Contributions à la précision et à la robustesse de la localisation visuelle dans un monde d'objets

Matthieu Zins

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Contributions à la précision et à la robustesse de la localisation visuelle dans un monde d'objets

THÈSE

présentée et soutenue publiquement le 9 décembre 2022

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Matthieu Zins

Composition du jury

Président :	Sylvain Lazard	Directeur de Recherche – Inria Nancy - Grand Est
Rapporteurs :	Vincent Lepetit	Professeur des Universités – École des Ponts ParisTech
	Eric Marchand	Professeur des Universités – Université de Rennes
Examinateur :	Gabriela Csurka	Principal Research Scientist – Naver Labs Europe
Encadrants :	Marie-Odile Berger	Directrice de Recherche – Inria Nancy - Grand Est
	Gilles Simon	Maître de Conférences HDR – Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Je tiens tout d'abord à remercier mes co-directeurs de thèse, Marie-Odile Berger et Gilles Simon, pour leur implication dans cette thèse et les nombreux conseils qu'ils m'ont donnés tout au long de ces trois années. Ils ont su trouver un rythme de travail très satisfaisant entre liberté et nécessité de résultats. Ils m'ont permis de découvrir le monde de la recherche dans des conditions idéales.

J'adresse tous mes remerciements à Vincent Lepetit, Professeur à l'École des Ponts ParisTech, et à Eric Marchand, Professeur à l'Université de Rennes, qui ont accepté d'être rapporteurs de cette thèse. Tout comme à Gabriela Csurka, Principal Research Scientist à Naver Labs, d'avoir accepté de participer à mon jury et à Sylvain Lazard, Directeur de Recherche à l'Inria, qui m'a fait l'honneur de le présider.

Je remercie toutes les personnes que j'ai pu côtoyer au LORIA. Ces trois années ont été pour moi une très belle expérience. Je tiens à exprimer ma reconnaissance aux membres permanents de l'équipe Tangram : Brigitte, Frédéric, Erwan, Pierre-Frédéric, Fabien, Isabelle, ainsi qu'à mes différents collègues de bureau qui ont rendu l'environnement de travail très agréable. Je pense notamment à Dasha et Vincent pour leur aide et leurs conseils à mes débuts et à Yasmine, Romain, Karim, Nariman et Youssef pour les parties de beach-volley, les sorties trail et rando. Je n'oublie pas tous ceux avec qui j'ai partagé un bout de chemin : Anastasiia, Thomas, Amandine, Hugo, Idriss, Radhouane, Léo, Daria, Nicolas et Timothée.

Un très grand merci à ma famille, en particulier à mes parents pour leur soutien inconditionnel depuis tant d'années, ainsi que pour la relecture minutieuse de ce manuscrit. Je remercie mon frère, Pierre, pour son aide et la motivation qu'il me donne, ma tante Nicole et mon oncle Jeannot pour leurs encouragements.

Enfin, je remercie Justine d'avoir été à mes côtés. Grâce à elle, j'ai pu trouver un juste équilibre entre travail et loisirs.

Sommaire

Introduction

1 Contexte et motivations

1.1 Contexte de la localisation visuelle

La localisation visuelle est un problème bien connu en vision par ordinateur, avec de nombreuses applications, qui consiste à estimer la position et l'orientation de la caméra dans une scène. En robotique, par exemple, l'estimation de la pose d'un système autonome (véhicules ou drones) est primordiale pour assurer la sécurité des utilisateurs. La localisation est également très importante en réalité augmentée, qui a connu un fort développement ces dernières années dans des domaines variés, tels que le divertissement, le tourisme, le commerce, l'architecture, le médical, l'enseignement, l'industrie ou encore la culture. Les musées sont, par exemple, devenus particulièrement friands de cette technologie, pour améliorer l'expérience de leurs visiteurs. La réalité augmentée permet également de créer des contenus éducatifs plus attractifs et de stimuler la collaboration entre les élèves et le professeur. Enfin, ses applications sont également nombreuses dans l'industrie avec, par exemple, l'aide à la fabrication ou à la maintenance, la sécurité ou encore la formation. L'augmentation de la réalité peut être faite par des lunettes spécifiques (HoloLens ou MagicLeap) ou sur un écran à travers lequel l'utilisateur voit la réalité (tablette ou smartphone). Des exemples d'applications sont illustrés dans la figure 1.

Une localisation visuelle précise est un élément clé dans ces applications. En effet, la position et l'orientation de la caméra doivent être estimées très précisément pour que les éléments virtuels intégrés dans les images apparaissent de manière cohérente avec la vue de l'utilisateur. Les applications de réalité augmentée doivent faire face à certains challenges, notamment la durée de l'utilisation, le mouvement de la caméra qui est laissé totalement libre à l'utilisateur et les environnements d'utilisation variés.

Bien que différents capteurs puissent être utilisés pour se localiser tels qu'un GPS, un capteur inertiel, une caméra de profondeur ou encore un Lidar, nous nous limitons à une caméra couleur. En effet, un GPS permet d'obtenir une position, mais ne fonctionne pas en intérieur ou à proximité de grands bâtiments. De plus, sa précision est limitée à quelques mètres. Les caméras de profondeur sont de plus en plus répandues, mais pas encore pour le grand public. Seuls quelques modèles de smartphone haut de gamme intègrent de tels capteurs. De plus, elles éprouvent des difficultés face à des surfaces très réfléchissantes telles que les structures métalliques dans des environnements industriels. Un Lidar, très utilisé pour la navigation autonome, est un capteur laser qui offre une très bonne précision mais n'est pas envisageable pour des applications grand public, notamment à cause de son prix très élevé. Au contraire, dans ce travail, nous cherchons à développer des méthodes de localisation à partir d'images couleurs acquises avec une caméra standard, disponible sur la plupart des téléphones portables modernes et permettant ainsi d'être utilisées par tout le monde.

(a) Pokémon GO [1] (b) Catalogue virtuel IKEA [2] (c) Google Maps AR [3]

(d) Histopad [4] (e) HoloLens pour le médical [5] (f) Digitalisation des usines [6]

FIGURE 1 – Exemples d'applications de réalité augmentée dans différents domaines.

Approches classiques de localisation visuelle. Les méthodes classiques de localisation visuelle se basent généralement sur un modèle de scène sous la forme d'un nuage de points 3D, qui peut être reconstruit par un système de *Structure-from-Motion* (SfM), tel que COLMAP [SF16] ou Bundler [SSS08] (voir Figure 2). La pose de la caméra est calculée par des mises en correspondance entre des points d'intérêt 2D extraits dans l'image et les points 3D du modèle de la scène. Cet appariement repose généralement sur de l'information locale extraite dans le voisinage des points d'intérêt dans les images. Malgré les avancées de nombreuses années de recherche, ces méthodes éprouvent toujours des difficultés dans certains environnements (absence de textures, structures répétitives, ...). Elles restent également sensibles à des perturbations visuelles (illumination, flou de mouvement, spécularités, ...) et à des changement importants de point de vue (orientation et échelle). De plus, les points sont généralement en grand nombre dans une image et leur mise en correspondance avec un modèle de scène pouvant contenir plusieurs milliers, voire millions, de points implique un coût de calcul élevé.

Approches basées sur de l'apprentissage profond. D'autres méthodes, plus récentes, se basent sur les avancées en apprentissage profond et proposent de prédire la pose de la caméra ou des coordonnées 3D de la scène directement à partir d'une image. Cependant, ces méthodes n'atteignent pas, pour l'instant, les mêmes niveaux de précision de pose que les approches classiques [SZPL19] et leur généralisation à d'autres points de vue de la scène que ceux utilisés en entraînement est limitée.

1. cnetfrance.fr/news/pokemon-go-a-change-la-realite-augmentee-a-tout-jamais-39854920.htm
2. leparisien.fr/high-tech/la-realite-augmentee-va-nous-en-faire-voir-17-09-2017-7266135.php
3. medialist.info/en/2019/05/12/google-maps-ar-navigation-with-augmented-reality/
4. club-innovation-culture.fr/chateau-amboise-400-histopad/
5. anthedesign.fr/autour-du-web/realite-augmentee-ra/
6. 4dcrea.com/realite-virtuelle-ou-augmentee-entreprise/

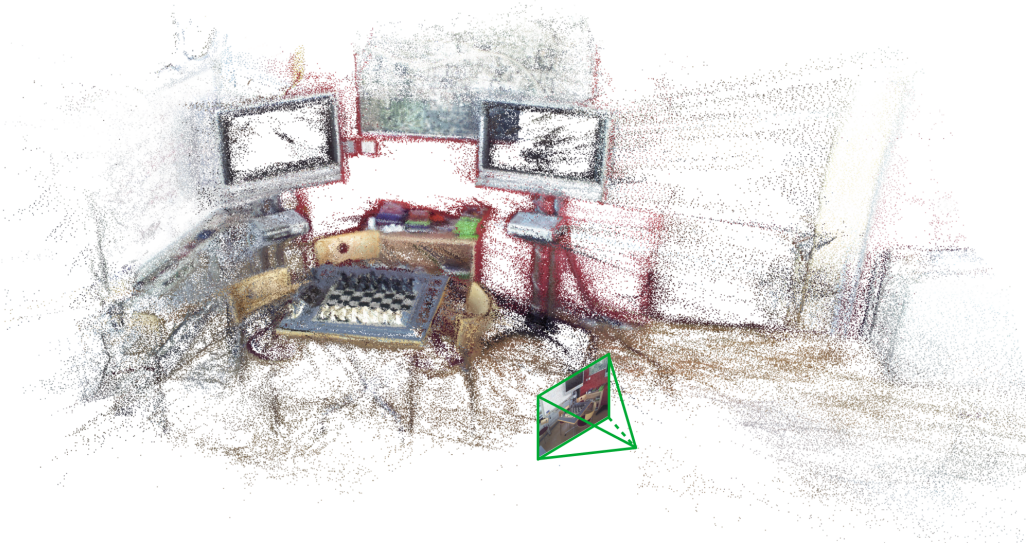


FIGURE 2 – Localisation visuelle à partir d'un nuage de points de la scène.

1.2 Localisation visuelle dans un monde d'objets

Pour surmonter ces difficultés, des chercheurs ont commencé à s'intéresser à l'utilisation d'information sémantique pour aider la localisation visuelle. Sünderhauf *et al.* ont, par exemple, utilisé des boîtes sémantiques proposées par un réseau de neurones convolutif pour reconnaître des lieux [Sün+15] et Weinzaepfel *et al.* ont développé un système de localisation basé sur des correspondances denses entre des objets plans tels que des tableaux dans un musée [WCCH19].

La prise en compte des objets présents dans une scène pour estimer la pose de la caméra est en effet avantageuse pour plusieurs raisons :

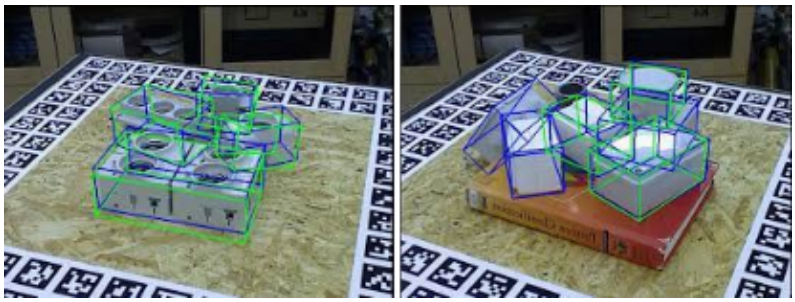
— Les avancées impressionnantes en termes de détection d'objet apportées par les réseaux de neurones permettent maintenant de les détecter dans une image de manière très robuste.

— Les objets constituent des balises sémantiques plus discriminantes que des points d'intérêt décrits par de l'information locale.

— Outre le problème de localisation, des cartes sémantiques d'objets sont également très utiles pour d'autres tâches, telles que la compréhension de la scène ou la saisie d'objets.

Estimation de pose avec des modèles précis des objets. De nombreux travaux se sont intéressés à l'estimation de la pose 6D d'un objet dans une image [Keh+17 ; RL17 ; Pav+17 ; XSNF18 ; Sun+18 ; SMDT20 ; TSF18 ; ORL18 ; PPV19 ; ZSI19 ; HHFS19], ce qui est équivalent à chercher la pose de la caméra par rapport à cet objet. Ces méthodes reposent fortement sur des réseaux de neurones convolutifs et de l'apprentissage intensif avec une couverture complète de tous les points de vue de l'objet considéré. Des modèles 3D précis des objets, parfois texturés, sont alors nécessaires pour générer des images d'entraînement synthétiques (voir Figure 3). Ces modèles peuvent être obtenus par design (CAO) ou par reconstruction 3D préalable, par exemple, avec un scanner de haute précision, mais ne sont généralement pas disponibles pour tous les objets d'une scène. Ces méthodes sont donc plutôt adaptées à des environnement contrôlés, par exemple, pour des tâches de dévracage d'un certain type d'objet industriel, mais leur utilisation est plus limitée pour des objets inconnus dans une scène.

(a) Échantillonnage de points de vue pour la génération d'images synthétiques d'entraînement.



(b) Estimation de poses 6D d'objets.

FIGURE 3 – Images synthétiques d'entraînement et poses 6D estimées pour les objets [Sun+18].

Modélisation approximative des objets. Dans la continuité des travaux de Gaudillière *et al.* [GSB19a ; GSB19b ; GSB20], nous cherchons à utiliser des objets dans des environnements non modélisés et nous nous attachons donc à proposer des méthodes ne nécessitant pas de modèles précis des objets. Au contraire, nous adoptons une représentation approximative mais plus générique des objets par des primitives géométriques virtuelles. Ces modélisations ont l'avantage d'être compactes et légères, n'ont pas besoin d'être parfaitement ajustées aux objets 3D mais donnent tout même une idée sur leur position, leur forme et leur taille.

Dans la littérature, on retrouve principalement deux formes de représentation approximative des objets en 3D : les cuboïdes et les ellipsoïdes (voir Figure 4). Gaudillière *et al.* ont exploré la représentation par des ellipsoïdes, qui a plusieurs avantages :

— La projection d'un ellipsoïde dans une image s'écrit de manière continue sous la forme d'une ellipse, dont on peut facilement obtenir l'équation, pour n'importe quel point de vue. Cela n'est pas possible pour un cuboïde dont la projection sous la forme d'un polygone dépend du point de vue et doit être traitée par morceaux.

— La reconstruction d'un ellipsoïde est possible à partir de trois ellipses et une solution analytique a été développée par Rubino *et al.* [RCB18]. Au contraire, la reconstruction d'un cuboïde 3D à partir de boîtes 2D est moins évidente.

— L'ellipsoïde a déjà été utilisé comme primitive géométrique pour décomposer des objets dans les travaux de Paschalidou *et al.* [PUG19]. Cela permet de rendre l'estimation de la pose de la caméra compatible avec de telles décompositions.

Qu'est-ce qu'un bon objet pour la localisation ? Dans ce travail, le terme « objet » est employé au sens large. En effet, il fait référence à n'importe quel élément statique d'une scène qui puisse être détecté dans une image. Cela comprend des objets très classiques, comme une table, une chaise ou une lampe, dont les images sont disponibles en très grande quantité dans

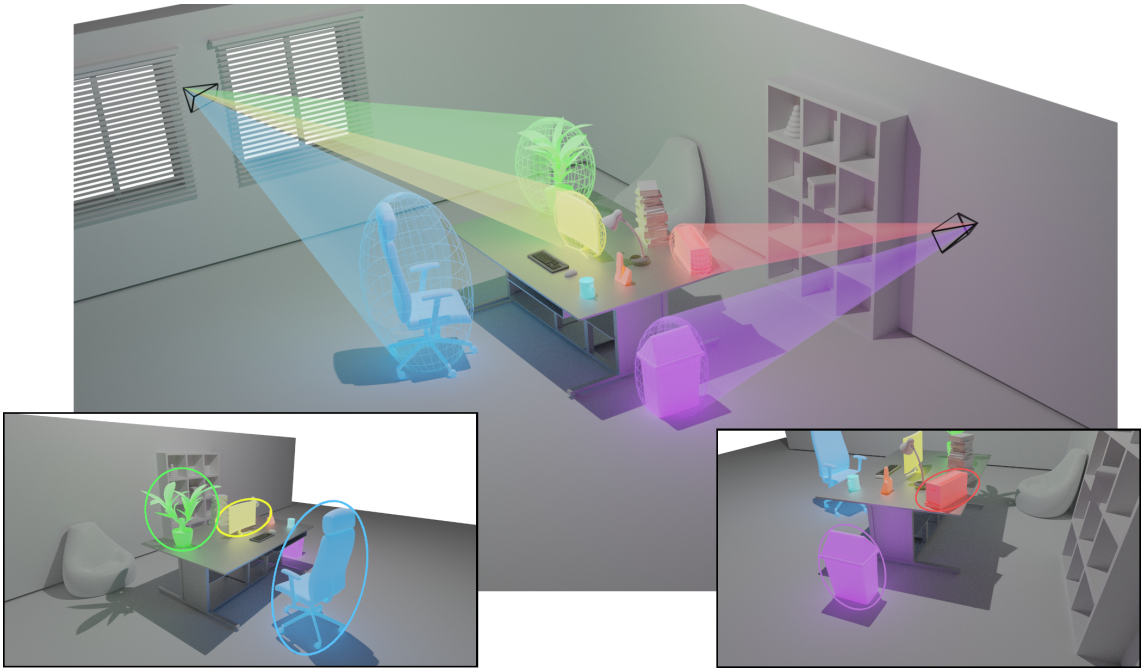


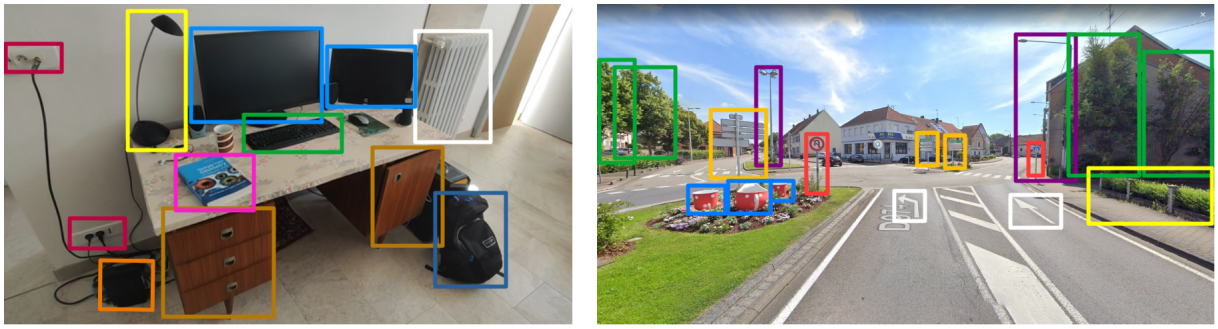
FIGURE 4 – Localisation visuelle basée sur des objets modélisés par des ellipsoïdes.

des bases de données telles que COCO [Lin+14a]. Ces objets peuvent être détectés en utilisant un réseau pré-entraîné. Étant donné qu'ils sont utilisés comme des balises pour le calcul de la pose de la caméra, des objets fixes sont privilégiés. En intérieur, les éléments du mobilier sont particulièrement utiles, par exemple, un évier, un interrupteur, une prise électrique, un cadre, un four, une télévision ou encore une armoire. Dans des scène extérieures, il peut s'agir d'arbres, de clôtures, de lampadaires, de panneaux et feux de signalisation ou des portes et des fenêtres des bâtiments.

Des objets plus spécifiques à certains environnements de travail peuvent également être utilisés. Dans un musée, on retrouve par exemple, les différentes pièces exposées, par exemple des statues, des peintures ou des sculptures, mais aussi les éléments du décor tels que des présentoirs, des panneaux ou des étiquettes.

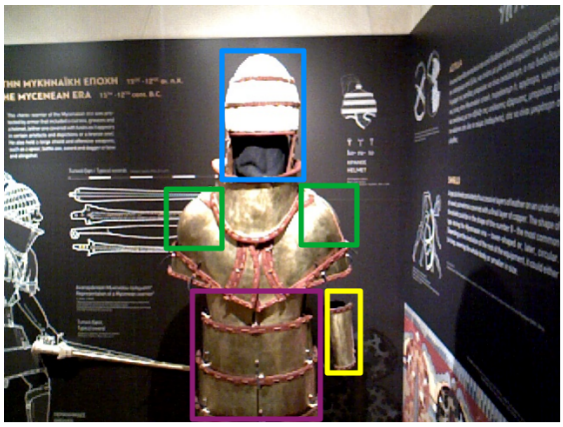
Enfin, la notion d'objet contient également un niveau de granularité pouvant être ajusté en fonction des besoins. Dans le cadre industriel, on préférera, par exemple, décomposer une grosse machine en plusieurs parties : vanne, moteur, capteur, levier ou voyant. La détection d'objets spécifiques ou de parties d'objets est possible en ajustant un réseau de détection pré-entraîné sur quelques dizaines d'images annotées. La figure 5 illustre la détection d'objets utiles pour se localiser dans différents environnements.

Motivations. Les méthodes existantes pour le calcul de la pose de la caméra à partir de correspondances ellipse-ellipsoïde, P1E [GSB19a ; GSB19b] et P2E [GSB20], utilisent des détections d'objets sous la forme d'ellipses inscrites dans des boîtes, qui sont donc nécessairement alignées avec les axes de l'image. Ce biais de détection constitue une source importante d'imprécision dans la pose estimée. De plus, dans ces travaux, la pose est calculée à partir d'un ensemble minimal de paires ellipse-ellipsoïde (une ou deux), les autres servant seulement à valider la pose. Ces limites actuelles constituent la première motivation à notre travail, dans lequel nous allons chercher à améliorer la précision et la robustesse de ces méthodes de positionnement visuel basées objet.

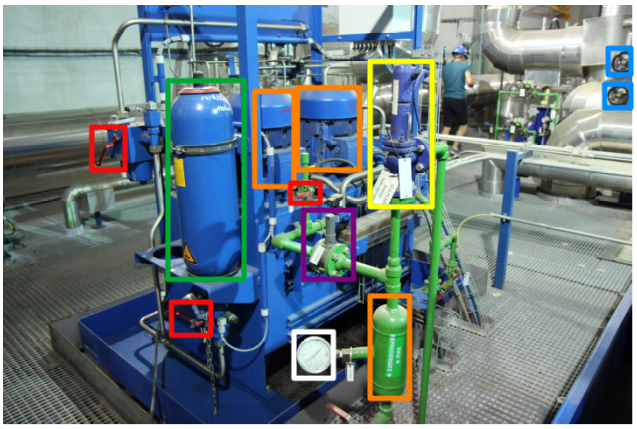


(a) Scène intérieure

(b) Scène extérieure



(c) Scène de musée



(d) Environnement industriel

FIGURE 5 – Détections d'objets (ou de parties) dans différents environnements pouvant servir à la localisation visuelle.

