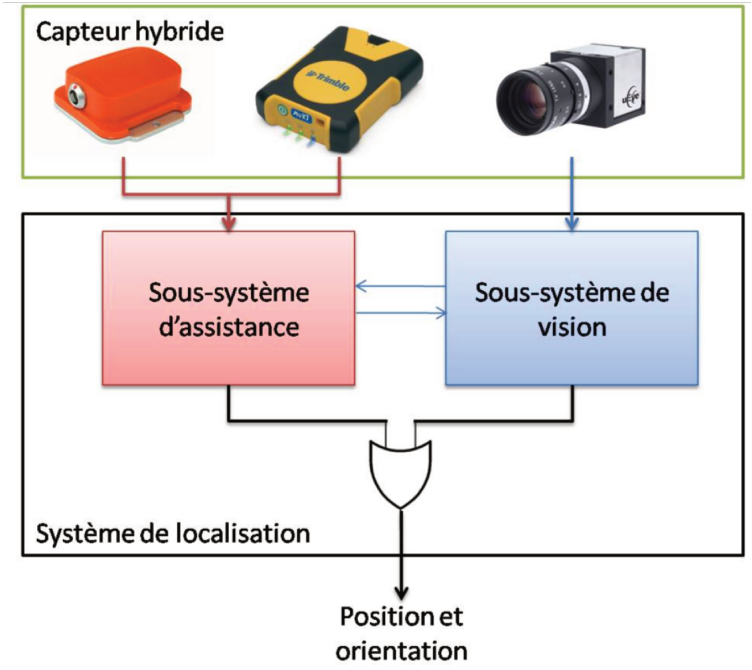


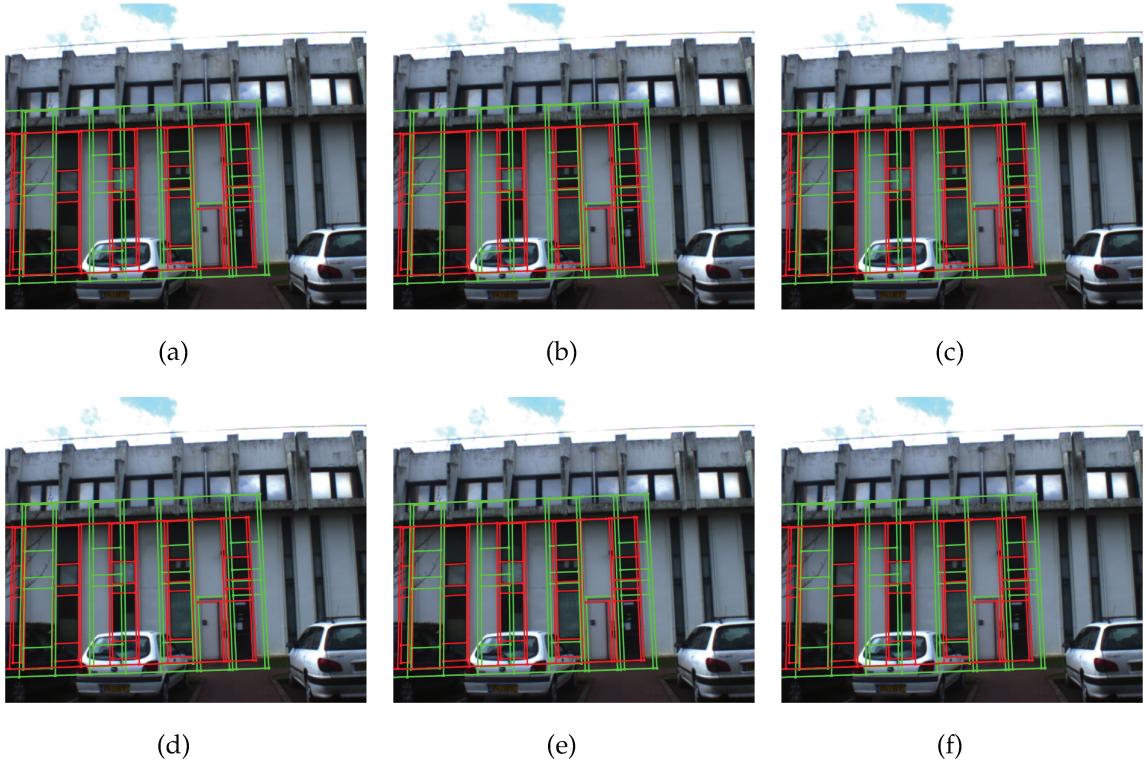
FIGURE 3.5 – *Approche par suppléance de données de Zendjebil (2010)*

Ce qui nous a le plus intéressé dans cette approche, c'est que tous les capteurs interagissent ensemble, un peu comme ce que nous proposons dans cette thèse.

En effet, les données fournies par certains capteurs sont utilisées pour valider les estimations que d'autres donnent. Dans l'autre sens, les mesures fournies par une partie des capteurs peuvent être corrigées en utilisant celles fournies par d'autres. De ce fait, et comme le montre la figure 3.5, le système est considéré muni de deux systèmes de localisation, non pas indépendants, mais qui interagissent en continu l'un avec l'autre.

Nous illustrons à la figure 3.6 quelques résultats de cette approche. Les auteurs présentent ces résultats de recalage réalisés à partir des deux systèmes de visualisation séparés : les données fournies par GPS, et les données d'orientation par vision, et ce à différentes positions et points de vue en projetant le modèle filaire.

En rouge les poses fournies par la vision et en vert celles obtenues grâce au GPS. Le décalage entre les résultats basés vision et ceux basés GPS ne sont pas très éloignés. L'évaluation du système montre une grande erreur, de l'ordre de 64.7935 pixels avec un écart type égale à 38.7278 pixels.



(a) (b) (c)

(d) (e) (f)

FIGURE 3.6 – *Résultat de recalage de Zendjebil (2010)*

3.3 CAPTEURS DE MOUVEMENT

Dans cette partie, nous allons présenter l'ensemble des capteurs de mouvements utilisés dans le cadre de nos travaux. En effet, dans les méthodes existantes, citées dans la partie 3.2, les capteurs utilisés sont parfois extrêmement précis (DGPS), séparés ou encore dupliqués (trois gyroscopes, etc.). Dans cette thèse, nous proposons d'utiliser les instruments embarqués dans le smartphone. Grâce à ces différents instruments, nous arrivons à créer une centrale inertielle.

En effet, dans le smartphone sont embarqués plusieurs capteurs et instruments. Dans ce qui suit nous les présentons. Nous commençons tout d'abord, dans la partie 3.3.1, par le capteur de localisation : le GPS . Puis, nous présenterons les différents capteurs inertiels utilisés qui nous permettent de calculer, en temps réel, l'évolution des vecteurs vitesses et son attitude : le roulis, le tangage et le lacet (en anglais *Roll, Pitch et Yaw*). Ces différents angles et vitesse angulaires seront estimées dans la partie 3.4.3. Enfin, nous introduirons dans la partie 3.3.5 le baromètre qui nous permet d'estimer une altitude relative. Voyons plus en détail le principe de fonctionnement de chacun de ces capteurs, afin de pouvoir les utiliser dans la suite du chapitre.

3.3.1 GPS

Bien que le système de géolocalisation européen *Galileo* ait été lancé en 2017, il n'est cependant pas encore opérationnel à 100% et n'est toujours pas intégré dans les appareils grand public, de type smartphone ou GPS portables. De ce fait, les smartphones se basent aujourd'hui sur le *GPS*, un système conçu par le département américain de la défense, permettant d'estimer une position globale dans le repère du monde, le système géodésique WGS84. Ce système se base sur un positionnement par satellites.

En effet, Le système fonctionne grâce au principe de triangulation des signaux reçus de ses 24 satellites placés en orbite terrestre et tournant autour de la terre en 24 heures, émettant continuellement un signal horodaté. La position est déduite à partir de la différence de temps entre les instants d'émission et de réception du signal. Une configuration particulière de ces satellites permet d'avoir au minimum 5 à 8 satellites visibles à partir de n'importe quel point sur la terre, à n'importe quel moment du jour ou de la nuit. La figure 3.7 illustre la disposition de ces satellites autour de l'orbite terrestre.

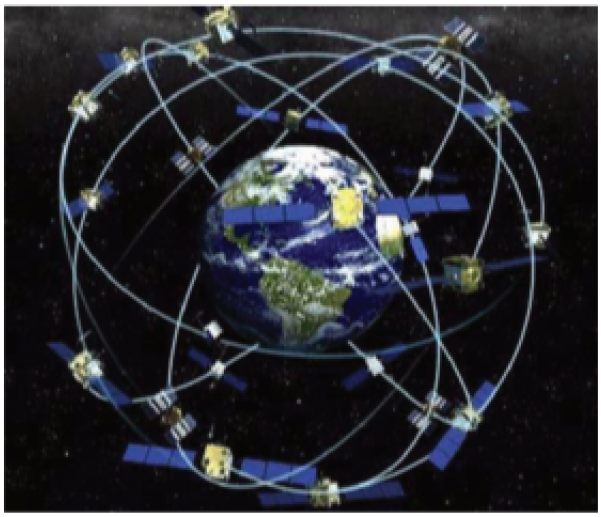


FIGURE 3.7 – *Satellite GPS*

Cependant, dans un contexte urbain comme le nôtre, il nous arrive parfois de nous trouver dans des situations appelées *"canyons urbains"*, où des bâtiments très *hauts* cachent et occultent les signaux reçus par ces satellites, et influent donc sur la qualité du positionnement et de la localisation. En effet, un bon positionnement est déterminé à partir d'au moins 3 satellites visibles (triangulation). Or, comme l'horloge du récepteur n'est pas précise et possède un décalage inconnu en plus des dérives des horloges atomiques des satellites, il faut au moins 4 satellites afin de déterminer la position de l'antenne réceptrice ainsi que le décalage de l'horloge.

La précision de ces systèmes est de l'ordre de 10 mètres pour les GPS grand pu-

blic, et peut atteindre le mètre pour les GPS professionnels. Par ailleurs, il existe une variété de GPS qui possèdent une précision de l'ordre du centimètre. Il s'agit des systèmes DGPS pour *Differential GPS*. Généralement, la localisation en altitude est moins précise que la localisation en latitude et en longitude. De plus, différentes causes peuvent altérer le signal GPS. Pour plus d'information, ces causes sont décrites en annexe **??**.

Dans cette étude, nous nous basons sur le GPS intégré dans le smartphone. Notre précision est donc de l'odre de 10 mètres (en longitude et latitude). En ce qui concerne l'altitude, nous proposons de combiner les mesures fournies par le GPS avec un autre capteur : le baromètre, qui sera détaillé dans la partie 3.3.5.

3.3.2 Accéléromètre

L'accéléromètre permet de mesurer les accélérations linéaires de l'objet auquel il est relié. On distingue deux grandes familles d'accéléromètres : les accéléromètres non asservis et les accéléromètres à asservissement.

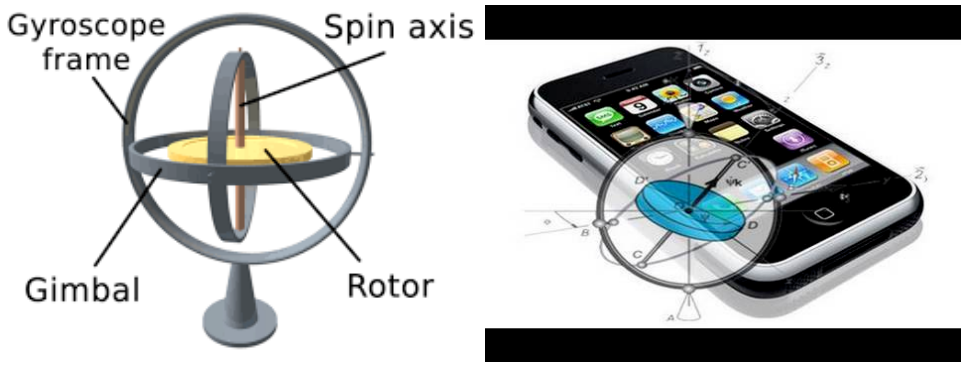
Brièvement, pour les capteurs de type non asservi (boucle ouverte), l'accélération est mesurée à partir de son image "directe", alors que pour les accéléromètres à asservissement, l'accélération est mesurée à la sortie d'une boucle de contre-réaction (asservissement) comportant un correcteur de type P.I.

La position dans le repère du monde peut être estimée à partir de ces données en effectuant une double intégration. La localisation est relative. Ceci implique la connaissance a priori d'une position de référence. Ce type de capteur est rapide, mais le processus d'intégration tend à accumuler les erreurs au cours du temps ce qui produit une dérive (drift) entre la position estimée et la position réelle.

3.3.3 Gyroscope

Les gyroscopes, dans le cas général, sont des systèmes qui se basent sur le principe physique de *conservation des moments angulaires*, supposant que tout corps en mouvement de rotation rapide, et en l'absence de moments externes, conserve ses moments angulaires. Mécaniquement, un gyroscope est une sorte de roue où l'orientation (du corps *sur / dans* lequel il est embarqué) est calculée à partir des angles reportés sur les encodeurs rationnels (3.8).

Le gyroscope n'a pas besoin d'un repère externe : les axes de rotation de la roue représentent ce repère de référence. Les gyroscopes ne fournissent pas les orientations de l'objet, comme le ferait un inclinomètre, mais plutôt les vitesses angulaires. En les intégrant, ces vitesses nous permettent de déduire les orientations. Toutefois, ces capteurs présentent l'inconvénient d'accumuler les erreurs au fil du temps.



(a) (b)

FIGURE 3.8 – *Gyroscope Mecanique embarqué dans un Smartphone*

3.3.4 Compas magnétique

C'est un capteur qui permet de mesurer le champ magnétique ambiant. Il est composé d'un liquide contenant des ions, placés dans une bobine. Ces ions oscillent sous l'effet du champ magnétique ambiant. En injectant un courant électrique intense dans la bobine, un champ magnétique supérieur au champ qu'on veut mesurer est créé. Ceci force les ions à s'orienter selon ce nouveau champ. Une fois le courant coupé brusquement, les ions reviennent à leur place initiale en oscillant autour de cette position. Cette oscillation crée à son tour un courant dans la bobine dont la période est fonction du champ magnétique mesuré. Le magnétomètre peut s'utiliser pour mesurer le champ émis d'une base fixe ou le champ magnétique terrestre. Partant de là, il est alors possible de faire agir le capteur comme une boussole afin d'indiquer le nord magnétique. Le capteur fournit alors son orientation par rapport au Nord et, comme il ne nécessite pas de base magnétique, a un rayon d'action à l'échelle de la planète, ce qui en fait un capteur privilégié pour opérer en extérieur. L'inconvénient de ce type de système reste toutefois sa sensibilité aux perturbations électromagnétiques qui peuvent induire des erreurs angulaires dans les mesures de l'orientation.

Les capteurs inertiels mécaniques traditionnels demeurent encombrants. L'avènement des nanotechnologies a permis de miniaturiser ces dispositifs appelés MEMS (Micro-Electro- Mechanical Systems). La méthode consiste à intégrer des éléments mécaniques, capteurs, actionneurs et leur électronique sur un substrat commun en silicium par le biais des technologies de micro-fabrication. L'électronique est fabriquée en utilisant les méthodes classiques de production des circuits intégrés tandis que les éléments micromécaniques sont obtenus par le biais de processus compatibles qui vont découper et retirer des parties du silicium pour constituer les micromécanismes. Dans le cas des accéléromètres, ceci a permis de réduire les coûts de fabrication par 10, tout en miniaturisant et en augmentant la fiabilité de ces derniers.

3.3.5 Baromètre

Dans la plupart des nouveaux dispositifs mobiles aujourd'hui (*iPhone 6S* et plus, *Samsung S8* et plus), de nouveaux capteurs sont embarqués, dont le baromètre. Une des applications possibles avec ce nouveau capteur, est la possibilité d'avertir l'utilisateur d'un changement brutal de temps (conditions météorologiques), et ce à travers un changement brusque de température déduit par un changement de la pression atmosphérique. En effet, le baromètre permet de mesurer, après calibration, la pression atmosphérique de l'environnement dans lequel se trouve le dispositif. D'autres applications, en intérieur, permettent à l'utilisateur de mieux se localiser dans un centre commercial et de détecter l'étage dans lequel il se trouve.

Ce qui nous intéresse ici est de pouvoir calculer l'*altitude relative* du dispositif et non pas une *altitude absolue*. En effet, pour le calcul de l'altitude, la plupart des GPS reposent sur des *API* qui permettent de fournir une altitude en un point donné. Ces API utilisent des cartes dans lesquelles l'altitude précise est donnée pour certains points de référence. L'altitude d'un point donné est calculée en utilisant une interpolation avec les points qui lui sont les plus proches, et qui sera appelée *altitude absolue*. Cependant, avec un baromètre, c'est l'altitude relative qui est calculée, ce qui signifie qu'on aura une meilleure précision lors de l'étape d'estimation de la pose (voir partie 3.4.3).

3.4 Approche proposée

Le système de RA en extérieur que nous proposons est un système mobile. Comme dispositif nous proposons un smartphone, faisant donc partie de la famille des dispositifs portés à la main, ce qui implique que les mouvements auxquels est assujetti le dispositif ne peuvent être supposés réguliers.

Nous orientons donc nos travaux vers une approche en deux temps : d'abord, une fusion de données permettant d'avoir une première approximation de la pose, ensuite une approche par suppléance de données, où la pose estimée est corrigée par un système basé vision. En effet, les instruments à eux seuls permettent d'avoir une bonne estimation de la position et de l'orientation. Le problème se pose lorsque ces capteurs fournissent des données entachées d'erreurs dues aux différentes perturbations, spécialement dans un environnement extérieur (champs magnétique, dérives de la pose, etc.). Pour cela, nous faisons intervenir l'image et nos algorithmes basés vision.

Dans ce qui suit, nous détaillons la première partie d'estimation de la pose basée instruments. De plus, afin de préparer le terrain pour le chapitres 5, nous présentons

le pipeline complet de l'approche, permettant, grâce à la pose estimée par fusion des données des capteurs, de placer une caméra virtuelle dans le modèle 3D de la ville, et de générer une vue théorique de ce que l'utilisateur devrait voir s'il avait été bien positionné. Nous présenterons donc, dans l'ordre :

1. **La géométrie de la caméra**, permettant d'avoir un modèle de la caméra qui se rapproche le plus du modèle de la caméra réelle captant l'environnement dans lequel évolue l'utilisateur. Cette étape sera détaillée dans la partie 3.4.1 du présent chapitre ;
2. **Les données** utilisées dans le cadre de cette thèse ;
3. L'estimation de **la pose** grâce aux différentes données récupérées des instruments ;
4. Le processus de **génération** de la vue théorique à partir des données **3D** et de la pose.

3.4.1 Géométrie de la caméra

Dans cette partie, nous allons présenter les concepts utilisés dans le domaine de la vision par ordinateur afin de créer un modèle de caméra. Ce modèle, dans notre cas, devra ressembler le plus possible au modèle de la caméra embarquée dans le smartphone. Pour cela, nous devons estimer les six degrés de liberté en translation et rotation, ou encore appelés paramètres extrinsèques, le modèle de projection utilisé ainsi que les paramètres internes de la caméra, ou encore appelés paramètres intrinsèques.

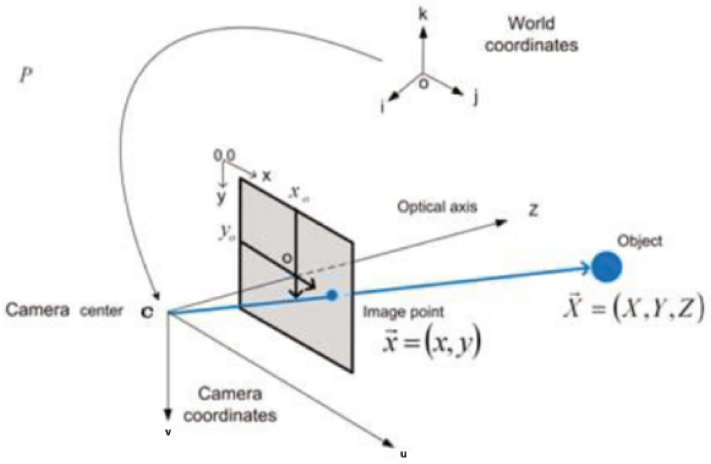


FIGURE 3.9 – *modèle de projection perspective : modèle de sténopé*

3.4.1.1 Introduction et définitions

La type de projection utilisé dans ce qui suit est la ***projection perspective***. Le modèle de caméra à la figure 3.9, présente le modèle sténopé, aussi dit modèle de projection perspective. Dans cette figure, il est représenté :

— *"P"* : Un plan rétinien représentant la caméra

— *"C"* : Centre optique correspondant au centre de projection

La projection orthogonale du point *"C"* sur le plan *"P"* : le point *"O"*. Ce point est appelé *point principal*. La droite (*"OC"*) est nommée : *"axe optique"*. La distance f = *"OC"* est nommée : *"distance focale"*.

Dans cette figure, un objet de coordonnées $\overrightarrow{X} = (X, Y, Z)$ se projette sur le plan image *"P"* en un point de coordonnées $\overrightarrow{x} = (x, y)$ selon une projection perspective sur centre *"O"*.

La projection monde/image se décompose donc en :

— Une projection de l'objet X dans le repère associé à la caméra selon les paramètres extrinsèques de la caméra.

— Le point résultant de cette transformation monde-caméra se projette sur le plan image selon les paramètres intrinsèques de la caméra.

3.4.1.2 Paramètres extrinsèques

Soit le point X de coordonnées $(X, Y, Z)^T$ exprimées dans le repère monde. Ses coordonnées dans le repère de la caméra sont notées $(X_c, Y_c, Z_c)^T$. Ces coordonnées sont calculées à l'aide de l'équation (3.1).

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = R * \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + t = [R|t] * \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.1}$$

Où $(X, Y, Z, 1)^T$ sont appelés coordonnées homogènes du point X dans le repère du monde.

R et t représentent le déplacement rigide entre les deux repères, avec :

— R étant la matrice de rotation

— t le vecteur de translation

Ces paramètres sont la position et l'orientation de la caméra dans le repère du monde. Ils représentent les paramètres extrinsèques de la caméra, que l'on nomme souvent ***"pose de la caméra"***. Nous les déterminons grâce à une procédure de fusion des données capteurs, détaillée en section 3.4.3.

3.4.1.3 Paramètres intrinsèques

La projection du point X, de coordonnées $(X_c, Y_c, Z_c)^T$ définies dans le repère caméra, est le point x de coordonnées $(x_c, y_c, z_c)^T$ définies dans le repère caméra à l'aide de l'équation 3.2.

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = f * \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ \end{pmatrix} \tag{3.2}$$

En passant à un repère en 2D, celui de l'image, le point x a comme coordonnées $(u, v)^T$ obtenues à l'aide de l'équation 3.3.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} k_u & -k_u cos\theta & u_0 \\ 0 & k_v sin\theta & v_0 \end{pmatrix} * \begin{pmatrix} x_c \\ y_c \\ 1 \end{pmatrix} \tag{3.3}$$

Où :
— k_u : La distance focale exprimée en pixels selon l'axe u ;
— k_v : La distance focale exprimée en pixels selon l'axe v ;
— $(u_0, v_0)^T$: les coordonnées pixel du centre C ;
— θ angle entre les deux axes (u, v), représentant l'angle de distorsion.

Tous ces paramètres sont les paramètres intrinsèques de la caméra.

q est l'angle entre les deux axes (u et v) dit aussi angle de distorsion. Les paramètres $ku, kv, f, u0, v0, q$ sont les paramètres intrinsèques de la caméra.

À partir des deux équations **??** et 3.2, nous obtenons la relation suivante :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_u & v_0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{3.4}$$

Où :

$\alpha_u = f * K_u$ et $\alpha_v = f * K_v$ et θ généralement estimé à $\pi/2$. Posons K la matrice des paramètres intrinsèques. L'équation **??** définit la relation entre le point X dont les coordonnées homogènes sont z dans le repère du monde et le point x dont les coordonnées pixeliques homogènes sont $(u, v, 1)^T$.

Référentiel	Abréviation	dimensions	axes
Image Coordinate Frame	ICF	2D	x, y
Camera Coordinate Frame	CCF	3D	X, Y, Z
World Coordinate Frame	WCF	3D	i, j, k

TABLE 3.1 – *Systèmes de coordoonées*

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K * [R|t] * \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.5}$$

La matrice $K * [R|t]$ est appelée matrice de projection perspective. Les deux matrices R et t ont été définies et détaillées dans la partie 3.4.1.2 et calculées en 3.4.3. Nous donnons en annexe **??** les détails de notre approche de calibration de la caméra.

3.4.1.4 Les systèmes de coordonnées

Dans cette thèse, trois systèmes de coordonnées différents sont utilisés.

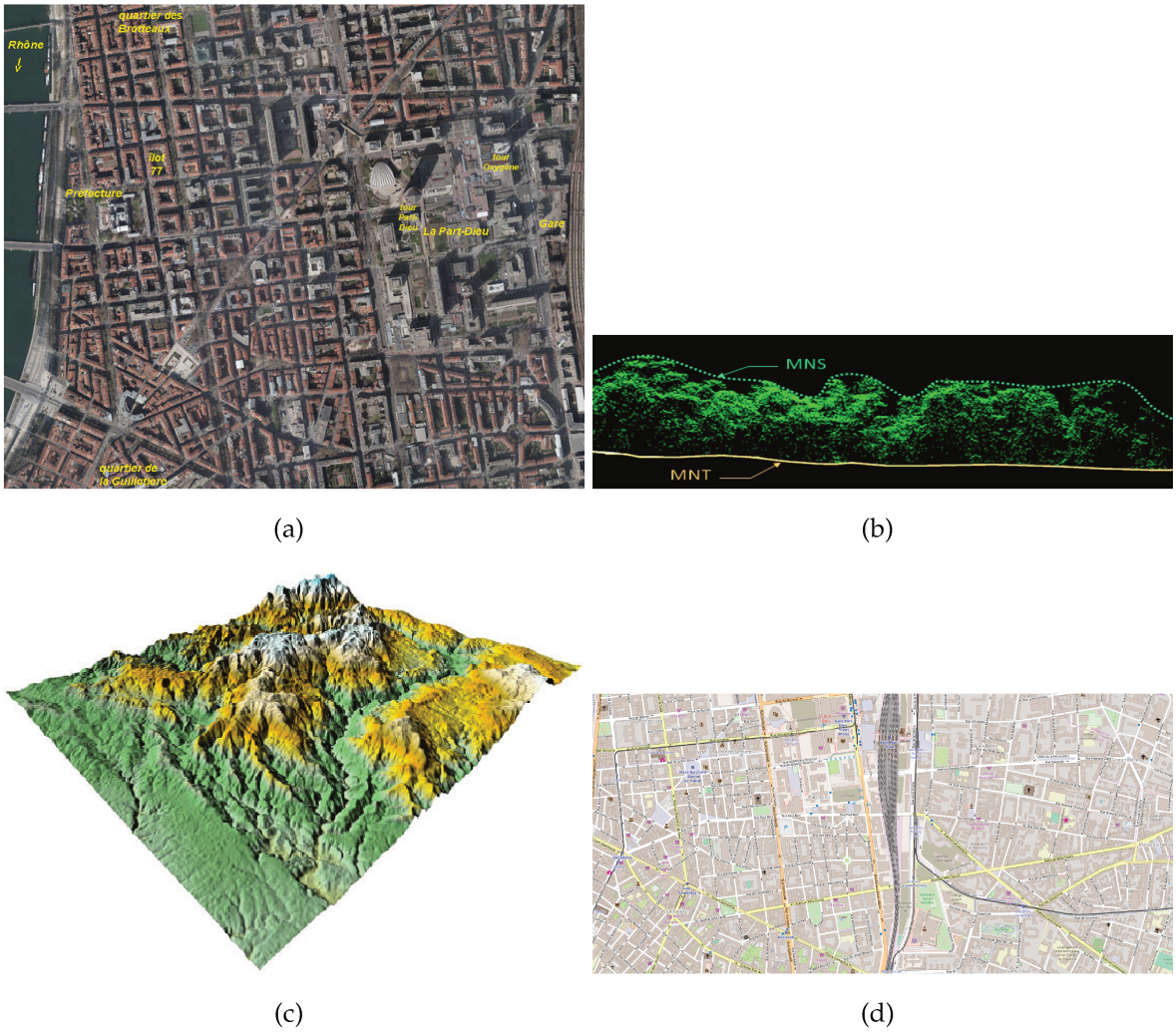
Le premier est situé au niveau de l'image et décrit l'emplacement du pixel en 2D : il est appelé *"Repère des Coordonnées Image"*, en anglais *"Image Coordinate Frame"* (ICF). Les origines de ce repère sont (x0,y0).
Le deuxième est relatif à la caméra, appelé *"Repère des Coordonnées Caméra"*, en anglais *"Camera Coordinate Frame"* (CCF). Les origines de ce repère sont fixées au centre de la caméra (centre de projection) et orientées selon la direction d'observation de la caméra. Les coordonnées sont exprimées en (X, Y, Z).
Le dernier système est le *"Repère des Coordonnées du monde"*, en anglais *"World Coordinate Frame"* (WCF) et il est utilisé pour décrire les positions du modèle 3D de la scène et la position des caméras dans le monde. Les coordonnées sont exprimées en (i, j, k).
Le tableau 3.1 donne un aperçu de ces différents systèmes de coordonnées qui seront utilisés tout au long de ces travaux de thèse.