Dans un second temps, nous visons également à mettre au point un système de cartographie automatique des objets, dans des environnements inconnus, pouvant être intégré à des applications de réalité augmentée. Notre seconde motivation vient du fait que les méthodes existantes de SLAM (*Simultaneous Localization And Mapping*) basé objet nécessitent une intervention manuelle pour l'association des objets, par exemple dans QuadricSLAM [NMS19] et SO-SLAM [Lia+22], ou requièrent des informations supplémentaires telles que la direction de la verticale dans EAO-SLAM [Wu+20].

2 Contributions et organisation du manuscrit

Le chapitre 2 est consacré à l'amélioration de la détection des objets, de manière plus cohérentes avec leurs modèles ellipsoïdaux. Nous y proposons, en particulier, un réseau de neurones de prédiction d'ellipse orientée, mis en œuvre pour chaque objet. Nous mettons en évidence la meilleure précision de pose obtenue avec cette détection à travers plusieurs expériences. Nous proposons également de calculer la pose de la caméra à partir des centres de trois paires ellipse-ellipsoïde, sous la forme d'un problème P3P. Comme pour P1E [GSB19a ; GSB19b] et P2E [GSB20], la pose est validée en mesurant le recouvrement entre les ellipsoïdes projetés et les ellipses de détection. À la fin de ce chapitre, nous proposons une méthode jointe de détection d'instances d'objets et de prédiction d'ellipses qui facilite sa mise en oeuvre, avec un seul réseau, et réduit le temps de calcul de la pose.

Dans le chapitre 3, nous nous intéressons à l'amélioration de la précision du calcul de la pose. Nous y développons une étape de raffinement de la pose par la minimisation d'une erreur de reprojection d'ellipsoïdes. Celle-ci permet de prendre en compte tous les objets détectés dans l'image et associés à des objets de la scène, contrairement à l'estimation initiale de la pose qui est faite à partir d'un ensemble minimal d'objets (2 ou 3). Nous proposons également une métrique ellipse-ellipse basée sur des ensembles de niveaux que nous comparons avec d'autres formulations de coût en termes de précision de pose obtenue et de traitement des objets partiellement détectés. Enfin, nous explorons l'estimation de l'incertitude associée à la prédiction d'un réseau de neurones, dans le but de pondérer les contributions des différents objets dans le raffinement de la pose.

Alors que dans les chapitres 2 et 3, nous nous sommes intéressés à l'estimation de la pose de la caméra par rapport à un modèle de scène créé lors d'une étape de pré-traitement, à partir de quelques annotations manuelles, dans le chapitre 4, nous cherchons à mettre au point un système de reconstruction automatique d'une carte des objets de la scène. Nous développons un SLAM monoculaire intégrant la notion d'objet, permettant leur cartographie à la volée, mais également d'en tirer avantage pour améliorer la robustesse de sa relocalisation.

Les principales contributions des travaux présentés dans ce manuscrit se résument ainsi :

— Une détection des objets cohérente avec la projection de leurs modélisations ellipsoïdales à travers un réseau de prédiction d'ellipse orientée.

— Un calcul de la pose de la caméra à partir de correspondances ellipse-ellipsoïde, traité comme un problème P3P en utilisant leurs centres. Nous montrons que l'erreur d'approximation reste faible et que la pose calculée est plus stable que celle obtenue avec la méthode existante à partir de deux paires ellipse-ellipsoïde.

— Un raffinement de la pose de la caméra cherchant à aligner les cônes de projection des ellipsoïdes avec les cônes de retroprojection issus des ellipses et qui permet de prendre en compte tous les objets détectés dans l'image et associés à des objets de la carte.

— Une métrique ellipse-ellipse basée sur des ensembles de niveaux qui dispose de caractéristiques avantageuses pour la gestion des objets partiellement détectés. Nous l'avons comparée avec d'autres formulations de coût entre deux ellipses.

— Une estimation de l'incertitude associée à la prédiction d'une ellipse par notre réseau de neurones, permettant de pondérer la contribution des objets dans le raffinement en fonction de la confiance de leur détection.

— Un système de SLAM monoculaire permettant une cartographie automatique des objets de la scène sous la forme d'ellipsoïdes.

— Une amélioration de la capacité de relocalisation d'un SLAM en faisant collaborer les points et les objets et ainsi bénéficier de leurs avantages respectifs de précision et de robustesse.

2.1 Publications et logiciels associés

Les travaux menés dans le cadre de cette thèse ont fait l'objet de trois publications à des conférences internationales et une publication dans un journal international :

[ZSB22c] Matthieu Zins, Gilles Simon et Marie-Odile Berger. "Object-Based Visual Camera Pose Estimation From Ellipsoidal Model and 3D-Aware Ellipse Prediction". In : *International Journal of Computer Vision* (2022)

[ZSB22b] Matthieu Zins, Gilles Simon et Marie-Odile Berger. "OA-SLAM : Leveraging Objects for Camera Relocalization in Visual SLAM". In : *2022 IEEE International Symposium on Mixed and Augmented Reality, ISMAR*. 2022

[ZSB22a] Matthieu Zins, Gilles Simon et Marie-Odile Berger. "Level Set-Based Camera Pose Estimation From Multiple 2D/3D Ellipse-Ellipsoid Correspondences". In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. 2022

[ZSB20] Matthieu Zins, Gilles Simon et Marie-Odile Berger. "3D-aware ellipse prediction for object-based camera pose estimation". In : *2020 International Conference on 3D Vision (3DV)*. 2020

Les différents codes développés au cours de cette thèse sont disponibles à l'adresse suivante : `gitlab.inria.fr/tangram`. Ils comprennent :

— `PyEllCV` : une bibliothèque écrite en C++ et Python pour la manipulation des ellipses et des ellipsoïdes et l'estimation de la pose de la caméra (Chapitres 2 et 3).

— `3D-Aware-Ellipses-for-Visual-Localization` : code Python pour la prédiction d'ellipses cohérentes aux modèles ellipsoïdaux des objets (Chapitre 2).

— `Level-Set-Based-Camera-Pose-Estimation` : code Python pour pour le raffinement de la pose de la caméra par la minimisation d'une erreur de reprojection entre ellipses (Chapitre 3).

— `OA-SLAM` : implémentation en C++ de notre système de SLAM basé objet (Chapitre 4).

Chapitre 1

Positionnement visuel

Sommaire

Le positionnement visuel d'une image consiste à estimer la position et l'orientation de la caméra qui l'a capturée, par rapport à un modèle de la scène. C'est un sujet important qui a été traité dans de nombreux travaux. Nous débutons ce chapitre en présentant les approches classiques basées sur la structure de la scène reconstruite sous la forme d'un nuage de points dans la section 1.1. Nous présentons ensuite des méthodes plus récentes de prédiction de pose ou de coordonnées de scène dans la section 1.2. Le positionnement par recherche d'image proche et estimation de pose relative entre images est abordé dans la section 1.3. Des méthodes d'estimation de pose de caméra basées sur des objets sont présentées dans la section 1.4. Enfin, nous terminons ce chapitre par quelques rappels sur la modélisation des objets par des ellipsoïdes et les méthodes existantes de calcul de pose de caméra à partir de paires ellipse-ellipsoïde dans la section 1.5.

1.1 Approches basées structure

Les approches les plus classiques de positionnement visuel se basent sur une représentation de la scène sous la forme d'un nuage de points 3D. L'estimation de la pose de la caméra est obtenue grâce à des associations entre des points d'intérêt 2D détectés dans l'image et des points 3D de la scène (voir Figure 1.1). Le nuage de points d'une scène est généralement reconstruit par des méthodes de Structure-from-Motion (SfM) telles que COLMAP [SF16] ou Bundler [SSS08]. Selon la taille de la scène et la densité du nuage de points, cette reconstruction peut nécessiter un temps de calcul très important. Elle est donc exécutée comme une étape de pré-traitement. Les points triangulés sont associés à leurs descripteurs (par exemple SIFT) dans les images où ils sont visibles.

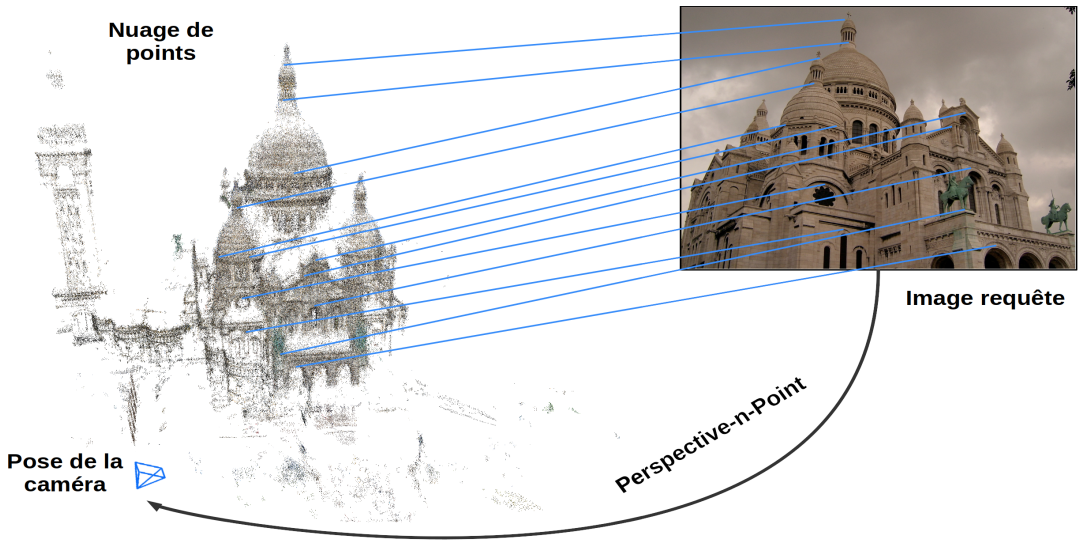


FIGURE 1.1 – Localisation visuelle basée structure.

Détection et descriptions de points d'intérêt

Le calcul de points d'intérêt dans une image se fait généralement en deux temps, avec d'abord une phase d'extraction qui détecte des points saillants, puis la description des points détectés en utilisant de l'information locale issue de leur voisinage. Plus précisément, le rôle du détecteur de points d'intérêt est de rechercher des indices de manière répétable dans l'image, c'est-à-dire qui puissent être détectés dans d'autres images. Ils doivent notamment être suffisamment invariants à des translations, des rotations, des changements d'échelle et des transformations photométriques.

L'un des premiers détecteurs de point d'intérêt a été développé par Moravec [Mor80], mais est fortement sensible au bruit dans l'image et n'est pas invariant à des rotations. Harris *et al.* l'ont amélioré en proposant un détecteur de coins [HS88], invariant à des translations, rotations et variations d'illumination. Ce détecteur reste cependant sensible à des changements d'échelle. Mikolajczyk *et al.* ont proposé une approche multi-échelle de ce détecteur [MS04]. Lowe a développé un détecteur basé sur des différences de gaussiennes [Low04] et y a intégré la notion d'espace d'échelle, ce qui rend la détection de points d'intérêt plus stable. Matas *et al.* ont proposé de définir des régions d'intérêt, nommées *Maximally Stable Extremal Regions* (MSER), invariantes aux transformations affines [MCUP02]. Le détecteur FAST [RD06] (*Features from Accelerated Segment Test*) a été développé par Rosten *et al.* et repose sur de l'apprentissage automatique pour offrir une détection de coins très rapide.

Afin d'établir des correspondances de points d'intérêt, dans plusieurs images ou par rapport à une carte de points 3D, des descripteurs sont calculés. Ils permettent de caractériser chaque point par une information locale extraite de leur voisinage. Un descripteur doit posséder les propriétés suivantes :

— Robustesse : l'information visuelle doit être invariante à des distorsions géométriques ou des changements d'illuminations.

— Discrimination : un descripteur doit être suffisamment informatif pour permettre de distinguer différents indices dans l'image.

— Efficacité : le descripteur doit avoir une faible empreinte mémoire et son calcul doit être rapide.

Le descripteur le plus connu est probablement SIFT [Low04] (*Scale-Invariant Feature Transform*) qui est couplé au détecteur par différence de gaussiennes. Il se base sur des histogrammes de gradients orientés pour fournir un vecteur de taille 128. Malgré son coût de calcul relativement important, SIFT est considéré comme une référence en matière d'appariement de points. Le descripteur SURF [BTG06] (*Speeded Up Robust Features*) est une version allégée de SIFT et est généralement utilisé lorsqu'une exécution en temps réel est nécessaire [CN10 ; QSHP16 ; Stu+16]. Le descripteur BRIEF [CLSF10] (*Binary Robust Independent Elementary Features*) calcule une signature binaire à partir de comparaisons entre les intensités des pixels au voisinage d'un point d'intérêt. Les descripteurs binaires ont l'avantage de pouvoir être comparés de manière très rapide en utilisant la distance de Hamming. Le descripteur ORB [RRKB11] (*Oriented FAST and Rotated BRIEF*) introduit une invariance à la rotation dans BRIEF et le combine avec une version orientée du détecteur FAST. ORB combine de très bonnes performances de description et un coût de calcul faible, ce qui en fait une très bonne alternative à SIFT et SURF. Il est notamment utilisé dans le système ORB-SLAM [MMT15].

Plus récemment, de nouvelles méthodes basées sur de l'apprentissage profond sont apparues, fournissant des alternatives à la détection [Sav+17 ; ZR18], à la description [BRPM16 ; Sim+15 ; SVZ14] ou aux deux [YTLF16 ; OTFY18 ; DMR18 ; Dus+19 ; RDHW19]

Les points d'intérêt sont généralement obtenus à partir d'une carte de saillance prédite. Savinov *et al.* ont développé Quad-Net [Sav+17], une approche non-supervisée de détection de points d'intérêt qui se base sur l'idée que le classement des points saillants extraits d'une image doit être préservé malgré des transformations appliquées à l'image.

Dans les méthodes de description apprises, le réseau de neurones choisit par lui-même les éléments importants à garder pour obtenir des signatures particulièrement discriminantes. Ces descripteurs sont généralement entraînés en utilisant une fonction de coût par triplets [Wan+14 ; HA15 ; BRPM16] ou contrastive [RTC16]. L'objectif est de minimiser la distance entre deux descripteurs correspondant à une association valide et, au contraire, de maximiser l'éloignement entre deux descripteurs non associés. Simo-Serra *et al.* ont utilisé un réseau de neurones siamois entraîné à partir de paires de sous-images correspondantes ou non [Sim+15].

Des méthodes apprises réalisant à la fois la détection et la description de points d'intérêt ont également été développées. Yi *et al.* ont proposé LIFT [YTLF16] (Learned Invariant Feature Transform), qui implémente la détection, l'estimation de l'orientation et la description, à partir d'une architecture de réseau profond différentiable de bout en bout. Ono *et al.* ont développé LF-NET [OTFY18], une méthode complète de détection et description, entraînable sans supervision. Pour cela, ils créent des cibles virtuelles à partir de données de profondeur des images et des poses relatives des caméras. DeTone *et al.* ont proposé SuperPoint [DMR18] une méthode auto-supervisée de détection et description de points d'intérêt, apprise à partir d'images générées artificiellement et contenant des éléments géométriques de base, tels que des coins et des arêtes. D2-Net [Dus+19] a été proposé par Dusmanu *et al.* et utilise également un unique réseau de neurones pour réaliser à la fois la détection et la description. La détection se base sur les maxima locaux à travers les canaux et les dimensions spatiales des cartes de caractéristiques. Dans R2D2 [RDHW19], Revaud *et al.* ont également proposé une détection et une description jointes et se différencient par la prédiction de la fiabilité et de la répétabilité des points d'intérêt et de leurs descripteurs.

Schönberger *et al.* ont réalisé une comparaison [SHSP17] entre ces méthodes et les descripteurs classiques sans apprentissage. Ils montrent que les versions les plus avancées des descripteurs classiques, telles que RootSIFT [AZ12] ou DSP-SIFT [DS15], ont des performances égales ou supérieures aux méthodes apprises, qui présentent quant à elles une variance importante entre les différents ensembles de données.

Mise en correspondance et calcul de pose

Une fois que des points d'intérêt ont été détectés et associés à des descripteurs, des correspondances 2D-3D entre ces points et ceux du modèle 3D de la scène sont recherchées. Une recherche exhaustive étant généralement trop coûteuse en calcul, de nombreux travaux ont tenté d'améliorer cette étape à travers une recherche par arbres [IZFB09], l'utilisation de mots visuels [IZFB09 ; SLK11 ; SLK12] ou encore une méthode d'indexation optimisée pour des descripteurs binaires [FFW15]. Des appariements inversés (3D-2D) [LSH10], bidirectionnels (2D-3D et 3D-2D) [SLK12 ; LSHF12] et cohérents en termes de covisibilité des points [Sat+15 ; LLD17] ont également été proposés, ainsi que diverses stratégies de rejet de mauvaises associations [SEOK14 ; SEKO16 ; ZSP15].

Le problème d'estimation la pose de la caméra à partir de correspondances 2D-3D est nommé *Perspective-n-Point* (PnP) [MUS15] (voir Figure 1.2). Différentes approches pour résoudre ce problème sont, par exemple, implémentées dans OpenCV [Bra00]. L'une des approches les plus utilisées consiste à combiner un algorithme P3P [GHTC03 ; KSS11] avec une boucle RANSAC [FB81].

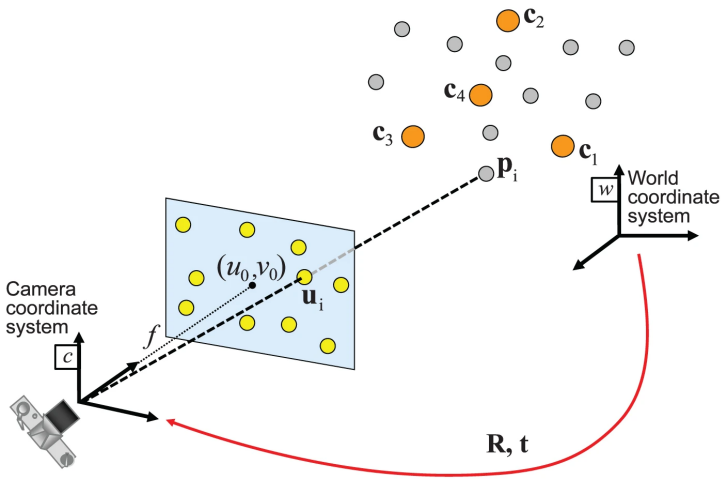


FIGURE 1.2 – Problème de Perspective-n-Point (PnP) pour l'estimation de la pose de la camréra à partir de correspondances de points 2D-3D [MUS15].

P3P est une version spécifique du problème PnP à partir de seulement trois correspondances. La plupart des approches fonctionnent en deux temps. Elles cherchent, tout d'abord à estimer la profondeur des trois points de l'image en résolvant une équation polynomiale de degré 4, puis recherchent la transformation rigide entre les points 3D exprimés dans le repère de la caméra et ceux dans le repère de la scène. Ces méthodes fournissent quatre hypothèses de poses et la prise en compte d'une quatrième correspondance permet de sélectionner la bonne hypothèse.

Une boucle RANSAC est utilisée pour éliminer les correspondances aberrantes. L'idée est de choisir aléatoirement un ensemble minimal de correspondances (3 pour P3P), d'estimer la pose de la caméra à partir de celles-ci, puis de calculer le nombre de correspondances cohérentes par rapport à la pose obtenue parmi toutes les correspondances disponibles. Ces étapes sont répétées et la pose avec le plus grand nombre d'*inliers* est choisie.

Cette approche (P3P+RANSAC) fournit une première estimation qui est généralement raffinée en minimisant la distance entre les points 2D et les projections des points 3D à partir des correspondances considérées comme *inliers*. Cette minimisation non linéaire peut être réalisée avec l'algorithme Gauss-Newton [Gau09] ou Levenberg-Marquardt [Mar63].

Méthodes de localisation basées structure

Irschara *et al.* ont combiné un modèle de scène 3D compressé avec une méthode d'indexation de descripteurs basée sur des arbres [IZFB09]. Ils génèrent également des vues synthétiques pour étendre la couverture de points de vue du modèle, et ainsi, étendre la robustesse de leur localisation. Dans [LSH10], les auteurs ont proposé d'inverser le sens de la mise en correspondance, des points 3D vers les images. Ils ont montré qu'il est possible de se localiser dans des scènes de la taille d'une ville à partir d'images issues d'Internet. Sattler *et al.* ont associé les points 3D du modèle de scène à des mots visuels pour une identification rapide de possibles correspondances, qui sont ensuite vérifiées de manière plus précise [SLK11]. Les mêmes auteurs ont par la suite amélioré leur méthode, en précision et en vitesse, par la recherche d'associations bidirectionelles, 2D vers 3D et 3D vers 2D [SLK12]. Li *et al.* ont également exploité la mise en correspondance bidirectionnelle et ont utilisé une hypothèse de co-occurence de points 3D dans un voisinage proche pour améliorer la recherche [LSHF12]. Leur méthode a permis de traiter des modèles de scène allant jusqu'à plusieurs dizaines de millions de points 3D.

Des chercheurs se sont également intéressés à la localisation visuelle sur des appareils mobiles avec une taille mémoire et des capacités de calcul limitées [MSUK14 ; Lyn+15]. Ces méthodes combinent une localisation globale à grande échelle réalisée sur un serveur externe et un suivi de la caméra local en temps-réel, de type SLAM, exécuté sur l'appareil mobile.

Lu *et al.* ont proposé une méthode de localisation à partir d'une courte vidéo [Lu+15]. Ils reconstruisent un modèle 3D à partir de cette vidéo et cherchent des associations 3D-3D avec des points du modèle de la scène. Feng *et al.* ont considérablement réduit la puissance de calcul nécessaire à la localisation grâce à l'utilisation de descripteurs binaires et à une nouvelle méthode d'indexation, très efficace pour la recherche approximative du plus proche voisin entre de tels descripteurs [FFW15]. Dans [Sat+15], les auteurs ont utilisé un graphe de covisibilité pour guider l'échantillonnage de RANSAC et rejeter des mauvaises associations. Svärm *et al.* ont développé une méthode de rejet de correspondances aberrantes leur permettant de traiter de grandes scènes avec un très fort taux d'*outliers* (99%) [SEOK14]. Dans [SEKO16], les mêmes auteurs y ont intégré la connaissance de la direction verticale qui, de nos jours, peut être fournie par les appareils photos ou les téléphones intégrant un capteur de gravité. Zeisl *et al.* ont proposé une stratégie de vote de Hough [Hou59] pour l'estimation de la pose de la caméra avec une complexité linéaire par rapport au nombre de correspondances. Cela leur permet de considérer beaucoup plus d'associations que les méthodes existantes de complexité quadratique [ZSP15].