3.4.2 Données et architecture du système

Dans la plupart des applications de réalité augmentée avec une **connaissance a priori de l'environnement** dans lequel évolue l'utilisateur, les bases de connaissances sont très importantes. Elles peuvent être de plusieurs types :

— **Photographie aérienne** : Ce sont des images acquises dans des campagnes d'acquisition aériennes prises par un drone ou un avion.

— **Modèle Numérique de Terrain (MNT)** : C'est une représentation de la topographie naturelle.
— **Modèle Numérique de Surface (MNS)** : C'est une extension du *MNT* en modélisant les *sursols*. Les modèles numériques de surface sont des représentations surfaciques représentant la topographie de l'environnement et des objets au sol. On peut citer à titre d'exemple : SRTM.
— **Données vectorielles** : Ce sont les empreintes (traces) au sol des bâtiments, comme OpenStreetMap [1].



(a) (b)

(c) (d)

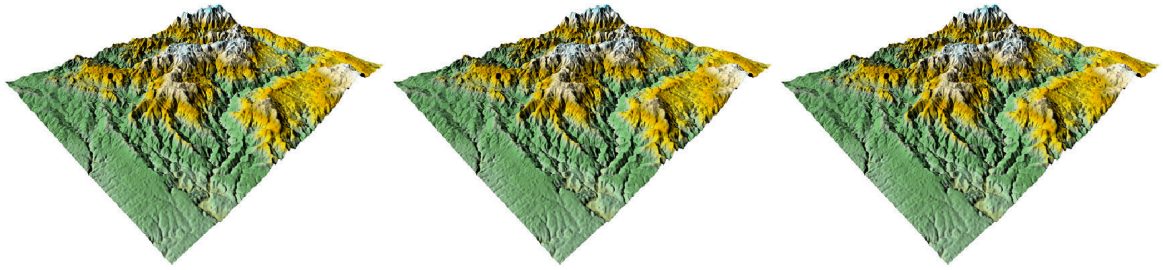
FIGURE 3.10 – *(a) photo aérienne, (b) MNT, (c) MNS (SRTM), (d) Donnee Vectorielles (OSM)*

Dans l'approche que nous proposons, nous avons besoin d'une représentation virtuelle 3D de l'environnement. En effet, contrairement aux applications de RV (Réalité Virtuelle), l'immersion doit être la plus fidèle possible pour être la plus en adéquation avec l'environnement réel que nous visualisons grâce à la caméra embarquée dans le smartphone et dans lequel l'utilisateur évolue.

1. https ://www.openstreetmap.org

La construction de modèles 3D simples et riches en informations à grande échelle est aujourd'hui un enjeu majeur pour la RA mobile. Spécialement en extérieur, la construction de ces modèles a connu ces dernières années une avancée majeure. Comme mentionné dans Gaillard et al. (2015), de nombreuses villes dans le monde possèdent aujourd'hui leur ***double virtuel***. Nous pouvons en citer quelques-unes très récentes, à titre d'exemples : le modèle 3D de New York de 2016, de Bruxelles en 2014 ou de Lyon en 2015. La production de ces données 3D géoréférencées est réalisable grâce à des processus basés sur des campagnes d'acquisition aériennes ou terrestres. Ces données deviennent de plus en plus précises et notamment open source. Ces modèles sont disponibles en plusieurs formats : 3DS, CityGML, KMZ, DirectX. Dans notre cas, nous utilisons le format CityGML de l'*Open Geospatial Consortium* - OGC [2]).

Avant de détailler cette base de connaissance, nous en donnons un exemple, à la figure 3.11, du modèle 3D de la ville de Lyon avec lequel nous validons l'ensemble des approches que nous proposons. Plusieurs points de vue, dont les coordonnées GPS sont données en légende, ont été utilisés pour générer ces images.



(a) (b) (c)

FIGURE 3.11 – *Rendu virtuel dans la ville de Lyon*
(a) (x,y), (b) (x,y), (c) (x,y)

Le scénario type pour l'application que nous proposons suit plusieurs étapes :

1. L'utilisateur se géolocalise dans la ville grâce à l'application ;
2. Une requête est envoyée à un serveur avec cette position GPS ;
3. Le serveur recherche dans la base de données tous les modèles 3D de chacun des bâtiments autour de la position, dans un périmètre prédéfini, et renvoie au client les données correspondantes au format *JSON* ;
4. plusieurs options se présentent à l'utilisateur :
   (a) un rendu virtuel de ces bâtiments ;
   (b) un rendu en réalité augmenté de ces bâtiments superposés au monde réel :

2. http ://www.opengeospatial.org

i. avec correction du skyline ;

ii. sans correction du skyline.

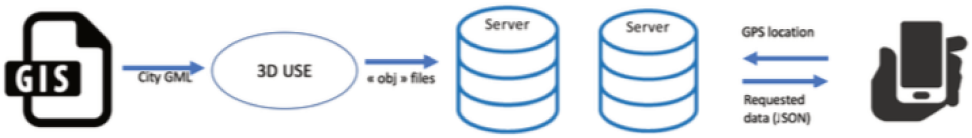
3.4.2.1 Données 3D

Les tests effectués dans le cadre de cette thèse ont été possibles grâce au modèle 3D de la ville de Lyon, fourni par le GrandLyon [3]. Les données couvrent toute la *Métropole de Lyon*, avec ses 52 communes, représentant ainsi environ *550 kilomètres carrés*. Ces données sont fournies sous forme de fichiers *CityGML*. Nous ne détaillons pas cette structure dans ce chapitre. Plus d'informations sur les aspects techniques sont décrits en annexe **??**.

D'une part, la lecture d'un fichier *XML* (*CityGML*) est très coûteuse et très couteuse en temps de calcul sur le plan informatique. De ce fait, une étape de prétraitement est nécessaire afin de construire une base de données dans laquelle le modèle 3D des bâtiments sera converti en fichiers texte (voir figure 3.12.a en partie 3.4.2.2).

D'autre part, le stockage de tout ce modèle sur un appareil mobile n'est évidemment pas possible. Les données géométriques et leurs textures représentant un peu plus de 400 Giga de données. Pour cela nous proposons une architecture légère client-serveur (voir figure 3.12.b en partie 3.4.2.2).

3.4.2.2 Pré-traitements et architecture



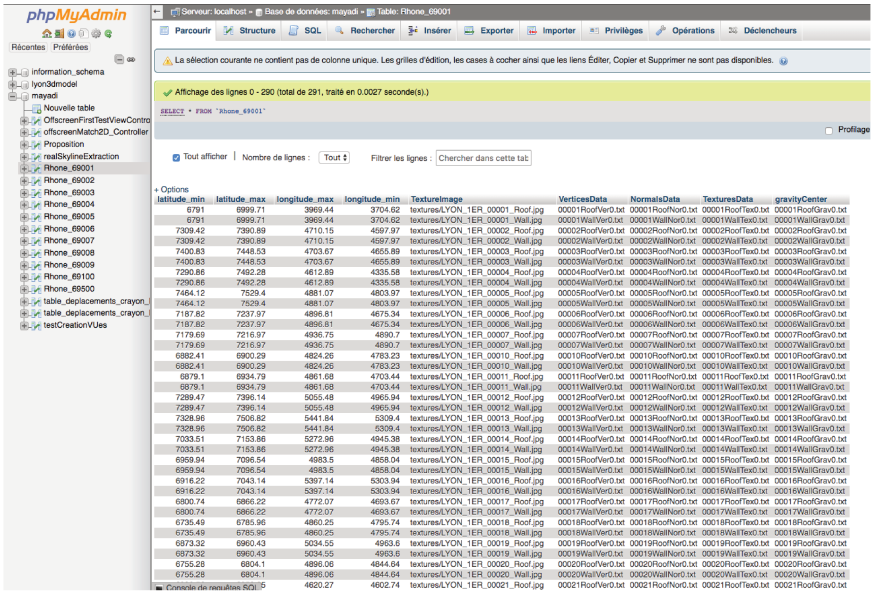
(a) (b)

FIGURE 3.12 – *(a) : Pré-traitements sur les données CityGML, (b) échange de données client-Serveur*

Cette étape nous permet d'alléger l'échange de fichiers entre le client et le serveur. En effet, les données brutes sont stockées sous forme de quartiers : chaque quartier de la ville est stocké dans un fichier *CityGML*. Notre étape de pré-traitement consiste à convertir toutes ces données et à les stocker sous forme de fichiers texte. Ces fichiers sont automatiquement découpés en *clusters* de taille fixe. Toutes ces opérations sont regroupées du côté serveur. En cas de modification des données (autres villes, autres arrondissements, etc.), les données sont recalculées facilement et automatiquement. En ce qui concerne les fichiers de textures, ils sont conservés en l'état, à savoir dans

3. https ://data.grandlyon.com

leur format original en *"png"*. Toutes ces données sont stockées dans une base de données, elle-même divisée en plusieurs tables correspondant aux différents quartiers de la ville. Cela nous permet d'avoir un accès très rapide aux données et un échange de données très fluide. Cette étape est une préparation à une des perspectives du prototype : le ***rendu progressif***. Nous en parlerons dans la partie **??** du chapitre 5.
Nous illustrons à la figure 3.13 un exemple de cette architecture, pour la ville de *Lyon*, *Bron* et *Villeurbanne*.



(a)

FIGURE 3.13 – *Exemple des différentes tables de la base de données*

3.4.3 Estimation de la pose

Une fois le modèle de la caméra construit, il nous faut la placer dans l'environnement virtuel (la ville virtuelle) à la même pose que l'utilisateur (position et orientation). Les six degrés de liberté de cette pose sont récupérés grâce aux différents instruments, qui ont été présentés dans la partie 3.3. Nous détaillons dans ce qui suit l'estimation de ces six degrés de liberté en translation et rotation.

3.4.3.1 Translations

Les translations, ou coordonnées en translation, sont calculées grâce aux données récupérées du GPS. En effet, le GPS nous renvoie des données sous la forme : (*longitude*, *latitude*). À titre d'exemple, nous donnons les coordonnées GPS de la Tour de la Part-Dieu, que les Lyonnais appellent le Crayon (long=*45.761109*, lat=*4.853895*) selon le référentiel WGS84, précédemment cité dans la partie 3.3.1. Nous changeons

de système de coordonnées : d'une part, ***WGS84*** n'est pas le plus précis à l'échelle de la *France*, et présente des dérives dues aux altérations linéaires. D'autre part, les données de la ville de *Lyon* sont elles-mêmes projetées selon un autre repère : le ***RGF93***. Nous rajoutons donc une étape de changement de repère, qui sera détaillée en Annexe **??**. Les coordonnées précédemment citées du *Crayon* deviennent alors : (long=*7379975.37m* , lat=*1826145.61m*). Les mesures dans ce nouveau repère sont le *mètre*. Nous choisissons dans ce nouveau repère la zone 46, vu les détails à la figure 3.14.

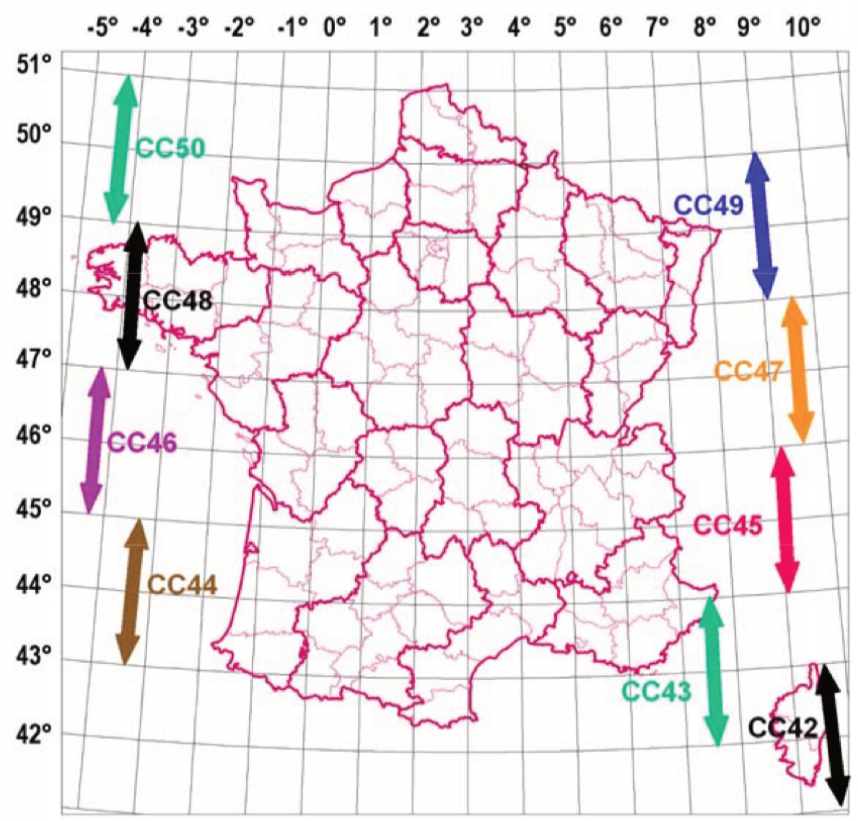


FIGURE 3.14 – *Carte des zones de projection à l'échelle de le France*

3.4.3.2 Rotations

Les angles de rotation que nous cherchons à estimer représentent l'orientation de la caméra. Nous les décrivons avec les trois angles d'*Euler*.
En effet, si nous prenons par exemple un objet quelconque, le centre de masse m de cet objet est une coordonnée dans notre espace à 3 dimensions. Tout point dans un espace à n dimensions peut être atteint par n translations. Ainsi, le point m est atteint par *3 translations*, quelle que soit l'origine du repère. De façon semblable à une *coordonnée*, une *orientation* dans cet espace tridimensionnel est obtenue par trois rotations. Ces rotations sont effectuées successivement dans un ***ordre arbitraire***, chacune autour

d'un des trois axes du repère. Cela revient à effectuer une rotation dans le plan formé par deux axes autour d'un troisième, et cela trois fois de suite.

En utilisant l'accéléromètre, nous calculons les angles d'*Euler* (et donc l'orientation de l'objet).

Pour rappel, un accéléromètre permet de mesurer l'accélération linéaire que subit un corps auquel ce capteur serait attaché. La sortie de l'accéléromètre peut être modélisée de la façon suivante :

$$\overrightarrow{a_S} = \begin{pmatrix} a_{Sx} \\ a_{Sy} \\ a_{Sz} \end{pmatrix} = \overrightarrow{a_B} + R * \overrightarrow{g}$$

où :

$\begin{cases} \overrightarrow{a_S} & \text{Sortie de l'accéléromètre} \\ a_{Sx} & \text{Acc. selon l'axe X} \\ \overrightarrow{a_B} & \text{Accélération subie par le dispositif} \\ \overrightarrow{g} & \text{Gravité terrestre} \\ R & \text{Matrice de rotation qui permet de projeter } \overrightarrow{g} \text{ du} \\ & \text{référentiel inertiel terrestre au référentiel du corps en mouvement} \end{cases}$

Lorsque le corps est au repos, $\overrightarrow{a_B} = \overrightarrow{0}$:

$$\overrightarrow{a_S} = \begin{pmatrix} a_{Sx} \\ a_{Sy} \\ a_{Sz} \end{pmatrix} = \overrightarrow{a_B} + R * \overrightarrow{g} = R * \overrightarrow{g} * \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Les matrices de rotation dans l'espace euclidien s'écrivent sous la forme :

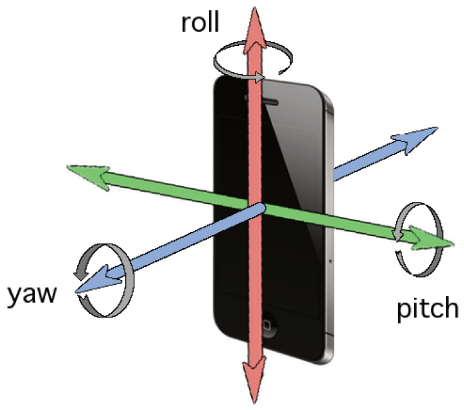


FIGURE 3.15 – *les axes de rotation de l'accéléromètre d'un Smartphone*

$$\text{Autour de l'axe X} : R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{pmatrix}$$

$$\text{de l'axe Y} : R_y(\theta) = \begin{pmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{pmatrix}$$

$$\text{Autour de l'axe Z} : R_z(\psi) = \begin{pmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Nous allons maintenant chercher à résoudre cette équation. Chacune de ces matrices correspond à une rotation autour d'un des axes du repère. Étant donné que nous voulons effectuer successivement les trois rotations, nous devons multiplier les matrices entre elles. Or, la multiplication de matrices n'est pas commutative. Nous devons choisir un ordre arbitraire à ces multiplications.

Le détail des calculs pour R_{xyz} est donné en Annexe **??**.

Le résultat de ces six combinaisons est donné en tableau 3.2.

$R_{xyz} * (0\ 0\ 1)^T = \begin{pmatrix} -sin(\theta) \\ sin(\phi)cos(\theta) \\ cos(\phi)cos(\theta) \end{pmatrix}$
$R_{yxz} * (0\ 0\ 1)^T = \begin{pmatrix} -sin(\theta)cos(\phi) \\ sin(\phi) \\ cos(\phi)cos(\theta) \end{pmatrix}$
$R_{xzy} * (0\ 0\ 1)^T = \begin{pmatrix} cos(\psi)sin(\theta) \\ cos(\theta)sin(\phi) + cos(\phi)sin(\psi)sin(\theta) \\ cos(\phi)cos(\theta) - sin(\theta)sin(\phi)sin(\psi) \end{pmatrix}$
$R_{yzx} * (0\ 0\ 1)^T = \begin{pmatrix} cos(\theta)sin(\phi)sin(\psi) - cos(\phi)sin(\theta) \\ cos(\psi)sin(\phi) \\ cos(\theta)cos(\phi) + sin(\theta)sin(\phi)sin(\psi) \end{pmatrix}$
$R_{zxy} * (0\ 0\ 1)^T = \begin{pmatrix} cos(\theta)sin(\phi)sin(\psi) - cos(\psi)sin(\theta) \\ cos(\psi)cos(\theta)sin(\phi) + sin(\theta)sin(\psi) \\ cos(\theta)cos(\phi) \end{pmatrix}$
$R_{zyx} * (0\ 0\ 1)^T = \begin{pmatrix} sin(\phi)sin(\psi) - cos(\phi)cos(\psi)sin(\theta) \\ cos(\psi)sin(\phi) + cos(\phi)sin(\psi)sin(\theta) \\ cos(\theta)cos(\phi) \end{pmatrix}$

TABLE 3.2 – *Résultat combinaison des rotations*

En interprétant ces résultats, on remarque que les équations des deux combinaisons de R_{xyz} et de R_{yxz} ne dépendent que des variables θ et ϕ. Toutes les autres combinaisons ont des équations qui dépendent de θ, ψ et ϕ, ce qui les rend inexploitables.

Nous faisons donc le choix de R_{xyz} et par conséquent nous obtenons le résultat suivant :

$$\overrightarrow{a_S} = \begin{pmatrix} a_{Sx} \\ a_{Sy} \\ a_{Sz} \end{pmatrix} = \begin{pmatrix} -sin(\theta) \\ sin(\phi)cos(\theta) \\ cos(\phi)cos(\theta) \end{pmatrix}$$

$$\text{d'où :} \quad a_{Sx} = -sin(\theta); \qquad a_{Sy} = sin(\phi)cos(\theta); \qquad a_{Sz} = cos(\phi)cos(\theta)$$

$$\frac{a_{Sy}}{a_{Sz}} = \frac{sin(\phi)cos(\theta)}{cos(\phi)cos(\theta)} = \tan(\phi) \;\; si \;\; \theta \neq \pi/2 \;\; et - \pi/2 ==> \boxed{\phi = \tan^{-1}(\frac{a_{Sy}}{a_{Sz}})}$$

$$a_{Sy}^2 + a_{Sz}^2 = cos(\theta)^2 * (sin(\phi)^2 + cos(\phi)^2) = cos(\theta)^2 \;\; ==> \;\; cos(\theta) = \sqrt{a_{Sy}^2 + a_{Sz}^2}$$

$$\tan(\theta) = \frac{sin(\theta)}{cos(\theta)} = \frac{-a_{Sx}}{\sqrt{a_{Sy}^2 + a_{Sz}^2}} \;\; ==> \;\; \boxed{\theta = \tan^{-1}(\frac{-a_{Sx}}{\sqrt{a_{Sy}^2 + a_{Sz}^2}})}$$

À travers ces différents calculs, nous avons récupéré deux des degrés de liberté en rotation de la pose à estimer.

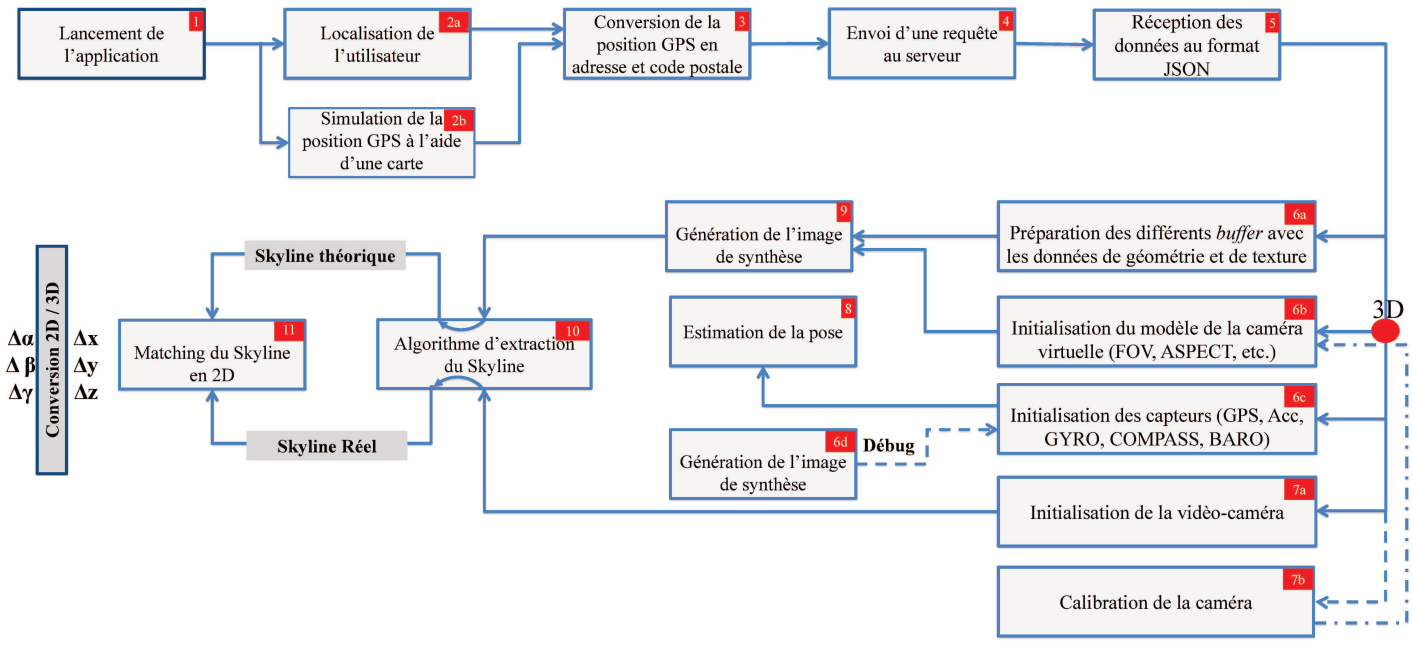
Les données provenant du compas magnétique nous servent à fixer le dernier angle.

3.4.4 Du calcul de pose à la RA

Nous proposons tout d'abord un récapitulatif. À ce stade, nous avons :

1. Construit le modèle de la caméra avec les paramètres :
   (a) **Intrinsèque** en extrayant les paramètres internes de la caméra (focale, ...) ;
   (b) **Extrinsèque** en calculant les paramètres de translation et rotation ;
2. Préparé les données 3D dans une base de données.

Nous illustrons à la figure 3.4.4, le schéma de l'application que nous proposons, et détaillons ci-après.


1
Lancement de l'application
2a
Localisation de l'utilisateur
2b
Simulation de la position GPS à l'aide d'une carte
3
Conversion de la position GPS en adresse et code postale
4
Envoi d'une requête au serveur
5
Réception des données au format JSON
6a
Préparation des différents buffer avec les données de géométrie et de texture
6b
Initialisation du modèle de la caméra virtuelle (FOV, ASPECT, etc.)
6c
Initialisation des capteurs (GPS, Acc, GYRO, COMPASS, BARO)
7a
Initialisation de la vidèo-caméra
7b
Calibration de la caméra
3D
9
Génération de l'image de synthèse
8
Estimation de la pose
6d
Génération de l'image de synthèse
Débug
10
Algorithme d'extraction du Skyline
11
Matching du Skyline en 2D
Skyline théorique
Skyline Réel
Conversion 2D / 3D
Δα
Δ β
Δγ
Δx
Δy
Δz

```
self.projectionMatrix = GLKMatrix4MakePerspective(GLKMathDegreesToRadians(fov), aspect, near, far);
```

(a)

```
self.rotMatrix = GLKMatrix4Rotate(self.rotMatrix,  self.RotationX + alpha, 1, 0, 0);
self.rotMatrix = GLKMatrix4Rotate(self.rotMatrix,  self.RotationY + beta, 0, 1,0);
self.rotMatrix = GLKMatrix4Rotate(self.rotMatrix,  self.RotationZ + gamma, 0, 0, 1);
```

(b)

FIGURE 3.16 – *(a) : Directive de construction de la matrice de projection (b) : Directive pour l'application de rotations*

Dans cette partie, nous allons détailler notre processus de rendu. En effet, comme mentionné dans l'introduction, la première version de cette application a été réalisée sur un iPhone 4s en 2014. Puis, grâce à un projet au sein du Laboratoire *LIRIS*, nous avons pu nous munir d'un panel de dix iPhones 6. Ce dispositif prend en charge les graphiques 2D et 3D hautes performances avec l'*Open Graphics Library* (*OpenGL*), et en particulier l'API avec laquelle la plupart de nos développements ont été effectués : *OpenGL-ES*. *OpenGL* est une *API* graphique multiplateforme qui spécifie une interface logicielle standard pour le matériel de traitement graphique 3D. Pour les développements logiciels, *iOS-SDK* et le logiciel spécifique d'Apple XCode sont utilisés, avec leur langage de programmation spécifique *Objective-C* d'abord, puis *Swift*. Pour la représentation d'objets 3D, deux formats de fichiers ont été utilisés : d'abord les fichiers *CityGML*, qui sont ensuite convertis en fichiers d'extension *".obj"*. Pour l'acquisition vidéo, on utilise les *API* internes au dispositif afin de pouvoir maîtriser la qualité du rendu vidéo, et la diminuer dans le cas d'une latence vidéo due aux différents traitements effectués.

Pour visualiser la RA, deux ***contextes*** séparés, issus de la classe *GLContext* de l'*API OpenGLES* et paramétrés en mode plein écran, sont utilisés : un premier pour le rendu réel, sur lequel est superposé le contexte dédié au rendu virtuel. Cette classe *GLContext* est dédiée pour l'affichage du rendu des modèles 3D avec toutes les options pour la gestion de la transparence et de la génération d'une image 2D. Cette *API* nous propose plusieurs outils permettant de construire le modèle de la caméra, mais aussi des fonctions pour appliquer les translations et rotations adéquates à l'objet visualisé. On peut citer à titre d'exemple, les directives permettant la *construction de la matrice de projection*, une fois l'étape de calibration terminée, ou alors les directives pour l'*application des rotations*.

Il est à noter que dans toute cette boucle, toutes les données (positionnement réel, propriétés de l'objet 3D), ont comme unité de mesures le mètre. Ce choix est fait, grâce à ce qui est détaillé en 3.4.3.1 et en **??**, et ce pour obtenir la taille naturelle du bâtiment.

3.5 EXPÉRIMENTATIONS ET RÉSULTATS

Nous présentons dans cette partie les expérimentations menées ainsi que les résultats que nous en avons obtenus. Le but de ces expérimentations est d'évaluer la précision de notre système de RA basé instruments, à savoir l'estimation de la pose basée sur les capteurs du smartphone. Pour cela, nous avons défini deux critères de performance, qui sont :

— Le temps de calcul pour le rendu virtuel ;
— L'erreur de reprojection, représentant l'écart entre les points 2D de l'image réelle et la projection de leurs correspondants 3D grâce aux paramètres de pose calculés.

Tous nos tests ont été effectués dans notre terrain de jeu : **la ville de Lyon**. Nous avons utilisé , en premier lieu un iPhone 6 puis un iPhone 7 Plus, à des fins de comparaisons de temps de calcul et de précision. Ces différents dispositifs sont munis des capteurs cités dans la partie 3.3 et de différentes caméras.

Nous présentons ci-dessous les caractéristiques des caméras utilisées dans ces tests, grâce à l'outil *Exiftool*.

Pour l'iPhone 6 :

— Focal Length : 4.2 mm (35 mm equivalent : 29.0 mm)
— Field Of View : 63.7 deg
— Image Size : 3264x2448
— Megapixels : 8.0
— Lens Info : 4.15mm f/2.2
— Lens Model : iPhone 6 back camera 4.15mm f/2.2

Pour la caméra l'iPhone 7 Plus :

— Focal Length : 4.0 mm (35 mm equivalent : 28.0 mm)
— Field Of View : 65.5 deg
— Image Size : 3024x4032
— Megapixels : 12.2
— Lens Info : 3.99-6.6mm f/1.8-2.8
— Lens Model : iPhone 7 Plus back dual camera 3.99mm f/1.8

3.5.1 Erreur de reprojection

Le premier critère d'évaluation pour un système de RA est l'erreur de reprojection. En effet, ce critère reflète la qualité de l'augmentation de la scène en termes de précision. Pour ce qui est de la robustesse (ou encore appelée stabilité), ce critère sera étudié en chapitre 5. Pour calculer cette erreur, nous avons mené plusieurs cam-

pagnes de test dans la ville, qui consistent à se déplacer dans l'environnement et à enregistrer à chaque instant :

— L'image réelle, captée par la caméra de l'utilisateur
— L'image virtuelle, générée à partir des paramètres de pose estimés
— Toutes les métadonnées qui peuvent nous être utiles (temps exécution, paramètres de la pose - six degrés de liberté, date et heure du test, les paramètres de la méthode d'extraction du skyline, etc..)

Ces différents jeux de données sont envoyés, lors du test et en temps réel grâce à une interaction avec l'application, sur un serveur. Ces données seront traitées a posteriori. Ces traitements consistent à calculer, pour chacune des images obtenues, l'erreur de reprojection, qui se traduit par **la distance entre les points 3D projetés sur l'écran du smartphone, et leurs correspondants 2D dans l'image réelle**.

Cette distance est calculée sur chacun des deux axes X et Y de l'espace image, et ce pour **plusieurs points choisis manuellement de sorte à en avoir le plus possible qui soient non-coplanaires**. En effet, **l'erreur de reprojection sur un plan peut différer d'un plan à l'autre, et ceci en raison de la projection perspective**. On essayera également, pour un même objet (bâtiment), de prendre les mêmes points d'une vue à l'autre (d'une image à l'autre).

Nous illustrons quelques exemples aux figures 3.17, 3.18 et 3.19 avec pour chacune des images, l'image réelle et virtuelle superposées (avec leur deux skyline, détaillé en chapitre 4), et les points utilisés pour calculer la distance entre les deux images. Cette distance sera notée δ_x selon l'axe X et δ_y selon l'axe Y. Dans cette figure, nous aurons plusieurs vues (plusieurs images, numérotées 13, 14, 16 et 17 dans notre base de données), et pour chacune nous donnons les coordonnées dans l'image réelle et virtuelle du point utilisé pour le calcul de l'écart.

Pour la figure 3.17 :

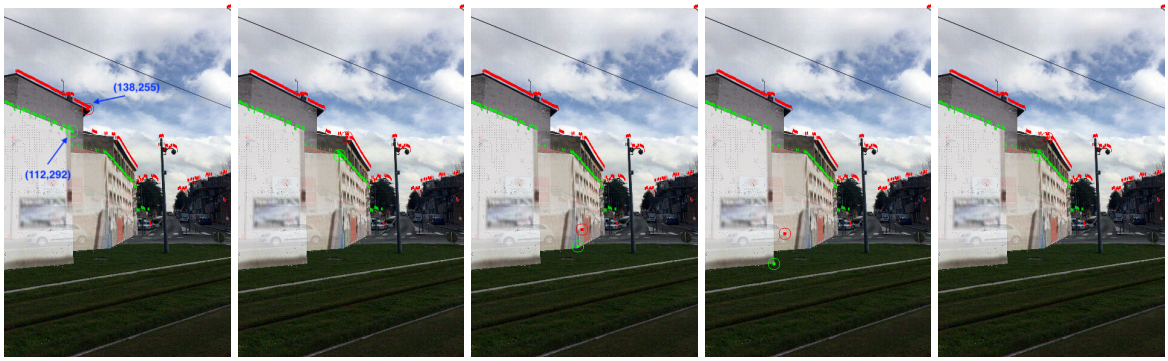
— (a) : Img13 : Point réel (138,255) ; Point virtuel (112,292) ==> (δ_x, δ_y) = (26 , -37)
— (b) : Img13 : Point réel (180,302) ; Point virtuel (168,332) ==> (δ_x, δ_y) = (12 , -30)
— (c) : Img13 : Point réel (183,455) ; Point virtuel (176,483) ==> (δ_x, δ_y) = (7 -28)
— (d) : Img13 : Point réel (132,462) ; Point virtuel (114,512) ==> (δ_x, δ_y) =(18 , -50)
— (e) : Img13 : Point réel (180,303) ; Point virtuel (165,332) ==> (δ_x, δ_y) =(15 , -29)

Pour la figure 3.18 :

— (a) : Img14 : Point réel (262,237) ; Point virtuel (240,266) ==> (δ_x, δ_y) =(22 , -29)
— (b) : Img14 : Point réel (157,187) ; Point virtuel (126,228) ==> (δ_x, δ_y) = (31 , -41)
— (c) : Img14 : Point réel (311,282) ; Point virtuel (304,303) ==> (δ_x, δ_y) =(7 , -21)
— (d) : Img14 : Point réel (323,441) ; Point virtuel (314,466) ==> (δ_x, δ_y) =(9 , -25)
— (e) : Img14 : Point réel (153,185) ; Point virtuel (124,230) ==> (δ_x, δ_y) =(29, -45)

Pour la figure 3.19 :

— (a) : Img16 : Point réel (131,280) ; Point virtuel (138,306) ==> (δ_x, δ_y) =(-7, -26)
— (b) : Img16 : Point réel (81,230) ; Point virtuel (80,259) ==> (δ_x, δ_y) =(1, -29)
— (c) : Img16 : Point réel (76,449) ; Point virtuel (78,481) ==> (δ_x, δ_y) =(-2, -32)
— (d) : Img17 : Point réel (214,282) ; Point virtuel (328,290) ==> (δ_x, δ_y) =(-114 -8)
— (e) : Img17 : Point réel (198,314) ; Point virtuel (292,323) ==> (δ_x, δ_y) =(-94, -9)

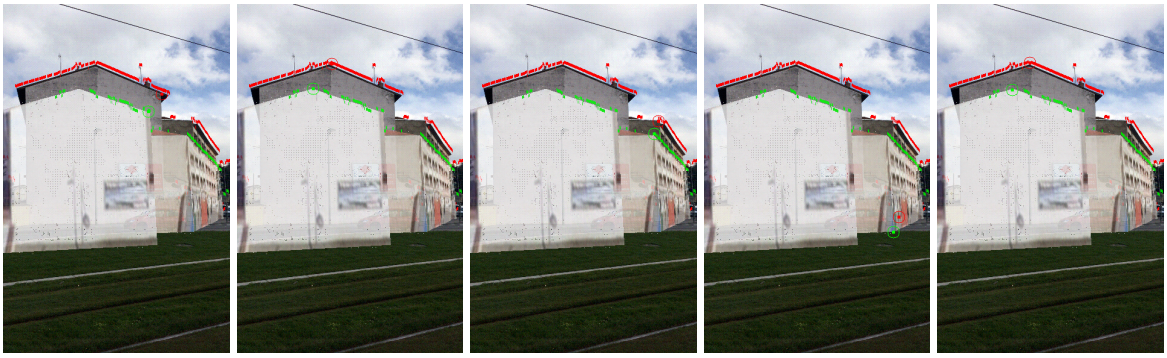


(a) (b) (c) (d) (e)

FIGURE 3.17 – *Plusieurs points pour le calcul de l'erreur de reprojection*

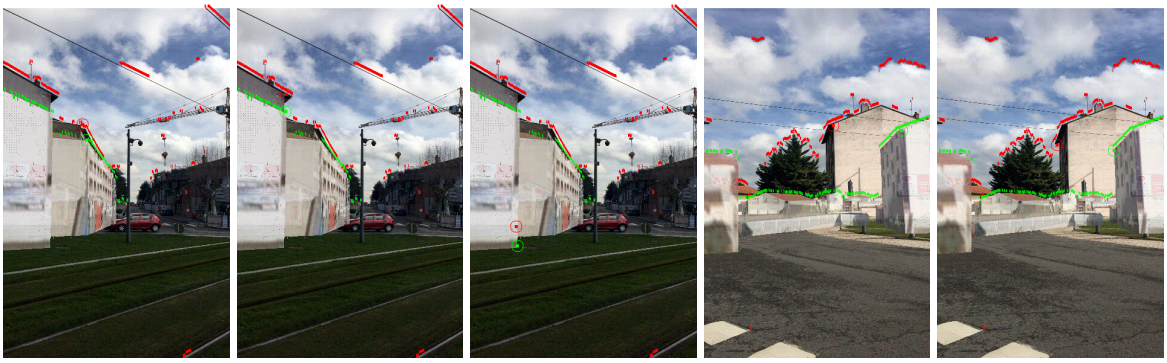
Sur un ensemble de plus de 1000 images constituant notre jeu de données, prises à différents endroits de la ville, à différents moments et dans différentes conditions, la figure 3.20 présente les erreurs moyennes obtenues par image, pour chacun des axes X et Y. L'écart type pour cette base de données, est égal à : σ_x, σ_y, =(46.7 ,17.2).

Les tests que nous avons menés nous montrent que les rendus réels et virtuels, munis de leurs skylines, se font en temps réel. De ce fait, nous n'avons pas besoin d'anticiper certains mouvements tels que les mouvements larges et brusques. De plus, les expérimentations conduites nous montrent aussi que le système est robuste face à ce type de mouvements. En d'autres termes, quand on effectue un mouvement très brusque, par exemple en balayant très rapidement de l'extrême droite vers l'extrême gauche, le rendu redevient rapidement très stable en quelques millisecondes.



(a) (b) (c) (d) (e)

FIGURE 3.18 – *Plusieurs points pour le calcul de l'erreur de reprojection*



(a) (b) (c) (d) (e)

FIGURE 3.19 – *Plusieurs points pour le calcul de l'erreur de reprojection*

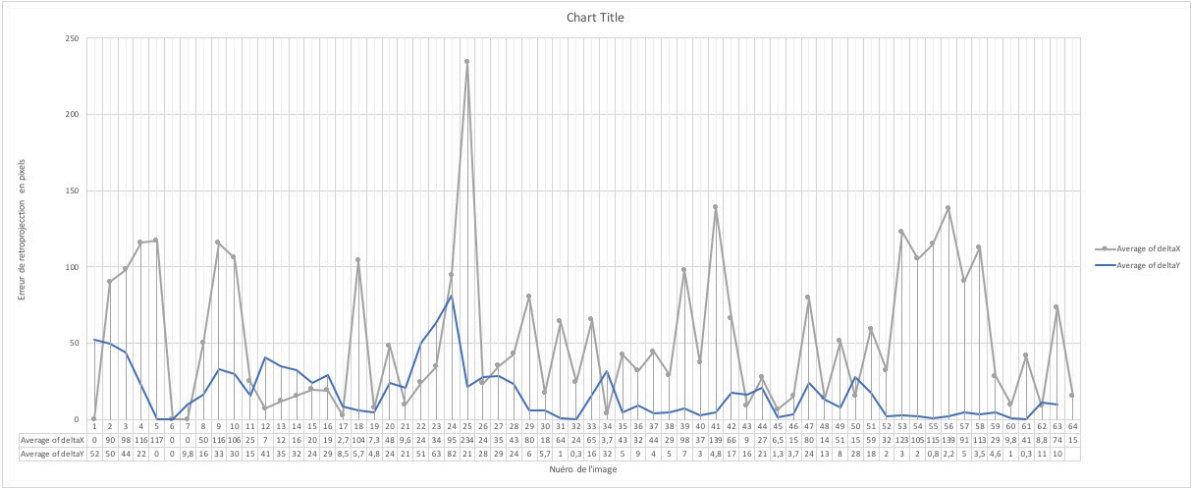


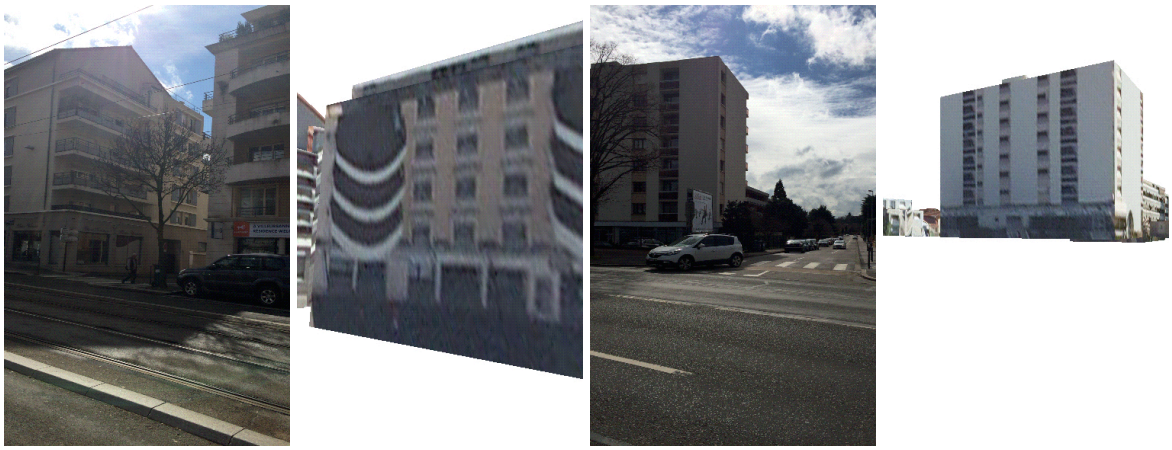
FIGURE 3.20 – *Réparatition de l'erreur moyenne par image*

Aucune latence n'est observée.

Visuellement, les erreurs sont en général inférieures à une *cinquantaine de pixels selon*

l'axe X et une *quinzaine selon Y*. Nous montrons un échantillon de ces résultats dans un tableau en annexe **??**.

En observant le tracé des erreurs de reprojection, nous constatons que certaines images présentent des erreurs très importantes. En interprétant les conditions dans lesquelles elles ont été prises, nous constatons que l'un des instruments a été fortement influencé. À titre d'exemple, nous donnons l'erreur de reprojection de deux images 24 et 55, dont les erreurs de reprojection respectives sont : ((δ_x, δ_y) = (234 , 81) et (138 , 0.75). Les deux images qui correspondent sont données à la figure **??**. Pour ces deux images, et bien d'autres prises à la même position, ce décalage est parfois présent, et d'autres pas ! Ceci est dû au passage d'un tramway à l'instant même où le test a été effectué, provoquant ainsi un champ magnétique plus ou moins important, et donc un décalage dans les mesures délivrées par le compas magnétique. Ces images sont, bien évidemment, gardées dans la base de données, même si on pourrait les trier, vu que la source du problème a été identifiée, et éviter leur influence sur les résultats.



(a) (b) (c) (d)

FIGURE 3.21 – *Erreur de reprojection*

3.5.2 Temps de calcul

Les temps de calculs, dans notre contexte, sont très importants et à prendre en considération avec toute la rigueur qu'il faut. En effet, visant une application de RA mobile en contexte urbain, il est important que l'utilisateur puisse avoir un rendu fluide, sans latence et avec la meilleure précision possible. Dans le cas contraire, l'application n'aurait pas d'avantages par rapport aux casques de réalité virtuelle, qui donneraient même une meilleure impression d'immersion. Pour cela, et afin d'avoir

un rendu réaliste en temps réel, il faudrait avoir 40 images par seconde. En d'autres termes, il faudrait que cette étape ne dépasse pas les milli-secondes. Cependant, il faudrait toujours garder en tête que le recalage visuel n'est pas encore pris en compte. En effet, les temps de calcul mentionnés ci-dessous sont la somme des temps de calculs de :

- *Récupération de l'image réelle ;*
- *le calcul de la pose ;*
- *La préparation des données 3D pour le rendu (envoi des données au buffer) ;*
- *Le rendu virtuel dans un contexte OpenGL-ES, en prenant en considération l'image réelle ;*
- *La construction d'une image 2D, étant la projection de ce modèle 3D sur l'écran du smartphone ;*
- *Enregistrement vidéo temps réel, à des fins d'évaluation.*

En effet, il faut prendre en considération que ces temps d'exécution dépendent du **nombre de points 3D** utilisés pour construire l'image. En d'autres termes, c'est le nombre de points composant le modèle 3D utilisé.

De plus, il est à noter que le modèle 3D de la ville de Lyon est proposé selon une structure en **redondance de sommets**. Ce qui signifie que deux triangles consécutifs [4], ayant deux sommets en commun, auront en totalité six sommets au lieu d'en avoir uniquement quatre avec une stratégie en **Trangle Strip** [5]. Ce qui nous donne un modèle de données important en termes de nombre de sommets à dessiner.

Un des *avantages* que nous avons pris en considération dans ces travaux, c'est le ***test de profondeur***. En effet, si deux points du modèle 3D de coordonnées $A = (x_1, y_1, z_1)$ et $B = (x_2, y_2, z_2)$ s'occultent l'un l'autre, l'occulté sera rejeté et ne sera pas utilisé pour faire le rendu. En d'autres termes, si $x_1 = x_2$, $y_1 = y_2$ et que $z_2 < z_1$, cela signifie que ***le point B cachera le point A***. De ce fait, il est inutile d'**alourdir** les temps de calcul et de **demander** à notre moteur de rendu **OpenGL-ES** de ***dessiner*** le point A.

Néanmoins, un *inconvénient* est à souligner. Notre approche permet de faire ce **rendu en temps réel panoramique (à 360 degrés)**. En effet, nous n'avons pas réussi à implémenter une méthode qui nous permette de ne faire le rendu que des bâtiments compris dans notre champ de vision (tout ce qui devant nous), et éliminer ceux qui sont derrière. Ceci nous permettrait de diminuer encore plus les temps de calcul et alléger encore plus notre application. Les bâtiments qui se trouvent derrière l'utilisa-

4. En 3D, on ne peut construire que des triangles. De ce fait, afin de construire un bâtiment composé de plusieurs façades, considérées comme des rectangles, il faut passer par ces triangles : un rectangle peut être "**dessiné**" au travers de deux triangles

5. https ://en.wikipedia.org/wiki/Trianglestrip

teur ne seront pas évidemment visualisés, vu que le *Frustum de rendu* les éliminera, vu leur présence derrière le plan *"**Near**"* de ce dernier.

Un autre aspect est de même à prendre en compte : à ce rendu virtuel ainsi que toutes les contraintes matérielles prises en compte, nous rajoutons une étape d'***enregistrement vidéo*** en temps réel, nous permettant d'avoir, a posteriori, une vidéo des tests réalisés par l'utilisateur, et de pouvoir les évaluer en cas de besoin.

Nous illustrons, à titre d'exemple, à la figure 3.22 (a,b,c et d,e,f), les bâtiments correspondant respectivement au campus de l'***Université Lyon 2 à Bron*** et au ***Crayon*** dans le ***quartier d'affaires de la Part-Dieu à Lyon***.

Pour le modèle du campus de Bron, il est composé de :

— 3120 sommets dessinant les toits

— 5838 sommets dessinant les murs

Pour le modèle du Crayon, il est composé de :

— 93 sommets dessinant les toits

— 1104 sommets dessinant les murs

Ces différentes opérations fusionnées ont un temps de calcul variable, selon, d'une part la complexité de la scène, et d'autre part, le nombre de sommets composant du modèle 3D. Les tableaux 3.3 et 3.4 présentent les temps d'exécution obtenus dans chacun des deux cas, illustrés à la figure 3.22, et ce pour les deux différentes plateformes de tests : iPhone 6 et iPhone 7 Plus. Nous y calculons la moyenne, le maximum et le minimum des temps d'exécution sur un corpus de 550 images.

image	moyenne	maximum	minimum
(a), (b) et (c)	19 ms	30 ms	14 ms
(d), (e) et (f)	20 ms	29 ms	13 ms

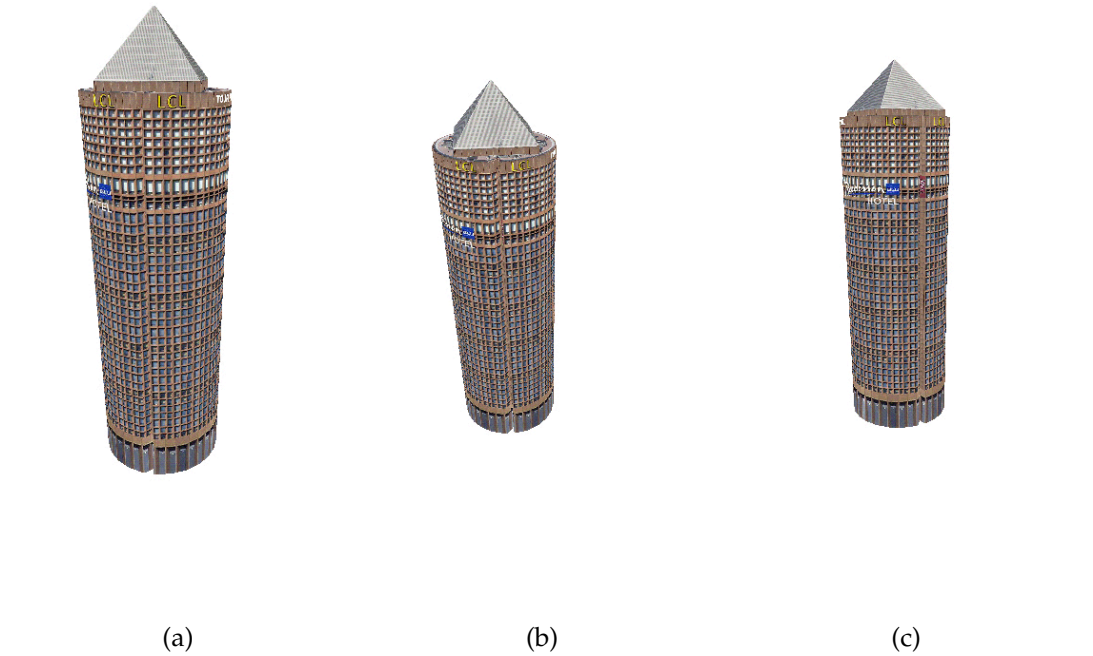
TABLE 3.3 – *Temps d'exécution iPhone 6*

image	moyenne	maximum	minimum
(a), (b) et (c)	12 ms	22 ms	9 ms
(d), (e) et (f)	11 ms	21 ms	8 ms

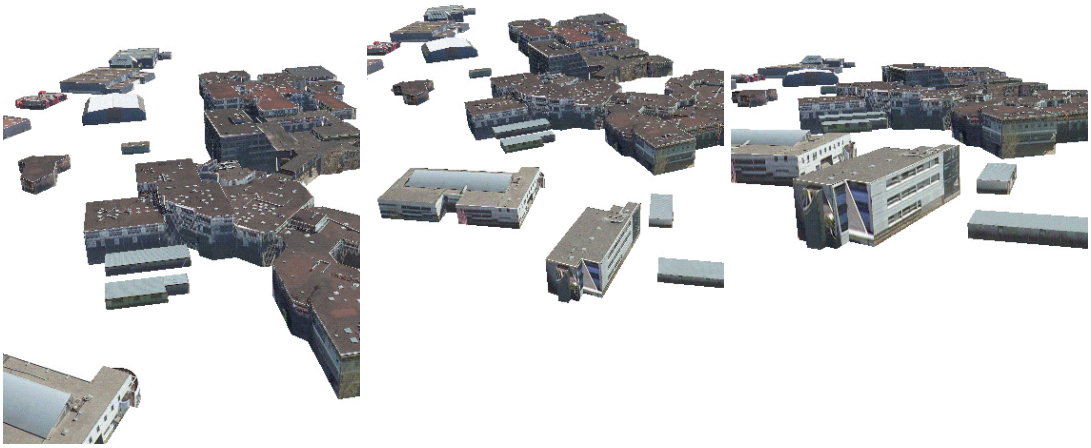
TABLE 3.4 – *Temps d'exécution iPhone 7 Plus*

CONCLUSION

Dans ce chapitre, nous nous sommes intéressé aux approches de RA basée capteurs. Ces différentes approches utilisent **différents capteurs**, à **différents niveaux**



(a) (b) (c)



(d) (e) (f)

FIGURE 3.22 – *Rendus virtuels*
(a-c) Rendu virtuel de la tour Crayon à Lyon
(d-f) Rendu virtuel du campus de l'Université Lyon 2 à Bron

d'échelle, pour **différents besoins** et à **différentes étapes** du processus. Ces différentes approches sont privilégiées dans les contextes extérieurs, où les approches basées vision donnent des résultats variables selon la situation. En effet, dans certaines situations, les approches basées vision proposent des résultats extrêmement bons pour le suivi de points caractéristiques dans l'image avec lesquelles approches

basées capteurs ne peuvent rivaliser. Dans d'autres situations, les conditions de l'extérieur influent tellement sur les résultats que ces systèmes divergent, d'où un besoin de réinitialisation manuelle, comme discuté dans la partie 3.2 du présent chapitre. Nous avons donc présenté dans cette partie, l'ensemble des approches basées capteurs pour l'estimation de la pose, avec une étape de correction basée vision, ou le contraire.

Puis, nous avons proposé une étude sur les différents capteurs utilisés tout au long du processus, et ce afin de comprendre leur fonctionnement, le type de données qu'ils fournissent, leurs différents systèmes de coordonnées, etc. Cette étude nous a permis, grâce aux calculs détaillés de la partie 3.4.3, de pouvoir combiner toutes ces données et d'en estimer une pose, qui sera le point de départ pour l'ensemble des traitements que nous proposerons dans le chapitre 5. En effet, nous avons opté pour un système de RA hybride, où, dans ce chapitre, nous ne faisons intervenir que les capteurs pour l'estimation de la pose, plus ou moins précise, qui sera ensuite corrigée grâce à notre système basé vision en chapitre 5.

Ensuite, nous avons présenté les données utilisées et l'architecture complète du système proposé, avec un échange fluide des données entre le client (smartphone) et le serveur (base du modèle 3D de la ville).

Enfin, nous avons présenté les différentes expérimentations qui ont été conduites, notre protocole d'évaluation ainsi que les résultats sur les performances de l'approche que nous avons mise au point. Ces résultats, préliminaires, seront améliorés lorsque le système basé vision entrera en action. Les différentes résultats de ces deux approches seront comparés entre eux pour démontrer l'apport du skyline dans le processus d'augmentation en termes de précision et de stabilité.

Dans le chapitre qui va suivre, nous allons proposer une méthode pour l'extraction du skyline, qui sera la base de notre système de vision. Nous présenterons donc cet objet immatériel avec une étude bibliographique, puis l'approche que nous proposons pour son extraction et son utilisation comme marqueur pour la RA.

4 Extraction du skyline

Plan du chapitre

Le présent chapitre propose un tour d'horizon sur les méthodes d'extraction du skyline. Il traite également des problématiques sous-jacentes nous permettant de comprendre et ainsi pouvoir manipuler cet objet immatériel et donc de cerner le contexte de cette thèse. En effet, le skyline a déjà été utilisé à plusieurs fins dans les travaux antérieurs (géolocalisation, description géométrique d'une scène, recalage d'images, etc.). Pour nous, il (le skyline) nous est tout aussi important, en proposant de l'utiliser comme un marqueur pour la RA mobile en contexte urbain.