En plus du temps de calcul, le passage à grande échelle induit également des ambiguïtés de pose causées par des zones similaires et faisant échouer les méthodes basées sur des comparaisons individuelles de descripteurs. Liu *et al.* ont proposé une approche globale basée sur un réseau de Markov [LLD17], qui tient compte non seulement des similitudes visuelles entre les correspondances 2D-3D, mais aussi de leurs compatibilités globales, mesurées en termes de covisibilité, parmi toutes les paires d'appariements dans la scène.

Des méthodes récentes se sont attaquées à la localisation à long terme [Tof+18 ; Tof+20]. Ce problème est rendu particulièrement difficile par les changements d'apparence importants des scènes, entre le moment de la création de leurs modèles et leur utilisation (jour-nuit, saison, météo, construction/destruction de bâtiment). Dans [Tof+18], Toft *et al.* ont, par exemple, utilisé l'information sémantique extraite de l'image requête sous forme d'une segmentation pour filtrer des mauvaises correspondances 2D-3D et ont montré que cela permettait d'améliorer largement les performances de localisation.

Très récemment, Sarlin *et al.* ont proposé d'apprendre des couches de caractéristiques avec un réseau de neurones convolutif et ont transformé le problème d'estimation de la pose de la caméra en un problème d'alignement d'images [Sar+21]. Les caractéristiques apprises leur ont permis d'obtenir un large bassin de convergence. Enfin, dans [GLB21], Germain *et al.* soutiennent que l'utilisation de correspondances 2D-3D éparses entraîne une perte d'information importante. Ils ont proposé de remplacer la formulation classique de l'erreur de reprojection entre points par une erreur de reprojection neuronale qui exploite des informations plus riches et élimine, par exemple, le besoin de choisir une fonction de coût robuste.

Des analyses plus approfondies des méthodes de calcul de pose basées structure sont disponibles dans [LF+05] et [MUS15].

1.2 Approches par apprentissage profond

Plus récemment, des méthodes basées sur de l'apprentissage profond, entraînées de bout en bout, ont émergées. Elles encodent la scène dans un réseau de neurones et permettent de régresser une pose de caméra absolue à partir d'une image d'entrée [KGC15 ; KC17 ; KC16 ; NB17 ; RVB18 ; Wal+17 ; Bra+18] ou de prédire des coordonnées 3D de la scène [Sho+13 ; Cav+17 ; BR18 ; BAIN18 ; Bra+17 ; BR22 ; BR19 ; Cav+19 ; Yan+19].

1.2.1 Prédiction de pose absolue

La régression de pose absolue est illustrée sur la figure 1.3. PoseNet [KGC15] a été la première méthode à prédire des poses de caméra absolues. Kendall *et al.* ont, par la suite, utilisé la notion de réseau bayésien pour estimer l'incertitude de la pose prédite [KC16] et ont remplacé la fonction de coût initiale, qui nécessitait l'ajustement d'hyper-paramètres pour équilibrer les termes de rotation et de translation, par une fonction de coût géométrique apprise [KC17]. Une architecture de type *Long Short-Term Memory* (LSTM) a été proposée par Walch *et al.* [Wal+17] pour limiter le problème de sur-apprentissage de PoseNet. Clark *et al.* ont développé VidLoc [Cla+17], dans lequel ils ont proposé de prédire la pose de la caméra pour une séquence d'images en utilisant un réseau de neurones récurrent. Dans MapNet [Bra+18], Brahmbhatt *et al.* ont également proposé d'encoder la scène dans un réseau de neurones profond. Cette méthode est inspirée de PoseNet mais considère, en plus, des contraintes géométriques entre des paires d'images au cours de l'entraînement. Ces contraintes de translation ou de rotation sont issues de données d'odométrie visuelle, de GPS ou de capteur inertiel.

Ces méthodes ont permis de traiter des conditions dans lesquelles les méthodes classiques échouent, comme les changements d'illumination ou le flou de mouvement. De plus, elles présentent un temps d'inférence constant, par rapport aux méthodes basées structure qui nécessitent généralement de lourds calculs de correspondance 2D-3D à l'intérieur d'une boucle RANSAC.

La régression de pose est cependant spécifique à une scène et nécessite donc d'être entraînée pour chaque nouvel environnement de travail. De même, elle ne s'applique pas bien à de nouveaux points de vue, différents de ceux utilisés pour l'apprentissage. Ces méthodes n'atteignent pas le même niveau de précision que les approches plus classiques présentées dans la section précédente. Sattler *et al.* ont d'ailleurs montré que la régression de pose absolue est plus proche d'une approximation de la pose par une recherche d'image similaire que d'une estimation précise de la pose par un raisonnement géométrique à partir de la structure 3D de la scène [SZPL19]. Enfin, elle est également limitée à de petites scènes par la capacité du réseau.

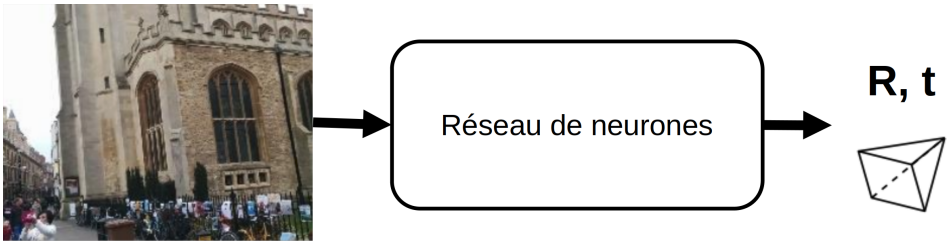


FIGURE 1.3 – Localisation visuelle par régression de pose absolue de caméra.

1.2.2 Prédiction de coordonnées de scène

Une autre approche consiste à prédire les coordonnées 3D de la scène des pixels d'une image d'entrée, puis à utiliser un algorithme PnP pour estimer la pose de la caméra (voir Figure 1.4). Cette approche permet d'éviter le besoin d'un paramètre d'équilibrage entre les termes de translation et de rotation dans la fonction de coût, qui est généralement compliqué à ajuster. Shotton *et al.* ont d'abord utilisé des forêts de régression pour prédire les coordonnées de scène [Sho+13]. Plus récemment, Brachmann *et al.* ont utilisé un réseau de neurones convolutif pour prédire ces coordonnées [BR18] et l'ont ensuite couplé avec une version différentiable de RANSAC [Bra+17], pour estimer la pose de la caméra. Bui *et al.* ont proposé une méthode similaire et prédisent, en plus, des incertitudes associées aux coordonnées, ce qui leur permet de rejeter des mauvaises prédictions [BAIN18].

Ces approches obtiennent une très bonne précision pour des scènes relativement petites mais, comme les méthodes de régression de pose absolue, ne s'étendent pas à des scènes de plus grande échelle à cause de la capacité limitée du réseau. Dans ESAC [BR19], Brachmann *et al.* ont amélioré le passage à l'échelle en entraînant un ensemble de réseaux, chacun spécialisé dans une partie de la scène. Li *et al.* ont développé une approche hiérarchique permettant de mieux gérer les ambiguïtés qui apparaissent dans des grands environnements. Ces deux méthodes restent cependant moins précises à grande échelle que les approches basées structure.

Tout comme les approches de prédiction de pose absolue, ces méthodes sont spécifiques à une scène et doivent être ré-entraînées pour chaque nouvelle scène. Pour y remédier, Yang *et al.* ont proposé SANet [Yan+19], une méthode de prédiction de coordonnées, indépendante de la scène. Pour cela, ils ont retiré la représentation de la scène des poids du réseau et régressent ses coordonnées 3D à partir d'un nuage de points de la scène fourni en entrée.

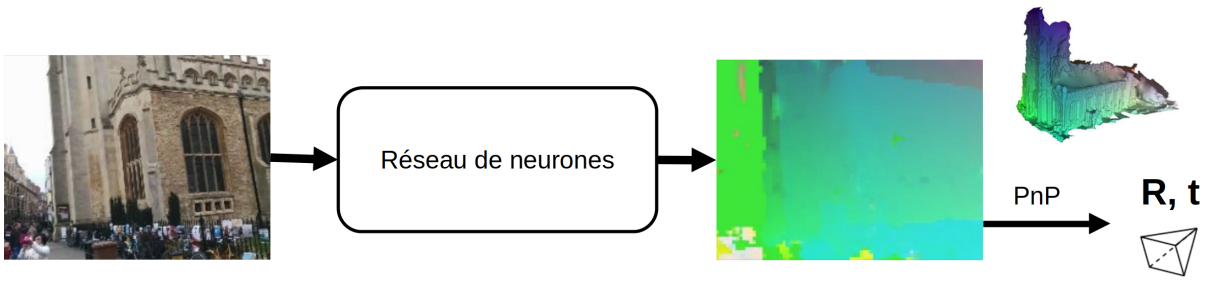


FIGURE 1.4 – Localisation visuelle par prédiction de coordonnées de scène.

1.3 Approches par recherche d'image et pose relative

Une autre approche de localisation visuelle se base sur la recherche d'une ou plusieurs images similaires à l'image requête, dans une base de données d'images géo-localisées de la scène. Cette approche est également connue sous le nom de reconnaissance de lieu [Low+15 ; Sün+15 ; Tor+15 ; CS13 ; TSPO13 ; JDF17 ; AZ14 ; SHSP16 ; WKP16 ; Che+17].

Pour permettre une recherche d'image efficace, des descripteurs globaux des images sont utilisés. GIST [OT01] a été l'un des premiers descripteurs fournissant une représentation globale d'une image. D'autres méthodes bien connues sont par exemple, les vecteurs de Fisher [PLSP10], les sacs de mots visuels [GT12b] (BoW) ou VLAD [JDSP10 ; DGJP13] (*Vector of Locally Aggregated Descriptors*), qui consistent en des agrégats de caractéristiques visuelles d'une image. Cummins *et al.* ont proposé, par exemple, FAB-MAP [CN08 ; CN10], un modèle probabiliste de recherche d'image proche basé sur l'apparence. Il repose sur un dictionnaire de mots visuels calculés sur des descripteurs SIFT [Low04] ou SURF [BTG06] et est notamment utilisé pour traiter les problèmes de fermeture de boucle et de relocalisation dans un SLAM.

Avec l'émergence des réseaux de neurones convolutifs, des descripteurs globaux appris sont apparus [BSCL14]. L'architecture NetVLAD [Ara+16] a été introduite par Arandjelovic *et al.* et a été utilisée de manière faiblement supervisée pour la reconnaissance de lieu. Elle a montré des résultats remarquables, surpassant l'état de l'art des représentations d'images non apprises. Ces descripteurs sont généralement combinés avec des méthodes de recherche efficaces [NS06 ; Phi+07].

Hausler *et al.* ont proposé Patch-NetVLAD [Hau+21], qui extrait des régions de l'espace des caractéristiques d'une image pour combiner les avantages d'une description locale et globale et l'ont appliqué à la reconnaissance de lieu. Dans [Sün+15], Sünderhauf *et al.* ont également proposé une approche hybride locale/globale. Ils calculent des descripteurs sur des régions extraites des images, en adaptant des techniques de proposition d'objets pré-entraînées et les utilisent comme indices pour la reconnaissance de lieu.

Dans [Tor+15], Torii *et al.* se sont attaqués au problème de reconnaissance de lieu à long terme dans des situations particulièrement compliquées, où la scène subit des changements d'apparence importants, par exemple, d'illumination (jour/nuit), des changements de saison ou des modifications de sa structure (construction/destruction de bâtiments). Ils ont proposé d'utiliser un descripteur dense, nommé DenseVLAD, créé par agrégation de descripteurs RootSIFT [AZ12] échantillonnés de manière dense dans toute l'image.

Cao *et al.* ont proposé une représentation de la base de données des images de référence sous la forme d'un graphe [CS13] et ont montré que sa richesse d'information permettait d'améliorer la reconnaissance de lieu basée sur des sacs de mots visuels. Les difficultés causées par les structures répétitives telles que les façades des bâtiments, les clôtures ou les marquages routiers ont été explorées par Torii *et al.* [TSPO13]. Kim *et al.* ont introduit un module de pondération contextuel qui prédit l'importance de certaines régions de l'image [JDF17]. Le passage à très grande échelle a également été exploré [AZ14 ; SHSP16 ; WKP16 ; Che+17].

Ces méthodes de localisation par recherche d'image sont considérées comme imprécises du point de vue du positionnement visuel, car elles approximent la pose de la caméra par celle de l'image de référence la plus proche. Elles sont tout de même très utiles pour réduire la zone de recherche des approches basées structure, permettant d'accélérer le calcul de la pose dans de grands environnements [SWLK12 ; Sat+17 ; LSHF12 ; CCPS18]. Sattler *et al.* ont, par exemple, combiné une recherche d'image avec des reconstructions locales de la scène [Sat+17].

La recherche d'image peut également être couplée avec une estimation de pose relative afin d'améliorer sa précision de localisation [LMKK17 ; BLP18 ; Din+19 ; Tai+18 ; PSDG19 ; ZSPL20 ;

FOBD18]. En comparaison des méthodes entraînées à prédire la pose absolue de la caméra ou des coordonnées 3D de scène, présentées dans la section 1.2, cette approche par pose relative ne nécessite pas d'encoder la géométrie de la scène et n'est donc pas spécifique à une scène. Laskar *et al.* ont proposé une telle approche, avec d'abord une recherche d'images proches, puis un calcul de poses relatives entre l'image requête et ces images [LMKK17]. La position de la caméra est triangulée à partir de deux estimations relatives de translation en utilisant une approche basée sur RANSAC. Les hypothèses multiples de l'orientation sont également filtrées avec RANSAC. Dans RelocNet [BLP18], Baltnas *et al.* ont introduit des informations de recouvrement de points de vue entre des paires d'images pour entraîner la description globale d'image. Cela leur permet de faire en sorte que la différence de descripteurs d'images représente bien un changement de pose de caméra. Ils entraînent également un réseau de régression de pose relative entre l'image d'entrée et les images de référence les plus proches. Ding *et al.* ont développé CamNet [Din+19], un système composé de trois modules : une recherche grossière d'images proches, une recherche de pose fine de la caméra par rapport à une image d'ancrage et une régression précise de pose relative.

Pour des environnements intérieurs, Taira *et al.* ont développé InLoc [Tai+18], qui combine une recherche d'image et une mise en correspondance dense pour estimer précisément la pose relative de la caméra. L'appariement dense est préféré à des points d'intérêt épars pour mieux gérer les zones faiblement texturées. Cette méthode a cependant un coût de calcul relativement élevé et nécessite de disposer des modèles 3D précis et denses de l'environnement. De manière similaire, Piasco *et al.* ont proposé une méthode qui combine une recherche d'image, une mise en correspondance dense et une prédiction de profondeur monoculaire [PSDG19]. Ils rendent leur approche légère et rapide, en utilisant une architecture commune pour calculer les descripteurs globaux des images et prédire la carte de profondeur.

Enfin, Zhou *et al.* ont récemment proposé une approche plus classique d'estimation de pose relative [ZSPL20], qui se base sur la mise en correspondance de points d'intérêt en utilisant des descripteurs SIFT. Les poses relatives sont calculées par rapport à un ensemble d'images proches, identifiées en utilisant des descripteurs globaux DenseVLAD [Tor+15]. Ils montrent que cette approche reste plus précise que les méthodes entraînées à régresser des poses relatives.

Finalement, une étude très complète sur toutes les méthodes d'estimation de la pose de la caméra (basées structure, par apprentissage profond ou par recherche d'image et calcul de pose relative) a été proposée par Xu *et al.* [Xu+22].

1.4 Approches basées objet

L'estimation de la pose d'un objet a été un domaine de recherche très actif au cours des dernières années. De nombreuses méthodes ont été développées pour estimer la pose d'un objet dans le repère de la caméra (voir Figure 1.5) [Keh+17 ; RL17 ; Pav+17 ; XSNF18 ; Sun+18 ; SMDT20 ; TSF18 ; ORL18 ; Cri+15 ; Cri+17 ; PPV19 ; ZSI19 ; HHFS19 ; PIL19], ce qui est équivalent à localiser la caméra par rapport à l'objet.

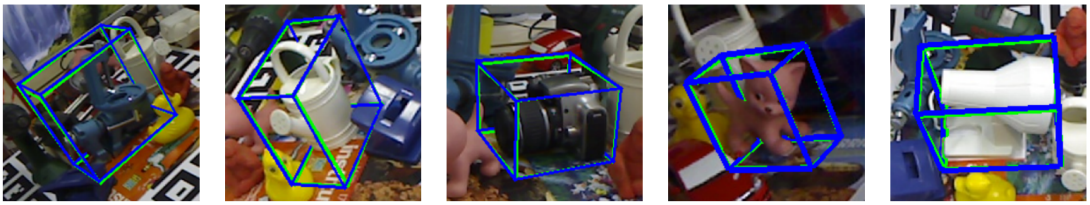


FIGURE 1.5 – Estimation de la pose à 6 dimensions (orientation et position) d'un objet.

Kehl *et al.* ont proposé SSD-6D [Keh+17] qui étend le détecteur d'objets SSD [Liu+16] et infère la pose 6D d'un objet par classification de points de vue discrets. Dans BB8 [RL17], Rad *et al.* détectent d'abord l'objet d'intérêt sous la forme d'une segmentation de l'image, puis prédisent les projections 2D des coins la boîte englobante de l'objet 3D. La pose de l'objet est ensuite calculée en résolvant le problème PnP. Ils génèrent des images synthétiques pour l'entraînement à partir des modèles 3D des objets et appliquent différentes stratégies, telles que la génération d'arrière-plans aléatoires, pour rendre l'estimation de la pose indépendante du contexte. Pavlakos *et al.* ont proposé de prédire une carte de saillance de points d'intérêt sémantiques et de les combiner avec un modèle d'objet déformable pour estimer sa pose [Pav+17]. Xiang *et al.* ont proposé de découpler la rotation et la translation [XSNF18] : avec la connaissance des paramètres intrinsèques de la caméra, la position d'un objet est calculée en utilisant les prédictions de son centre 2D dans l'image et de sa distance par rapport à la caméra. Son orientation est régressée sous la forme d'un quaternion.

Sundermeyer *et al.* ont proposé un auto-encodeur entraîné sur des vues synthétiques d'un modèle 3D d'un objet [Sun+18 ; SMDT20]. Ils appliquent du bruit aléatoire (illumination, réflection, arrière-plan, contraste, luminosité, flou, couleurs et occultations) sur l'image d'entrée et cherchent à reconstruire une image propre en sortie, ce qui pousse l'auto-encodeur à apprendre des codes invariants à ces éléments et caractérisant le point de vue sur l'objet. À l'inférence, ils recherchent un code similaire à celui obtenu, dans une base de données construite auparavant et qui contient les codes associés à tous les points de vue échantillonnés sur une sphère autour de l'objet. Ils calculent la position de l'objet à partir de son centre dans l'image et du rapport entre les diagonales de sa boîte de détection dans l'image requête et celle dans l'image associée au code correspondant dans la base de données qui montre l'objet du même point de vue.

Tekin *et al.* ont proposé un réseau d'estimation de pose d'objets [TSF18], inspiré de l'architecture du réseau de détection YOLO [RDGF16]. Ils prédisent, pour chaque objet détecté, les projections 2D de son centre et des coins de sa boîte englobante 3D. Comme dans BB8, la pose est ensuite calculée en résolvant le problème PnP à partir des correspondances 2D-3D. Ils montrent que leur méthode est suffisamment précise pour ne pas nécessiter d'étape de raffinement, contrairement à SSD-6D [Keh+17]. Leur approche est bien plus rapide que SSD-6D et BB8 et s'exécute en temps-réel.

Dans [ORL18], Oberweger *et al.* ont également proposé de calculer la pose d'un objet à partir de prédictions des projections 2D des coins de sa boîte englobante 3D. Ils notent cependant que les méthodes existantes de prédiction de ces points sont assez sensibles aux occultations, même lorsque de telles occultations sont présentes dans les données d'entraînement. Pour y remédier, ils proposent d'accumuler une multitudes de cartes de chaleurs indépendantes prédites à partir de sous-images et permettant de localiser les projections des coins.

Également dans le but d'améliorer leur robustesse aux occultations, plusieurs méthodes ont proposé de ne plus traiter un objet comme une globalité. Crivellaro *et al.* ont, par exemple, décomposé un objet en parties [Cri+15 ; Cri+17]. Il estiment ensuite la pose de chaque partie

grâce à la prédiction des projections 2D de points de contrôles. Dans Pix2Pose [PPV19] et DPOD [ZSI19], les auteurs ont proposé une prédiction dense de coordonnées de son modèle 3D pour chaque pixel de l'objet (voir Figure 1.6). Sa pose est ensuite calculée avec l'algorithme EPnP [LMF09]. Une nouvelle fonction de coût permettant de traiter les vues ambiguës des objets symétriques a également été développée dans Pix2Pose [PPV19].

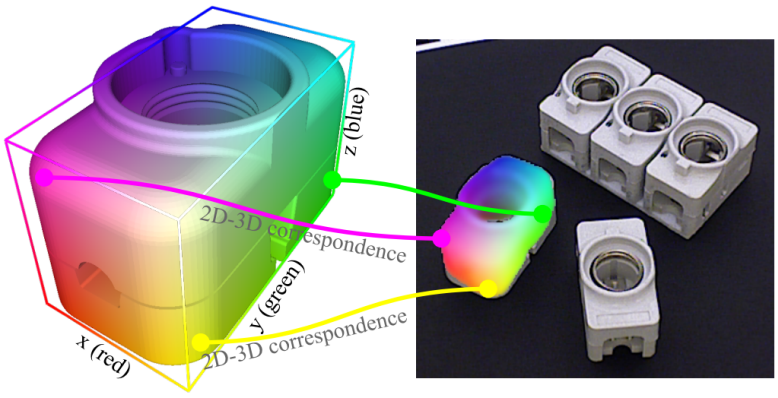


FIGURE 1.6 – Prédiction de coordonnées 3D dense, pour chaque pixel de l'objet [PPV19].