Pour commencer, nous nous intéresserons aux différents algorithmes d'extraction du skyline. Nous dresserons, tout d'abord, un **état de l'art** sur les méthodes existantes qui se déclinent en deux grandes familles : les **approches basées contours** et les **approches basées régions**. Nous présenterons ensuite une **étude comparative** sur la base de cette étude bibliographique et nous présenterons les limites des méthodes existantes. La seconde partie du chapitre détaillera les grandes lignes de l'approche que nous proposons ainsi qu'une liste des problématiques traitées tout au long de sa mise en œuvre. Pour finir, nous présenterons les résultats obtenus, le protocole d'évaluation proposé ainsi qu'une conclusion et une discussion sur tout ce qui est établit.

4.1 INTRODUCTION

Le processus d'extraction du skyline a été largement étudié en littérature. L'idée n'est pas donc nouvelle. Plusieurs disciplines s'y sont intéressé : géographie, photographie, informatique, urbanisme, architecture, etc. Évidement, chacune en a une définition et une perception différente, comme ce qui sera discuté par la suite.

Le besoin initial motivant nos travaux de recherche a été exprimé par des urbanistes et géographes afin de décrire, en termes de caractéristiques géométriques, la forme globale créée par les bâtiments artificiels dans les zones urbaines. Pour cela, le skyline, est l'outil immatériel permettant d'établir cette caractérisation.

Dans les différents contextes où le skyline a été étudié, l'étape d'extraction n'est qu'un premier pas à différentes fins : la navigation de drones dans des environnements naturels ; la géo-localisation de l'utilisateur en environnements naturels ou urbains ; le recalage d'images synthétiques avec des images réelles ; etc. Et dans la plupart de ces études, l'horizon est défini, de façon générale, comme la *limite entre les objets terrestres et le ciel*. Nous l'interprétons donc, en *traitement d'image*, comme un problème de segmentation d'images. Néanmoins, les différents contextes, environnements ou conditions (etc.), prouvent que cette définition ne répond pas forcément à tous les besoins, dont les notre. Nous avons donc besoin d'une *définition plus opérationnelle* du skyline, permettant de fournir le skyline le mieux adapté à la grande variabilité des scènes, aux différentes conditions d'observation et en fonction de l'intérêt de l'utilisateur (horizon lointain formés par des montagnes en arrière plan, bâtiments intermédiaires, constructions proches, etc.). De ce fait, plusieurs skylines peuvent exister, à différents niveaux de profondeur, rendant son extraction plus variable et un peu plus complexe.

De plus, nos différents travaux s'insèrent comme une suite au projet *ANR Skyline* porté par les géographes et urbanistes. Les interactions entre les différentes disciplines concernées a permis de faire émerger le besoin d'outils et de techniques spécifiques, susceptibles d'avoir des retombées sur l'ensemble des communautés concernées. L'un de ces outils est un outil paramétrique d'extraction du (des) skyline(s) afin de caractériser géométriquement le paysage visualisé.

L'objectif de ce chapitre est de proposer une approche paramétrique permettant d'extraire le skyline. Nous ne nous intéressons pas à la caractérisation géométrique, qui fait l'objet d'une autre thèse parallèlement à celle-ci, où des variables objectives (extraite du skyline) sont reliées à des évaluations subjectives traitées dans le cadre du projet *Skyline*.

Nous proposons tout d'abord une approche paramétrique permettant d'extraire plusieurs skylines à plusieurs niveaux de profondeurs. Ceci nous permet de

nous adapter aux différentes conditions atmosphériques, aux bruits éventuels dans l'image et aux différentes conditions d'acquisition d'image (orientation, etc.). Plus tard, nous l'utiliserons comme un marqueur pour notre application de réalité augmentée mobile, qui a pour objectif d'aider les urbanistes, les architectes et essentiellement le grand public à visualiser l'impact de différents scénarios de construction dans la ville.

Ce chapitre est organisé comme suit : nous commençons tout d'abord par définir le skyline en illustrant, à partir des résultats de nos travaux, les différents skylines correspondants à chacune des définitions. Nous utilisons, certes un peu tôt nos résultats, à des fin de compréhension de la suite de la thèse. Puis, nous proposons un tour d'horizon sur les différentes approches qui existent aujourd'hui et les limitations justifiant le besoin d'une nouvelle méthode. Ensuite, nous détaillons l'approche que nous avons mis au point. Enfin, nous présentons nos résultats et des discussions sur ce qui est proposé.

4.2 DÉFINITIONS

Il nous faut, tout d'abord, définir le ***skyline***. Dans la littérature, plusieurs définitions existent pour cet objet immatériel. De manière théorique et même philosophique, dans Yusoff et al. (2014), le skyline est défini comme **l'empreinte unique d'une ville** et comme un objet qui ***abstrait l'identité d'une ville en termes de ses structures spatiales, historiques, sociales, culturelles et économiques au fil du temps***. D'autres définitions plus opérationnelles existent :

— Le contour unidimensionnel qui représente la frontière entre le ciel et les objets terrestre [Johns et Dudek (2006)] ;
— L'horizon artificiel que crée la structure globale d'une ville [1].

Comme discuté dans la partie 4.3 du présent chapitre, plusieurs travaux se sont intéressés à cette problématique. La plupart utilisent la première définition, qui résout le problème de l'extraction et le ramène à un problème de segmentation d'image en deux classes : ciel et non-ciel (objets terrestres). Cette première définition ne prend donc pas en considération le type des objets terrestres : bâtiments, montagne, etc.

Cependant, du point de vue des géographes, et du notre aussi, cette première définition n'est pas satisfaisante. En effet, dans une même image, nous pourrions extraire plusieurs skylines à plusieurs niveaux de profondeurs. La deuxième définition le prouve, en limitant les objets terrestres à ce qui est fait par la main de l'homme (les objets artificiels, la ville et les bâtiments).

1. See Wikipédia : https ://en.wikipedia.org/wiki/Skyline

Afin de mieux comprendre ces deux définitions, nous illustrons à la figure 4.1, le résultat d'extraction des skylines pour deux images de la ville de *Vancouver* au *Canada*, nommées *Vancouver 1* et *Vancouver 2*. Pour chacune, les skylines sont extraits grâce à la méthode que nous proposons. Les résultats sont différents à chaque fois et permettent de caractériser le paysage différemment.

(a) Skyline naturel - Vancouver 1

(b) Skyline urbain - Vancouver 1

(c) Skyline naturel - Vancouver 2

(d) Skyline urbain - Vancouver 2

FIGURE 4.1 – *Résultats extraction skyline - Vancouver 1 et 2*

Cette première étape d'extraction n'est qu'un première étape dans un processus complet. Le skyline obtenu peut être utilisé dans plusieurs applications : la géo-localisation, la détection d'obstacles pour la conduite de véhicules autonomes aériens (drones) ou terrestres (robots), caractérisation de paysage, amélioration du positionnement, etc. Nous proposons dans ce qui suit une revue de la littérature pour des différentes méthodes d'extraction existantes aujourd'hui.

4.3 Revue de la littérature

L'idée d'extraction du skyline n'est pas nouvelle. Au cours des deux dernières décennies, il y a eu un intérêt croissant pour trouver des algorithmes et des techniques pour extraire automatiquement l'horizon dans des images numériques. L'extraction d'horizon n'est qu'un premier pas dans une chaîne entière où la forme spécifique de l'horizon peut agir comme une signature pour la localisation géo-spatiale. Plusieurs méthodes proposent de baser les algorithmes d'extraction sur les méthodes classiques de traitement d'images, et d'autres, plus récentes, utilisent les réseaux de neurones et l'apprentissage profond (en Anglais *deep learning*) pour cette même tâche.

Nous nous intéressons, tout d'abord, à cette étape d'extraction. Pour cela, les travaux antérieurs sont clairement considérés comme des problèmes de segmentation d'images et classés en deux grandes familles :

— Méthodes basées régions ;
— Méthodes basées contours ;

Dans ce qui suit, nous proposons une synthèse de ces différentes approches. Plusieurs d'entre elles ont des points et des notions communes. Nous nous concentrons sur les spécificités de chacune d'entre elles.

4.3.1 Méthodes basées régions

Plusieurs travaux d'extraction du skyline orientés régions existent dans la littérature : Fang et al. (1993), Luo et Etz (2002), Baatz et al. (2012) et Liu et al. (2017).

Fang et al.

L'approche proposée par Fang et al. (1993) fait partie des tous premiers travaux qui se sont intéressé à l'utilisation du skyline comme un outil d'aide à la correction du positionnement. Elle est pionnière dans ce domaine et l'ensemble des travaux postérieurs s'en sont vu inspirés, y compris les notre.

Dans Fang et al. (1993), les auteurs proposent une méthode d'extraction du skyline basée région. Cette approche est composée de deux phases : une phase *hors ligne* et une phase *en ligne*.

— La phase hors ligne est considérée comme une phase de préparation de données. Une vidéo de la scène y est d'abord capturée et enregistrée. Ensuite, chacune des images de la vidéo est traitée et son skyline est extrait. Une base de données est alors créée dans laquelle chaque image est associée à son skyline et à la position à laquelle elle a été acquise.

— Plus tard, dans la phase en ligne, lorsque le véhicule explore la scène, les images en temps réel sont traitées afin d'en extraire leurs skylines. Une comparaison entre ce dernier et tous ceux présents dans la base de données est effectuée. La meilleure correspondance donne l'emplacement le plus probable du véhicule dans le réseau routier.

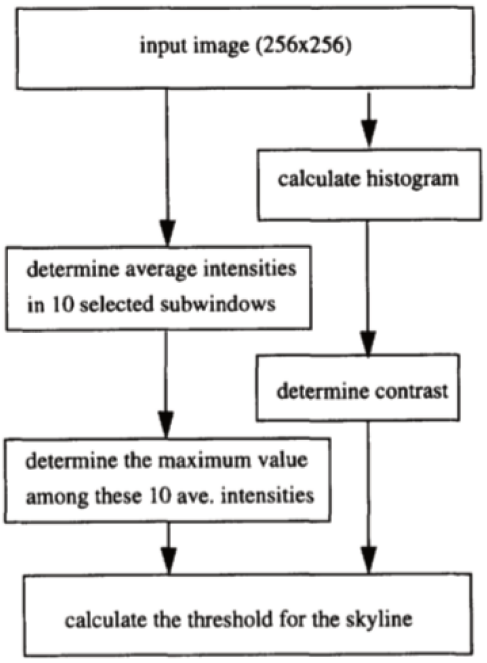


FIGURE 4.2 – *Extrait de Fang et al. (1993)*

L'algorithme d'extraction repose sur un seuillage de l'intensité des gradients. En effet, les images sont tout d'abord redimensionnées en taille 256x256 afin de réduire les temps de calcul. Puis, à partie de la région supérieure de l'image, dix petites sous-fenêtres de taille (10 x 5 pixels) sont réparties de manière équidistante. Les intensités moyennes de chacune de ces dix sous-fenêtres sont calculées et la valeur maximale d'entre elles est déterminée. En même temps, un histogramme de l'image est calculé afin d'en déterminer le contraste. Ce qui est gênant, c'est que, sans tout autre détail, ces travaux déterminent le seuil final en combinant l'intensité moyenne maximale des dix-sous fenêtres avec le contraste de l'image. Un schéma de principe est fourni à la figure 4.2, décrivant la procédure suivie pour déterminer la valeur de ce seuil.

Le principe qui est repris dans l'ensemble des travaux est le suivant : // Une fois ce seuil déterminé, une recherche, colonne par colonne, de haut en bas, est lancée sur l'image. Pour chacune des étapes (colonne), le premier point dont l'intensité est inférieure à la valeur du seuil précédemment déterminé, est considéré comme faisant partie du skyline. L'ensemble de ces points choisis forment un skyline brut. Un filtre médian est ensuite appliqué afin d'obtenir un skyline lissé.

Néanmoins, cette méthode reste simple et donne de moins bons résultats dans les cas où des nuages sont présents et éparpillés dans l'image. En d'autres termes, si des nuages sont présents dans la partie supérieure droite de l'image, mais pas dans la partie gauche, les différents seuils calculés ne permettent pas de segmenter convenablement l'image et ainsi le skyline.

Elle nous semblait tout de même être intéressante vu le pipeline complet qu'elle propose : l'extraction du skyline et son utilisation pour le recalage pour la navigation de véhicules. Un dernier point intéressant sera discuté dans le chapitre 5, où, dans le cadre de l'étape de recalage, l'extraction du skyline est dédoublées pour pouvoir extraire les deux skylines dans les deux directions : en regardant à droite et à gauche. Ceci permet d'augmenter la robustesse de la méthode aux changements de sens de navigation du véhicule (vers l'avant ou vers l'arrière).

Luo et al.

L'algorithme proposé par Luo et Etz (2002) repose sur l'utilisation d'un réseaux de neurones dont le résultat est passé comme entrée à une méthode d'extraction de région fournissant plusieurs skylines possible qui sont filtrés (approuvés ou rejetés) par un modèle issu de la physique.

Cette approche est donc divisée en trois étapes :

— Tout d'abord, un réseau de neurone est entrainé permettant de fournir, au niveau régions et non pas au niveau pixel, une *carte de croyance* d'appartenance à la classe ciel. Ce réseau est basé sur les caractéristiques de couleur. À chaque région de l'image, le réseau associe une valeur plus ou moins grande, selon sa probabilité d'appartenance au ciel. L'utilisation unique de ce réseau basé sur les caractéristiques de couleurs est insuffisante. Pour cela, les auteurs rajoutent deux autres étapes.

— Le résultat fourni précédemment est une carte de croyance. L'objectif ici est de trouver un seuil pour cette carte afin d'en générer une carte binaire où : un pixel à "1" est potentiellement un bon candidat à faire partie de la classe ciel, et un pixel à "0" fait certainement partie de la classe non-ciel. Pour cela, l'histogramme de la carte de croyance est calculé afin d'en déduire le nombre de régions contenant du ciel. La figure 4.3 présente l'exemple utilisé dans ces travaux pour déterminer le seuil de cette carte de croyance. Dans cette figure, nous constatons une première *vallée* suivie de *trois sommets*. Dans une image où il existe uniquement une seule région ciel et une autre non-ciel, nous aurions dû avoir uniquement deux sommets avec une vallée entre les deux. Néanmoins, dans l'exemple de la figure 4.3, les trois sommets ne signifient pas que l'image contient deux ciels, mais plutôt plusieurs régions séparées contenant

du *bleu ciel* (un lac, etc.). De ce fait, le seuil global qui sera utilisé est celui de la première vallée trouvée dans cet histogramme, et par conséquent, plusieurs skylines vont exister.
Les auteurs ont fait le choix de garder l'ensemble de ces candidats au skyline, et de laisser la tâche de les discriminer (accepter, rejeter) au modèle physique de la dernière étape.
Cette étape de recherche de seuil est suivie d'une étape d'analyse en composantes connexe, où des les composantes sont reliées entre elles et affinés grâce à aux opérations de *"Split and Merge"*. Ces opérations de ***croissance de région*** permettent de combler les trous et étendre les limites : les pixels peuvent avoir des valeurs de probabilité d'appartenance au ciel ne dépassant pas le seuil global précédemment déterminé, mais sont assez proches des valeurs de croyance de leurs voisins qui eux ont réussi à passer ce seuillage. Pour les pixels avec des valeurs de croyance inférieures au seuil, la croissance de la région est guidée par la continuité des valeurs de croyance ainsi que la continuité des valeurs de couleur.

— Enfin, un modèle issu de la physique est construit permettant de fournir un ensemble de règles, qui elles-même permettent de dire si une trace ressemble à une trace du ciel ou pas (skyline ou pas). Ce modèle permet de valider la signature du ciel et de déterminer le skyline.

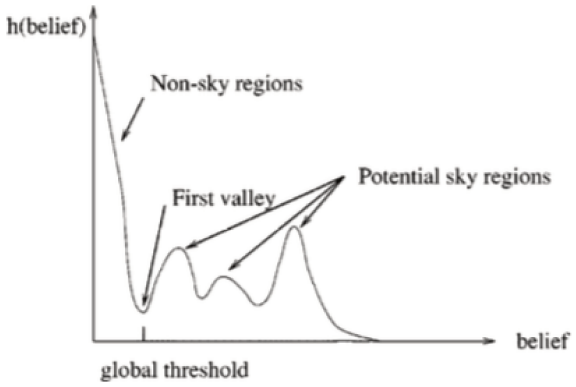


FIGURE 4.3 – *Approche de détermination du seuil de* **?**

L'inconvénient de cette méthode est l'utilisation d'un modèle de la physique afin de valider et discriminer les candidats, et ainsi dire si c'est un bon skyline ou pas. Cette approche dans un contexte urbain ne peut être utilisée vu la grande variabilité des scène : bâtiment de tout genre (type), architecture particulière, alterner l'absence et la présence de bâtiments, présence d'éléments artificiels et naturels (bâtiments + végétation), etc. De plus, l

Saurer, Baatz et al.

Dans une première version en Baatz et al. (2012), puis une seconde en Saurer et al. (2016), les auteurs présentent une approche automatique pour la géo-localisation visuelle à très grande échelle d'une image quelconque dans un contexte naturel. Dans cette approche, les auteurs exploitent les informations visuelles (contours) et les contraintes géométriques (orientation cohérente) en même temps. En effet, l'approche proposée est basée sur de l'apprentissage automatique utilisant précisément un *SVM (Support Vector Machine)*, pour segmenter l'image et extraire l'horizon. Un SVM est un classifieur permettant de discriminer et classifier et donc de segmenter l'image, grâce à des descripteurs particuliers. Les descripteurs utilisés dans cette étude sont essentiellement basés sur la **couleur**, les caractéristiques **statiques** et les informations de **localisation des arêtes**. Cette étape est suivie d'une étape de programmation dynamique pour relier les discontinuités, précédemment extraite par le classifieur.

La première partie de cette approche d'extraction basée région, est assez proche de celle qu'on propose dans nos travaux, ainsi qu'à celles de Lie et al. (2005) et Bazin et al. (2009). En effet, pour chaque colonne de l'image, du haut vers le bas, l'objectif est de trouver le premier candidat répondant à un critère particulier. Pour cela, une fonction d'énergie est proposée, permettant de calculer un coût à chaque point. Ce coût est proportionnel à la somme de tous les pixels en dessous et en dessus de ce candidat. L'hypothèse qui devra donc être vérifiée est : en traversant le skyline (la ligne d'horizon) extrait, il devrait y avoir une évidence locale en terme de gradient orthogonal entre ce qui en est au dessous et en dessous.

La fonction d'énergie considérée est donnée en équation 4.1 :

$$E = \sum_{x=1}^{width} E_d(x) + \lambda * \sum_{x=1}^{width-1} E_s(x, x+1) \tag{4.1}$$

Liu et al.

Dans Liu et al. (2017), les auteurs proposent une méthode pour l'extraction du skyline en quatre temps.

D'abord, une première étape de pré-traitement (redimensionnement de l'image, etc...) qui a pour objectif principale de réduire les temps de calcul de l'algorithme afin de toujours respecter les exigences temps réel. En effet, le contexte dans lequel se situent ces travaux est la prédiction de tout changement d'attitude d'un avion à travers son système de commande de vol automatique (drone). Pour cela, les temps de calculs global des différents algorithmes mis en jeu doit être inférieure à celui de

l'arrivée des images (fllux vidéo), à savoir 33 milli-secondes. Ceci n'est pas considéré comme du temps réel, mais plutôt du temps interactif.

Puis, une deuxième étape consiste à donner une sélection des points candidats à faire partie du skyline. Cette étape se base sur un traitement colonne par colonne de l'image, en recherchant, pour chacune, les zones homogènes. Ceci résulte en un nombre de zones, noté n, pour chaque colonne notée m.

Ensuite, la troisième étape consiste à filtrer les candidats précédemment identifiés, en éliminant ceux qui ont réussi à passer l'étape précédente mais qui restent tout de même improbables. Ce filtrage se fait selon un critère d'adjacence. En effet, comme le skyline a une région lumineuse au-dessus de lui et une région sombre en dessous, les points candidats pour faire partie du skyline doivent avoir une intensité (valeur) de pixels plus élevée dans la région au-dessus. De ce fait, un pixel du skyline doit présenter un changement d'intensité entre ses pixels et les pixels qui lui sont voisins, sur la même colonne ;

Enfin, la dernière étape est la connexion des points candidats au skyline. En effet, il résulte de l'étape précédente un ensemble de points formant le skyline, horizontalement successifs, mais verticalement séparés. Plusieurs cas, selon l'amplitude du saut vertical, sont traités afin de connecter cet ensemble de points.

4.3.2 Méthodes basées contours

Lie et al.

Dans Lie et al. (2005), la méthode proposée se décline en deux étapes : une première étape qui consiste à détecter les contours avec un seuillage sur les intensités des gradients, suivi d'une étape de programmation dynamique.

En effet, la première étape de détection de contours permet d'obtenir gradients des arêtes supérieures au seuil fixé.

La deuxième étape, qui nous intéresse dans ce qui est présenté dans ces travaux, est une étape de programmation dynamique. En effet, le skyline étant défini par un ensemble de points appartenant au contours détecté, s'étend souvent d'un côté de l'image à l'autre. L'extraction du skyline est donc modélisée comme un problème de programmation dynamique qui recherche un chemin en optimisant les coûts. L'idée est de construire un graphe permettant de trouver ce chemin. Tout d'abord, le graphe est construit sur la base de l'image des gradients obtenue. Ce graphe est noté $G = V, L, \phi, \psi$ tel que :

V est l'ensemble des sommets $v_{i,j}$ composants le graphe, et qui correspondent au pixels $b_{i,j}$ de l'image des gradients tel que $v_{i,j} = b_{i,j} = 1$ ou 0 selon si le pixel en question appartient au contour ou pas.

L est l'ensemble des liens entre deux sommets consécutifs. Dans ce graphe, chaque sommet $v_{i,j}$ et chaque lien $l_{h,k,j}$ entre deux sommets consécutifs ($V1_{h,j}$ et $V2_{k,j+1}$) ont un coût, tel que :

Le coût d'un sommet est : $v_{i,j} = \begin{cases} (i+1)^2 & \text{si } j = 1 ou 0 \\ 0 & \text{sinon.} \end{cases}$

Le coût d'un lien est : $\phi(h,k,j) = \begin{cases} |h-k| & \text{si } b_{h,j} = b_{k,j+1} = 1 et |h-k| < \sigma \\ \infty & \text{sinon.} \end{cases}$

où σ est un seuil prédéfini, illustré à la figure 4.4

Un coût nul est donné au tout premier et dernier sommets, notés s et t, tel que tout sommet du deuxième ou de l'avant dernier étage est nul. En d'autres termes : $\psi(s,k,0) = \psi(h,t,N) = 0$, avec N la longueur du graphe (largeur de l'image).

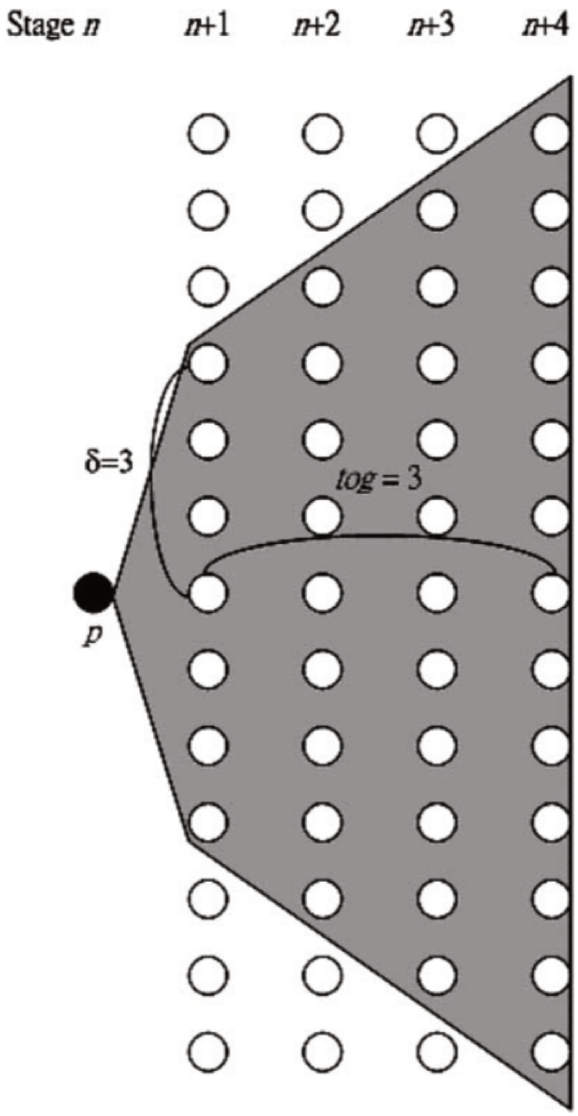


FIGURE 4.4 – *Graphe extrait de Lie et al. (2005)*

Tout d'abord, l'exploration dans ce graphe n'est autorisée que dans un seul sens. En effet, partant d'un sommet, et voulant rejoindre le sommet suivant, une explora-

tion n'est permise que vers la droite. De ce fait, les concavités ou convexités, dues à des problèmes de perspective lors de la prise de vue ou à des architectures originales, ne peuvent être extraites. On peut citer à titre d'exemple, les porte à faux, que nous verrons plus en détails dans la partie 4.4.2 du présent chapitre, et que l'approche que nous proposons est capable d'extraire.

Ensuite, dans cette approche, lorsqu'un chemin ne peut être établit entre deux sommets consécutifs séparés par une distance verticale supérieure au seuil prédéfini, une interpolation linéaire est effectuée. Cela ne reflète pas dans tous les cas la forme réelle de la scène.

Enfin, le coût des liens entre deux sommets consécutifs V1 et V2 est calculé en se basant sur la position du sommet cible (V2). En effet, le coût est d'autant plus grand, que les deux pixels V1 et V2 sont verticalement loin l'un de l'autre. Ce choix a une influence forte sur la capacité de l'algorithme à détecter des sauts verticaux importants, qui sont particulièrement présente dans des images urbaines (formées par les bâtiments).

Ramalingam et al.

L'approche proposée dans Ramalingam et al. (2009) et Ramalingam et al. (2010) concerne, contrairement à ce qui est étudiée dans cette thèse, des omnidirectionnels. En effet, au lieu d'avoir une image dans laquelle le skyline se trouve dans la partie supérieure de l'image, ou à droite ou à gauche dans le cas d'une image en paysage, les auteurs traitent des images dans lesquelles le skyline est au milieu de l'image, entouré par les bâtiments. Nous illustrons à la figure 4.5, un exemple de ce type de skyline. Pour avoir un tel résultat, les auteurs placent une caméra omnidirectionnelle (en anglais *"fisheye camera"*) regardant vers le ciel selon l'axe Z.

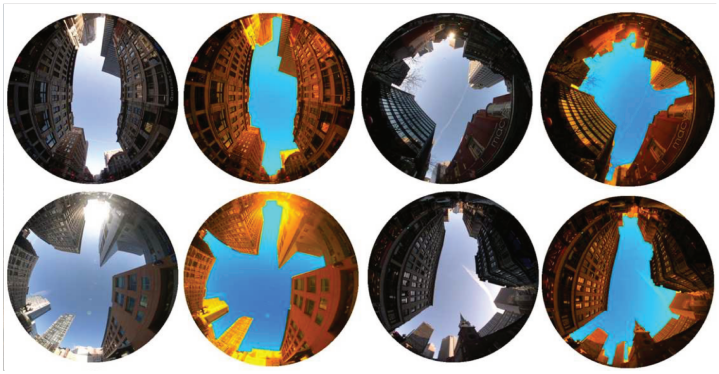


FIGURE 4.5 – *Sky detection results : Original and segmented fisheye images are shown*

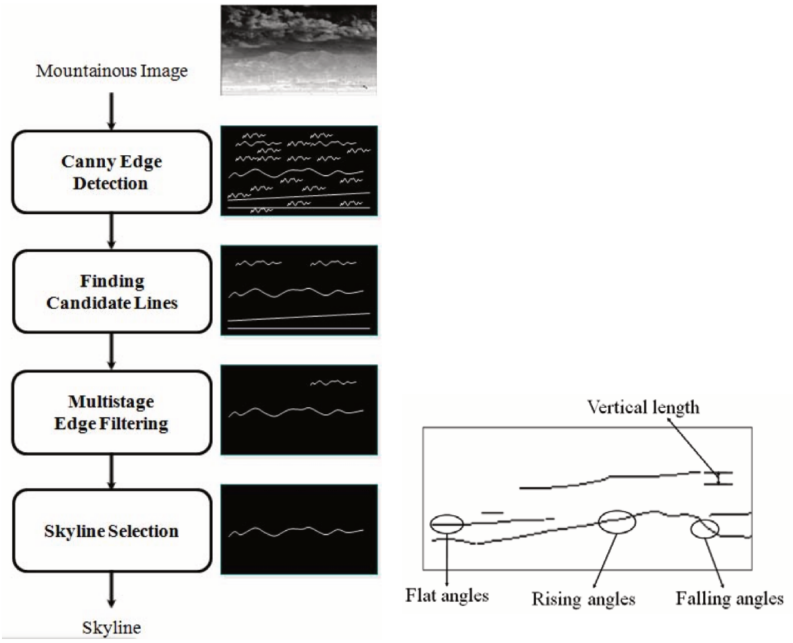
L'approche proposée pour l'extraction du skyline se fait en deux temps. La première étape est une étape d'apprentissage dans laquelle le modèle proposé est entrainé sur un corpus d'images de vérité terrain segmentée manuellement. Cet en-

trainement permet de trouver le paramètre de la fonction d'énergie proposée. La seconde étape consiste en une optimisation des paramètres trouvés, et ce en proposant un modèle gaussien pour les classes ciel et non-ciel et en calculant la moyenne et la covariance à l'aide des images de vérité terrain sur lesquelles le modèle a été entrainé.

Kim et al.

Dans Kim et al. (2011) , une méthode d'extraction de skyline a été développée basée sur une technique de filtrage en plusieurs étapes (en anglais *"MEF : Multistage Edge Filtering"*).

Dans un premier temps, un filtrage grâce au filtre de Canny est appliqué aux images d'entrée suivi d'une étape d'élimination des candidats au skyline basée sur la longueur horizontale de chacun. Enfin, le skyline est sélectionné parmi les candidats restants basé sur une mesure bien définie. à la figure 4.6, l'algorithme proposé montrant l'ensemble des étapes suivies ainsi que les critères de selection des différents candidats.



(a) (b)

FIGURE 4.6 – *Extraits de Kim et al. (2011) : (a) algorithme, (b) critères de selection*

La première étape de cet algorithme est l'application d'un filtre de canny avec des petits paramètres afin de garder tous les candidats au skyline. Tout d'abord, l'étape de recherche des candidats est basée sur l'élimination des lignes horizontales dont la longueur est plus petite qu'un seuil défini. Puis, vient l'étape d'élimination des candidats aberrants basée sur trois mesures particulières :

Chaque pixel candidat au skyline est appelé : ***horixel*** *(pour* ***Hori****zon Pi****xel****).*

Angle horizontal L'idée poursuivie est intéressante : tous les *horixels* composant *une ligne candidate à être le skyline* sont filtrés par le filtre de *Sobel* de taille 3x3 afin d'extraire l'angle de chacun des *horixel*. Cette orientation est calculée selon 8 angles en degrés : 0, 22.5, 45, 67.5, 90, 112.5, 135 et 157.5, appelés $MOD(k)$ avec $k \in [0..7]$. Étant en contexte naturel, les auteurs font l'hypothèse que le skyline a tendance à être horizontal, ce qui ne serait pas le cas dans un environnement urbain. De ce fait, si l'angle *"0"* est dominant par rapport à tous les autres angles définis, ce candidat au skyline est, pour l'instant, retenu.

Ratio de symétrie Les skylines n'étant jamais une ligne horizontale parfaite mais contenant nécessairement des oscillations plus ou moins importantes, ascendantes ou descendantes. Un ratio est calculé selon l'équation 4.2. De ce fait, si ce ratio est plus petit qu'un deuxième seuil préalablement défini, c'est que ce candidat pourrait sûrement faire un bon skyline.

$$Ratio = \frac{min[\sum_{i=1,2,3} MOD(k), \sum_{i=1,2,3} MOD(k)]}{max[\sum_{i=4,5,6} MOD(k), \sum_{i=1,2,3} MOD(k)]} \tag{4.2}$$

Mesure de l'étendu vertical Si un candidat au skyline présente une étendue verticale beaucoup trop courte, les auteurs estiment qu'il peut s'agir d'un nuage ou tout autre bruit. Pour cela, si cette étendue est inférieure à un seuil, le candidat est éliminé. Cette étendue n'est autre que la différence entre la plus grande ordonnée et la plus petite des "horixels" du candidat.

Sélection finale Une fois tous les différents critères ci-dessus appliqués, la sélection parmi les candidats restant se faire sur :

- La mesure de la longueur horizontale du candidat par rapport à l'image
- Mesure du contraste des zones de l'image au dessus et en dessous du candidat
- Mesure de l'homogénéité des différentes horixels au sein du candidat même
- Le Skyline final répondant le mieux à tous ces critères est choisi.

4.3.3 Apports et besoin d'une nouvelle méthode

Les méthodes proposées dans l'état de l'art ont chacune leurs avantages et inconvénients.

Tout d'abord, dans le système de RA global que nous proposons dans cette thèse, nous avons la chance d'avoir l'utilisateur dans la boucle afin de pouvoir interagir

avec l'application. Puis, les interactions que nous proposons sont très intuitives : deux curseurs correspondants aux seuils haut et bas du filtre de Canny peuvent être affichés et cachés selon le besoin. Nous avons mené plusieurs compagnes de tests dans lesquelles les utilisateurs ont très rapidement pris en main l'application proposée et sont arrivés après quelques essais à extraire le skyline très rapidement, avec une précision permettant d'établir un matching sans grande difficulté (voir chapitre 5).

Ensuite, une fois que l'utilisateur paramètre l'algorithme d'extraction, il peut garder les mêmes paramètres tout au long de sa déambulation dans la ville. En effet, les paramètres permettant de détecter le skyline séparant le ciel des objets terrestres sont généralement les mêmes, vu que les conditions météorologiques (nuages, luminosité, etc.) ne changent presque jamais brusquement. Si cela venait à arriver, un ajustement (affinage) très rapide des paramétrés permet de retrouver le bon skyline à nouveau.

Enfin, cet aspect paramétrique de notre approche nous permet essentiellement de pouvoir nous adapter au besoin de l'utilisateur. En effet, cela nous permet de pouvoir extraire différents skyline à différents niveaux de profondeurs correspondant à des objets proches (bâtiments) ou à des objets de fond (montagne en arrière plan). De plus, les approches ci-dessus mentionnées considèrent que le skyline peut présenter des concavités ou des convexités, mais jamais de V-convexité (détaillée dans la partie 4.4.2). Dans certaines, les courbes raides ne sont pas admises dans la construction des skylines qui sont susceptibles d'être présentes dans les paysages urbains, en raison des structures verticales des bâtiments. De ce fait, l'algorithme qui doit être proposé doit correctement gérer ces différents cas.

Pour finir, le temps d'exécution est un critère important vu le contexte de mobilité que nous proposons, et donc des plateformes mobiles pour lesquelles les performances en calcul et en mémoire sont très importantes.

4.4 Méthode proposée

L'approche que nous proposons dans ce qui suit a pour objectif l'extraction paramétrique du skyline permettant d'extraire, dans une seule image, plusieurs skylines à plusieurs niveaux de profondeur. Cette approche est développée et intégrées dans une application, dont l'objectif global est de faire de la RA en contexte urbain. Cette étape d'extraction du skyline est donc très importante pour la suite des travaux menés.

Néanmoins, ces dernières années ont connues une vraie révolution dans les méthodes de traitement d'image, faisant passer la plupart des travaux menés dans le monde du Deep Learning. En effet, cette étape d'extraction a été traitée dans les travaux les plus récents (de 2014 jusqu'à aujourd'hui) grâce aux réseaux de neurones

de tout types et de toutes sortes. De plus, ces différentes techniques de Machine Learning, qui se présentaient jusqu'à quelques mois uniquement sur des plateformes Desktop dont les puissance de calculs sont très honorable, se déclinent aujourd'hui en des librairies compatibles avec les dispositifs mobile : le réseau est entrainé sur des grandes machines, et le réseau dont la taille n'excède pas les quelques *Mo*, est embarqué dans le smartphone et les tests peuvent donc être menés.

Pour l'approche proposée, et comme en littérature, certaines hypothèses doivent d'abord être vérifiées : nous supposons que la ligne d'horizon est une courbe imaginaire composée de plusieurs points séparant les objets au sol du ciel. Ajouté à cela, le ciel doit être situé dans la partie supérieure de l'image.

Afin d'avoir une idée brève, mais très claire, sur ce qui est proposé ci-dessous, nous proposons la figure 4.7 montrant les différentes de notre algorithme, illustrée avec des exemples d'images et de résultats réels de nos travaux.

Tout d'abord, nous transformons l'image originale (fig :4.7(a)) en niveaux de gris. Puis, nous appliquons un filtrage paramétrique à l'aide du filtre de Cannny. Le réglage manuel de ses deux paramètres permet de cibler différents horizons à différents niveaux de profondeurs (fig : 4.7(b) et 4.7(c)), selon la définition subjective de chaque utilisateur. Ensuite, l'étape d'extraction de l'enveloppe supérieure nous permet d'obtenir un skyline discontinu (fig : 4.7(d) et 4.7(e)). Enfin, nous lançons un algorithme de recherche du plus court chemin dans un graphe, obtenant ainsi un skyline continu (fig : 4.7(g) et 4.7(f)).

Dans les sous-sections suivantes, nous détaillons ces différentes étapes une à une.

4.4.1 Détection des contours et enveloppe supérieure

Dans cette étape, les images d'entrées sont converties dans l'espace colorimétrique de niveaux de gris. Nous appliquons ensuite un filtrage pour détecter les gradients. Le filtre utilisé est le filtre de Canny Canny (1986). L'utilisateur interagit avec ses deux paramètres au travers de deux curseurs. Le choix doit être fait de sorte à conserver le plus de pixels faisant réellement partie du skyline. À partir de cette image, nous extrayons une enveloppe supérieure afin que ses pixels en soient connectés (voir partie 4.4.3).

Détection de contours :

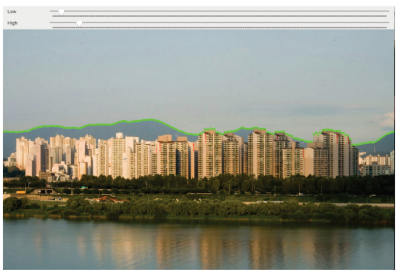
Dans la méthode que nous proposons, nous basons notre processus de détection de contours sur le filtre de Canny, qui a deux paramètres : un paramètre haut, et un paramètre bas ;

Ces deux paramètres permettent de sélectionner les points qui vont faire partie des contours finaux, candidats au skyline. Une contrainte est à prendre en considé-

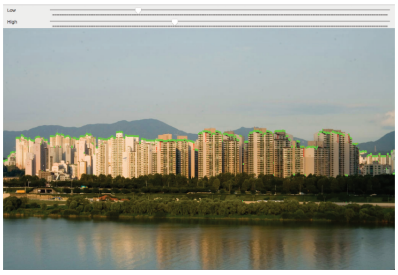
(a) Vancouver original image

(b) Canny (64,26)

(c) Canny (292,205)



(d) Natural upper envelope



(e) Urban upper envelope

(f) Natural skyline

(g) Urban skyline

FIGURE 4.7 – *Approche détaillée*

ration : le seuil bas doit toujours être inférieur au seuil haut ; Si l'utilisateur fourni le contraire, le filtre inverse automatiquement les valeurs des seuils pour que la contrainte redevienne vérifiée.

Avec ces deux seuils, trois cas se présentent. Si intensité du pixel de l'image est :

— Inférieure au seuil bas, le point est rejeté ;

— Supérieur au seuil haut, le point est accepté dans le contour final

— Entre le seuil bas et le seuil haut, le point est accepté s'il est connecté à un point déjà accepté. Cette étape est appelée chainage.

Dans les figures 4.7(b) et 4.7(c), les deux couples de paramètres choisis sont respectivement : (64,26) et (292,205).
Lorsque ces deux valeurs sont petites nous obtenons une carte des gradients d'intensité où tous les détails les plus fins sont gardés, qui est le cas de la figure 4.7(b). Dans le cas des grandes valeurs du seuil, les détails les plus fins sont supprimés, ne laissant que les gradient les plus forts et donc les contours les plus saignant.

Extraction de l'enveloppe supérieure :

Une fois l'étape de détection de contours appliquée, nous obtenons une image binaire dans laquelle les pixels blancs représentent les contours acceptés. Dans cette étape, nous allons essayer d'extraire une *enveloppe supérieure* pouvant représenter la forme de la scène. En effet, notre traitement se fait selon un balayage de gauche à droite, et un traitement colonne par colonne.

Pour chaque colonne, nous parcours l'image de haut en bas, en ne gardant que le point le plus haut dans l'image de gradient. Un premier ensemble de points est ainsi obtenu.

4.4.2 V-convexité

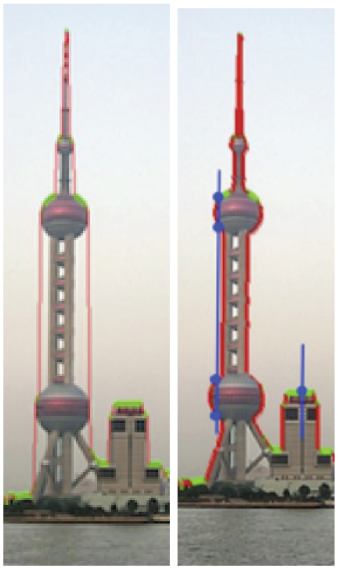
Une fois l'enveloppe supérieure extraite, on obtient un ensemble de points qui présente l'inconvénient d'être déconnecté puisque deux colonnes voisines de l'image peuvent définir des points verticalement très éloignés, essentiellement dans notre contexte urbain, où les courbes raides dues à la présence de bâtiments avec une très grande hauteur.

Une première approche naïve nous permettrait, par exemple, de les connecter par un segment de droite pour obtenir une courbe continue. Cette approche n'est évidemment pas satisfaisante. En effet, la ligne d'horizon extraite alors ne reflète pas toujours la forme réelle des objets de la scène visualisée, comme l'illustre la figure 4.8(a). Pour formaliser ce problème, nous avons introduit le concept de ***v-convexité***.

Une ligne d'horizon, qui définit les deux régions *ciel* et *non-ciel*, est dite v-convexe si et seulement si l'intersection de toute ligne verticale avec la région *non-ciel* est limitée à ***un seul segment de droite***.

Ces cas où la ligne d'horizon n'est pas ***v-convexe*** sont dus à plusieurs facteurs :

Tout d'abord, les **problèmes de perspectives** : la **distorsion** de la caméra du dispositif d'acquisition de l'image peut transformer une **ligne verticale (bâtiment) en une ligne légèrement oblique**.



(a) Approximation v-convexe (b) non v-convex skyline

FIGURE 4.8 – *Illustration des non-v-convexité*

Ensuite, l'**orientation de l'image** : l'utilisateur pourrait prendre une image où l'angle d'**inclinaison** de la caméra serait non nul (différent de $\pi/2$).

Enfin, les images peuvent contenir des objets artificiels et des bâtiments communs et architecturale-ment banals, mais aussi des éléments architecturaux présentant des **porte à faux**. Pour illustrer ce principe, nous présentons la figure 4.8(b) présentant la *Pearl TV Tower* à *Shanghai*.

Le skyline est alors appelé non v-convex. Ce principe reste toujours valable dans un contexte naturel, où notre algorithme suit les concavités et les convexités décrites par les arbres, comme illustré dans la figure 4.9(a) et 4.9(b).

(a) Image naturelle (b) Segmentation arbre

FIGURE 4.9 – *Exemple de résultatsArbreEtNaturel*

Nous détaillons dans ce qui suit, section 4.4.3, la méthode proposée pour relier les points de l'enveloppe supérieure entre eux, en utilisant un algorithme de recherche

du plus court chemin dans un graphe, basé sur une méthode de programmation dynamique.

4.4.3 Liaison des discontinuités

Cette dernière étape est destinée à la liaison des discontinuités. En effet, comme illustré à la figures 4.9(a) et 4.9(b), les points verts représentent les pixels de l'enveloppe supérieure, qui doivent être connectés pour obtenir une courbe continue définissant le skyline. Les segments de droite de couleur rouge représentent le chemin suivi par l'algorithme de recherche, ci-dessous détaillé.

4.4.3.1 Programmation dynamique

Nous traitons chacune de ces discontinuités comme un graphe pondéré. Tout d'abord, nous utilisons les pixels de l'enveloppe supérieure, précédemment calculée en section 4.4.1, pour la construction de ce graphe.

Sur ce graphe, on dénote :

— w = largeur du graphe, égale à la largeur de l'image

— h = hauteur du graphe, égale à la hauteur de l'image

— Chaque pixel est noté " $p_{i,j}$ " où : $p_{i,j}$ = 1 ou 0, avec i = *dans* [1. .w] et j = *dans* [1..h].

Nous construisons alors ce graphe, illustré à la figure 4.10.

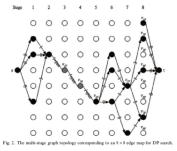


FIGURE 4.10 – *graphe de liaison des discontinuités*

Les composants de ce graphe, noté G, sont {V, E, ψ } :

1. **V = sommet :** Chaque point $p_{i,j}$ dans l'image (pixels verts dans la figure **??** ou **??**) correspond à un sommet $v_{i,j}$ en V ;

$$V(i,j) = \begin{cases} \textit{point appartenant au contour}, & \textit{if b(i,j)=1} \\ \textit{point n'appartenant pas au contour}, & \textit{if b(i,j)=0} \end{cases} \tag{4.3}$$

2. **E = Points du contour :** Aux stades consécutifs (i et $i+1$), ayant les deux sommets $V(i,j)$ et $V(i+1,k)$, un le lien doit être établi. Ainsi, en fonction de certains critères définis ci-après, deux cas sont possibles :

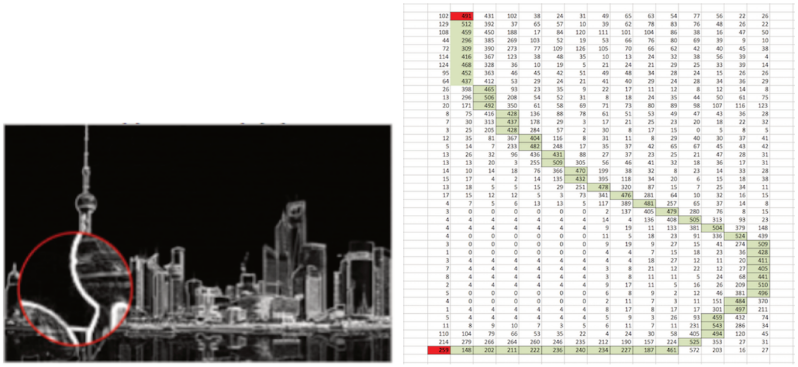
   $|k-j| < 2$: Le saut vertical entre ces deux sommets horizontalement consécutifs est inférieur à deux pixels : dans ce cas, aucun coût n'est calculé, et une simple ligne entre les deux est dessinée.

$|k-j| \geq 2$: Le saut vertical est supérieur à deux pixels : nous utilisons un algorithme de recherche du plus court chemin en utilisant une fonction de coût permettant de trouver le chemin optimal entre le sommet de d'origine $V(i,j)$ et le sommet cible $V(i+1,k)$.

3. ψ **= fonction de coût** : les coûts sont calculés en utilisant la magnitude du gradient de l'image originale filtrée avec le filtre *Sobel*, comme illustré dans la figure 4.11(a).

   Dans cette recherche, chaque pixel est connecté à ses 8 voisins directs. Pour chacun d'entre eux, nous associons une valeur de coût en fonction du niveau de contraste de l'image filtrée avec le filtre de Sobel. Le coût est d'autant plus faible que le contraste est élevé.

   Nous illustrons, à titre d'exemple, à la figure 4.11(b), grâce à une feuille Excel, le chemin suivi pour l'algorithme de recherche dans le cas de la Pearl TV Toer. Les cellules rouges correspondent aux pixels source et destination. Les cellules vertes correspondent au chemin trouvé. Dans ce cas, nous avons un chemin d'une taille de 52 pixels, où les coordonnées du pixel source étant (184,457), et celles de la cible étant (185,417).



(a) Zoom sur l'image de Sobel (b) Illustration d'une feuille excel avec les coûts

FIGURE 4.11 – *Chemin poursuivi pour un exemple de v-convexité*

4.4.3.2 Complexité

Dans cette partie, nous donnons une analyse de la complexité des algorithmes que nous proposons, étape par étape.

Pour la détection des contours et le seuillage, les temps de calcul ne dépendent que de la dimension des images d'entrées. Le contenu de l'image (texture) n'a pas d'influence sur la complexité de cette partie de l'approche.

(a) Image originale (b) vérité terrain par Baatz et al. (2012) (c) Vérité terrain proposée

FIGURE 4.12 – *Les vérités terrains*

Cependant, le coût de calcul de l'étape de liaison de discontinuité basée sur notre approche de programmation dynamique correspond à la complexité de l'algorithme de Dijkstra. Nous l'avons implémenté en nous basant sur un tas binaire où les nœuds correspondent aux pixels voisins et les arcs au gradient. La complexité, dans le meilleur des cas est de : $O(n^2)$, alors que dans le pire des cas, elle est de O(1), en raison d'une recherche éventuellement étendue. Le complexité en mémoire est de $O(n^2)$.

4.5 RÉSULTATS OBTENUS ET DISCUSSION

4.5.1 Données

Nous avons conduits plusieurs expériences sur plusieurs bases de données.

Tout d'abord, deux bases de données, nommées CH1 et CH2, fournies par Baatz et al. (2012) contenant notamment des images naturelles prises dans des conditions réelles dans les *Alpes*. Les images de vérité de terrain ne sont données que pour la première base de données nommée *CH1*. Nous avons donc dû créé manuellement toutes les vérités terrain pour toute la base de données CH2.

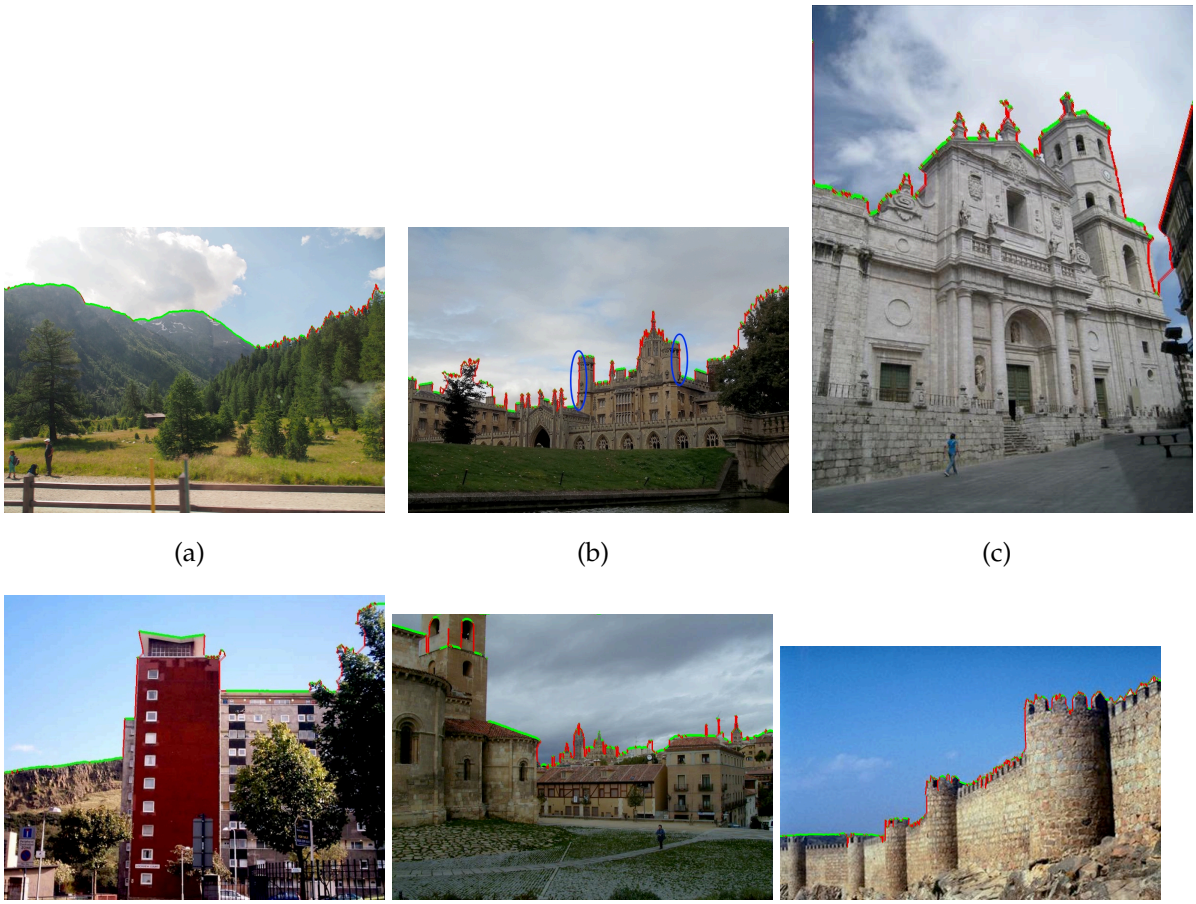
Cependant, pour certaines vérités terrain de la base CH1, nous avons constaté que la segmentation n'était pas aussi parfaite que cela. En effet, cette séparation ciel / non-ciel ne correspond pas vraiment à la forme réelle décrite par les objets de la scène. Nous donnons un exemple à la figure 4.12 illustrant la différence entre notre nouvelle image de vérité de terrain et l'image de vérité terrain fournie par Baatz et al. (2012). En fait, la vérité au sol ne prend pas en considération les concavités et convexités que les éléments de l'image pourraient créer.

Puis, nous avons utilisé la base de données de Tighe et Lazebnik (2010), où des images diverses dans un contexte urbain munies de leurs vérité de terrain sont four-

nies. Les résultats de l'algorithme proposé sont également fournis. Ces images sont surtout prises dans les zones urbaines de la ville de *Barcelone*. Nous avons pu extraire les horizons avec précision dans cet ensemble de données et comparer notre résultat à Tighe et Lazebnik (2010).

Ensuite, nous avons une base de données personnelles d'environ 550 images, que nous mettons à disposition de la communauté.

Toutes ces images sont prises dans différentes conditions (météo, luminosité, bruit...). Nous représentons quelques-unes d'entre elles (différentes conditions, différents contextes, ..) à la figure 4.13, avec les résultats que nous obtenons.



(a) (b) (c)

(d) (e) (f)

FIGURE 4.13 – *Exemple de résultats d'extraction*

4.5.2 Évaluation de l'horizon extrait

Afin d'évaluer les performances des algorithmes d'extraction, nous proposons d'utiliser plusieurs mesures : la *Précision*, le *Rappel*et la *distance de Hamming*. Pour cela nous définissons tout d'abord :

— w est la largeur de l'image ;

— S_g est la ligne d'horizon de la vérité terrain ;

— S_e est la ligne d'horizon extraite.

L'objectif ici étant d'extraire le skyline, le résultat est une image binaire dans laquelle deux classes sont définies : la classe *ciel*, et la classe *non-ciel*, et par conséquent nous notons :

Vrai positif :	un pixel	detecté	**ciel**	qui est réellement	**ciel**
Faux positif :	un pixel	detecté	**ciel**	qui est réellement	**non-ciel**
Vrai négatif :	un pixel	detecté	**non-ciel**	qui est réellement	**non-ciel**
Faux négatif :	un pixel	detecté	**non-ciel**	qui est réellement	**ciel**

Précision :

Nous pourrions définir la précision, dans le cas général, comme étant le rapport entre **le nombre de vraies réponses données** et le *nombre total des réponses possibles*.