Hu *et al.* ont, quant à eux, proposé une segmentation de l'objet où chaque partie visible contribue à la prédiction des projections des points clés [HHFS19]. Dans PVNet [Pen+19], Peng *et al.* ont également repris cette idée de prédiction dense pour améliorer la robustesse à des occultations et ont proposé de prédire, pour chaque pixel, une direction indiquant la position d'un point clé projeté. L'ensemble de ces vecteurs permet de voter pour les positions des points clés dans l'image, qui sont ensuite utilisés pour estimer la pose.

Pitteri *et al.* ont développé CorNet [PIL19], une méthode d'estimation de pose pour des objets avec des coins saillants qui ne nécessite pas d'entraînement spécifique pour de nouveaux objets. Leur méthode repose sur les coins de ces objets qu'ils détectent et pour lesquels ils prédisent une pose 3D (voir Figure 1.7). Ces poses sont obtenues par la prédiction des projections 2D de 7 points de contrôles. Les coins sont ensuite mis en correspondance avec ceux du modèle CAO de l'objet en utilisant une approche de type RANSAC. Le fait de calculer la pose de chaque coin de l'objet dans l'image leur permet d'estimer sa pose globale à partir de seulement un ou deux coins visibles et ainsi d'être robuste à des occultations.

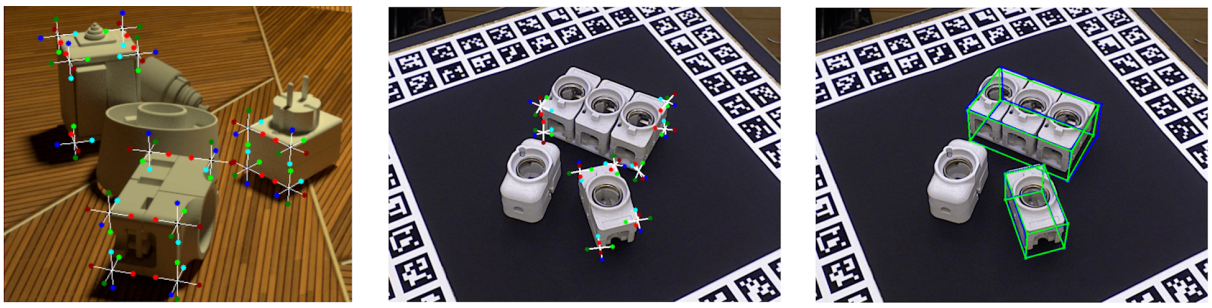


FIGURE 1.7 – Prédiction de 7 points de contrôle pour chaque coin des objets et utilisés pour estimer leur pose [PIL19].

Ces méthodes permettent d'estimer la pose complète (orientation et position) de la caméra à partir d'un seul objet mais nécessitent généralement d'avoir accès au modèle 3D précis de cet objet, parfois texturé, pour calculer la pose ou pour générer des images d'entraînement synthétiques. Cela limite l'utilisation d'objets inconnus et donc leur application dans des environnements qui n'ont pas été modélisés précisément.

Pour remédier à ce besoin, Wang *et al.* ont proposé NOCS [Wan+19b], une approche basée sur des catégories d'objets. Ils utilisent un espace de coordonnées normalisé pour représenter tous les objets d'une même catégorie (voir Figure 1.8). L'utilisation d'une grande base de données d'images synthétiques et réelles leur permet de se généraliser à des objets inconnus d'une catégorie connue. Le calcul de la pose d'un objet nécessite tout de même l'utilisation d'une image de profondeur, qu'ils combinent avec une carte de coordonnées prédites de l'objet.

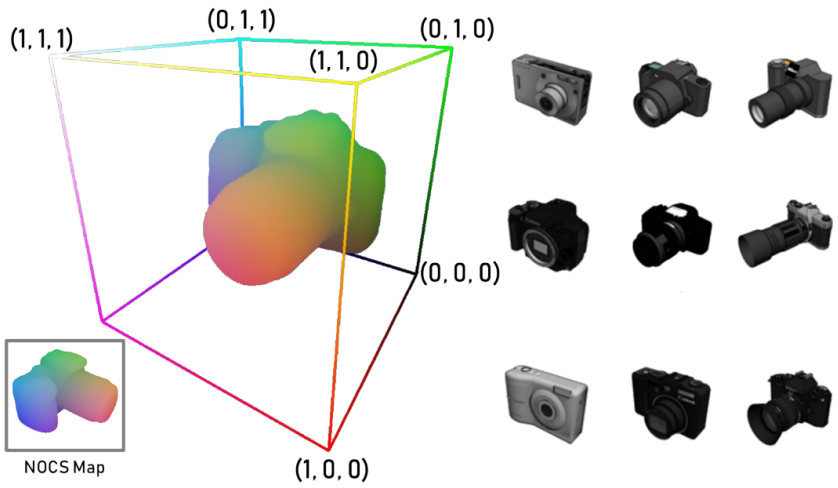


FIGURE 1.8 – Système de coordonnées normalisé (NOCS) utilisé pour représenter tous les objets d'une certaine catégorie [Wan+19b].

Les méthodes présentées ci-dessus traitent chaque objet de manière indépendante, dans leur propre repère. Plus proche de notre objectif, d'autres méthodes créent une carte des objets d'une scène et cherchent à estimer la pose de la caméra par rapport à cette carte.

Weinzaepfel *et al.* ont utilisé des images de références, dont la pose est connue, et ont développé une méthode de localisation visuelle à partir de correspondances denses 2D-3D prédites entre les objets présents dans l'image requête et ceux des images de référence [WCCH19] (voir Figure 1.9). Cette méthode est cependant limitée à des objets plans. Labbé *et al.* ont proposé CosyPose [LCAS20], dans un contexte multi-caméra et multi-objet. Ils génèrent tout d'abord des hypothèses de poses de chaque objet détecté dans les images puis les recalent de manière robuste, entre images, afin d'estimer conjointement les poses des caméras et les poses des objets dans une seule scène cohérente. Ils ont également développé un ajustement de faisceaux basé objet qui permet un raffinement global de la scène.

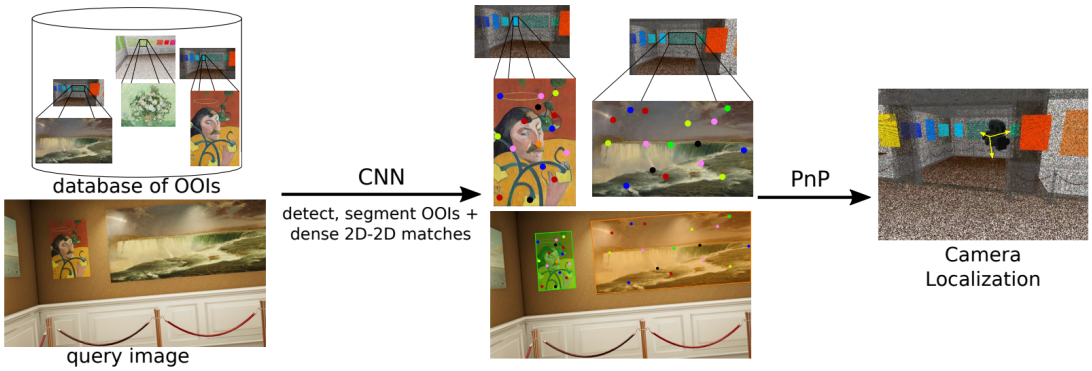


FIGURE 1.9 – Localisation à partir d'objets planaires et de correspondances 2D-2D denses [WCCH19].

Crocco *et al.* ont proposé une solution analytique au problème de *Structure-from-Motion* à partir d'objets *SfMO* [CRB16] (voir Figure 1.10). Ils permettent de reconstruire les objets sous la forme d'ellipsoïdes et d'estimer la pose de la caméra à partir de leurs détections dans les images par des ellipses inscrites dans des boîtes. Leur méthode se limite cependant au cas de projections orthographiques. Gay *et al.* ont étendu cette méthode avec une approche probabiliste nommée *PSfMO* [GRBD17] qui utilise des modèles CAO des objets pour contraindre leurs reconstructions ellipsoïdales. Cette méthode reste également limitée à une caméra affine.

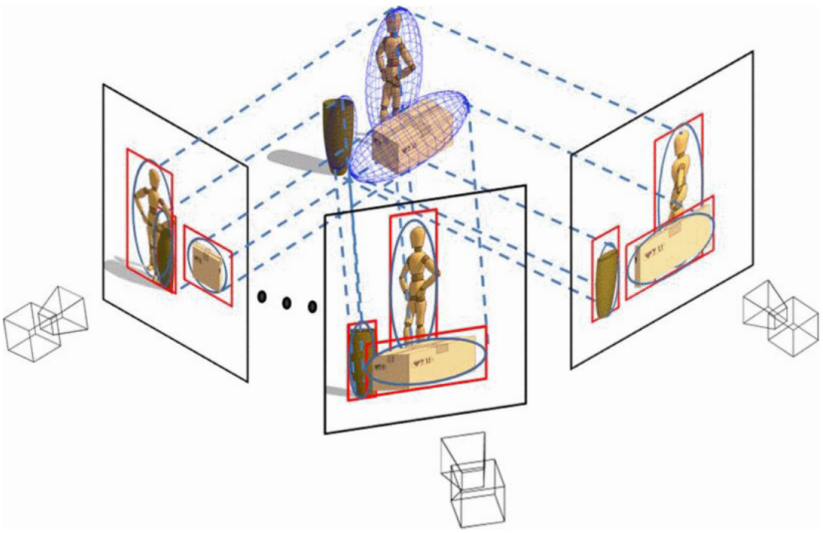


FIGURE 1.10 – *Structure-from-Motion* à partir d'objets (*SfMO*) reconstruits par des modèles ellipsoïdaux [CRB16].

Rubino *et al.* ont étendu la reconstruction des objets sous forme d'ellipsoïdes à la projection perspective [RCB18] (voir Figure 1.11).

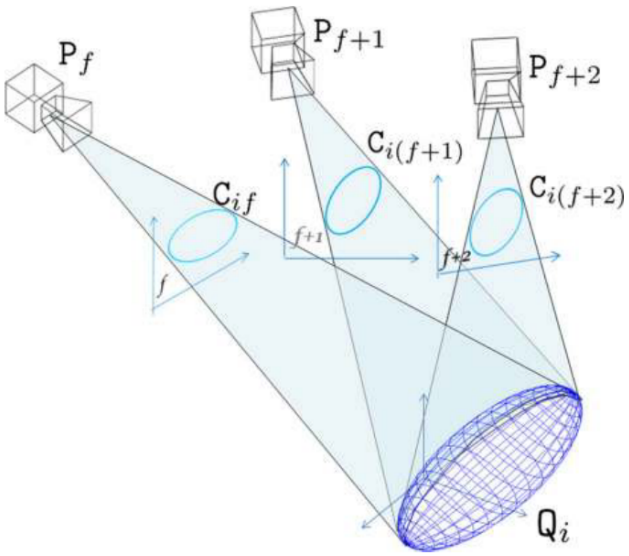


FIGURE 1.11 – Reconstruction d'un ellipsoïde par projection perspective à partir d'observations elliptiques dans trois vues [RCB18].

Des systèmes de cartographie et localisation simultanées (SLAM) ont également intégré l'utilisation d'objets [NMS19 ; Hos+18 ; Wu+20 ; Lia+22 ; YS19]. Par exemple, les objets sont représentés par des cuboïdes dans CubeSLAM [YS19], alors qu'une modélisation sous forme d'ellipsoïdes est utilisée dans QuadricSLAM [NMS19] et SO-SLAM [Lia+22]. Cette modélisation d'objet par un ellipsoïde virtuel présente des caractéristiques avantageuses :

— Une représentation légère et compacte d'un objet nécessitant seulement 9 paramètres.

— La reconstruction d'un ellipsoïde à partir de trois observations d'ellipses a une solution analytique développée dans [RCB18].

— Un ellipsoïde (Q^*) se projette dans l'image sous la forme d'une ellipse (C^*) qui peut s'écrire de manière continue en fonction des paramètres de projection de la caméra : $C^* = PQ^*P^T$. En comparaison, le polygone décrit par la projection d'un cuboïde dépend du point de vue et ne possède pas d'équation continue, ce qui induit une combinatoire importante lors de la mise en correspondance des coins de la boîte 3D avec les arêtes d'une boîte 2D.

— Une représentation compatible avec des méthodes de décomposition d'objet en primitives ellipsoïdales [PUG19].

Enfin, des travaux récents ont proposé une solution analytique au calcul de pose à partir d'hypothèses de correspondances ellipse-ellipsoïde, sans avoir besoin d'une estimation initiale [GSB19a ; GSB19b]. Gaudillière *et al.* ont d'abord montré que la position de la caméra pouvait être obtenue à partir d'une correspondance 2D-3D ellipse-ellipsoïde en connaissant son orientation [GSB19a ; GSB19b]. Ils ont, par la suite, étendu leur méthode au calcul de la pose complète de la caméra dans P2E [GSB20]. Ces deux méthodes sont détaillées dans la section 1.5.2.

1.5 Modélisation des objets par des ellipsoïdes et calcul de pose

1.5.1 Rappels mathématiques : ellipse, ellipsoïde et cône

Une ellipse est une forme spécifique de conique qui s'écrit

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} = 1, \tag{1.1}$$

où α et β correspondent à la taille de ses demi-axes. On la définit généralement par son équation euclidienne :

$$(X - O)^T A (X - O) = 1, \tag{1.2}$$

où $O \in \mathbb{R}^2$ est son centre et $X \in \mathbb{R}^2$ est un point appartenant à son contour. La matrice A est symétrique et définie positive. Elle peut être décomposée en une matrice diagonale caractérisant la taille de l'ellipse et une matrice de rotation R_θ caractérisant son orientation. L'équation de l'ellipse s'écrit alors

$$(X - O)^T R_\theta \begin{bmatrix} 1/\alpha^2 & 0 \\ 0 & 1/\beta^2 \end{bmatrix} R_\theta^T (X - O) = 1, \tag{1.3}$$
$$\text{avec } R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

où α et β sont les tailles de ses demi-axes et θ correspond à l'orientation de l'ellipse. Cette équation peut également s'écrire sous la forme homogène :

$$X^T T_{-O}^T R_\theta \begin{bmatrix} 1/\alpha^2 & 0 & 0 \\ 0 & 1/\beta^2 & 0 \\ 0 & 0 & -1 \end{bmatrix} R_\theta^T T_{-O} X = 0, \tag{1.4}$$
$$\text{avec } T_{-O} = \begin{bmatrix} 1 & 0 & -o_x \\ 0 & 1 & -o_y \\ 0 & 0 & 1 \end{bmatrix},$$

où $O = [o_x, o_y]$ est le centre l'ellipse, $[\alpha, \beta]$ sont les tailles de ses demi-axes et θ est son orientation. $X = [x, y, 1]^T$ est un point de son contour exprimé en coordonnées homogènes. Une ellipse peut donc être totalement définie par l'équation

$$X^T C X = 0, \tag{1.5}$$

où C est une matrice symétrique 3×3 définie par les 6 éléments de sa partie triangulaire supérieure, à une échelle près. Une ellipse dispose donc de 5 degrés de liberté.

Finalement, une ellipse peut également être représentée par sa conique duale [HZ03], définie par la matrice $C^* = adj(C)$. Comme illustré sur la figure 1.12, cette forme duale est définie par un ensemble de droites l tangentes au contour de l'ellipse et qui forment son enveloppe. Ces droites tangentes vérifient

$$l^T C^* l = 0. \tag{1.6}$$

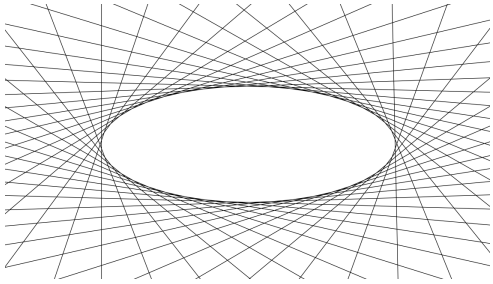


FIGURE 1.12 – Forme duale d'une ellipse définie par un ensemble de droites tangentes à son contour.

De son côté un ellipsoïde est une forme spécifique de quadrique, du type

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} + \frac{z^2}{\gamma^2} = 1, \tag{1.7}$$

où α, β et γ sont les tailles de ses demi-axes. Son équation euclidienne s'écrit

$$(X - O)^T R \begin{bmatrix} 1/a^2 & 0 & 0 \\ 0 & 1/b^2 & 0 \\ 0 & 0 & 1/c^2 \end{bmatrix} R^T (X - O) = 1, \tag{1.8}$$

où $X \in \mathbb{R}^3$ est un point appartenant à sa surface, $O \in \mathbb{R}^3$ est son centre, $[\alpha, \beta, \gamma]$ sont les tailles de ses demi-axes et $R \in SO(3)$ est son orientation. Sa forme homogène s'écrit

$$X^T Q X = 0, \tag{1.9}$$

où $X = [x, y, z, 1]$ est un point appartenant à sa surface, exprimé en coordonnées homogènes, et Q est une matrice symétrique 4×4 définie par les 10 éléments de sa partie triangulaire supérieure, à une échelle près. Un ellipsoïde dispose donc de 9 degrés de liberté. De manière similaire à une ellipse, un ellipsoïde dispose également d'une forme duale, définie par la matrice $Q^* = adj(Q)$. Cette matrice caractérise l'ensemble des plans tangents à la surface de l'ellipsoïde et qui vérifient

$$\pi^T Q^* \pi = 0. \tag{1.10}$$

Avec le modèle de caméra sténopé, un ellipsoïde se projette dans une image sous la forme d'une ellipse. La projection par une matrice de caméra $P = K[R|t]$ relie les matrices des formes duales de l'ellipsoïde et de l'ellipse [HZ03], tel que

$$C^* = P Q^* P^T. \tag{1.11}$$

Enfin, un cône est caractérisé par l'équation

$$(X - E)^T B (X - E) = 0, \tag{1.12}$$

où $X \in \mathbb{R}^3$ est un point du cône, $E \in \mathbb{R}^3$ est son sommet et B est une matrice symétrique 3×3 qui définit les proportions et l'orientation du cône.

1.5.2 Solutions analytiques au calcul de pose par paires ellipse-ellipsoïde

Dans cette section, nous détaillons les méthodes de calcul de pose de caméra basées sur une ou plusieurs paires ellipse-ellipsoïde, développées par Gaudillière *et al.* et qui constituent l'une des bases à notre travail.

P1E [GSB19a ; GSB19b] : Dans un premier temps, les auteurs ont développé une solution analytique au calcul de la position de la caméra à partir d'une paire ellipse-ellipsoïde en connaissant son orientation. Ils se placent dans le cas illustré sur la figure 1.13, avec un ellipsoïde défini par une matrice A et de centre C, un centre de projection E et un plan de projection de normale N dans lequel l'ellipsoïde se projette sous la forme d'un ellipse de centre K et de demi-axes a et b. Les directions principales de l'ellipse sont représentées par des vecteurs unitaires U et V et forment, avec N, une base orthonormée $\{U, V, N\}$.

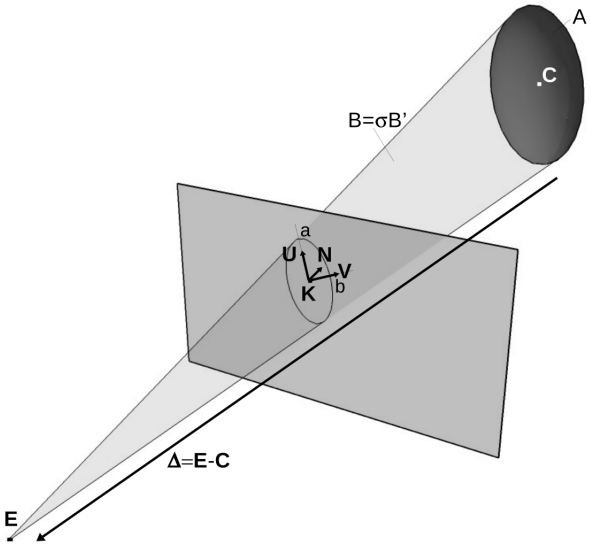


FIGURE 1.13 – Illustration de P1E pour le calcul de la position de la caméra à partir d'une paire ellipse-ellipsoïde.

Ils montrent que les points X du cône de projection de sommet E et tangent à l'ellipsoïde vérifient

$$(X - E)^T B(X - E) = 0, \tag{1.13}$$

où

$$B = A\Delta\Delta^T A - (\Delta^T A\Delta - 1)A, \tag{1.14}$$

et $\Delta = E - C$. Ils montrent également que les points X du cône de rétroprojection issu du point E et passant par le contour de l'ellipse dans le plan image vérifient

$$(X - E)^T B'(X - E) = 0, \tag{1.15}$$

où

$$B' = P^T M P - Q, \tag{1.16}$$

avec

$$\begin{cases} M = & UU^T/a^2 + VV^T/b^2 \\ W = & N/(N \cdot (K - E)) \\ P = & I - (K - E)W^T \\ Q = & WW^T \end{cases} \tag{1.17}$$

Les cônes de projection et de rétroprojection coïncident s'il existe une valeur non nulle σ, telle que $B = \sigma B'$, c'est-à-dire

$$A\Delta\Delta^T A - (\Delta^T A\Delta - 1)A = \sigma B'. \tag{1.18}$$

Les auteurs ont montré que cette égalité peut être simplifiée en

$$A\Delta = \sigma B'\Delta. \tag{1.19}$$

Le vecteur recherché Δ est donc un vecteur propre généralisé du couple $\{A, B'\}$. Ce couple a deux valeurs propres généralisées réelles et distinctes, l'une de multiplicité 1, notée σ_1 et l'autre de multiplicité 2, notée σ_2. Les auteurs ont montré que σ_1 est la solution de l'équation 1.19. Finalement, Δ est estimé en utilisant le fait qu'il soit proportionnel au vecteur propre Δ_1 de norme unitaire associée à σ_1. Ils remplacent Δ par $k\Delta_1$ dans l'équation 1.18 et estiment k tel que

$$k^2(A\Delta_1\Delta_1^T A - (\Delta_1^T A\Delta_1 A) = \sigma_1 B' - A. \tag{1.20}$$

Seule la valeur de k qui place le centre de l'ellipsoïde devant la caméra est valide et est retenue.

P2E [GSB20] : Dans un second temps, ils étendent leur méthode à l'estimation de la pose complète de la caméra (orientation et position) en utilisant deux paires ellipse-ellipsoïde. La figure 1.14 illustre ce cas. L'orientation de la caméra est recherchée entre les bases orthonormées *caméra* $\{i_{cam}, j_{cam}, k_{cam}\}$ et *monde* $\{i_w, j_w, k_w\}$. Leur méthode repose sur les deux hypothèses suivantes :

1. L'angle de roulis de la caméra est nul.
2. La droite reliant les centres des ellipsoïdes se projette sur la droite reliant les centres des ellipses projetées.