La précision est définie dans l'équation 4.4 .

$$Precision_{Ciel} = \frac{\sum VraiPositif}{\sum VraiPositif + \sum FauxPositif}; \tag{4.4}$$

Rappel :

Nous pourrions définir la précision, dans le cas général, comme étant le rapport entre **le nombre de vraies réponses données** et le *nombre total des réponses possibles*.

$$Rappel_{Ciel} = \frac{\sum VraiPositif}{\sum VraiPositif + \sum FauxNegatif}; \tag{4.5}$$

F-mesure :

La F-mesure qui combine la précision et le rappel est leur moyenne harmonique.

$$Compromis = 2 * \frac{Precision_{ciel} * Rappel_{ciel}}{Precision_{ciel} + Rappel_{ciel}} \tag{4.6}$$

Distance de Hamming :

La distance de Hamming nous permet d'avoir la distance entre le skyline extrait et sa vérité terrain, et ce en utilisant l'équation 4.7.

$$hamming = \frac{\sum FauxNegatif + \sum FauxPositif}{w} \tag{4.7}$$

4.5.3 Résultats

Nous calculons pour chaque ensemble de données, l'ensemble des mesures définies, avec leur moyenne, maximum et minimum. Une vue d'ensemble de nos résultats est donnée dans le tableau 4.1, donnant le nombre d'images de chaque ensemble de données, la moyenne et le calcul du temps maximum, la moyenne et le minimum de la **Précision** , du **Rappel** et de la **distance de Hamming**.
De même, nous comparons notre méthode aux différentes méthodes citées.

Afin de générer ces résultats, nous avons mené nos expériences sur une machine de bureau. L'étape d'extraction de l'enveloppe supérieure se fait en temps réel. En fonction de la complexité de la scène, le temps CPU nécessaire pour l'étape de liaison des discontinuités varie entre 0,4 et 19 secondes. Nous avons également mené nos expériences sur un smartphone. L'algorithme fonctionne en temps réel et permet d'extraire l'enveloppe supérieure sans latence. Comme pour l'interface graphique de bureau, les discontinuités de connexion prennent plus de temps et dépendent également des performances du smartphone. Le prototype développé est prêt à être utilisé dans un contexte de réalité augmentée, comme ce qui sera proposée dans le chapitre 5.

		CH1	CH2	Barcelona	
		[*our*]	[*our*]	Tighe et Lazebnik (2010)	[*our*]
images		175	80	52	52
Processing time	average	0,98	0,7 sec		0,71
	max	19 sec	10 sec		5 sec
Precision	average	0,999	0,988	0,959	0,977
	min	0,97	0,293	0,32	0,36
Recall	average	0,99	0,989	0,95	0,55
	min	0,94	0,56	0,95	0,67
Hamming distance	average	2,54	12,59	25,75	23,85
	min	8,31	0,59	143	195

TABLE 4.1 – *Resultats d'extraction du skyline*

CONCLUSION

Dans le présent chapitre, nous venons de proposer une revue de la littérature des différents algorithmes d'extraction du skyline qui sont clairement considérées comme des problèmes de segmentation d'images et classées en deux grandes familles : les approches basées région et celles basées contours. L'approche que nous

proposons est basée contours et se veut paramétrique permettant de fournir le skyline le mieux adapté à la grande variabilité des scènes, des conditions d'observation et des préférences de l'utilisateur (plusieurs skylines peuvent exister, à différents niveaux de profondeur).

En effet, après avoir donné les différentes définitions du skyline, nous avons proposé une revue des approches d'extraction. À partir de ces différentes études, nous avons pu cerner les différentes problématiques et limitations. Comme nous l'avons vu, ces problématiques axent essentiellement sur la robustesse des algorithmes face aux différentes conditions d'acquisition, la possibilité d'extraction des non-v-convexités et l'aptitude à extraire plusieurs skylines selon la perception (définition) que l'utilisateur en a.

Comme nous l'avons cité auparavant, nous nous intéressons au problème de localisation dans le cadre d'un système de RA mobile. Pour cela, ce skyline va nous servir comme marqueur géométrique naturel pour corriger une première pose estimée à l'aide des différents capteurs (voir chapitre 3). Pour ce faire, dans le chapitre qui va suivre, nous allons présenter dans un premier temps les approches de recalage basées sur le skyline. Dans un second temps, nous allons détailler notre approche, les différentes mesures de similarités utilisées et les étapes suivies en termes de simplification et recherche de solution optimale.

5 Matching des skylines

Plan du chapitre

Dans les précédents chapitres, nous avons étudié le skyline. Nous avons commencé tout d'abord par en donner quelques définissions, pour ensuite proposer une approche d'extraction (chapitre 4). Cette étape d'extraction nous a permis d'obtenir, à partir d'images réelles, un skyline réel. Ensuite, à partir des résultats du chapitre 3, où une une image virtuelle est générer, un skyline virtuel est extrait grâce au même algorithme. Dans le présent chapitre, nous nous intéressons aux méthodes de recalage d'images basées sur les caractéristiques géométriques des éléments la scène, et plus particulièrement : le skyline.

Dans un premier temps, nous allons nous intéresser aux approches existantes en littérature. Puis, nous présenterons notre processus de recalage de skylines. Ensuite, nous détaillerons les différentes mesures de similarité proposées ainsi qu'une discussion pour chacune d'entre elles. Enfin, le reste du chapitre détaillera les différentes expérimentations menées et les résultats obtenus ainsi qu'une comparaison avec les méthodes de la littérature.

5.1 INTRODUCTION

Au cours des deux dernières décennies, le besoin d'informations géographiques détaillées (MNT, MNS, Géométrie 3D, texture, etc.) s'est considérablement accru, et essentiellement ceux concernant les modèles 3D de villes. La mise à jour de ces données est de même très importante vu les différents changements et développements qui s'opèrent au sein des villes du monde. Ces différentes données sont acquises et créée grâce aux différentes techniques du numérique, et leurs différents domaines d'application tournent principalement autour de la navigation, la gestion du territoire ou l'évaluation de l'environnement. De plus, les modèles 3D remplacent de plus en plus les données 2D traditionnelles qui ne sont pratiquement plus utilisées de nos jours. Différentes formes de visualisations et d'interactions sont possibles, permettant d'en avoir une meilleure compréhension.

Cette visualisation peut se faire grâce aux techniques de RV (réalité virtuelle) ou de RA (réalité augmentée). En RV, l'environnement réel est modélisé en 3D et visualisé grâce à des casques de RV : toutes la procédure de visualisation est donc virtuelle. En revanche, en RA, l'objet 3D est visualisé en même temps que la scène réelle grâce à une seule image (flux vidéo), et par conséquent une procédure de recalage entre la vidéo du monde réel et l'objet 3D inséré est nécessaire.

Dans ce problème de recalage, ou aussi appelé fusion entre les vidéos et les modèles de bâtiments, le défi majeure consiste à estimer la pose de la caméra. Comme détaillé en chapitre 3, cette pose est estimée et une caméra virtuelle est placée dans le repère du monde à travers les six degrés de liberté estimés. Il est alors facile, grâce aux deux images réelles et virtuelles, de trouver la relation entre un pixel de l'image réelle et son correspondant dans l'image virtuelle. Ce recalage est essentiellement basé sur les caractéristiques des bâtiments visualisés dans l'image : les façades, les bords, les fenêtres ou d'autres caractéristiques particulières qui peuvent être utilisées pour identifier un bâtiment dans l'image. Dans cette thèse, une caractéristique géométrique très particulière est utilisée : le skyline.

5.2 REVUE DE LA LITTÉRATURE

Plusieurs travaux se sont penchés sur la question du recalage d'une image réelle, acquise par la caméra d'un quelconque dispositif, et d'une image de synthèse, générée grâce au modèle 3D des mêmes objets visualisés dans la scène réelle. Dans notre contexte (urbain), ces objets 3D ne sont autres que les bâtiments de la ville, et donc nous aurons besoin du modèle virtuel de la ville. Traditionnellement, les méthodes de recalage font correspondre plusieurs éléments de texture extraits à partir

des images réelles et virtuelles. Ces méthodes sont assez coûteuses en temps de calcul et dépendent fortement de la qualité des images et du type de caractéristiques extraites (SIFT, SURF, etc.). Ces méthodes ne sont pas adaptées à notre contexte pour au moins deux raisons :

— D'une part, dans ce contexte de RA mobile, les performances et l'espace mémoire sont très limités (smartphone, tablettes, etc.). De ce fait, une extraction de points *SIFT* par exemple, qui reste une méthode néanmoins très rapide, ne pourrait être envisagée. L'ensemble des étapes composant le processus est trop lourde (extraction dans l'image réelle, extraction dans l'image virtuel, vecteurs caractéristique de chacun, comparaison, etc.) ;

— D'autre part, les modèles 3D de villes, aujourd'hui démocratisés, sont en général géométriquement plus ou moins précis, mais la qualité des textures est très variable. Ces modèles sont évalués selon les niveaux de détails qu'ils proposent (voir figure 5.1 extrait de Löwner et al. (2013)).

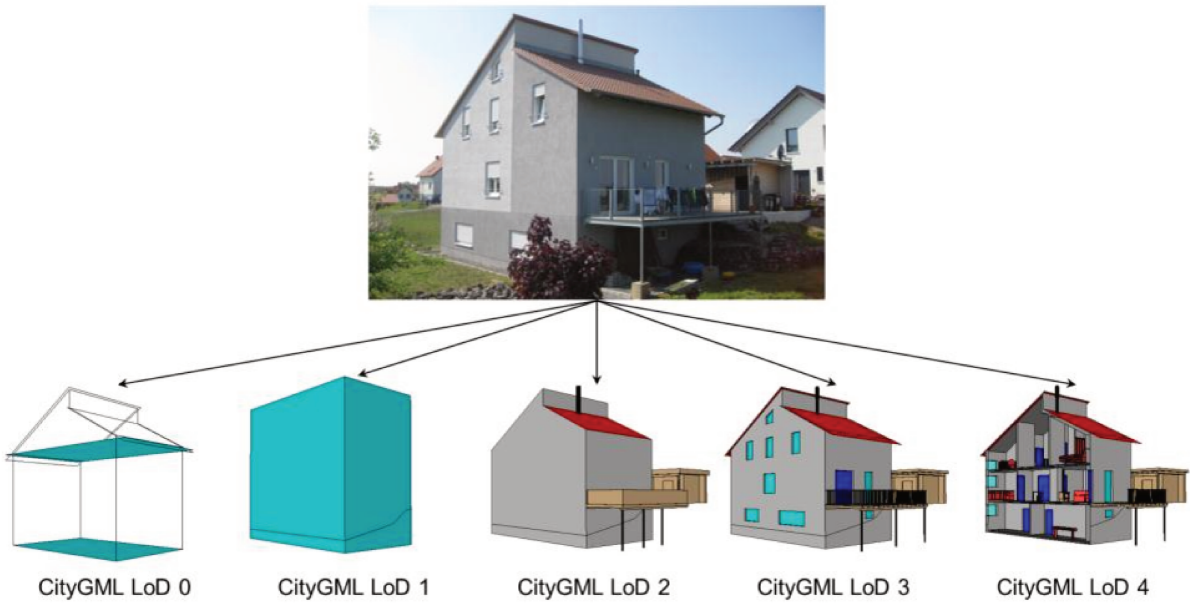


FIGURE 5.1 – *Niveaux de détails, extrait de Löwner et al. (2013)*

Le modèle 3D que nous utilisons pour valider notre thèse est celui de la ville de *Lyon*, qui présente deux niveaux de détails. Ce modèle fournit une géométrie plus ou moins précise munie d'une texture qui n'est pas toujours exacte. Ceci ne nous permet pas d'utiliser des méthodes basées couleur, points d'intérêts, etc.

C'est pour ces raisons, que dans nos travaux, nous proposons d'exploiter certaines ***caractéristiques géométriques*** de la scène : le *skyline*. En effet, la plupart des images acquises en milieu urbain contiennent une ligne d'horizon qui est définie comme la frontière entre la région "ciel" et la région "non ciel". L'hypothèse qui doit toujours

être vérifiée, comme en chapitre 4 est qu'il existe toujours une région significative du ciel à l'intérieur de l'image.

Les approches que nous détaillons ci-après sont connues sous le nom de méthodes *"de matching"* ou *"d'appariement"* ou *"de recalage"* de skylines. Nous étudierons donc les travaux de Fang et al. (1993), de Johns et Dudek (2006), de Ramalingam et al. (2010), de Nüchter et al. (2011), de Baatz et al. (2012), de Zhu et al. (2012), de Zhu et al. (2013) ou de Hofmann et al. (2014).

5.2.1 Fang et al.

L'appariement de skylines proposé dans Fang et al. (1993) est basé sur mesure de similarité assez simple. En effet, pour deux skylines X_1 et X_2, cette mesure est donnée par l'équation 5.1.

$$D(X_1, X_2) = ||X_1 - X_2|| \tag{5.1}$$

Néanmoins, une stratégie qui nous parait intéressante a été adoptée dans le cadre de ces travaux. En effet, il a été remarqué que la précision du positionnement du véhicule basé sur le matching de skyline était fortement influencée par la direction de conduite du véhicule (angle de vision de la caméra). Pour cela, l'expérimentation menée consiste à munir le véhicule de deux capteurs vidéo acquérant les deux skylines des deux côtés (gauche et droit) en même temps, et ce afin de corriger les erreurs de positionnement. Ceci peut être traduit par l'acquisition d'une image omnidirectionnelle, comme ce qui est proposé dans Ramalingam et al. (2010).

Dans la phase hors ligne, une base de données est créée, où, à chaque endroit où sont acquises les images, les deux skylines (droite et gauche) sont extraits et enregistrés.

Dans la phase d'appariement, le skyline instantané est apparié avec chacun des deux skylines (gauche et droit) séparément avec pour but d'obtenir l'horizon correspondant dans la base de données. Deux cas de figure se présentent alors :

— Si le véhicule a la même direction de conduite que celle qu'il avait lors de la construction de la base de données, appelé ***sens avant***, les deux correspondances indiquent approximativement la même position.

— Sinon, si le skyline de droite indique une position du véhicule derrière celle de la ligne d'horizon de gauche, on peut déduire que le véhicule est en train de se déplacer dans le sens contraire, appelé le ***sens arrière***.

En général, en utilisant les informations des skylines de gauche et de droite, la position du véhicule peut être déterminée plus précisément, et la direction de conduite actuelle peut être déduite.

Cette stratégie, couplée avec la mesure de similarité présentée, permet d'avoir une précision longitudinale allant jusqu'à 1 mètre.

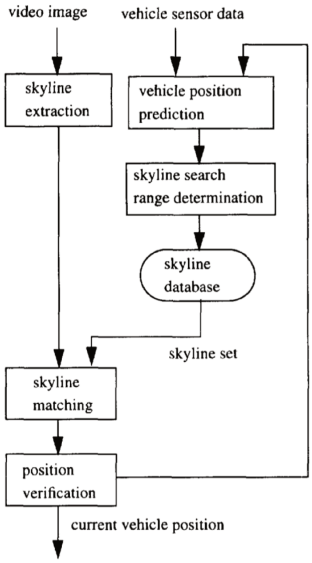


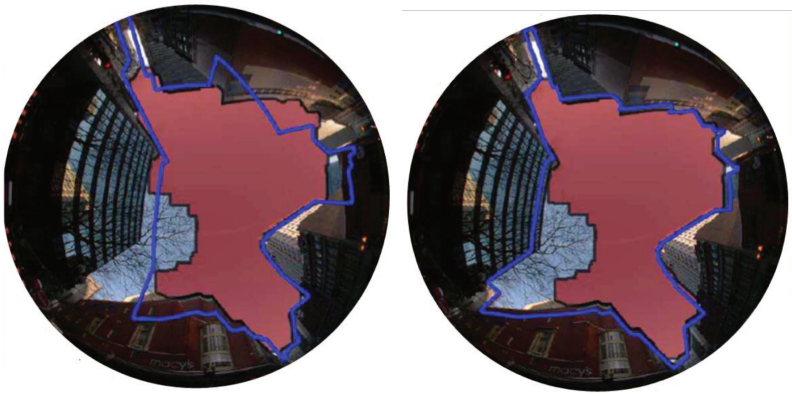
FIGURE 5.2 – *pipeline Fang1993*

5.2.2 Ramalingam et al.

Dans Ramalingam et al. (2010), une première étape d'extraction du skyline est proposée, détaillée en 4.3.2 du chapitre 4. Ci-après, nous détaillons la deuxième étape de l'algorithme proposé concernant le matching des skylines. Les auteurs proposent deux approches pour le matching de skylines : une première approche basée sur un appariement à l'aide de la distance de chanfrein, et une deuxième approche par recherche de plus court chemin.

5.2.2.1 Distance de chanfrein

Les auteurs présentent une méthode basée sur une **distance de chanfrein** (en anglais *"Chamfer distance"*). Cette distance est une distance discrète fournissant une approximation de la distance euclidienne. Cette méthode est possible grâce à la qualité des images acquises.
Le premier appariement entre le skyline réel et virtuel est fait manuellement. Ensuite, la pose de la caméra virtuelle est légèrement variée en changeant les paramètres extrinsèques de la caméra aux alentours de la pose de départ. À chaque fois, la distance est calculée, convergeant vers le minimum recherché.
Afin de gérer les occlusions qui pourraient gêner cette étape de matching, les auteurs combinent cette étape de matching avec un algorithme de RANSAC permettant de supprimer les résultats aberrants (voir figure 5.3).



(a) Skyline détecté par distance de chamfrein (b) Skyline détecté en rajoutant RANSAC

FIGURE 5.3 – *Résultats extrats de Ramalingam et al. (2010)*

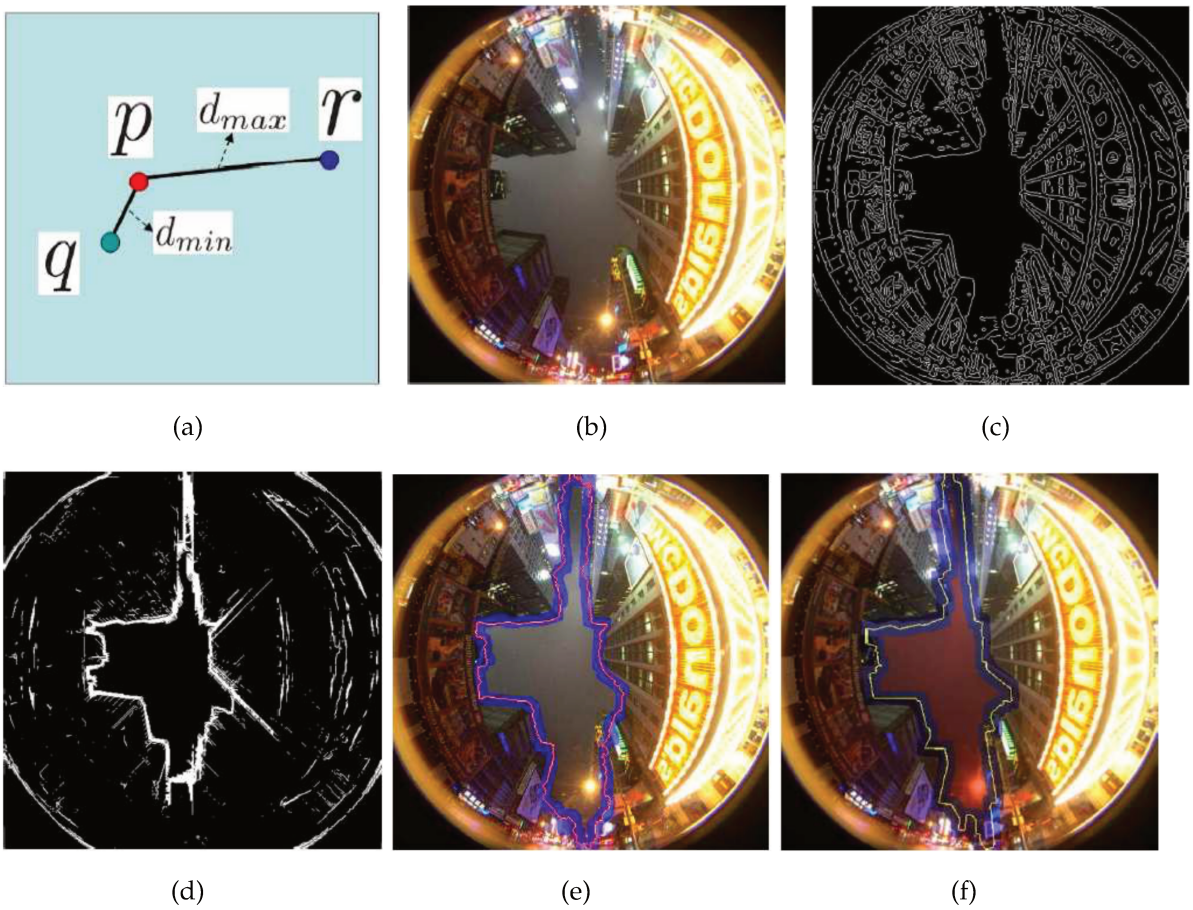
5.2.2.2 Recherche du plus court chemin

Dans cette approche, un ensemble de skylines virtuels est d'abord généré pour diverses poses. Chacun de ces candidats au skyline est apparié au skyline réel et une fonction de coût permet de mesurer une distance entre ces deux images. La figure 5.4 illustre l'approche proposée.

Tout d'abord, (fig.5.4.c) un filtre de *Canny* est appliqué sur l'image réelle (fig.5.4.b) acquise par la caméra *fisheye*. Puis, un opérateur de max-min est appliqué à cette image, en utilisant la relation de l'équation 5.2. Cette approche permet de ne garder que les pixels de contours candidats à faire partie du skyline. En d'autres termes, cela met l'accent sur les limites proches d'un grand espace libre, dans notre cas : la partie centrale étant le ciel. Ensuite, le skyline virtuel à une pose particulière est superposé au skyline réel que nous venons d'extraire.

$$M(p) = \frac{d_{max}}{d_{min}} \tag{5.2}$$

Enfin, à l'aide de la méthode de Mortensen et Barrett (1998), un coût (distance) est calculé entre les deux images des skylines réel et candidat virtuel. L'idée de cette méthode est de traduire ces deux images en deux graphes, dans lesquels les nœuds correspondent aux pixels à l'intérieur des deux skylines (*la partie centrale de l'image délimitée par le skyline*). Les poids entre deux nœuds du graphe est inversement proportionnel à M(p). Grâce à cette fonction de distance, un chemin circulaire à l'intérieur de ces zone est calculé. Le coût du chemin le plus court augmente avec le nombre de discontinuités, ce qui signifie que le meilleur skyline répondant aux critères est celui permettant de trouver le chemin le plus court sans discontinuités.



(a) (b) (c)

(d) (e) (f)

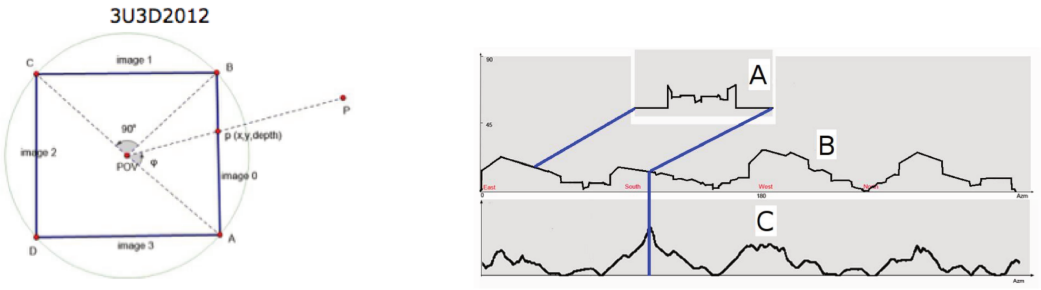
FIGURE 5.4 – *Extrait de Ramalingam et al. (2010)*

5.2.3 Zhu et al.

Les travaux de Zhu et al. (2012) et Zhu et al. (2013), tournent autour de l'extraction et du matching de skyline en environnement urbain, ce qui est très proche de ce qui est proposé dans cette thèse.

L'idée poursuivie dans cette recherche est la recherche du skyline capté par la caméra dans un skyline panoramique généré à partir du modèle Système d'information Géographie (en anglais *"GIS :Geographic Information System"*) et ce à la même position GPS. à la figure 5.5, nous illustrons un des exemples traités dans ces travaux.

En effet, un skyline panoramique (360) est généré autour d'un point de vue (qui est la position de la caméra de l'image réelle). Ensuite, les deux skylines réel et panoramique sont comparés afin de trouver une correspondance entre eux, de sorte que le skyline de l'image réelle corresponde à une sous-partie de l'horizon panoramique. Cette correspondance est effectuée grâce à une mesure de similarité appelée *Cross Similarité*. Cette mesure est donnée par l'équation 5.3, qui est basée sur le calcul de la somme des valeurs absolues (en anglais *"SAD : sum of absolute differences"*).



(a) (b)

FIGURE 5.5 – *Zhu et al. (2013) approche cross similarité*

$$CS(fg)[n] = \sum_{i=\infty}^{\infty} \begin{cases} 1 - \frac{|f[m]-g[n+m]|}{f[m]} & \text{si } 1 - \frac{|f[m]-g[n+m]|}{f[m]} < 0.75 \\ 0 & \text{sinon.} \end{cases} \tag{5.3}$$

Cette étude nous ait parue intéressante à citer vu, tout d'abord le contexte dans lequel elle est proposée (urbain, extérieur), mais surtout afin de comparer nos résultats et les mesures de similarité que nous proposons avec celles qui y sont. Cette comparaison sera détaillée dans la partie 5.4.

5.3 APPROCHE PROPOSÉE

L'approche proposée se décline en plusieurs étapes, détaillées une à une ci-dessous. Nous illustrons, à des fins de clarté, à la figure 5.6(a) et à la figure 5.6(c), deux images qui sont supposés être acquise et générée à la même position de la caméra et au même instant. Néanmoins, les instruments avec lesquels la pose a été estimée et qui a permis de générer l'image virtuelle (figure 5.6(c)) sont entachés d'erreurs, et donc les skylines résultant (figures 5.6(b) et 5.6(d)) sont décalés en translation et en rotation. L'objectif ici est de recaler ce skyline virtuel sur le skyline réel afin de trouver la correction à appliquer à la pose de l'utilisateur. Pour ce faire, nous nous appuyons sur la similitude de leurs formes.

5.3.1 Description de l'approche

L'objectif ici est de déplacer le skyline virtuel dans le plan image selon les paramètres x, y et θ afin de le faire correspondre au skyline réel, et ainsi faire correspondre les coordonnées (pose) estimées aux coordonnées réelles.

Approximations et hypothèses :

(a) Image réelle (b) Image virtuelle (c) Skyline réel (d) Skyline virtuel

— Étant donné que notre système tourne en temps réel (ou du moins en temps interactif), et que la génération de la vue virtuelle suit parfaitement l'arrivée des images réelles (flux vidéo), nous considérons que les rotations dans le repère du monde sont estimées comme des translations dans le plan de l'image.
— L'approche et l'application proposée permet de visualiser un objet 3D (futur bâtiment) incrusté dans la scène réelle, afin d'avoir une idée sur son impact sur le paysage complet. De ce fait, l'utilisateur doit être (position) à une certaine distance de l'objet visualisé et des bâtiments réels. Ainsi, les erreur de positionnement dues aux imprécisions du GPS (qui ne sont pas totalement aberrantes) peuvent être approximées en des translations dans le plan image.

Le problème est alors ramené à la minimisation d'une distance, en fonction des paramètres x, y et θ. L'algorithme de minimisation choisi est une descente de gradient, utilisé à partir des modules d'optimisation de la bibliothèque *OpenCV*, nommé *Downhill Solver*. Les étapes de rendu synthétique, d'extraction des deux skylines et de leurs mise en correspondance sont effectuées en temps réel. Nos premiers tests ont été réalisés sur un iPhone 6. Sur la fin de la thèse, un projet au sein du laboratoire nous a permis l'acquisition d'un iPhone 7 Plus, et les mêmes tests y ont été réalisé, à des fins de comparaisons (temps de calculs, etc.). Nous définissons également trois mesures de similarité pour déterminer la distance entre eux. Il est clair que, si la pose estimée correspondait exactement à la pose réelle de la caméra, ces métriques devraient être nulles. Dans l'autre sens, plus les skylines sont éloignés l'un de l'autre, plus les valeurs de ces métriques sont importantes. De plus, dans notre cas, ces mesures doivent être robustes à de plusieurs facteurs : bruits dus à des erreurs dans le

processus d'extraction du skyline, données 3D incomplètes, présence de végétation, données manquantes, etc.

Le pipeline de notre approche contient 3 grandes étapes : Tout d'abord, nous commençons par un processus de simplification du skyline, détaillé en section 5.3.2. Cela nous permet d'éliminer les pixels aberrants qui proviendraient d'une mauvaise extraction du skyline ou dus à la présence de bruit dans l'image. Ensuite, nous appliquons une approximation polygonale permettant de passer d'un vecteur dont la taille est égale à la largeur de l'image (w) à un vecteur d'une taille beaucoup moins importante (une cinquantaine de points en moyenne). Ceci nous permet de diminuer considérablement la complexité de l'algorithme(de O(n) où n=w à O(n) où n$\simeq$50). Enfin, nous lançons notre algorithme de recherche basé sur l'algorithme du simplexe. Cette recherche repose sur l'une des trois mesures de similarité proposées dans la partie 5.3.3.