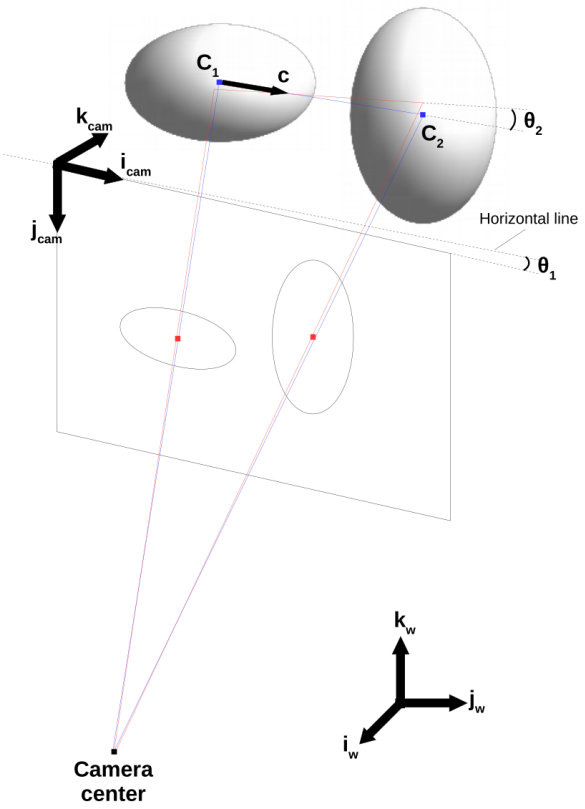


FIGURE 1.14 – Illustration de P2E pour l'estimation de la pose de la caméra à partir de deux paires ellipse-ellipsoïde.

Pour calculer l'orientation de la caméra, ils distinguent les cas où le vecteur c, reliant les centres des deux ellipsoïdes, et le vecteur i_{cam}, correspondant à l'axe x de la caméra, sont colinéaires ou non. Ils ont montré que, dans les deux cas, l'orientation de la caméra dispose d'un seul degré de liberté et peut s'exprimer comme une fonction d'un angle α. Ils utilisent une recherche exhaustive de ce paramètre en discrétisant l'intervalle $[0°, 360°]$ de manière uniforme. Pour chaque orientation possible, la position de la caméra est calculée avec P1E et un coût de reprojection basé sur la distance de Jaccard est calculé entre les ellipses de détection et les ellipsoïdes projetés avec la pose estimée de la caméra. La meilleure pose est choisie comme étant celle avec le coût de reprojection le plus bas.

En pratique, il y a souvent plus de deux objets détectés dans une image et les correspondances entre ellipses et ellipsoïdes ne sont pas connues. Les auteurs ont proposé une procédure de type RANSAC, sous la forme d'une boucle permettant de tester les différentes combinaisons de deux paires ellipse-ellipsoïde. Les correspondances sont contraintes par les classes des objets.

Chapitre 2

Prédiction d'ellipse guidée par la 3D pour améliorer le calcul de pose

Sommaire

Dans ce chapitre, nous nous intéressons au calcul de pose de caméra basé objet et cherchons à proposer une méthode nécessitant peu de données d'entraînement. Nous nous attachons tout particulièrement à proposer un système complet pouvant être utilisé dans des environnements inconnus et non modélisés, pour lesquels les modèles 3D précis des objets présents ne sont pas disponibles. Nous débutons ce chapitre en décrivant les limites des méthodes existantes, notamment au niveau de l'observation des objets dans les images sous formes de boîtes englobantes, alignées avec les axes de l'image dans la section 2.1. Nous proposons ensuite un réseau de neurones permettant d'obtenir des détections d'objets sous forme d'ellipses plus cohérentes avec les modèles ellipsoïdaux 3D de ces objets dans la section 2.2. Nous y étudions également deux fonctions de coût utilisées pour entraîner ce réseau. Le système complet, avec l'annotation automatique des données, l'entraînement du réseau et le calcul de la pose de la caméra, est décrit dans la section 2.3. Enfin, dans la section 2.4, nous comparons notre méthode avec celles existantes, nous illustrons les gains de précision apportés par la détection améliorée des objets et nous démontrons sa robustesse. Les travaux décrits dans ce chapitre ont fait l'objet des publications [ZSB20] et [ZSB22c].

2.1 Limites des méthodes existantes

Dans le contexte de la localisation visuelle, l'observation d'indices 2D dans l'image joue un rôle important. Les points d'intérêt utilisés traditionnellement consistent simplement en une coordonnée et peuvent être localisés de manière précise dans l'image, par exemple, au niveau de zones de fort gradient formant des coins. Au contraire, les objets que nous utilisons comme balises pour le calcul de pose apparaissent comme des formes 2D dans l'image. En particulier, avec la modélisation choisie, ils sont observés sous la forme d'ellipses et comprennent donc à la fois une position, une taille et une orientation. La difficulté liée à l'observation d'un objet vient de la plus grande variabilité spatiale de sa détection.

2.1.1 Observation d'objet alignée avec les axes de l'image

Dans les travaux existants, notamment ceux de Gaudillière *et al.* [GSB19b ; GSB19a ; GSB20] et Nicholson *et al.* [NMS19] (QuadricSLAM), les observations des objets se basent sur des boîtes 2D alignées avec les axes de l'image. Ces boîtes correspondent directement à la sortie des détecteurs d'objets classiques tels que YOLO [RDGF16 ; BWL20] ou Faster R-CNN [RHGS17]. Dans le cas de QuadricSLAM, ces boîtes sont directement utilisées, alors que Gaudillière *et al.* considèrent les ellipses inscrites dans ces boîtes et qui sont donc également alignées aux axes de l'image. Ces observations d'objets dépendent de l'orientation de l'image et donc de la caméra. Par exemple, l'observation d'un objet parfaitement centré dans l'image change lorsque l'image subit une rotation. Les méthodes existantes de calcul de pose de caméra basées ellipse-ellipsoïde visent à aligner le cône de projection de l'ellipsoïde avec le cône de rétroprojection formé par l'observation de l'objet dans l'image. Elles sont donc sensibles à des observations imprécises des objets.

Les figures 2.1 et 2.2 illustrent ce problème et la dégradation de la précision de la pose estimée pour la caméra. Nous nous plaçons dans un scénario synthétique, sans bruit, avec une paire ellipse-ellipsoïde. L'orientation de la caméra est connue et sa position est estimée en utilisant P1E [GSB19a]. La caméra est placée en face de l'ellipsoïde à une distance de 8 m et subit une rotation autour de son axe principal. Les courbes de la figure 2.3 montrent la position estimée de la caméra (sa coordonnée Z) en utilisant l'ellipse inscrite dans la boîte de détection alignée

aux axes de l'image. La position estimée en utilisant comme observation l'ellipse correspondant effectivement à la projection de l'ellipsoïde dans l'image est montrée en référence. La dégradation de la précision du calcul de pose dépend également de la forme de l'objet (allongé ou non) et de son orientation dans l'image.

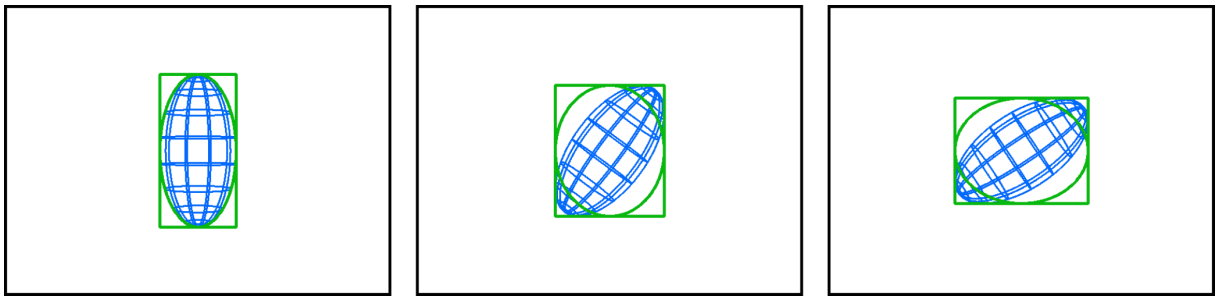


FIGURE 2.1 – Illustration de l'imprécision liée à l'utilisation de l'ellipse inscrite dans la boîte de détection d'un objet (en vert), alignée aux axes de l'image, pour différentes rotations de la caméra autour de son axe principal. À gauche, l'alignement entre la détection et la projection est précis. Au milieu et à droite, les alignements sont mauvais car l'objet apparaît incliné dans l'image.

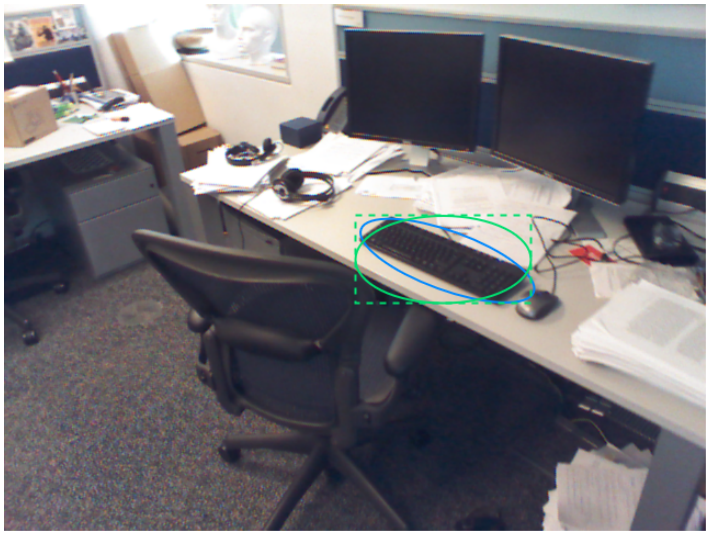


FIGURE 2.2 – Illustration, dans un cas réel, de l'imprécision de l'ellipse inscrite dans une boîte de détection (en vert), alignée aux axes de l'image, dans un cas réel. L'ellipse bleue correspond à la projection du modèle ellipsoïdal de l'objet.

2.1.2 Observation ajustée au contour de l'objet

Les réseaux de détection d'objets sont entraînés à prédire la boîte d'aire minimale englobant l'objet, c'est-à-dire que leurs côtés sont tangents au contour de l'objet. On pourrait imaginer étendre cela aux ellipses en considérant l'ellipse d'aire minimale englobant l'objet. Cependant les modèles ellipsoïdaux des objets que nous utilisons sont virtuels et n'ont pas vocation à être parfaitement ajustés à l'objet en 3D. Au contraire, l'objectif de nos travaux est de pouvoir se

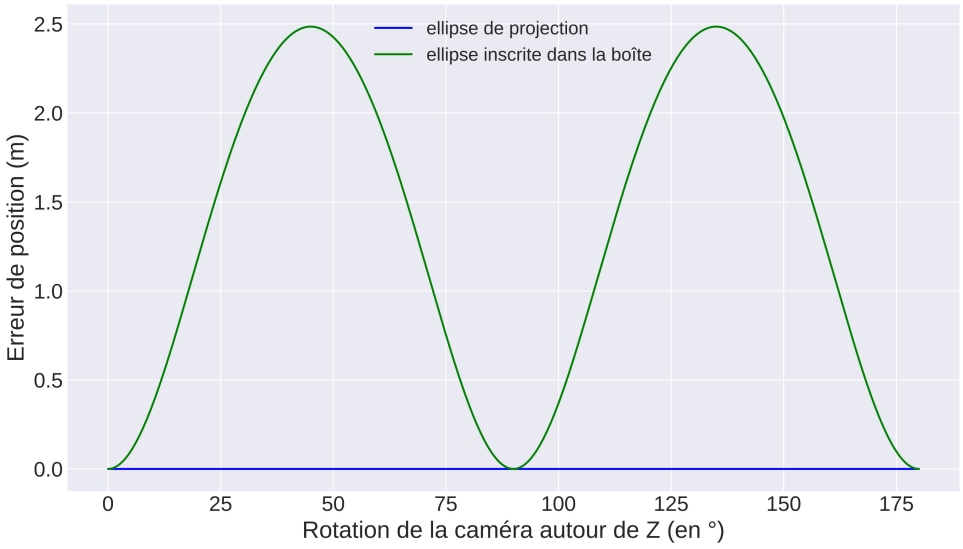


FIGURE 2.3 – Erreur commise sur la position estimée de la caméra avec P1E en utilisant l'ellipse inscrite dans la boîte de détection de l'objet (en vert), alignée aux axes de l'image, pour différentes rotations de la caméra autour de son axe principal. La courbe bleue montre que la position peut être estimée, sans erreur, en utilisant l'ellipse correspondant à la projection de l'ellipsoïde de l'objet.

contenter de modèles d'objets assez grossiers, pouvant être obtenus facilement par reconstruction à partir de quelques vues, en utilisant la méthode développée par Rubino *et al.* [RCB18] par exemple. Il est également nécessaire de pouvoir représenter n'importe quel objet avec un ellipsoïde, y compris des objets n'ayant pas une forme ellipsoïdale.

Ces modèles ellipsoïdaux approximatifs des objets, combinés avec les déformations dues à la perspective produisent des ellipses projetées assez éloignées d'une ellipse ajustée aux contours d'un objet dans l'image. La figure 2.4 illustre ce problème. En bas de l'image, trois détections « parfaites » (en terme d'ajustement au contour de l'objet) dans différentes vues de l'objet sont utilisées pour reconstruire un modèle ellipsoïdal de celui-ci. La ligne du haut montre l'objet à partir de trois autres points de vue dans lesquels on peut clairement voir la différence entre la projection de l'ellipsoïde et les détections « parfaites » de l'objet. Ces détections, bien que parfaitement ajustées au contour de l'objet, ne sont pas cohérentes avec la projection du modèle ellipsoïdal de l'objet. Au niveau du calcul de pose de la caméra, cela engendre une incohérence entre le cône de projection et celui de rétroprojection et entraîne donc une imprécision.

2.1.3 Importance de la supervision 3D

Pour surmonter les limites présentées dans les deux paragraphes précédents, nous proposons donc d'utiliser de l'information 3D au niveau de la détection des objets. Plus précisément, nous voulons être capable de prédire une ellipse qui corresponde à la projection du modèle abstrait de l'objet. Cette supervision 3D a deux principaux avantages :

1. Cela permet d'obtenir des observations d'objet plus cohérentes avec leur modèle ellipsoïdal 3D (des cônes de projection et rétroprojection cohérents) et donc d'améliorer la précision du calcul de pose de caméra basé objet.

2. Cela permet de dissocier le modèle ellipsoïdal de la forme de l'objet. L'ellipsoïde devient donc totalement abstrait et peut être utilisé pour représenter n'importe quel objet. De plus, nous montrons par la suite qu'il n'est pas nécessaire qu'il soit ajusté de manière très précise à l'objet, ce qui est, de toute façon, compliqué à obtenir en pratique à partir de quelques vues de l'objet.

Nous proposons ainsi d'entraîner un réseau de neurones pour prédire la projection de ce modèle ellipsoïdal d'un objet.

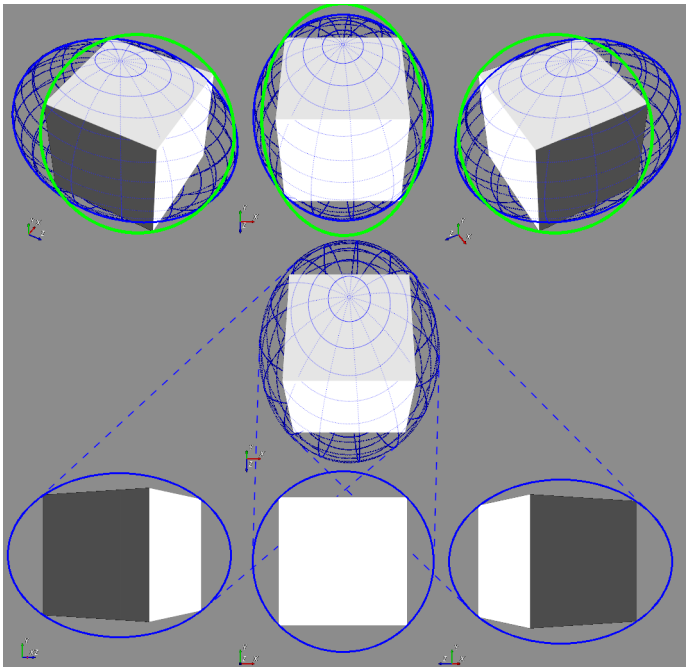


FIGURE 2.4 – Illustration de l'importance de la supervision 3D. Trois détections elliptiques « parfaites » de l'objet (ligne du bas) sont utilisées pour reconstruire un modèle ellipsoïdal de l'objet (au milieu). La différence entre les projections de cet ellipsoïde et des détections « parfaites » dans trois nouvelles vues est illustrée sur la ligne du haut.

2.2 Réseau de prédiction d'ellipse orientée guidée par la 3D

Dans cette section, nous nous intéressons à la prédiction d'une ellipse de détection d'un objet, guidée par la 3D, à partir d'une sous-image. Cette amélioration de la détection d'un objet sous forme d'une ellipse orientée constitue une étape importante de notre travail. Son intégration à notre système complet de localisation visuelle basée objet sera décrit plus tard dans la section 2.3.

2.2.1 État de l'art

Dong *et al.* ont introduit Ellipse R-CNN [DRPI21], un réseau de régression d'ellipse basé sur l'architecture de Mask R-CNN [HGDG17], pour la détection d'objets de forme elliptique. Ils paramétrisent une ellipse par son centre, ses axes et son orientation. Leur réseau comprend également un module de structure U-Net [RFB15] pour la gestion des occultations. Une autre méthode de prédiction d'ellipse a été proposée par Pan *et al.* pour la détection de nœuds de bois [Pan+21]. Ils ont adapté le réseau Faster R-CNN [RHGS17] en interprétant une ellipse

comme une distribution gaussienne 2D et ont utilisé la distance de Wasserstein pour entraîner leur réseau. Par rapport à notre objectif de prédiction d'ellipse cohérente avec la projection du modèle ellipsoïdal d'un objet, ces deux travaux se focalisent sur la détection 2D d'objets de forme elliptique, comme par exemple, des fruits dans des arbres ou des nœuds de bois.

2.2.2 Paramétrisation d'ellipse

Une ellipse peut être représentée de différentes manières :

— par les cinq paramètres de la matrice symétrique représentant sa forme quadratique.

— par trois points correspondants au centre et aux extrémités des deux demi-axes.

— par son centre, la taille de ses deux axes et son orientation.

Les paramètres de la forme quadratique d'une ellipse fournissent une représentation minimale car une ellipse dispose de cinq degrés de liberté. L'inconvénient de cette représentation vient de son manque d'interprétabilité au sens géométrique. Au contraire, utiliser trois points (centres et extrémités des demi-axes) pour représenter une ellipse a un réel sens physique. Cependant, cette représentation n'est pas de forme minimale car elle nécessite six paramètres. De plus, il est nécessaire d'ajouter une contrainte d'orthogonalité entre les vecteurs formés par le centre et les extrémités des demi-axes, ce qui n'est pas toujours possible lors d'une prédiction.

Nous avons donc opté pour la troisième solution, c'est-à-dire sous la forme d'un centre, de tailles de demi-axes et d'une orientation. Cette représentation a un sens physique, est minimale (cinq paramètres), mais est néanmoins sujette à l'ambiguïté due à la symétrie d'une ellipse. Pour obtenir une forme canonique, nous sélectionnons le premier axe comme étant le plus grand et son extrémité est choisie dans la moitié de droite de l'ellipse. Le second axe est le plus petit avec une extrémité dans la moitié basse de l'ellipse. L'orientation de l'ellipse est mesurée par l'angle entre le grand axe et l'horizontale. Cette paramétrisation d'ellipse est illustrée dans la figure 2.5.

La difficulté de cette représentation vient du fait qu'elle implique une discontinuité au niveau du paramètre angulaire. En effet, à cause de la symétrie de l'ellipse, les valeurs possibles d'orientation varient entre -90° et +90°, ce qui implique que deux ellipses proches de la verticale peuvent avoir des valeurs d'angle totalement opposées. La figure 2.6 illustre ce problème où deux images très proches d'un objet, simplement tournées de quelques degrés, donnent lieu à des ellipses ayant des valeurs d'orientation très différentes.

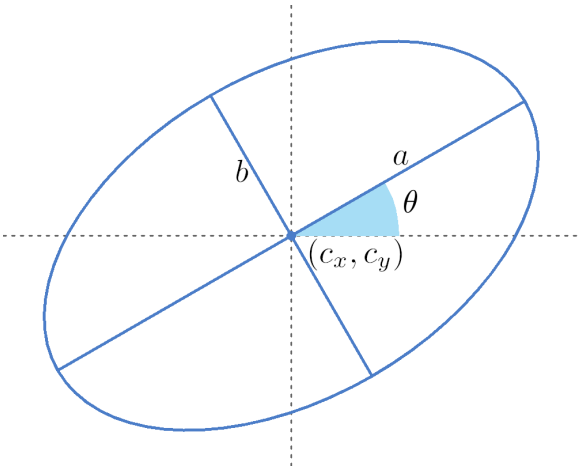


FIGURE 2.5 – Paramétrisation choisie d'une ellipse avec un centre (c_x, c_y), une orientation θ et une taille des axes (a, b).

FIGURE 2.6 – Illustration la discontinuité du paramètre angulaire de l'ellipse. Les deux images sont très similaires. L'objet est légèrement tourné mais les valeurs d'orientation des ellipses correspondantes sont très différentes, $+88°$ à gauche et $-86°$ à droite.

2.2.3 Fonction de coût *Multi-Bin*

Avec la paramétrisation d'ellipse choisie, notre réseau de neurones doit donc être capable de prédire un centre, deux distances et un angle. Une fonction de coût combinant ces paramètres est donc nécessaire et est minimisée au cours de l'apprentissage. Nous avons, dans un premier temps, tenté d'utiliser une fonction de coût classique mesurant la distance euclidienne entre les paramètres prédits $(c_{pred}, d_{pref}, \theta_{pred})$ et les valeurs de vérité terrain $(c_{gt}, d_{gt}, \theta_{gt})$, telle que

$$\mathcal{L} = \left(\|d_{gt} - d_{pred}\|_2^2 + \alpha \|c_{gt} - c_{pred}\|_2^2 + \beta \|\theta_{gt} - \theta_{pred}\|_2^2 \right). \tag{2.1}$$

Cette formulation combine des distances entre différentes grandeurs et nécessite donc des termes de pondération α et β. Nous avons cependant observé de grosses chutes de performance lorsque l'ellipse à prédire est proche de la verticale. Ce problème est causé par la discontinuité du paramètre angulaire de l'ellipse décrit dans le paragraphe précédent.

Afin limiter cet effet, nous avons proposé une approche nommée *Multi-Bin*, inspirée des travaux de Mousavian *et al.* [MAFK17]. Avec cette méthode, le problème de prédiction de l'orientation de l'ellipse est divisé en une classification puis une régression. L'espace angulaire $[\frac{-\pi}{2}, \frac{\pi}{2}]$ est divisé de manière régulière en n boîtes. Celles-ci se chevauchent partiellement afin d'éviter des effets aux bords. Dans un premier temps, notre réseau classifie la boîte correspondant à l'angle de vérité terrain puis régresse un petit angle de correction appliqué au centre de la boîte. La figure 2.7 illustre ce procédé, dans lequel la boîte correspondant à σ_2 est d'abord choisie puis un angle de correction Δ_{θ_2} est appliqué pour obtenir l'angle θ. Avec cette approche l'angle regressé reste petit, compris entre $\frac{\texttt{-taille bin}}{2}$ et $\frac{\texttt{taille bin}}{2}$.

Nous utilisons l'entropie croisée comme fonction de coût pour la classification, définie par

$$\mathcal{L}_{boite} = -\sum_{i}^{n} o_i \log \left(\frac{e^{s_i}}{\sum_j^n e^{s_j}} \right), \tag{2.2}$$

où s_i représente le score de sortie de la i$^{\text{ème}}$ boîte, n est le nombre de boîtes qui divisent l'espace angulaire et o_i indique si la i$^{\text{ème}}$ contient l'angle de vérité terrain (0 ou 1).

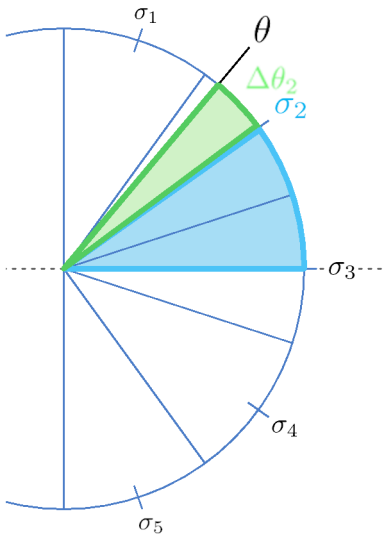


FIGURE 2.7 – Découpage de l'espace angulaire $[-\frac{\pi}{2}, \frac{\pi}{2}]$ en boîtes. Les σ_i sont les angles moyens des boîtes, Δ_{θ_2} est l'angle de correction ajouté à l'angle moyen de la 2$^{\text{éme}}$ boîte pour obtenir la valeur finale de l'angle prédit θ. Un chevauchement nul des boîtes est illustré ici. En pratique, nous avons utilisé un recouvrement de 4°.

La fonction de coût utilisée pour l'angle de correction est définie par

$$\mathcal{L}_{correction} = -\frac{1}{n_{\theta^*}} \sum_{i}^{n} o_i \cdot cos(\theta^* - \sigma_i - \Delta\theta_i), \tag{2.3}$$

où θ^* est la valeur de vérité terrain de l'angle, n_{θ^*} est le nombre de boîtes qui contiennent l'angle de vérité terrain (1 ou 2) et o_i indique si la i$^{\text{ème}}$ boîte contient l'angle de vérité terrain (0 ou 1). σ_i est l'angle central de la i$^{\text{ème}}$ boîte et Δ_{θ_i} correspond à l'angle de correction prédit et qui est appliqué à l'angle central de cette boîte. Cette fonction de coût n'a d'effet que sur l'angle de correction correspondant à la boîte contenant l'angle vérité terrain (ou éventuellement aux deux boîtes le contenant en cas de chevauchement).

Pour le centre et les dimensions de l'ellipse, nous continuons d'utiliser la distance euclidienne entre les valeurs prédites et les valeurs de vérité terrain, telle que

$$\mathcal{L}_{centre} = \|c_{gt} - c_{pred}\|_2^2 \tag{2.4}$$

$$\mathcal{L}_{dimensions} = \|d_{gt} - d_{pred}\|_2^2 \tag{2.5}$$

où c_{gt} et c_{pred} sont respectivement les centres de vérité terrain et prédits, d_{gt} et d_{pred} sont les dimensions de vérité terrain et prédites. Finalement, la fonction de coût totale de notre réseau est une combinaison pondérée des coûts individuels et devient

$$\mathcal{L} = \alpha(\mathcal{L}_{centre} + \mathcal{L}_{dimensions}) + (\mathcal{L}_{boites} + \beta\mathcal{L}_{correction}). \tag{2.6}$$

Les paramètres α et β permettent d'équilibrer les valeurs des coûts entre différentes quantités. Ils ont été déterminés empiriquement et nous les avons fixés respectivement à 0.01 et 1.0.

2.2.4 Première architecture du réseau

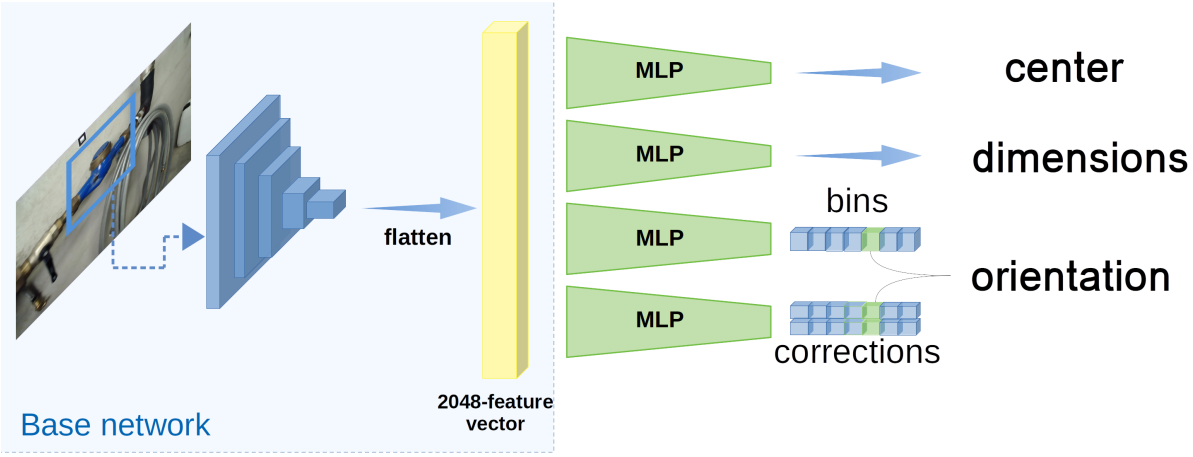


FIGURE 2.8 – Première architecture du réseau de prédiction d'ellipse, composée d'un réseau de base (VGG-19) qui permet d'extraire un vecteur de caractéristiques de taille 2048 et quatre branches parallèles permettant de régresser le centre, les dimensions et l'orientation de l'ellipse.

L'architecture de la première version de notre réseau, utilisant la fonction de coût *Multi-Bin* décrite précédemment, est détaillée sur la figure 2.8. Celui-ci prend en entrée une sous-image rognée autour de l'objet, qui est passée dans un réseau convolutionnel chargé d'extraire des caractéristiques. Nous avons utilisé le réseau VGG-19 [SZ15] pré-entraîné pour une tâche de classification sur la base de données ImageNet [Den+09]. Seule la dernière partie du réseau, constituée d'un *Softmax*, a été retirée. Les caractéristiques extraites sont aplaties en un vecteur de taille 2048 qui est passé en entrée de 4 branches parallèles. Ces branches sont composées de trois couches entièrement connectées avec des couches d'activation non linéaires de type *ReLU*. Les deux premières branches produisent chacune deux valeurs de sorties correspondantes respectivement aux coordonnées du centre de l'ellipse et aux tailles de ses demi-axes. Les deux branches restantes sont utilisées pour prédire l'orientation de l'ellipse. Une première branche nommée *bins* produit n valeurs, qui sont passées dans une couche *Softmax* et transformées en probabilités que l'angle prédit appartienne à une certaine boîte. n correspond au nombre de boîtes qui divisent l'espace angulaire. Nous avons déterminé expérimentalement que l'utilisation de 7 boîtes avec un chevauchement de 4° donne les meilleurs résultats. Enfin, la branche *corrections*, prédit $2n$ valeurs correspondant respectivement au cosinus et sinus de l'angle de correction. Pour garantir des valeurs de cosinus et sinus valides, une étape de normalisation a été ajoutée.

2.2.5 Fonction de coût basée sur des ensembles de niveaux

L'approche *Multi-Bin*, présentée précédemment, nous a permis de limiter les effets de la discontinuité du paramètre angulaire de l'ellipse. Cependant, nous observons toujours quelques difficultés pour la prédiction d'ellipses proches de la verticale. De plus, cette fonction de coût combine différentes grandeurs (une régression de distance, une classification et une régression d'angle) qui ne sont pas directement comparables et doivent donc être équilibrées par des termes de pondération dans son expression finale.

Afin de résoudre ce problème, nous avons proposé une formulation différente de la fonction de coût, basée sur une fonction de plongement. Nous représentons alors une ellipse par une fonction

de plongement 2D $\Phi : \Omega \subset \mathcal{R}^2 \rightarrow \mathcal{R}$. La distance entre l'ellipse de vérité terrain et l'ellipse prédite est alors définie entre leurs fonctions de plongement respectives Φ_{gt} et Φ_{pref} par

$$\mathcal{D}^2(\mathcal{E}_{pred}, \mathcal{E}_{gt}) = \int_{\Omega} (\Phi_{pred}(\mathbf{x}) - \Phi_{gt}(\mathbf{x}))^2 \mathbf{dx}. \tag{2.7}$$

En pratique, cette distance est calculée de manière discrète à des positions $\mathbf{x_i}$ échantillonnées régulièrement sur une grille 2D couvrant toute l'image d'entrée. Nous approximons cette distance sur une grille carrée de taille $N = 25 \times 25$, telle que

$$\mathcal{D}^2(\mathcal{E}_{pred}, \mathcal{E}_{gt}) \approx \sum_{i=1}^{N} (\Phi_{pred}(\mathbf{x_i}) - \Phi_{gt}(\mathbf{x_i}))^2. \tag{2.8}$$

Choix de la fonction de plongement Cette nouvelle formulation du calcul de distance entre ellipses se base donc sur une représentation par fonction de plongement 2D. Celle-ci n'est pas unique pour un contour donné et différentes formulations peuvent être utilisées. Dans le contexte de correspondance de formes, une fonction de plongement classique est la distance signée au contour [RBCS06], définie par

$$\Phi(\mathbf{x}) = \begin{cases} \mathbf{d}(\mathbf{x}, C) & \text{si } \mathbf{x} \text{ à l'intérieur } C \\ -\mathbf{d}(\mathbf{x}, C) & \text{si } \mathbf{x} \text{ à l'extérieur } C \\ 0 & \text{if } \mathbf{x} \in C \end{cases}, \tag{2.9}$$

où C est le contour de la forme et $\mathbf{d}$ est la distance euclidienne minimale entre un point et un contour. Dans notre cas d'une ellipse, le calcul de la distance minimale au contour n'est pas évidente et des fonctions plus simples et plus efficaces en temps de calcul peuvent être utilisées. La fonction la plus naturelle à utiliser est simplement l'équation quadratique de l'ellipse, donnée par

$$\Phi(\mathbf{x}) = (\mathbf{x} - c)^T R(\theta) \begin{bmatrix} \frac{1}{\alpha^2} & 0 \\ 0 & \frac{1}{\beta^2} \end{bmatrix} R(\theta)^T (\mathbf{x} - c), \tag{2.10}$$

où c est le centre de l'ellipse, θ son orientation et $[\alpha, \beta]$ sont les tailles de ses demi-axes. Cette fonction définit une carte de distances non-isotropes à partir du centre de l'ellipse et qui représente des ensembles de niveaux. En particulier, la courbe de niveau 1 correspond au contour de l'ellipse.

Nous avons cependant observé des instabilités numériques au cours de l'entraînement du réseau lorsque cette fonction 2D est utilisée pour représenter les ellipses. De très grandes valeurs de gradient ainsi que de fortes irrégularités dans les valeurs de coût sont visibles sur la première ligne de la figure 2.9. Celles-ci peuvent être expliquées par les expressions sur la diagonale de la matrice centrale dans l'équation 2.10 $[\frac{1}{\alpha^2}, \frac{1}{\beta^2}]$ et par leur dérivées respectives $[\frac{-2}{\alpha^3}, \frac{-2}{\beta^3}]$ qui peuvent prendre de très grandes valeurs lorsque α et β sont petits.

Afin d'éviter ce problème d'instabilité numérique, et puisque le choix de la fonction de plongement est libre, nous avons testé trois autres formulations légèrement différentes en modifiant les éléments de la diagonale de la matrice centrale dans l'équation 2.10 :

$$\begin{bmatrix} \frac{1}{\alpha} & 0 \\ 0 & \frac{1}{\beta} \end{bmatrix}, \begin{bmatrix} \alpha^2 & 0 \\ 0 & \beta^2 \end{bmatrix} \text{ et } \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}. \tag{2.11}$$

Leur évaluation, en terme d'IoU entre les ellipses prédites et celles de vérité terrain, est disponible dans l'expérience de la section 2.4.3 et montre que la forme avec $[\alpha, \beta]$ sur la diagonale donne les meilleurs résultats. Cette formulation permet d'éviter les problèmes d'instabilité numérique

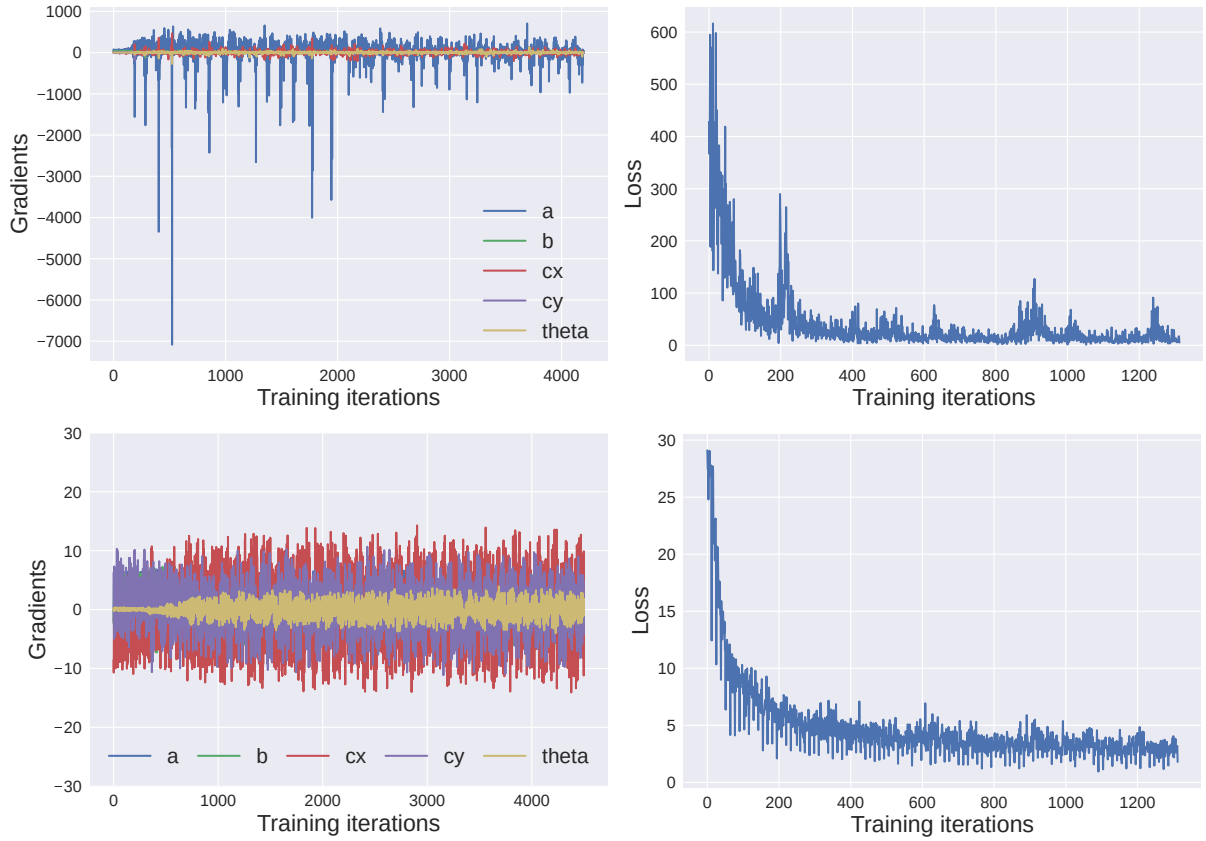


FIGURE 2.9 – Instabilité numérique du réseau pendant l'apprentissage. La colonne de gauche montre l'évolution des gradients, calculés par rapport aux valeurs de sortie du réseau, au cours de l'entraînement. La colonne de droite montre l'évolution du coût au cours de l'entraînement. La ligne du haut correspond à l'utilisation de la fonction de plongement décrite par l'équation 2.10 dans la fonction de coût et la ligne du bas correspond à l'utilisation de la version modifiée de cette fonction, décrite dans l'équation 2.12. Il faut noter la différence d'échelle entre les deux lignes.

(voir la deuxième ligne de la figure 2.9). On peut effectivement remarquer qu'en utilisant cette deuxième formulation les très grosses valeurs de gradient disparaissent et le coût décroît de manière plus lisse. Nous avons donc simplifié l'expression initiale par la suivante, qui résout les problèmes d'instabilité numérique au cours de l'entraînement et donne de meilleurs résultats :

$$\Phi(\mathbf{x}) = (\mathbf{x} - c)^T R(\theta) \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} R(\theta)^T (\mathbf{x} - c). \tag{2.12}$$

Finalement, en comparaison de la fonction de coût *Multi-Bin*, développée dans un premier temps, cette nouvelle formulation a plusieurs avantages :

— Il n'y a plus de problème de discontinuité de l'angle puisque la fonction de plongement que nous utilisons est elle-même également symétrique. Les fonctions générées pour des orientations de 90° et −90° sont équivalentes et donc les coûts également.

— Elle permet de combiner de manière naturelle tous les paramètres de l'ellipse sans réglage fin de termes de pondération qui sont généralement nécessaires lorsque différentes quantités sont présentes dans un même coût.

— Elle permet de gérer naturellement le cas des ellipses circulaires. Dans ce cas, le paramètre d'orientation n'a pas de sens, mais un coût angulaire était pourtant bien calculé avec la formulation précédente, ce qui n'est plus le cas.

2.2.6 Seconde architecture du réseau

Avec cette nouvelle formulation de notre fonction de coût, basée sur des ensembles de niveaux, nous avons légèrement modifié l'architecture du réseau de prédiction d'ellipse. Une base VGG-19, suivie de couches entièrement connectées, est toujours utilisée. Les modifications ont été apportées au niveau des branches de sortie du réseau.

Une première branche prédit les quatre valeurs de distance, correspondant aux coordonnées du centre de l'ellipse et aux tailles de ses demi-axes. Une couche d'activation finale de type *Sigmoid* est utilisée pour obtenir des valeurs comprises entre 0 et 1, que l'on interprètent comme étant normalisées par rapport à la taille de l'image d'entrée (256×256). Les coordonnées du centre et les dimensions des demi-axes des ellipses d'entraînement sont normalisées de la même façon. Une seconde branche est utilisée pour prédire l'orientation de l'ellipse. Du fait de sa symétrie, nous avons défini cet angle entre l'horizontale et l'extrémité du grand demi-axe se trouvant dans la moitié droite de l'ellipse. Il prend donc des valeurs entre $-\frac{\pi}{2}$ et $\frac{\pi}{2}$. La branche d'orientation se termine par une couche d'activation de type *Tangente hyperbolique* qui produit des valeurs comprises entre -1 et 1, que l'on interprète comme son sinus. L'architecture est détaillée dans la figure 2.10.

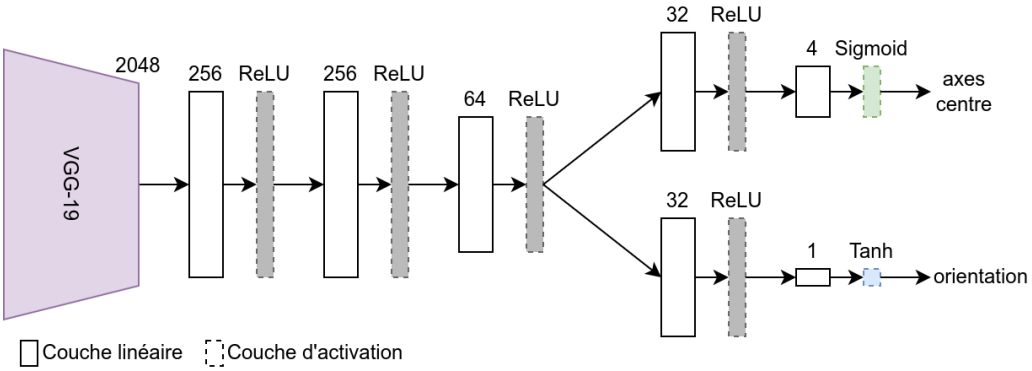


FIGURE 2.10 – Seconde architecture du réseau de prédiction d'ellipse. Un réseau de base (VGG-19) extrait à nouveau un vecteur de caractéristiques de taille 2048, qui est ensuite passé à une série de couches entièrement connectées communes. Enfin, deux branches parallèles sont utilisées pour prédire le centre, les dimensions et l'orientation de l'ellipse.

2.3 Système complet de calcul de pose et entraînement du réseau

2.3.1 Système complet

Les seuls prérequis à notre système sont des images acquises de la scène à partir de points de vue variés. Ces images doivent être calibrées, c'est-à-dire que les paramètres intrinsèques de la caméra utilisée sont connus ainsi que les positions et orientations de chaque image dans le système de coordonnées global de la scène. En pratique, ces annotations de pose, peuvent être obtenues de différentes manières, par exemple, en utilisant un système de GPS différentiel couplé avec

un capteur d'orientation, un procédé de Structure-from-Motion (SfM) tel que Colmap [SZPF16 ; SF16], ou encore des marqueurs [Yan+20]. Relativement peu d'images sont nécessaires mais cela dépend de la taille de la scène et de la couverture des points de vue souhaitée. Par exemple, pour l'expérience décrite dans la section 2.4.4, nous avons utilisé trois séquences de prises de vues de 1000 images chacune pour l'ajustement du détecteur d'objets et pour l'entraînement des réseaux de prédiction d'ellipses.