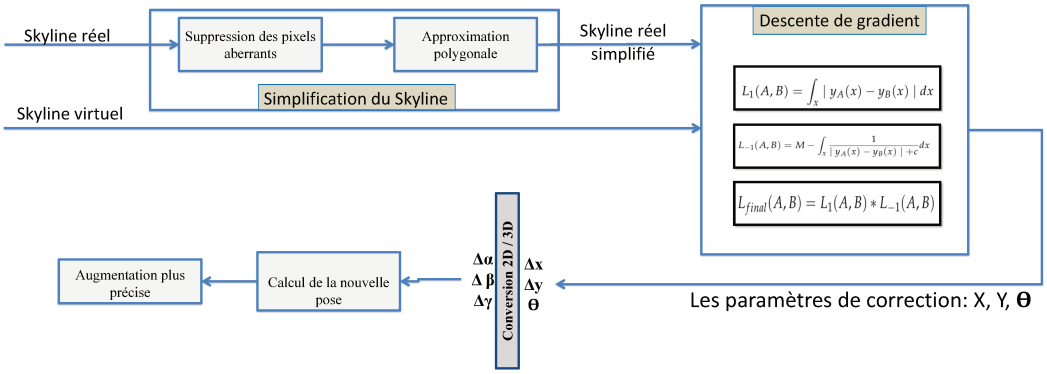


FIGURE 5.6 – *Schéma général de l'approche*

5.3.2 Simplification du skyline

Le processus d'extraction du skyline n'étant pas précis à 100%, les skylines contiennent, dans la plupart des cas, de nombreuses lignes verticales. Ceci est dû à la présence de pixels aberrants à cause de bruits dans l'image (nuages, ...) ou de gradients forts résistant au filtre de *Canny*, comme détaillé dans le chapitre 4. De plus, les deux skylines (réel et virtuel) diffèrent à plusieurs endroits de l'image pour plusieurs raisons, dont principalement la présence d'éléments existant dans le monde réel et non modélisé dans le modèle 3D de la ville. Nous citons, à titre d'exemple, les éléments naturels (arbres, nuages, etc.) ou des éléments artificiels (lampadaires, feu de circulation, etc.). D'autres détails architecturales ne sont pas modélisés dans le modèle de la ville (cheminée, etc.).

De plus, le skyline extrait initialement contient un peu plus de 300 pixels ($\simeq$ largeur de l'image). Lancer notre processus d'appariement (recalage) avec une telle

taille de vecteurs compromettrait la contrainte la plus importante en RA : le temps réel. Un processus permettant la simplification du skyline est alors nécessaire afin d'éliminer les pixels issus de bruits divers, et de réduire la taille du vecteur. Ce processus est divisé en deux grandes étapes : l'élimination des pixels aberrants puis une approximation polygonale du résultat obtenu. Cela réduit le taille des vecteurs à une cinquantaines de points qui définissent toujours très clairement la forme de la scène.

Nous détaillons dans les parties suivantes (5.3.2.1 et 5.3.2.2), le processus de suppression des pixels aberrants suivi d'une approximation polygonale appliquée sur le skyline initial.

5.3.2.1 Suppression des pixels aberrants

Ce processus de suppression des pixel aberrants (en anglais "outliers") est nécessaire. En effet, nous illustrons aux figures 5.7(c) et à 5.7(d), respectivement les deux skylines non simplifiés et simplifiés extraits des deux images réelle et virtuelle des figures 5.7(a) et 5.7(b).

Nous remarquons, à la figure 5.7(c), plusieurs sauts verticaux dans les skylines résultats (en vert, le skyline réel ; en rouge, le syline virtuel). Au cours de nos premières expériences, ces sauts ont fait échouer, de manière systématique, le processus d'appariement selon deux modalités : la contrainte temps réel n'est plus respectée, et même si des fois elle l'avait été, le résultat de l'appariement est absurde.

Durant cette étape de simplification, les deux skylines se voient privés, en moyenne (sur toutes les images de la base), d'une quinzaine de pixels pour les images de skylines réels et de deux pixels pour les images de skylines virtuels. Ceci est compréhensible, vu l'existence de beaucoup plus de bruits dans les images réelles, et un peu moins dans les virtuelles.

Le principe de notre algorithme de simplification est donné dans l'algorithme ci-dessous.

Algorithme 1 : Suppression des pixels aberrants

```
1 for i = 1i++, whilei < sizeof(input) do
2 |   if |input[i+1].y + input[i-1].y - 2 * input[i].y| < precision then
3 |   |   output ← input[i] ;
4 |   end
5 end
```

5.3.2.2 Approximation polygonale

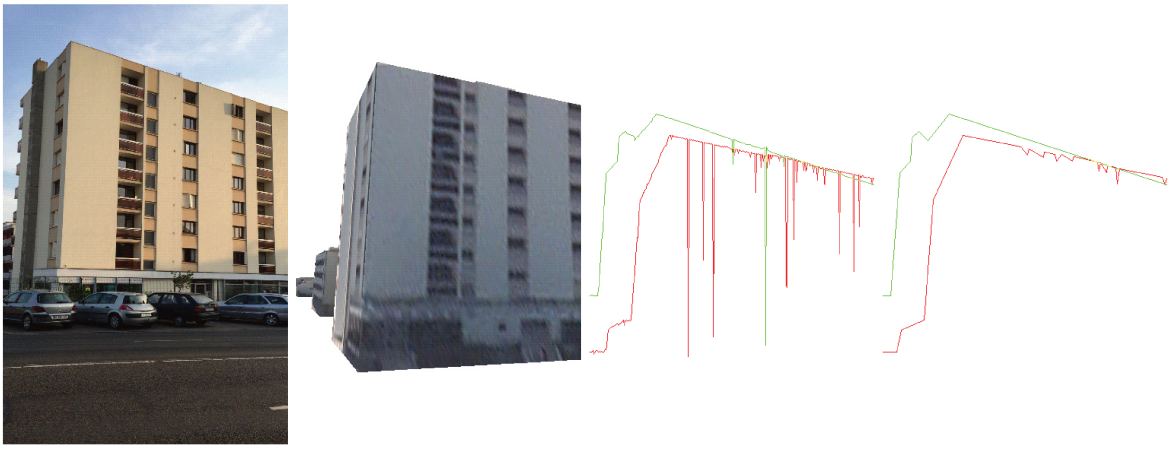
Étant donné la taille du vecteur skyline à la sortie de l'étape de simplification (en moyenne 314 points pour le réel, et 265 pour le virtuel), il nous faut mettre au point une approche permettant de la réduire et ainsi respecter la contrainte temps réel. En effet, dans le tableau 5.1, nous présentons un comparatif des tailles de chacun des vecteurs avant et après chacune des phases de simplification : la taille du vecteur initial, la taille après suppression des pixels aberrants et la taille finale du vecteur après la simplification polygonale pour les différents valeurs de précision. En effet, il est à savoir que cette approximation polygonale dépend d'un paramètre, à savoir la ***précision***. Nous donnons en annexe un détail de l'algorithme d'approximation polygonale utilisé. En effet, ce processus repose sur l'algorithme de *Douglas-Peucker* et permet d'approximer le vecteur d'entrée (skyline initial) en un vecteur avec moins de sommets, de sorte à ce que la distance entre le vecteur d'entrée et le résultat soit inférieure ou égale à la précision spécifiée.

Nous remarquons dans le tableau 5.1, que plus le paramètre de précision augmente plus la taille du vecteur skyline réel diminue considérablement. Par contre, la taille des vecteurs de skylines virtuels ne diminue pas avec le même facteur que celui du réel.

Ceci est compréhensible, vu que dans le monde réel, l'extraction du skyline peut être gênée par une multitude de bruits (nuages, mauvais paramétrage de l'algorithme, etc.) ou par l'aspect endenté des arêtes des bâtiments. D'un autre côté, la simple modélisation des bâtiments virtuels en droites parfaites fait que l'approximation n'a pas une grande influence (et une approximation polygonale d'une droite restera la même).

		image 1		base de données (moyenne)	
		réelle	virtuelle	réels	virtuels
Taille initiale		366	306	342	268
Suppression pixels aberrants		336	300	314	265
Precision		291	214	269	201
	1	122	214	110	200
	2	81	209	65	199
	3	61	209	47	198
	4	43	209	38	198
	5	40	209	30	197

TABLE 5.1 – *Comparatifs des tailles de vecteurs*



(a) imgRéelle - tab 5.1 sans approxPoly (b) imgVirtuelle - tab5.1 avec approxPoly (c) imgSkyRéel - tab5.1 sans approxPoly (d) imgSkyVirtuel - tab 5.1 avec approxPoly

FIGURE 5.7 – *image 1 et 2 du tableau 5.1*

La taille de chacun des vecteurs de skylines (réel et virtuel) est variable et dépend de l'étape de simplification (élimination des aberrations + approximation polygonale). Afin de pouvoir recaler le virtuel sur le réel, nous devons nous munir d'une ou de plusieurs distances (ou mesure de similarité / dissimilarité).

5.3.3 Mesures de similarité

Les mesures de similarité sont assez connues dans le monde de l'appariement d'image. L'intérêt principal est de donner une valeur numérique traduisant la ressemblance entre deux images. Cette notion de ressemblance est totalement subjective. Chaque individu pourrait en donner un classement différent, qu'il pourrait baser sur : la couleur, la forme ou l'utilisation ?

Dans notre problématique de recalage, nous recherchons une mesure qui permet de véfirier la mise en correspondance entre deux skylines. D'autres méthodes proposent une similarité pixel à pixel, basée sur la texture par exemple. Et bien d'autres méthodes que le lecteur pourra trouver dans les travaux de thèse de Chambon (2005) par exemple, ou celles de L. Robinault (2009).

Avant de commencer à présenter les mesures que proposons, nous précisons que, par abus de langage, nous parlerons dans ce qui suit de mesure de *similarité*, alors que réellement certaines d'entre elles sont plutôt des mesures de *dissimilarité*. En effet, pour certaines mesures, la similarité entre les deux courbes augmente lorsque la valeur de la mesure augmente. Dans ce cas, le terme *"mesure de similarité "* prend tout son sens. À l'inverse, d'autres mesures retournent une valeur décroissante lorsque la similarité augmente. Nous devrions alors parler de *"mesure de dissimilarité"*.

Dans cette partie, nous cherchons à minimiser une distance entre deux courbes selon trois paramètres : x, y et θ. La solution préconisée est une recherche dynamique basée sur l'algorithme du *simplexe*, qui sera présenté en détails en annexe , et présenté brièvement dans la partie 5.3.3.1. Les distances utilisées sont celles présentées dans les parties 5.3.3.2, 5.3.3.2 et 5.3.3.4. L'implémentation utilisée est celle fournie par la librairie *OpenCV*.

5.3.3.1 Algorithme de minimisation

L'algorithme de descente de gradient a pour objectif de trouver les valeurs optimales d'une fonction paramétrée. C'est un algorithme itératif, qui cherche à ajuster les paramètres de sorte à minimiser une fonction de coût particulière. Cet algorithme est souvent utilisé en apprentissage machine (en anglais *machine learning*), dans les problèmes de régression linéaire vu la rapidité avec laquelle il arrive à trouver une solution acceptable, et parfois optimale, à des problèmes très complexes . [1].

Pour utiliser un tel algorithme, plusieurs paramètres et fonctions sont à définir :

— La fonction de coût, qui permet de calculer la distance entre les deux entités à recaler, ce qui correspond, en général à des mesures de similarité ;
— Le pas (ou le gradient), avec lequel l'algorithme incrémente ou décrémente la valeur du paramètre ;
— Le ou les critère(s) d'arrêt permettant de dire que la solution optimale a été atteinte.

Le pas :

Le pas d'avancement, ou dans le cas général le gradient, est le pas avec lequel les paramètres de la fonction (les variables) sont incrémentées ou décrémentées. Ce pas nous permet donc d'explorer l'espace des paramètres à la recherche de la solution optimale. Ce pas est différent pour chacun de nos paramètres (x, y et θ). En effet, nous choisissons de faire ***bouger*** *ou* ***glisser*** le skyline dans le plan image, à chaque itération de l'algorithme, d'***un nombre variable de pixels selon l'axe X*** noté α, et d'***un nombre variable de pixels selon l'axe Y***, noté β et de le (skyline) faire ***pivoter*** d' ***un nombre variable de degrés autour l'axe Z***, noté γ.

Nous choisissons de lancer notre algorithme de recherche cinq fois (5 étapes). Au départ, les valeurs de α, β et γ sont très grandes, obtenant ainsi une première solution grossière. Au fur et à mesure des étapes, nous diminuons la valeur des différents "pas", afin d'affiner la solution encore et encore. À la fin, nous obtenons la solution (x, y et θ) permettant de passer du skyline virtuel au skyline réel.

Nous adoptons cette stratégie de recherche à pas variables afin de nous prémunir des minimas locaux dans lesquels l'algorithme pourrait être bloqué. Dans toutes nos

1. https ://eric.univ-lyon2.fr/ricco/cours/slides/gradientdescent.pdf

expérimentations, les étapes d'extraction du skyline réel, de génération d'une vue de synthèse texturée, d'extraction du skyline théorique et de la recherche paramétrée sont faites en temps réel. De ce fait, nous obtenons un rendu en réalité augmentée fluide.

La fonction de coût :

La fonction de coût utilisée dans notre approche est elle aussi variable. En effet, nous en avons testé plusieurs. Nous en détaillons quelques unes dans les parties 5.3.3.2 à 5.3.3.4.

Critères d'arrêt :

Les critères d'arrêts de l'algorithme sont :

— Un nombre maximal de pas (d'itérations), égal à 1000 itérations ;

— La précision (ou la distance entre les courbes)))) au delà de laquelle l'algorithme s'arrête, égale à 0.0001.

Limitation de l'espace de recherche :

Les résultats obtenus dans le chapitre 3 nous permettent de dire que la pose estimée à partir des instruments n'est pas aussi aberrante que cela. En effet, la pose est clairement imprécise et entachée d'erreurs. Néanmoins, la solution optimale (la pose réelle) n'est pas très loin de celle-ci. Pour cela, nous choisissons de limiter notre espace de recherche, uniquement pour les deux variables x et y.

En effet, parmi nos hypothèse, l'utilisateur doit se tenir (se positionner) à une certaine distance des objets visualisés. Nous pouvons donc trouver une relation entre l'erreur maximale dans le plan image et la taille de celle-ci. Dans toutes les expérimentations menées, à différents endroits, à différentes conditions et à différentes distances des objets visualisés, nous remarquons que l'erreur en translation est du même ordre de grandeur que la moitié de l'image. Nous faisons l'hypothèse que l'erreur maximale en x et y, dans le plan image, est toujours strictement inférieure à la moitié de la largeur / hauteur de l'image.

5.3.3.2 Distance L1

La première idée explorée est d'utiliser la distance L_1, définie par l'équation 5.4.

$$L_1(A,B) = \int_x \mid y_A(x) - y_B(x) \mid dx \tag{5.4}$$

Cette distance est tout simplement la surface entre les deux courbes et nous donne des résultats satisfaisants pour des horizons décalés mais qui restent similaires. Cette première métrique nous donne d'assez bons résultats pour certains cas. Sauf que, les données d'entrée (les deux skylines) contiennent parfois des différences importantes, et ce dû à plusieurs raisons :

— Le manque de précision dans la modélisation des bâtiments, qui sont parfois réduits à leur forme architecturale la plus simple ;
— Certains bâtiments récents ne sont pas encore représentés dans le modèle 3D (version actuelle de 2015) ;
— L'absence d'éléments artificiels, qui agissent sur la forme finale du skyline réel : lampadaires, câbles électriques, lignes électrifiées (tramway, etc.), feu de circulation et autres artefacts dans le modèle 3D ;
— L'absence d'éléments naturels : arbres, nuages, etc.

Lorsque les skylines présentent de telles différences, le processus d'appariement échoue. Pour cela, nous introduisons une deuxième mesure de similarité.

5.3.3.3 Distance L_{-1}

L'objectif ici est de trouver une distance permettant, d'une part, de gérer les différences citées dans la partie 5.3.3.2, et d'autre part, ayant les caractéristiques suivantes :

— Les courbes très proches sont fortement valorisées ;
— Les courbes éloignées ne sont pas trop pénalisées : les courbes doivent pouvoir s'écarter en un point si cela leur permet d'être identiques en beaucoup d'autres ;
— Les lignes éloignées et les lignes très éloignées ont la même importance.

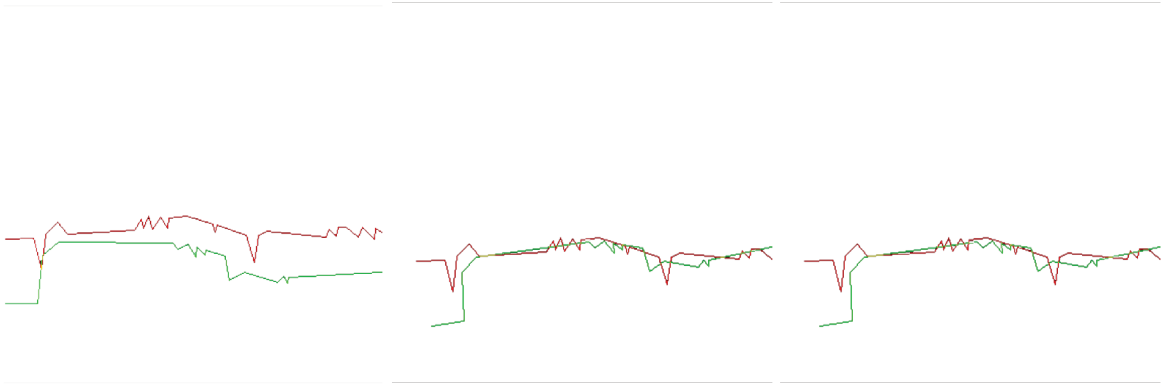
Parmi les métriques testées, une respecte bien ces critères. Nous la définissons dans l'équation 5.5.

$$L_{-1}(A,B) = M - \int_x \frac{1}{\mid y_A(x) - y_B(x) \mid + c} dx \tag{5.5}$$

Nous désignons A et B les deux skylines réels et virtuels. La constante c évite les divergences et permet d'ajuster le comportement : plus elle sera petit, plus elle appliquera fortement les critères au risque de créer beaucoup de minimums locaux affectant la convergence des algorithmes de minimisation. Au contraire, plus elle sera grande, plus on se rapprochera du comportement de la distance L_1.
M est une autre constante qui permet de vérifier que L_{-1} est toujours positive.

Cette mesure suit bien les critères définis ci-dessus et est résiste beaucoup mieux aux écarts parfois importants entre les skylines, où la mesure L_1 échoue. La figure 5.8(a), présente l'image des deux skylines de départ, recalés avec la mesure L_1 à la figure 5.8(b) et avec L_{-1} à la figure 5.8(c).

Cependant, cette mesure a tendance à ne pas bien faire correspondre les skylines contenant des lignes verticales. D'une part elle répond aux critères mentionnés ci-

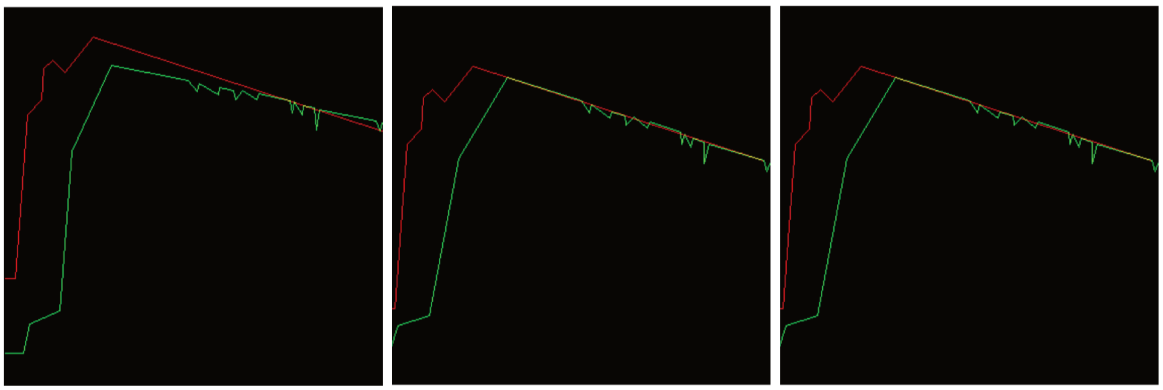


(a) Skylines de départ (b) Recalage avec L_1 (c) Recalage avec L_{-1}

FIGURE 5.8 – *Recalage avec les mesures L_1 et L_{-1}*

dessous, mais néanmoins échoue pour des cas dont les entrées (les deux skylines) sont assez simples, et où notre première métrique L_1 trouve de meilleures solutions.

Il est donc nécessaire de proposer encore une fois une nouvelles métrique pour pouvoir l'utiliser.



(a) Skylines de départ (b) Recalage avec L_1 (c) Recalage avec L_{-1}

FIGURE 5.9 – *Recalage avec les mesures L_1 et L_{-1}*

5.3.3.4 Distance finale

La première distance fonctionne assez bien pour les cas simples (skylines similaires), mais pas pour les cas compliqués. À l'inverse, la deuxième distance fonctionne mieux pour les cas compliqués (skylines différents en termes de forme), mais pas pour tous les cas simples. Pour améliorer les résultats, il est proposé tout simplement de composer ces deux distances (mesures).

Traditionnellement, pour de combiner deux fonctions (dans le cas général) en une seule, il est de nature à les composer linéairement en y rajoutant des pondérations par l'intermédiaire de paramètres α et β. Par exemple :

$$L_{final} = \alpha * L_1 + \beta * L_{-1}. \tag{5.6}$$

Deux hypothèses différentes pourraient être formulées dans ce sens :

— Nous avons plus de cas simples que de cas compliqués, et donc le choix serait fait de sorte à augmenter la valeur de α et diminuer celle de β : α=0,7 et β = 0,3 ;

— Inversement : α=0,3 et β = 0,7

Néanmoins, nous avons choisi de nous abstraire de telles hypothèses, et proposer une distance finale, détaillée en équation 5.7.

$$L_{final}(A, B) = L_1(A, B) * L_{-1}(A, B) \tag{5.7}$$

5.4 RÉSULTATS ET DISCUSSIONS

Comme ce qui a été discuté dans la partie 3.5.1 du chapitre 3, le protocole d'évaluation de notre système repose sur deux critères : la précision, au travers du calcul de **l'erreur de reprojection**, et le stabilité au travers du calcul de **la plage de flottement**.

La première partie de cette thèse, consistant à estimer la pose de l'utilisateur basée sur la fusion des différents données d'instruments, nous a permis d'avoir de premiers résultats en termes d'augmentation d'images. Les résultats de cette approche sont donnés dans les figures 5.10(b), 5.13(b), 5.13(e), 5.13(h) et 5.13(k).

Ces résultats nous permettent d'extraire les skylines réel et virtuels (lignes rouges et vertes). Puis, le skyline virtuel sera recalé sur le skyline réel afin de trouver la correction à appliquer permettant de rapprocher la pose estimée de la pose réelle. Cette étape de recalage est réalisée grâce aux métriques précédemment décrites.

5.4.1 Évaluation manuelle

Nous illustrons, à titre d'exemple, à la figure 5.10(c) le résultat du recalage utilisant la métrique L_1, à la figure 5.10(d) celui avec la métrique L_{-1} et à la figure 5.10(e) avec L_{final}. les coordonnées des points réel et virtuels sont respectivement :

— Pour L_1 : réel (247,165), virtuel (248,111) ==> (Δx, Δy) = (1, 54)

— Pour L_{-1} : réel (244,172), virtuel (206,138) ==> (Δx, Δy) = (38, 34)

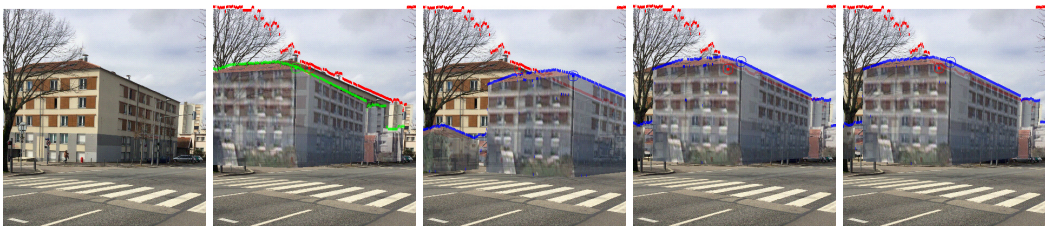
— Pour L_{final} : réel (245,166), virtuel (244,150) ==> (Δx, Δy) = (1, 16)

(a) Original image (b) AR with instruments (c) Match L_1 (d) Match L_{-1} (e) Match L_{final}

FIGURE 5.10 – *Recalage avec différentes métriques image 46*

(a) Original image (b) AR with instruments (c) Match L_1 (d) Match L_{-1} (e) Match L_{final}

FIGURE 5.11 – *Recalage avec différentes métriques image 62*



(a) Original image (b) AR with instruments (c) Match L_1 (d) Match L_{-1} (e) Match L_{final}

FIGURE 5.12 – *Recalage avec différentes métriques, image 58*

5.4.2 Comparaison au résultat utilisateur

Sur un corpus de 550 images, nous détaillons en tableau 5.2 la moyenne et l'écart type des erreur de reprojection selon les axes X et Y, pour les images illustrés dans les figures 5.10 et 5.11.

(a) Original image (b) AR with instruments (c) Match L_{final} (d) Original image (e) AR with instruments (f) Match L_{final}

(g) Original image (h) AR with instruments (i) Match L_{final} (j) Original image (k) AR with instruments (l) Match L_{final}

FIGURE 5.13 – *Résultats des images 16,17, 20 et 43*

image			Fukuda et al. (2014)	ours to real image			ours to GT-users		
				L_1	L_{-1}	L_{final}	L_1	L_{-1}	L_{final}
img.1	Δ X		28,5	-98	-15,66	-20,5	-106	-29,25	-30,66
	Δ Y		-73,5	56,33	32,5	49	45	41	46,66
img.2	Δ X		-3,5	-9	6,33	8,25	-10	9,75	8,66
	Δ Y		11	30	-12	-7,33	30,5	-8,66 7	-6,66
database (550 images)	Δ X	Absolute average	10,49	3,79	10,48	3,11	4,24	9,46	1,95
		Std Dev	70,49	82,52	66,98	66,30	92,99	61,61	59,89
	Δ Y	Absolute average	15,58	20,66	14,25	14,65	24,65	17,30	18,60
		Std Dev	18,83	36,07	28,55	28,97	38,82	32,66	30,82

TABLE 5.2 – *Reprojection error*

image		3D_pano				real-Pano			
		Zhu et al. (2013)	L_1	L_{-1}	L_{final}	Zhu et al. (2013)	L_1	L_{-1}	L_{final}
img 3		2576	3	3	2286	702	2828	2154	2154
img 37		1160	0	0	592	303	205	380	391
database	average	640	7	7	1154	619	610	918	947
	stdev	628	5	5	709	519	618	711	673

TABLE 5.3 – *Erreur en pixels*

(a) Real image (b) Virtual image (c) Real image (d) Virtual image (e) Real image (f) Virtual image

(g) Panoramic image

FIGURE 5.14 – *Panoramic matching*

5.4.3 Évaluation panoramique

Afin d'évaluer les performances de nos mesures de similarité, nous les comparons aux travaux de Zhu et al. (2013). En effet, dans cette approche, une mesure de similarité reponsant sur une formule de Cross-simlarité est proposée. Pour cela, il nous faut tout d'abord générer un skyline panoramique, pour y trouver plus tard la position (la direction d'observation) à laquelle l'image réelle a été prise. Nous illustrons à la figure 5.14(g), un exemple d'une image panoramique. Aux figures 5.14(a) à 5.14(f), les différentes images prises à différents endroits (translation) et à différentes orientations (*Roll, Pitch et Yaw*) autour de là où l'image panoramique a été générée.

Pour le même corpus utilisé précédemment (base de données), nous utilisons nos différentes métriques pour essayer retrouver la position des images réelles et virtuelles dans les images panoramiques. Le tableau 5.3 présente nos différents résultats.