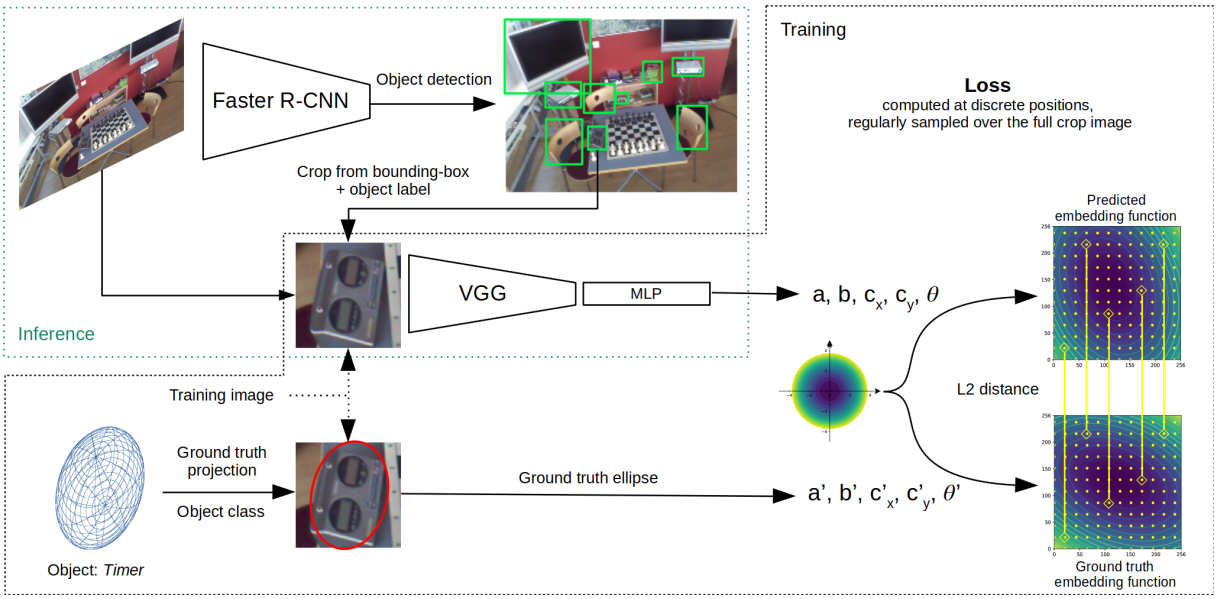


FIGURE 2.11 – Illustration de notre système complet de localisation visuelle basée objet. Le réseau de détection d'objets Faster R-CNN est utilisé pour extraire les sous-images contenant des objets, qui sont passées au réseau de prédiction d'ellipse. Les données de vérité terrain des ellipses et des classes des objets, nécessaires à l'entraînement, sont obtenues en projetant les modèles ellipsoïdaux des objets dans les images d'entraînement. La fonction de coût basée sur les ensembles de niveaux est utilisée pour entraîner le réseau de prédiction d'ellipse. Les données annotées sous forme d'ellipses peuvent être transformées en boîtes et utilisées pour ajuster finement le réseau de détection d'objets.

2.3.2 Reconstruction du modèle de scène

Nous utilisons un modèle de scène ellipsoïdal, dans lequel chaque objet est représenté par un ellipsoïde associé à une classe. Pour reconstruire ce modèle ellipsoïdal d'un objet, nous adoptons une approche simple nécessitant une intervention manuelle minimale.

1. Un minimum de trois images sont choisies, ayant des points de vue variés sur l'objet.
2. Une boîte alignée autour de l'objet est définie dans chaque image.
3. Les ellipses alignées aux axes de l'image et inscrites dans ces boîtes sont extraites.
4. L'ellipsoïde est construit à partir de ces trois ellipses et des matrices de projection des images correspondantes. Une solution analytique a été proposée par Rubino *et al.* [RCB18] pour résoudre ce problème.

Il est important de noter que les ellipsoïdes obtenus dépendent évidemment des images choisies initialement et donc des points de vue utilisés. De plus, nous utilisons simplement les ellipses

inscrites dans les boîtes dessinées autour des objets et alignées aux axes de l'image. Cela ne pose pas de problème ici, car le modèle ellipsoïdal de l'objet est approximatif et son ajustement précis, ou non, en 3D n'a pas de réelle influence. Ceci sera démontré plus tard dans l'expérience de la section 2.4.5.

2.3.3 Annotation automatique à partir du modèle de scène

Une fois le modèle de scène construit, notre système permet d'annoter de manière automatique les images d'entraînement. Pour cela le modèle ellipsoïdal d'un objet est simplement projeté dans les images d'entraînement, dont les poses de caméras sont connues. La projection d'un ellipsoïde se fait par l'équation

$$C^* = PQ^*P^T, \tag{2.13}$$

dans laquelle P correspond à la matrice de projection de la caméra, Q^* est la matrice de forme duale de l'ellipsoïde et C^* est la matrice duale de l'ellipse obtenue.

Les occultations entre objets peuvent également être facilement détectées en comparant leurs profondeurs par rapport à la caméra. Seules les occultations causées par d'autres parties de la scène, non modélisées, nécessitent une intervention manuelle. Cela revient à retirer quelques annotations, dans les cas où l'objet en question est caché dans l'image. Si des cartes de profondeur sont disponibles, cela peut également être traité de manière automatique. Les ellipses obtenues par projection constituent la vérité terrain pour le réseau de prédiction d'ellipse et sont également utilisées pour générer des données d'ajustement fin (*fine-tuning*) du réseau de détection d'objets. Pour cela, les boîtes alignées aux axes de l'image et ajustées aux ellipses sont utilisées.

2.3.4 Couplage entre la détection d'objet et la prédiction d'ellipse

Comme illustré dans la figure 2.11, notre système de détection d'objets améliorée sous la forme d'ellipses orientées est composé de deux étapes. Dans un premier temps, les objets sont détectés sous forme de boîtes alignées aux axes de l'image, puis les ellipses cohérentes avec les modèles 3D des objets sont prédites. Nous utilisons le réseau de neurones Faster R-CNN [RHGS17] pour réaliser la détection des objets. Celui-ci fournit une boîte 2D ainsi qu'une classe pour chaque objet détecté. La boîte de détection d'un objet est contrainte à être carrée. Pour cela, son centre reste inchangé et la valeur maximale entre sa hauteur (H) et sa largeur (L) est retenue comme taille des côtés du carré. Une sous-image carrée, centrée sur l'objet détecté, est extraite et redimensionnée, en utilisant une interpolation bicubique, à une résolution de 256×256 qui correspond à la taille d'entrée du réseau de prédiction d'ellipse. Cette méthode de redimensionnement permet d'éviter de déformer la sous-image. Celle-ci est ensuite passée au réseau de prédiction d'ellipse. À l'inférence, les coordonnées du centre de l'ellipse prédite sont multipliées par $\frac{max(H,L)}{256}$ et translatées par le coin supérieur gauche de la boîte de détection carrée utilisée pour extraire la sous-image donnée en entrée du réseau de prédiction d'ellipse. Les dimensions de l'ellipse sont également remises à l'échelle par le même facteur. Cela permet d'exprimer l'ellipse prédite dans l'image complète.

Les deux réseaux sont donc couplés dans notre système, mais n'opèrent pas au même niveau d'objets. D'un côté, Faster R-CNN est capable de détecter des objets et de reconnaître leur classe. De l'autre, le réseau de prédiction d'ellipse est entraîné à prédire la projection d'un certain modèle ellipsoïdal d'un objet spécifique. L'un travaille donc au niveau de la classe des objets (le détecteur), alors que l'autre opère au niveau de l'instance d'objet (le prédicteur d'ellipse). Comme une scène peut contenir plusieurs instances d'un certain type d'objet, il peut être nécessaire de prédire plusieurs ellipses par objet. Par exemple, lorsqu'un objet de classe c est détecté, les

réseaux de prédiction d'ellipses entraînés pour les différents instances d'objets de classe c qui se trouvent dans le modèle de scène sont utilisés. Cela donne lieu à plusieurs hypothèses d'ellipses qui sont résolues au niveau du calcul de pose, ce qui sera décrit dans la section 2.3.6.

Cette difficulté intervient principalement lorsqu'un détecteur d'objet pré-entraîné est utilisé. Il est cependant possible d'ajouter une étape d'ajustement du détecteur avec des classes d'objets affinées et qui se rapprochent d'instances spécifiques. Par exemple, la classe chaise peut être affinée en « fauteuil », « chaise avec accoudoirs » ou « chaise à roulettes sans accoudoirs » (voir figure 2.12). D'autres caractéristiques peuvent également être ajoutées afin de passer d'une détection de classes d'objets à une détection d'instances, tant que celles-ci restent suffisamment discriminatives visuellement.

FIGURE 2.12 – Illustration de différents type de chaises, pouvant être considérées comme une seule ou plusieurs classes. Les images sont issues de la base de données Objectron [Ahm+21].

2.3.5 Augmentation de données pour le réseau de prédiction d'ellipse

L'augmentation de données joue un rôle important lors de l'entraînement d'un réseau de neurones, notamment lorsqu'un nombre limité de données d'entraînement sont disponibles. Cela permet, en effet, d'introduire une bien plus grande variance dans les données vues par le réseau et ainsi d'améliorer de manière significative sa capacité de généralisation ou d'introduire une invariance à certains éléments. Dans notre cas, nous avons utilisé les différentes stratégies d'augmentation de données suivantes :

— Un bruit de couleur qui modifie de manière aléatoire la luminosité, le contraste et la saturation des images afin de simuler des changements d'illumination.

— Du flou qui est introduit dans les images avec filtre à noyau gaussien de taille aléatoire afin de tenir compte des résolutions différentes dues à l'éloignement de l'objet.

— Une translation aléatoire qui fait en sorte que l'objet ne soit pas toujours parfaitement centré dans les images données au réseau. Cela permet de mieux gérer des détections imparfaites de l'objet, qui causent des décalages des sous-images extraites.

— Des rotations dans le plan ainsi que des déformations perspectives (homographies) aléatoires qui sont utilisées pour générer de nouvelles vues de l'objet. Elles peuvent, par exemple, simuler une caméra qui n'est pas tenue droite ou qui ne regarde pas le centre de l'objet.

Ces différentes transformations sont résumées dans la figure 2.13. Elles sont appliquées de manière aléatoire, sur chaque image, tout au long de l'entraînement du réseau.

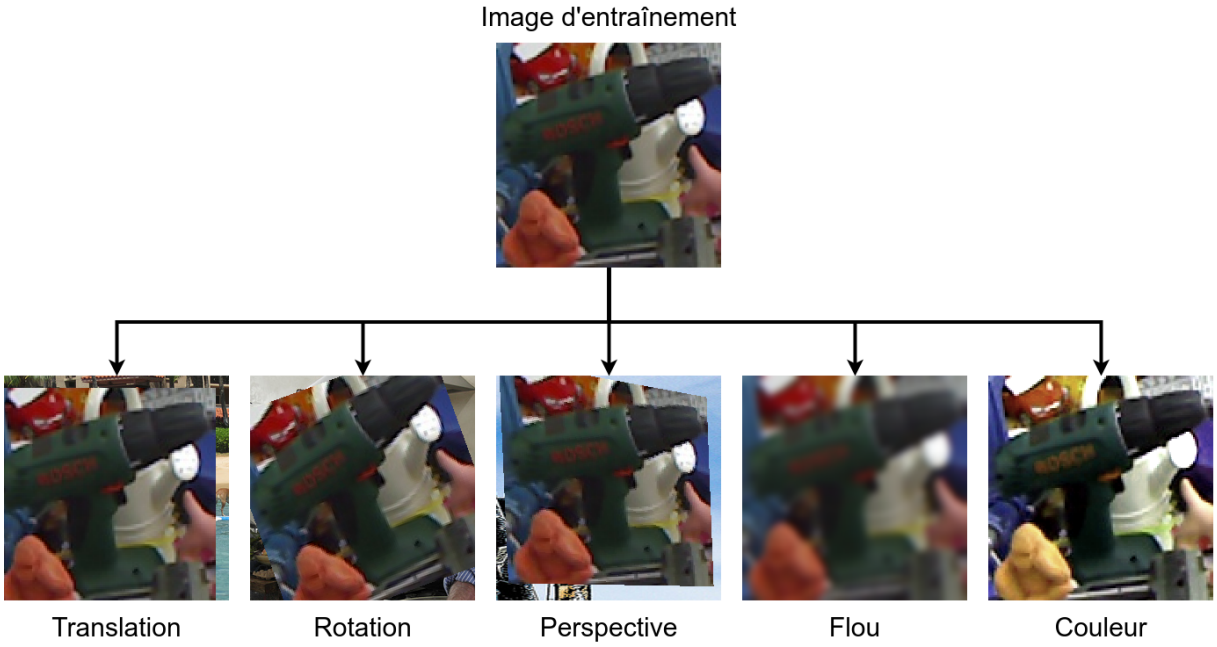


FIGURE 2.13 – Illustrations des stratégies d'augmentation de données utilisées : translation, rotation, perspective, flou et couleur. Celles-ci sont appliquées de manière aléatoire aux images tout au long de l'entraînement.

2.3.6 Calcul de la pose de la caméra

Le système présenté précédemment permet donc de détecter des objets dans les images sous forme d'ellipses orientées. En les combinant aux modèles ellipsoïdaux des objets de notre modèle de scène, il est possible d'estimer la position et l'orientation de la caméra.

Notre méthode de calcul de pose, inspirée des travaux précédents de Gaudillière *et al.* [GSB20 ; GSB19a ; GSB19b], passe en revue les combinaisons possibles des différentes associations 2D-3D entre les ellipses dans l'image et les ellipsoïdes dans la scène. Dans le même esprit que l'algorithme RANSAC, nous avons besoin d'une méthode directe de calcul de pose à partir d'un ensemble minimal de correspondances ellipse-ellipsoïde. Une telle méthode est décrite dans [GSB20] à partir de seulement deux paires ellipse-ellipsoïde, mais sous la condition que l'angle de roulis de la caméra soit nul. Cette contrainte est relativement forte pour des applications de réalité augmentée qui supposent un mouvement totalement libre de l'utilisateur. Nous avons donc adopté une autre stratégie basée sur un algorithme Perspective-3-Point (P3P) [KSS11]. Nous proposons d'appliquer cet algorithme de calcul de pose sur les centres de trois ellipses et ellipsoïdes, comme illustré dans la figure 2.14.

Supposer que le centre d'un ellipsoïde se projette sur le centre de son ellipse de projection n'est pas exact en théorie, mais est néanmoins réaliste en pratique. L'erreur reste relativement faible, seulement de quelques pixels, dans le cas d'une caméra avec un champ de vision standard, comme démontré par l'expérience disponible dans la figure 2.15.

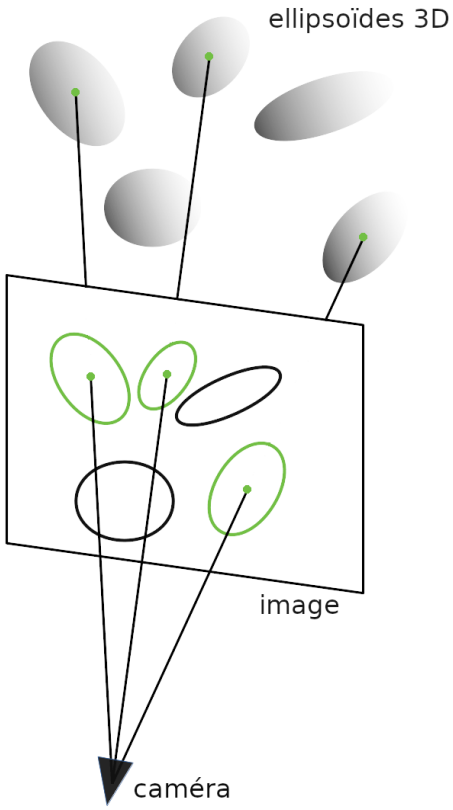


FIGURE 2.14 – Utilisation de l'algorithme Perspective-3-Point (P3P) sur les centres des ellipses et ellipsoïdes de trois objets. Cela permet d'obtenir 4 hypothèses de pose de la caméra.

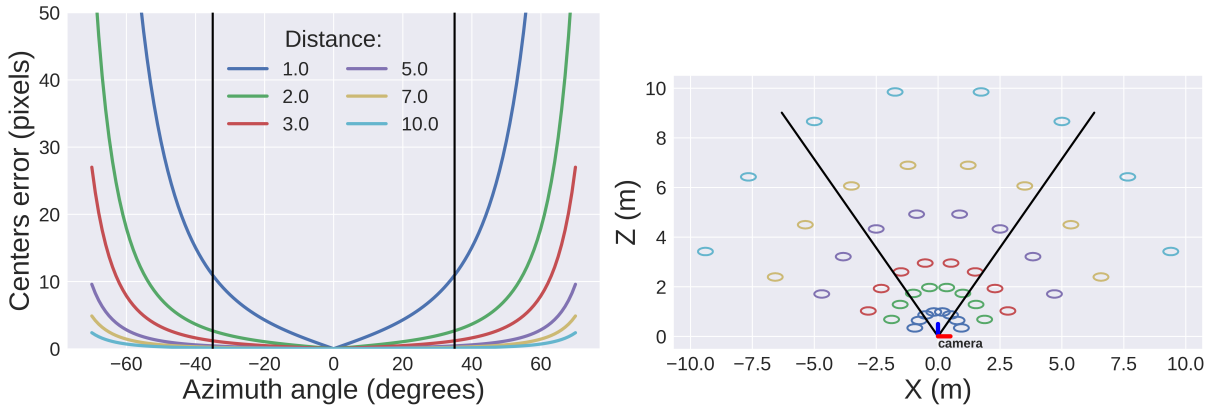


FIGURE 2.15 – Illustration de l'erreur commise par la supposition que le centre d'un ellipsoïde se projette sur le centre de son ellipse de projection. Les courbes de gauche montrent la distance entre le centre de l'ellipse de projection et la projection du centre de l'ellipsoïde pour différents angles de vue et distances entre l'objet et la caméra. Ces différents cas sont illustrés sur la figure de droite, avec une vue d'en haut.

L'algorithme P3P fournit quatre hypothèses de pose possibles lorsqu'il est utilisé avec trois points. Un quatrième point pourrait permettre de réduire ce nombre de solutions, mais nous nous limitons à trois points étant donné que le calcul de pose doit utiliser un nombre minimal de correspondances 2D-3D. Ces quatre hypothèses de pose sont calculées pour toutes les combinaisons possibles de paires ellipse-ellipsoïde et constituent un ensemble de poses plausibles. Parmi ces hypothèses, la meilleure pose est déterminée à partir d'un critère d'adéquation entre les détections des objets dans l'image et les projections des objets du modèle scène avec la pose considérée. Ce critère est calculé par l'Intersection-sur-l'Union (IoU) entre l'ellipse de détection et celle de projection. Contrairement à l'approche classique utilisée dans la méthode RANSAC qui consiste à maximiser le nombre d'inliers, nous adoptons une approche inspirée de MLESAC [TZ00] qui consiste à minimiser

$$\mathcal{C} = \sum_{i,j} 1 - \rho(\text{IoU}(\mathcal{E}_{det}^{i}, \mathcal{E}_{proj}^{j})), \tag{2.14}$$

où

$$\rho(\mathbf{x}) = \begin{cases} \mathbf{x} & \text{si } \mathbf{x} \geq T \\ 0 & \text{si } \mathbf{x} < T \end{cases}, \tag{2.15}$$

et T définit le seuil de recouvrement minimal entre une ellipse de détection $\mathcal{E}_{det}$ et la projection d'un ellipsoïde $\mathcal{E}_{proj}$ pour considérer la paire comme valide. Dans cette approche, le coût d'une donnée considérée comme valide est continu et permet de mieux départager certaines solutions. En effet, dans notre cas d'application le nombre d'associations valides possibles reste relativement restreint puisqu'il correspond, au maximum, au nombre d'objets détectés dans l'image. Il est donc probable de trouver des hypothèses de pose ayant le même nombre d'associations considérées comme valides. Avec l'approche classique de l'algorithme RANSAC, ces hypothèses auraient le même score et ne pourraient pas être départagées.

Toutes les paires d'ellipses-ellipsoïdes sont contraintes par les classes des objets et des détections. La méthode complète de calcul de pose à partir d'au moins trois objets est résumée dans l'algorithme 1.

Dans le cas où seulement deux objets, existants dans le modèle de scène, sont détectés dans l'image, notre système utilise la méthode P2E proposée dans [GSB20] et suppose que l'angle de roulis de la caméra est nul. Enfin, si un seul objet du modèle de scène est détecté dans l'image, il est tout de même possible de calculer la position de la caméra, à partir de seulement une paire ellipse-ellipsoïde, en utilisant la méthode décrite dans [GSB19b]. Cela suppose que l'orientation de la caméra soit connue, ce qui peut être obtenu, en pratique, avec un capteur externe de type gyroscope ou en utilisant une méthode automatique de détection de points de fuite telle que [SFB18]. Le tableau 2.1 résume les différentes approches de calcul de pose de notre système en fonction du nombre d'objet détectés et existants dans le modèle de scène.

Nombre d'objets détectés et existants dans le modèle de scène	Méthode utilisée	Contraintes / Approximations
>= 3	P3P à l'intérieur d'une boucle testant toutes les combinaisons possibles de paires ellipse-ellipsoïde	— Le centre de l'ellipsoïde se projette sur le centre de son ellipse de projection.
2	P2E [GSB20]	— L'angle de roulis de la caméra est nul. — La droite reliant les centres des ellipsoïdes se projette sur la droite reliant les centres des ellipses projetées.
1	P1E [GSB19a ; GSB19b]	— L'orientation de la caméra est connue.

TABLE 2.1 – Résumé des méthodes utilisées pour le calcul de la pose de la caméra en fonction du nombre d'objets détectés dans l'image.