Conclusion

Nous avons présenté en détail notre système de réalité augmentée utilisant le skyline comme marqueur. Ce système combine un sous-système de fusion de capteurs (comme détaillé en chapitre 3) avec un sous-système basé vision, utilisant une approche orientée sur des points d'intérêts très particuliers : le skyline. Le premier sous système nous permet de disposer en continu d'une estimation de la pose de la caméra. Le deuxième sous système est complémentaire au premier permettant de corriger cette pose grâce à une procédure d'appariement basée sur le skyline : le skyline virtuel est recalé sur le skyline réel.

En effet, les capteurs sont le composant principal qu'utilise notre système pour fournir une bonne estimation de la pose. Cependant, cette estimation est en général entachée d'erreurs à cause des différents types de bruits présents dans l'environnement extérieur. Associé au modèle 3D de la ville, cette pose nous permet d'avoir un rendu grossier de ce que l'utilisateur voit à cette position. Grâce à notre algorithme d'extraction du skyline présenté dans le chapitre 4, les deux skylines sont extraits afin d'être apparié. Pour cela, nous avons mis au point une approche automatique de recherche basée sur une descente de gradient, et plusieurs métriques selon les cas d'utilisation. Les basculements d'un sous-système à un autre se basent sur la qualité du recalage / appariement trouvé.

Les expérimentations menées, sur les différentes contributions prises séparément (extraction du skyline, recalage, etc.) et sur le système en sa globalité, ont démontré que notre approche est faisable en milieu extérieur. La procédure est testée dans des

conditions réelles dans la ville de Lyon. À chacune des étapes du processus, une base de données est créées et mise à disposition de la communauté.

6 Conclusion générale et perspectives

6.1 Travaux réalisés

Dans le cadre de cette thèse, nous nous sommes focalisés sur les aspects de la localisation en milieu extérieur. Cette problématique représente un enjeu important pour de nombreux domaines tels que la réalité augmentée qui connait depuis quelques temps un intérêt grandissant dans divers domaines d'applications. La localisation en milieu extérieur comporte plusieurs verrous scientifiques. Nous nous sommes particulièrement intéressés à ceux entourant la mise en œuvre de systèmes multi-capteurs notamment sur la stratégie à adopter pour la combinaison des capteurs, les approches de calibration du capteur hybride ainsi que la prédiction des erreurs des données issues des capteurs. Dans un premier temps, nous nous sommes intéressés aux approches de localisation basées vision utilisées en réalité augmentée. Nous nous sommes orientés vers une approche sans marqueurs utilisant des points d'intérêts. Brièvement, celles-ci consistent à identifier la projection des points 3D sur le plan image pour calculer les paramètres de la pose en minimisant l'erreur de reprojection. Notre tâche s'est concentrée sur la phase d'appariement entre points 3D issus du modèle et points 2D extraits en ligne. Ainsi, nous avons mis au point une approche d'initialisation qui requière l'intervention de l'utilisateur. L'approche proposée utilise des descripteurs SURF associés aux points 3D. Cette approche est suppléée par un suivi 2D/2D afin d'identifier les projections des points 3D dans le flux d'images et ainsi maintenir la localisation en temps réel. Cependant cette approche a des limites. En effet, sa précision dépend, d'une part, du nombre d'appariements utilisés dans le processus d'estimation, et d'autre part de la précision du suivi visuel qui dépend des conditions de travail (occultation, mouvement brusque et variations de luminosité). De ce fait, la vision a besoin d'être suppléée par d'autres capteurs afin d'améliorer la précision et la robustesse de la localisation. Suite à l'étude réalisée sur différents systèmes multi-capteurs, nous avons dégagé une taxonomie qui se base sur la stratégie de combinaison. Nous avons recensé deux types de stratégies qui sont la fusion de données et la suppléance de données. L'étude comparative effectuée entre ces deux classes, nous a conduit à plusieurs constatations. D'un côté, nous avons remarqué que les approches de fusion se basent sur des modèles cinématiques qui ne prennent pas en compte certains types de mouvements (mouvement brusque). D'un autre côté,

les approches de vision présentent souvent des performances satisfaisantes lorsque les conditions d'utilisation de la caméra sont favorables (mouvement lisse, éclairage contrôlé, etc.). Ainsi, l'utilisation de la suppléance qui consiste à remplacer la vision par d'autres capteurs tant qu'elle est dans l'incapacité de fournir une estimation correcte de la localisation, nous parait intéressante. Notre choix s'est porté donc sur ce type d'approche pour, 153 154 CHAPITRE 5. LOCALISATION BASÉE SUPPLÉANCE MULTI-CAPTEURS d'une part, proposer un système palliatif à la vision et d'autre part pouvoir tester sa faisabilité en environnement extérieur. Le fait d'utiliser une approche de suppléance, nous a permis de concevoir une solution "logicielle" pour se localiser en milieu extérieur. Ainsi, notre système de localisation est subdivisé en deux sous-systèmes : un sous-système de vision (principal) et un sous-système d'assistance à la localisation (palliatif). La mise en œuvre de notre système multi-capteurs repose sur la résolution de plusieurs problématiques. La première concerne la procédure de calibration qui permet d'unifier les données issues des différents capteurs et de les réexprimer dans le même référentiel. Etant donné que notre capteur hybride comprend une caméra, une centrale inertielle et un récepteur GPS, nous avons proposé deux processus de calibration qui se basent sur les modèles obtenus du couplage inertiel/Caméra et GPS/Caméra. Nos approches ont l'avantage d'être simples à mettre en œuvre et ne requièrent pas de lourdes hypothèses. De plus, elles sont génériques et peuvent fonctionner dans différentes configurations. Par exemple l'approche de calibration Inertiel/Caméra peut être utilisée aussi bien pour déduire les orientations absolues que relatives. La deuxième problématique abordée concerne le fonctionnement et les interactions de nos deux sous-systèmes. Pour cela, nous nous sommes inspirés d'une des caractéristiques de l'architecture logicielle utilisée pour le développement de notre système. En effet, le système ARCS se base sur un automate à états fini pour décrire le fonctionnement d'une application. Nous avons ainsi utilisé le concept d'automate pour modéliser notre système de localisation. Les états sont définis selon les traitements principaux effectués par le système. Le système passe d'un état à un autre selon les critères associés à chaque traitement, comme par exemple l'échec du suivi visuel fait passer le système de l'état associé à l'exploitation de la vision à l'état associé à l'assistance. Par ailleurs, nous avons aussi mis au point une approche de réinitialisation automatique qui permet de retrouver les appariements des points 3D dans l'image courante après un échec du suivi visuel. Cette approche a prouvé son efficacité et sa précision. Les contributions faites dans cette thèse ont permis de mettre au point un système qui a la faculté de s'adapter aux conditions extérieures en changeant son état interne. Nous avons décelé plusieurs problématiques auxquelles nous avons apporté des solutions que ce soit au niveau de la vision (approche d'initialisation et de réinitialisation), ou du système de suppléance. Les

résultats obtenus à l'issue des différentes expérimentations nous ont démontré que l'utilisation d'une approche de suppléance pouvait être intéressante. En effet, l'ajout d'un sous-système de suppléance permet de pallier les problèmes de défaillance de la vision qui sont fréquents dans ce type de milieu. A partir des résultats que nous avons obtenus, nous pouvons recenser plusieurs améliorations à apporter à notre système. Certes l'approche basée vision que nous avons mise au point est simple, elle fournit des résultats satisfaisants mais elle a des limites. Son problème principal réside dans le fait que la méthode ne peut suivre que les points identifiés lors de la phase d'initialisation. Une amélioration serait d'avoir la possibilité d'identifier de nouveaux points du modèle 3D et de les inclure en temps réel dans la phase de suivi. De plus, il serait fort intéressant d'utiliser d'autres approches basées vision telle que les contours ou les segments. Dans cette perspective, nous avons pensé à utiliser une approche basée segments dans un environnement urbain et en utilisant des données extrait des SIG

6.2 PERSPECTIVES

(Systèmes d'information géographiques) ou bien des contours appariés avec des données extraits du Modèle Numérique de terrain (MNT) pour un système fonctionnant dans un environnement panoramique représenté par une chaine de montagne. Le lecteur intéressé peut entrevoir les pistes dégagées pour ces scénario dans l'annexe C et l'annexe B. A cela s'ajoute le fait que l'utilisateur est toujours restreint à évoluer dans la partie modélisée de l'environnement. Pour pallier ce problème, il serait fort intéressant de se pencher sur des approches de type SLAM qui offrent la possibilité d'estimer simultanément les paramètres de la pose et de la structure de la scène. Ceci permettra de compléter les connaissances a priori dont nous disposons de l'environnement ou bien d'apporter cette connaissance pour des environnements inconnus. Concernant le sous-système d'assistance, nous pouvons envisager des améliorations à plusieurs niveaux. Par exemple, au niveau de la prédiction, il serait intéressant de l'appliquer sur l'erreur de recalage étant donné que l'un des objectifs principaux dans les systèmes de réalité augmentée et le recalage réel/virtuel. En effet, au lieu d'apporter la correction au niveau de la localisation celle-ci sera appliquée directement sur le recalage afin de le faire coïncider avec ce que pourrait donner la vision. Enfin, afin que le système de localisation fonctionne quelles que soient les conditions, il faut prendre en considération la défaillance des capteurs composant le système AL. Par exemple si le récepteur GPS se trouve dans une zone où il n'y a pas de couverture satellitaire, il fraudera trouver un moyen de prédire cette position. Pour cela, nous

pourrions utiliser de nouvelles technologies telles que les réseaux de téléphonie mobile ou wifi afin de développer de nouvelles méthodes de localisation

6.3 PUBLICATIONS

BIBLIOGRAPHIE

M. Angermann et P. Robertson. Footslam : Pedestrian simultaneous localization and mapping without exteroceptive sensors—hitchhiking on human perception and cognition. *Proceedings of the IEEE*, 100(Special Centennial Issue) :1840–1848, May 2012. ISSN 0018-9219. (Cité page 20.)

Michael Aron, Gilles Simon, et Marie-Odile Berger. Use of inertial sensors to support video tracking : Research articles. *Comput. Animat. Virtual Worlds*, 18(1) :57–68, Février 2007. ISSN 1546-4261. (Cité page 42.)

Mehdi Ayadi, Loreta Suta, Mihaela Scuturici, Serge Miguet, et Chokri Ben Amar. A parametric algorithm for skyline extraction. Dans Jacques Blanc-Talon, Cosimo Distante, Wilfried Philips, Dan Popescu, et Paul Scheunders, éditeurs, *Advanced Concepts for Intelligent Vision Systems*, pages 604–615, Cham, 2016. Springer International Publishing. ISBN 978-3-319-48680-2. (Cité page 8.)

Mehdi Ayadi, Leo Valque, Serge Miguet, Mihaela Scuturici, et Chokri Ben Amar. The skyline as a marker for augmented reality in urban context. Dans George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Fatih Porikli, Sandra Skaff, Alireza Entezari, Jianyuan Min, Daisuke Iwai, Amela Sadagic, Carlos Scheidegger, et Tobias Isenberg, éditeurs, *Advances in Visual Computing*. Springer International Publishing, 2018. (Cité page 8.)

Ronald T. Azuma. A survey of augmented reality. *Presence : Teleoper. Virtual Environ.*, 6(4) :355–385, Août 1997. ISSN 1054-7460. (Cité pages xiii, 11, 22 et 23.)

Georges Baatz, Olivier Saurer, Kevin Köser, et Marc Pollefeys. Large scale visual geo-localization of images in mountainous terrain. Dans Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, et Cordelia Schmid, éditeurs, *Computer Vision – ECCV 2012*, pages 517–530, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33709-3. (Cité pages 83, 87, 100 et 108.)

D. L. Baggio, S. Emami, D. M. Escriva, K. Ievgen, N. Mahmood, J. Saragih, et R. Shilkrot. *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing, Limited, 2012. ISBN 9781849517829. recommended : advanced OpenCV project support/examples inc. iOS and Android examples. (Cité pages xiii et 17.)

Herbert Bay, Tinne Tuytelaars, et Luc Van Gool. Surf : Speeded up robust features. Dans Aleš Leonardis, Horst Bischof, et Axel Pinz, éditeurs, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33833-8. (Cité page 18.)

J. Bazin, I. Kweon, C. Demonceaux, et P. Vasseur. Dynamic programming and skyline extraction in catadioptric infrared images. Dans *2009 IEEE International Conference on Robotics and Automation*, pages 409–416, May 2009. (Cité page 87.)

A. H Behzadan. *ARVISCOPE : Georeferenced Visualization of Dynamic Construction Processes in Three-Dimensional Outdoor Augmented Reality*. PhD thesis, Department of Civil and Environmental Engineering, University of Michigan, 2008. (Cité pages xiii, 20 et 21.)

A. H. Behzadan et V. R. Kamat. Visualization of construction graphics in outdoor augmented reality. Dans *Proceedings of the Winter Simulation Conference, 2005.*, pages 7 pp.–, Dec 2005. (Cité pages xiii, 20 et 21.)

Gabriele Bleser. *Towards Visual-Inertial SLAM for Mobile Augmented Reality*. Thèse de doctorat, l'Université technique de Kaiserslautern, Mars 2009. (Cité pages xiv, 42, 44, 45, 47 et 48.)

Wolfgang Broll, Irma Lindt, Jan Ohlenburg, Michael Wittkämper, Chunrong Yuan, Thomas Novotny, Fatah gen, C. Mottram, et Andreas Strothmann. ARTHUR : A Collaborative Augmented Environment for Architectural Design and Urban Planning. *Journal of Virtual Reality and Broadcasting*, 1(1), 2004. (Cité pages xiii, 30, 31 et 42.)

L. Bruno et P. Robertson. Wislam : Improving footslam with wifi. Dans *2011 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–10, Sept 2011. (Cité page 20.)

J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6) :679–698, Nov 1986. ISSN 0162-8828. (Cité pages 18 et 94.)

Ludovico Carozza, David Tingdahl, Frédéric Bosché, et Luc van Gool. Markerless Vision-Based Augmented Reality for Urban Planning. *Computer-Aided Civil and Infrastructure Engineering*, 29(1) :2–17, jan 2014. ISSN 10939687. (Cité pages xiv, 34, 35, 36 et 42.)

Sylvie Chambon. *Color stereo matching with occlusions*. Theses, Université Paul Sabatier - Toulouse III, Décembre 2005. (Cité page 117.)

Jean-Marc Cieutat. *A few applications of augmented reality : New modes of information and communication technologies. Effects on perception, cognition and action*. Habilitation à diriger des recherches, Université Paul Sabatier - Toulouse III, Mars 2013. (Cité pages xiii et 28.)

Arnis Cirulis et Kristaps Brigis Brigmanis. 3D Outdoor Augmented Reality for Architecture and Urban Planning. *Procedia Computer Science*, 25 :71–79, 2013. ISSN 18770509. (Cité pages xiv, 33, 34, 35 et 42.)

B. Drost, M. Ulrich, N. Navab, et S. Ilic. Model globally, match locally : Efficient and robust 3d object recognition. Dans *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 998–1005, June 2010. (Cité page 19.)

M. Fang, M. . Chiu, C. . Liang, et A. Singh. Skyline for video-based virtual rail for vehicle navigation. Dans *Proceedings of the Intelligent Vehicles '93 Symposium*, pages 207–212, July 1993. (Cité pages xiv, 83, 84 et 108.)

Jacques Feldmar, Nicholas Ayache, et Fabienne Betting. 3d–2d projective registration of free-form curves and surfaces. *Computer Vision and Image Understanding*, 65(3) : 403 – 424, 1997. ISSN 1077-3142. (Cité page 26.)

P. Fuchs et G. Moreau. *Le traité de la réalité virtuelle : Fondements et interfaces comportementales*. Presses de l'Ecole des Mines de Paris, 2003. (Cité page 11.)

Tomohiro Fukuda, Tian Zhang, et Nobuyoshi Yabuki. Improvement of registration accuracy of a handheld augmented reality system for urban landscape simulation. *Frontiers of Architectural Research*, 3(4) :386–397, dec 2014. ISSN 20952635. (Cité pages xiv, 34, 35, 42 et 125.)

Markus Funk. *Augmented Reality at the Workplace : A context-aware assistive szstem using in-situ projection*. PhD thesis, University of Stuttgart, 2016. (Cité pages xiii et 27.)

Reitmayr G et Drummond T. W. Initialisation for visual tracking in urban environments. Dans *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 161–172, Nov 2007. (Cité pages 47 et 48.)

T. W. Drummond G. Reitmayr. Going out : robust model-based tracking for outdoor augmented reality. Dans *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 109–118, Oct 2006. (Cité pages xiv, 42 et 46.)

Jérémy Gaillard, Alexandre Vienne, Rémi Baume, Frédéric Pedrinis, Adrien Peytavie, et Gilles Gesquière. Urban data visualisation in a web browser. Dans *Proceedings of the 20th International Conference on 3D Web Technology*, Web3D '15, pages 81–88, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3647-5. (Cité page 60.)

Mar González-Franco, Julio Cermeron, Katie Li, Rodrigo Pizarro, Jacob Thorn, Paul Hannah, Windo Hutabarat, Ashutosh Tiwari, et Pablo Bermell-Garcia. Immersive augmented reality training for complex manufacturing scenarios. *CoRR*, abs/1602.01944, 2016. (Cité pages xiii, 28 et 29.)

Chris Harris et Mike Stephens. A combined corner and edge detector. Dans *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988. (Cité pages 15 et 18.)

Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, et Kurt Konolige. Going further with point pair features. Dans Bastian Leibe, Jiri Matas, Nicu Sebe, et Max Welling, éditeurs, *Computer Vision – ECCV 2016*, pages 834–848, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46487-9. (Cité page 19.)

S. Hofmann, D. Eggert, et C. Brenner. Skyline matching based camera orientation from images and mobile mapping point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-5 :181–188, 2014. (Cité page 108.)

J. D. Hol, T. B. Schon, F. Gustafsson, et P. J. Slycke. Sensor fusion for augmented reality. Dans *2006 9th International Conference on Information Fusion*, pages 1–6, July 2006. (Cité page 42.)

Honkamaa. Interactive outdoor mobile augmentation using markerless tracking and gps. *In Proc. Virtual Reality International Conference (VRIC)*, 2007. (Cité pages 32 et 42.)

P.V.C. Hough. Method and means for recognizing complex patterns. *Google Patents*, 12 1962. (Cité page 15.)

Tobias Höllerer, Steven Feiner, Tachio Terauchi, Gus Rashid, et Drexel Hallaway. Exploring mars : developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers Graphics*, 23(6) :779 – 785, 1999. ISSN 0097-8493. (Cité pages 4, 29 et 30.)

J. Vallino. *Interactive augmented reality*. PhD thesis, University of Rochester, New York, jan 1998. (Cité page 11.)

D. Johns et G. Dudek. Urban position estimation from one dimensional visual cues. Dans *The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, pages 22–22, June 2006. (Cité pages 81 et 108.)

H. Kato et M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. Dans *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pages 85–94, Oct 1999. (Cité page 15.)

Wadim Kehl, Federico Tombari, Nassir Navab, Slobodan Ilic, et Vincent Lepetit. Hashmod : A hashing method for scalable 3d object detection. Dans *BMVC*, 2015. (Cité page 19.)

E. Kerrien, M. O. Berger, E. Maurincomme, L. Launay, R. Vaillant, et L. Picard. Fully automatic 3d/2d subtracted angiography registration. Dans Chris Taylor et Alain Colchester, éditeurs, *Medical Image Computing and Computer-Assisted Intervention – MICCAI'99*, pages 664–671, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48232-1. (Cité page 26.)

Byung-Ju Kim, Jong-Jin Shin, Hwa-Jin Nam, et Jin-Soo Kim. Skyline extraction using a multistage edge filtering. *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, 5(7) :792 – 796, 2011. ISSN eISSN :1307-6892. (Cité pages xv et 91.)

G. Klein et T. Drummond. Robust visual tracking for non-instrumental augmented reality. Dans *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 113–122, Oct 2003. (Cité page 47.)

Christian Koch, Matthias Neges, Markus König, et Michael Abramovici. Performance Study on Natural Marker Detection for Augmented Reality Supported Facility Maintenance. *Australasian Journal of Construction Economics and Building - Conference Series*, 2(1) :23, jan 2014. ISSN 2200-7679. (Cité page 18.)

L. Robinault. *Mosaïque d'images multi résolution et applications*. PhD thesis, Université Lumiére Lyon 2, 2009. (Cité page 117.)

Wen-Nung Lie, Tom C.-I. Lin, Ting-Chih Lin, et Keng-Shen Hung. A robust dynamic programming algorithm to extract skyline in images for navigation. *Pattern Recognition Letters*, 26(2) :221 – 230, 2005. ISSN 0167-8655. (Cité pages xiv, 87, 88 et 89.)

Yun-Jiun Liu, Chung-Cheng Chiu, et Jia-Horng Yang. A Robust Vision-Based Skyline Detection Algorithm Under Different Weather Conditions. *IEEE Access*, 5 :22992–23009, 2017. ISSN 2169-3536. (Cité pages 83 et 87.)

Mark a Livingston, Lawrence J Rosenblum, Dennis G Brown, Gregory S Schmidt, Simon J Julier, Yohan Baillot, J. Edward Swan, Zhuming Ai, et Paul Maassel. Military Applications of Augmented Reality. Dans *Handbook of Augmented Reality*, pages 671–706. Springer New York, New York, NY, 2011. ISBN 978-1-4614-0063-9. (Cité pages 29 et 30.)

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2) :91–110, Nov 2004. ISSN 1573-1405. (Cité page 18.)

Marc-O. Löwner, Joachim Benner, Gerhard Gröger, et Karl-Heinz Häfele. New concepts for structuring 3d city models - an extended level of detail concept for citygml buildings. Dans *ICCSA*, 2013. (Cité pages xv et 107.)

Jiebo Luo et S. P. Etz. A physical model-based approach to detecting sky in photographic images. *IEEE Transactions on Image Processing*, 11(3) :201–212, March 2002. ISSN 1057-7149. (Cité pages 83 et 85.)

Héctor Martínez, Seppo Laukkanen, et Jouni Mattila. A New Flexible Augmented Reality Platform for Development of Maintenance and Educational Applications. *International Journal of Virtual Worlds and Human Computer Interaction*, 2 :18–27, 2014. ISSN 23686103. (Cité pages 27 et 28.)

Paul Milgram. A TAXONOMY OF MIXED REALITY VISUAL DISPLAYS. *IEICE Transactions on Information Systems*, E77-D(12) :1–15, 1994. ISSN 0916-8532. (Cité pages xiii, 10 et 13.)

Eric N. Mortensen et William A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5) :349 – 384, 1998. ISSN 1077-3169. (Cité page 110.)

Andreas Nüchter, Stanislav Gutev, Dorit Borrmann, et Jan Elseberg. Skyline-based registration of 3d laser scans. *Geo-spatial Information Science*, 14(2) :85, May 2011. ISSN 1993-5153. (Cité page 108.)

Rafael Radkowski. Object Tracking With a Range Camera for Augmented Reality Assembly Assistance. *Journal of Computing and Information Science in Engineering*, 16(1) :011004, 2016. ISSN 1530-9827. (Cité page 19.)

S. Ramalingam, S. Bouaziz, P. Sturm, et M. Brand. Geolocalization using skylines from omni-images. Dans *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 23–30, Sept 2009. (Cité page 90.)

S. Ramalingam, S. Bouaziz, P. Sturm, et M. Brand. Skyline2gps : Localization in urban canyons using omni-skylines. Dans *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3816–3823, Oct 2010. (Cité pages xv, 90, 108, 109, 110 et 111.)

Guido Maria Re, James Oliver, et Monica Bordegoni. Impact of monitor-based augmented reality for on-site industrial manual operations. *Cognition, Technology & Work*, 18(2) :379–392, May 2016. ISSN 1435-5566. (Cité pages xiii et 27.)

Jun Rekimoto et Katashi Nagao. The world through the computer : Computer augmented interaction with real world environments. Dans *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, UIST '95, pages 29–36, New York, NY, USA, 1995. ACM. ISBN 0-89791-709-X. (Cité pages xiii et 23.)

E. Rosten et T. Drummond. Fusing points and lines for high performance tracking. Dans *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1508–1515 Vol. 2, Oct 2005. (Cité page 18.)

M. Roth, C. Brack, R. Burgkart, A. Czopf, H. Götte, A. Schweikard, et A. Multi-view contourless registration of bone structures using a single calibrated x-ray fluoroscope, 1999. (Cité page 26.)

Tobias Lossing Rune Nielsen, Thomas Fabian Delman. The harbour game - a mixed reality game for urban planning. Dans *the Computers in Urban Planning and Urban Management conference*, 06 2005. (Cité pages xiii, 17, 31 et 42.)

Olivier Saurer, Georges Baatz, Kevin Köser, L'ubor Ladický, et Marc Pollefeys. Image Based Geo-localization in the Alps. *International Journal of Computer Vision*, 116(3) : 213–225, feb 2016. ISSN 0920-5691. (Cité page 87.)

Fernando Suárez-Warden, Eduardo González Mendívil, Ciro A. Rodríguez, et Salvador Garcia-Lumbreras. Assembly operations aided by augmented reality : An endeavour toward a comparative analysis. *Procedia Computer Science*, 75 :281 – 290, 2015. ISSN 1877-0509. 2015 International Conference Virtual and Augmented Reality in Education. (Cité pages xiii et 27.)

Joseph Tighe et Svetlana Lazebnik. Superparsing : Scalable nonparametric image parsing with superpixels. Dans Kostas Daniilidis, Petros Maragos, et Nikos Paragios, éditeurs, *Computer Vision – ECCV 2010*, pages 352–365, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15555-0. (Cité pages 100, 101 et 103.)

V. Havard. *Développement de méthodes et outils basés sur la réalité augmentée et virtuelle pour l'assistance ou l'apprentissage d'opérations dans un contexte industriel*. PhD thesis, Université de Rouen, Feb 2018. (Cité pages xiii, 16 et 27.)

Charles Woodward, Jani Lahti, Jukka Rönkkö, Petri Honkamaa, Mika Hakkarainen, Jani Jäppinen, Kari Rainio, Sanni Siltanen, et Jouko Hyväkkä. Case digitalo-a range

of virtual and augmented reality solutions in construction application. *International Journal of Environmental Science and Technology*, 01 2007. (Cité pages xiii, xiv, 32, 33 et 42.)

Nobuyoshi Yabuki, Kyoko Miyashita, et Tomohiro Fukuda. An invisible height evaluation system for building height regulation to preserve good landscapes using augmented reality. *Automation in Construction*, 20(3) :228 – 235, 2011. ISSN 0926-5805. Augmented and Virtual Reality in Architecture, Engineering and Construction (CONVR2009). (Cité pages xiii et 17.)

Simon Julier Yohan, Simon Julier, Yohan Baillot, Marco Lanzagorta, Dennis Brown, et Lawrence Rosenblum. Bars : Battlefield augmented reality system. Dans *In NATO Symposium on Information Processing Techniques for Military Systems*, pages 9–11, 2000. (Cité page 30.)

S. You, U. Neumann, et R. Azuma. Orientation tracking for outdoor augmented reality registration. *IEEE Computer Graphics and Applications*, 19(6) :36–42, Nov 1999. ISSN 0272-1716. (Cité pages xiv, 42, 43 et 44.)

N.A.H. Yusoff, A.M. Noor, et R Ghazali. City skyline conservation : Sustaining the premier image of kuala lumpur. *Procedia Environmental Sciences*, pages 83–592, 2014. (Cité page 81.)

Imane Zendjebil. *3D localization based assistance scheme using multi-sensors for mobile outdoor augmented reality*. Theses, Université d'Evry-Val d'Essonne, Octobre 2010. (Cité pages xiv, 42, 48, 49 et 50.)

Imane Zendjebil, Fakhr-Eddine Ababsa, Jean-Yves Didier, Jacques Vairon, Luc Frauciel, Martin Hachet, Pascal Guitton, et Romuald Delmont. Réalité Augmentée en extérieur : Enjeux et Etat de l'Art. Dans *2èmes journées de l'Association Française de Réalité Virtuelle*, Luminy, France, 2007. (Cité page 36.)

S. Zhu, L. Morin, M. Pressigout, G. Moreau, et M. Servières. Video/gis registration system based on skyline matching method. Dans *2013 IEEE International Conference on Image Processing*, pages 3632–3636, Sept 2013. (Cité pages xv, 108, 111, 112, 125 et 127.)

Shupeng Zhu, Muriel Pressigout, Myriam Servières, Luce Morin, et Guillaume Moreau. Skyline Matching : A robust registration method between Video and GIS. Dans *Conference of the European COST Action TU0801 - Semantic Enrichment of 3D City Models for Sustainable Urban Development Location : Grad Sch Architecture*, pages 1–6, Nantes, France, Octobre 2012. (Cité pages 108 et 111.)