Algorithm 1 Calcul de la pose de la caméra à partir d'au moins trois correspondances ellipse-ellipsoïde

Input : Detections (ellipses), Objects (ellipsoids)
Output : Pose

$\mathcal{D} \leftarrow$ BuildTriplets(Detections)
$best\_pose \leftarrow I_{3\times4}$
$min\_cost \leftarrow \infty$
for (det_i, det_j, det_k) **in** $\mathcal{D}$ **do**
  $\mathcal{O} \leftarrow$ BuildPossibleTriplets($\mathcal{D}$, Objects) ▷ constrained by objects labels
  for (obj_u, obj_v, obj_w) **in** $\mathcal{O}$ **do**
    $poses \leftarrow$ P3P($dets_{ijk}, obj_{uvw}$) ▷ P3P using centers
    $min\_cost\_p3p \leftarrow \infty$
    $best\_pose\_p3p \leftarrow I_{3\times4}$
    for p **in** $\{0, 1, 2, 3\}$ **do**
      $Rt \leftarrow poses[p]$
      $\mathcal{E} \leftarrow$ Project(objects, Rt) ▷ obtain the projection ellipses $\mathcal{E}$
      $cost \leftarrow$ ComputeOverlapCost($\mathcal{E}$, detections) ▷ computed with IoU
      if $cost < min\_cost\_p3p$ **then**
        $min\_cost\_p3p \leftarrow cost$
        $best\_pose\_p3p \leftarrow Rt$
      end if
    end for
    if $min\_cost\_p3p < min\_cost$ **then**
      $min\_cost \leftarrow min\_cost\_p3p$
      $best\_pose \leftarrow best\_pose\_p3p$
    end if
  end for
end for
return $best\_pose$ ▷ return the estimated pose

2.4 Expériences et résultats

2.4.1 Jeux de données

Dans cette section, nous présentons les différents jeux de données utilisés pour évaluer notre système.

LINEMOD

Cette base de données fournit des images couleur et des cartes de profondeur de quinze objets avec des poses 6D annotées. Les modèles 3D colorés des objets sont également fournis sous forme de maillages. Ils ont été obtenus manuellement ou en utilisant des systèmes de reconstruction de surface 3D tels que KinectFusion [New+11]. Chaque objet est associé à un ensemble d'images de test montrant une instance de cet objet annoté et plusieurs autres objets placés autour de manière aléatoire et créant quelques occultations. Pour chaque objet, nous avons sélectionné environ 200 images d'entraînement et 200 images de test. Quelques échantillons d'images sont disponibles dans la figure 2.16.

Ce jeu de données a été utilisé pour évaluer la qualité de la prédiction d'ellipse et le calcul de la position de la caméra uniquement, étant donné que seul l'objet central reste fixe dans les séquences. Les autres objets sont régulièrement déplacés à l'intérieur d'une même séquence et ne peuvent donc être utilisés pour estimer la pose complète de la caméra.

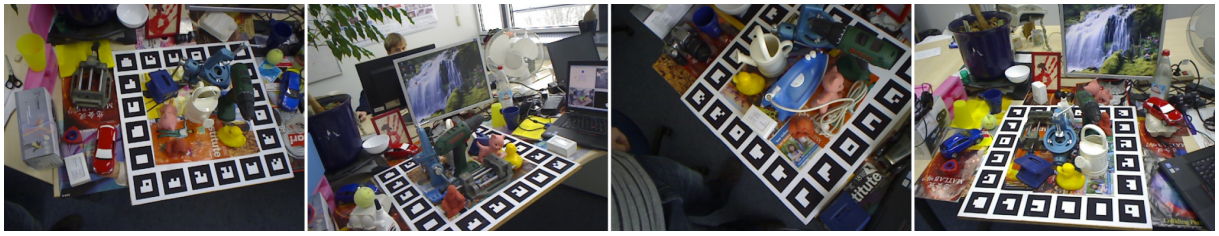


FIGURE 2.16 – Échantillons d'images du jeu de données LINEMOD.

WatchPose

Les données LINEMOD présentées précédemment sont des images réelles mais dans un environnement assez contrôlé, avec des objets placés sur un bureau. Nous avons donc également utilisé un jeu de données nommé WatchPose [Yan+20], qui fournit des images, avec pose annotée, de dix objets dans un environnement industriel. Les annotations de poses ont été obtenues en utilisant des marqueurs placés dans la scène. Pour chaque objet, environ 200 images sont disponibles. La moitié d'entre elles ont été acquises proches de l'objet (à environ 60 cm) et l'autre moitié de plus loin (à environ 1.4 m). Nous avons donc utilisé ces données pour évaluer la qualité de la prédiction d'ellipse de notre réseau, et en particulier, l'influence d'un éloignement de la caméra différent entre les images d'entraînement et celles de test. Au niveau du calcul de pose, seule la position a été calculée, parce qu'un seul objet est visible dans les images.

Ce jeu de données nous permet d'évaluer notre système dans des conditions proches de celles d'un cas réel d'application. En effet, les images ont été acquises avec un smartphone, tenu à la main, sans se soucier du bruit de mouvement, des reflets et des ombres. Des échantillons d'images sont disponibles dans la figure 2.17.

FIGURE 2.17 – Échantillons d'images du jeu de données WatchPose.

T-LESS

Le jeu de données T-LESS [Hod+17] fournit 20 scènes d'objets de type industriel, non-texturés, placés sur une planche et des images acquises tout autour, avec la caméra placée sur une demi-sphère de rayon 75 cm. La pose de chaque objet présent dans les images est annotée et les modèles CAO de ces objets sont également disponibles. Comme pour le jeu de données LINEMOD, ces scènes restent assez contrôlées, avec plusieurs objets centrés sur un support. La différence étant qu'ici tous les objets restent fixes et peuvent donc être utilisés pour le calcul de la pose 6D complète de la caméra. En particulier, nous avons utilisé une scène avec six objets dans nos expériences. Des échantillons d'images sont disponibles dans la figure 2.18.

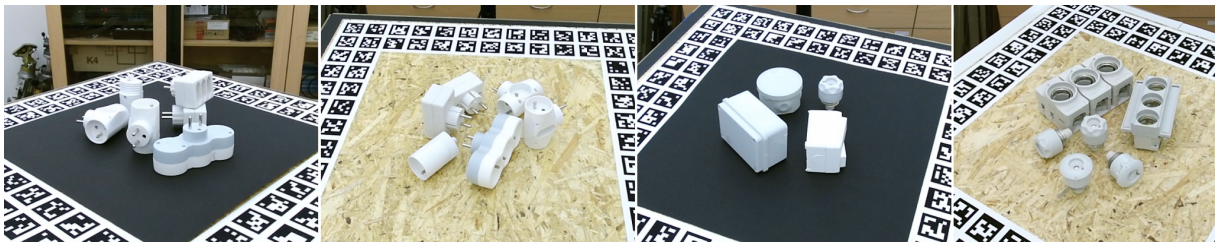


FIGURE 2.18 – Échantillons d'images du jeu de données T-LESS.

7-Scenes : Chess

7-Scenes est un jeu de données créé par Microsoft pour la localisation visuelle en intérieur [GISC13]. Il comprend 7 scènes scannées avec une caméra RGB-D de type Kinect. Pour chaque scène, plusieurs séquences d'images couleur et de cartes de profondeur sont fournies, ainsi que des annotations de poses vérité terrain de la caméra. Nous avons utilisé la scène nommée *Chess*, car elle représente bien le type d'environnement dans lequel les objets constituent des balises intéressantes pour le calcul de pose de la caméra. Nous avons divisé les données disponibles en séquences d'entraînement (1, 4, 6) et séquences de test (2, 3, 5), résultant en deux ensembles de 3000 images chacun. Nous avons choisi d'inclure 11 objets dans notre modèle de scène, répartis en 7 catégories : écran de télévision, console de jeu, dossier de chaise, pendule d'échecs, pile de boîtes de jeu, manette de jeu et interrupteur. Quelques images sont disponibles dans la figure 2.19.

Cette scène permet d'illustrer le fonctionnement de notre système dans des conditions réelles, de la construction du modèle de scène, à l'annotation automatique de données pour l'entraînement ou l'ajustement fin des réseaux, jusqu'au calcul de pose de la caméra. Seules les images couleur et leurs poses annotées ont été utilisées.

FIGURE 2.19 – Échantillons d'images de la scène *Chess* du jeu de données 7-Scenes.

Scènes virtuelles

Enfin, nous avons également créé quelques scènes virtuelles en utilisant les objets de la base de données YCB [Çal+15], qui fournit des maillages texturés d'objets. Ces scènes virtuelles sont particulièrement intéressantes car elles nous ont permis d'avoir un contrôle complet sur les environnements et les prises de vues. Nous avons, par exemple, pu analyser de manière plus précise l'influence de l'arrière-plan des images sur la prédiction d'ellipse (voir section 2.4.3). De même, il nous a été possible de varier de manière significative les points de vue utilisés en entraînement par rapport à ceux de test. Quelques images de ces scènes sont disponibles dans la figure 2.20.

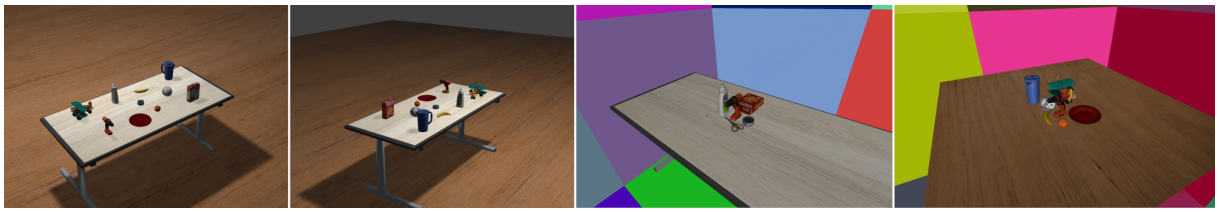


FIGURE 2.20 – Échantillons d'images des scènes virtuelles utilisées.

2.4.2 Détails d'implémentation et d'entraînement

Pour nos expériences sur la scène *Chess*, nous avons entraîné le réseau de prédiction d'ellipse pendant 100 époques pour chaque objet, avec un pas d'apprentissage initial de 5×10^{-5}, réduit de moitié après 50 époques. La taille d'un batch a été fixée à 16 et nous avons utilisé la méthode d'optimisation Adam [KB15]. Pour la détection des objets, nous avons utilisé le réseau Faster R-CNN [RHGS17], pré-entraîné sur la base de données COCO [Lin+14a], puis ajusté sur nos 7 classes d'objets pendant 2000 itérations avec un pas d'apprentissage de 2.5×10^{-4}. Seule sa dernière couche a été modifiée afin d'obtenir 7 classes de sortie. Les deux réseaux sont implémentés en Python avec la bibliothèque PyTorch [Lin+14b].

L'estimation de la pose de la caméra a été implémentée en Python et C++ pour certaines parties, notamment la boucle traitant les combinaisons possibles entre paires ellipse-ellipsoïde, pour des raisons de performances. L'ensemble du code est disponible sur GitLab à l'adresse suivante : `gitlab.inria.fr/tangram/3d-aware-ellipses-for-visual-localization`.

Pour chacun des jeux de données utilisés, les modèles ellipsoïdaux des objets ont été reconstruits à partir de quelques annotations manuelles (une simple boîte autour de l'objet) dessinées dans quelques images (au minimum 3 par objet).

2.4.3 Évaluation de la prédiction d'ellipse

Dans cette première partie d'évaluation, nous analysons spécifiquement les performances de la prédiction d'ellipse.

Comparaison régression directe vs. Multi-Bin vs. ensembles de niveau

Nous avons tout d'abord évalué l'apport de la fonction coût *Multi-Bin* en comparaison d'une régression directe des cinq paramètres d'une ellipse, qui consisterait à minimiser l'erreur quadratique moyenne. Pour cela, nous avons entraîné le réseau de prédiction d'ellipse sur un objet virtuel représentant une boîte en carton, issue de la base de données YCB. Cet objet est placé au centre de la scène et des positions de caméra ont été échantillonnées de manière régulière sur une demi-sphère autour de l'objet. 100 valeurs d'azimut et 20 valeurs d'élévation ont été utilisées pour générer les données d'entraînement. Les points de vue des images de test sont générés de la même manière, mais décalés, à des positions intermédiaires, comme illustré sur la figure 2.21. L'arrière-plan de chaque rendu est généré aléatoirement, à partir des images de la base de données COCO. Les rendus ont été générés avec la bibliothèque VTK [SMLK06]. La figure 2.22 montre quelques images d'entraînement générées ainsi que des prédictions d'ellipses obtenues en utilisant l'approche *Multi-Bin*.

Pour évaluer la qualité de la prédiction de l'ellipse, nous calculons l'IoU entre l'ellipse prédite et celle vérité terrain. Les résultats obtenus pour différents nombres d'époques sont disponibles dans la figure 2.23 et montrent l'avantage de l'approche *Multi-Bin* pour prédire l'orientation de l'ellipse. Il est important de noter que les valeurs d'IoU relativement élevées qui ont été obtenues sont liées à la large couverture de points de vue des images d'entraînement.

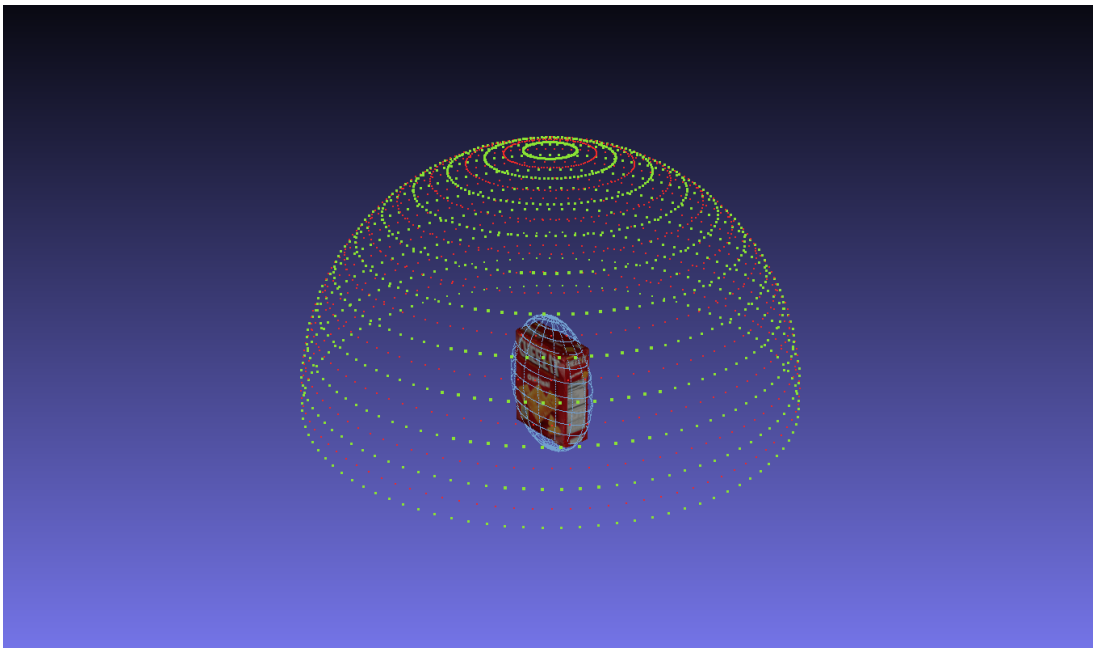


FIGURE 2.21 – Illustration des positions de la caméra utilisées pour générer les images d'entraînement (en rouge) et les images de test (en vert).

FIGURE 2.22 – Illustration des images d'entraînement synthétiques générées et des prédictions d'ellipses obtenues. Les ellipses prédites sont en vert et la vérité terrain en rouge.

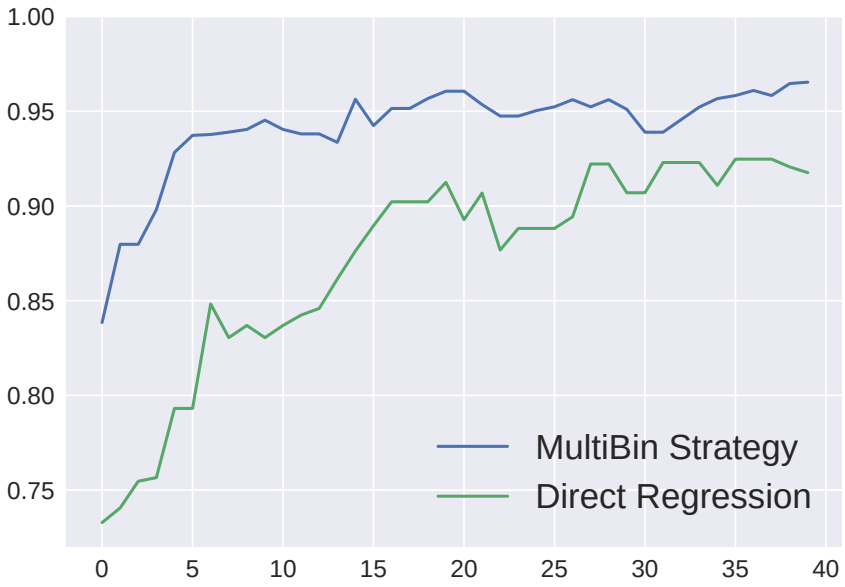


FIGURE 2.23 – Comparaison des évolutions du recouvrement moyen (IoU) entre l'ellipse prédite et celle de vérité terrain, obtenu sur les images de test au cours de l'entraînement, avec la régression directe et l'approche *Multi-bin*, pour l'objet de type boîte.

Malgré ces avantages de l'approche Multi-Bin, certaines erreurs persistent, notamment lors de la prédiction d'une ellipse allongée, pratiquement verticale. La deuxième version de fonction de coût, basée sur des ensembles de niveaux, a donc été proposée. Cette fois, nous avons comparé les deux fonctions de coût sur des données réelles. Pour cela, les 11 objets de la scène *Chess* ont été utilisés. Des exemples d'ellipses prédites pour ces objets sont illustrés dans la figure 2.24. La qualité de la prédiction d'ellipse a été mesurée sur la séquence de test 2, à nouveau en termes d'IoU entre les ellipses. Les résultats sont disponibles dans le tableau 2.2 et montrent clairement l'avantage de cette nouvelle formulation de la fonction de coût. En effet, le réseau entraîné avec la fonction de coût utilisant des ensembles de niveau obtient de meilleurs résultats pour chaque objet.

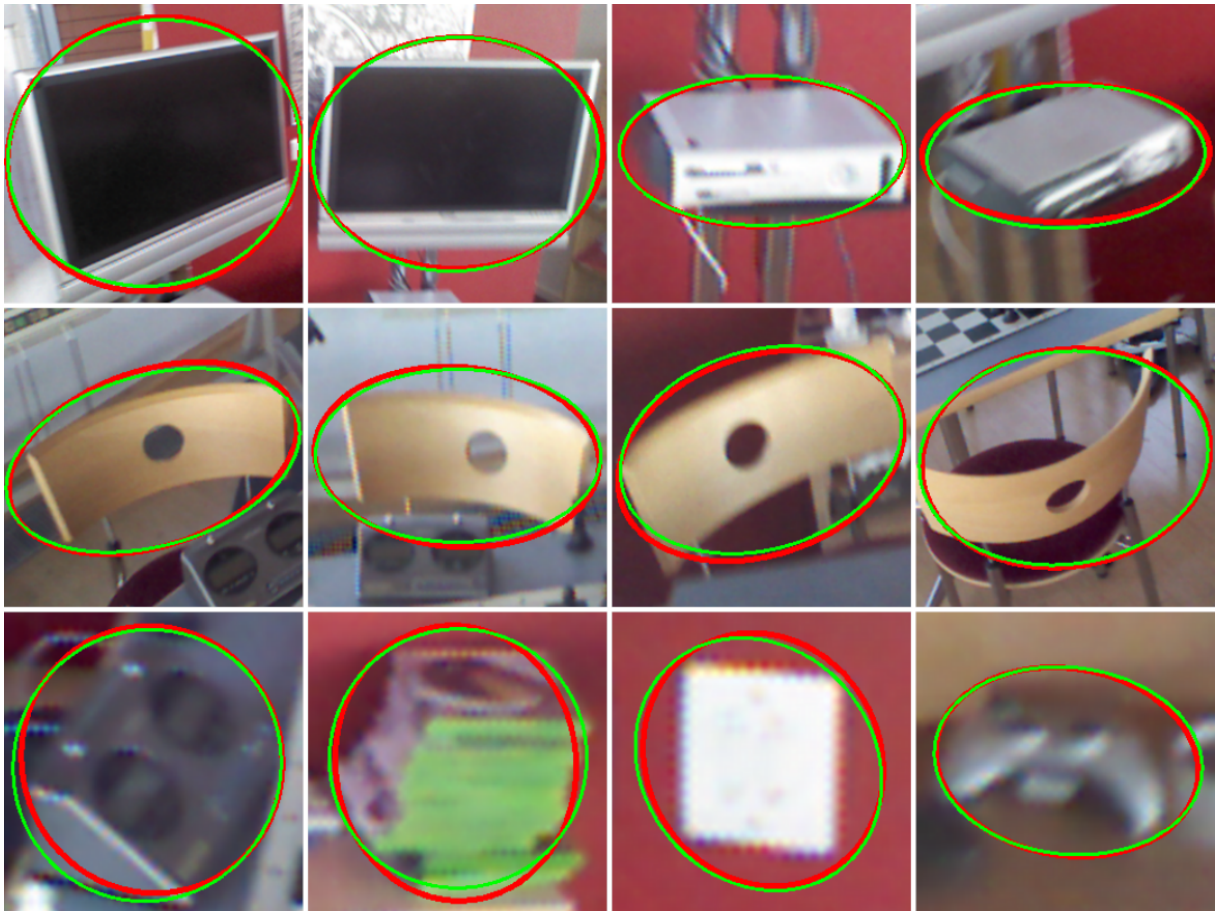


FIGURE 2.24 – Illustration de prédictions d'ellipses, obtenues avec la seconde architecture du réseau, pour les objets de la scène *Chess*. Les ellipses prédites sont en vert et les ellipses de vérité terrain sont en rouge.

Objets	*Multi-Bin*		Ensembles de niveaux	
	IoU moyen	Pourcentage d'IoU > 0.8	IoU moyen	Pourcentage d'IoU > 0.8
Télévision (gauche)	0.912	1.0	**0.96**	1.0
Télévision (droite)	0.897	1.0	**0.936**	1.0
Console de jeu (gauche)	0.869	0.788	**0.943**	0.995
Console de jeu (droite)	0.854	0.8	**0.952**	0.997
Dossier de chaise (milieu)	0.906	0.948	**0.95**	1.0
Dossier de chaise (gauche)	0.888	0.802	**0.902**	0.831
Dossier de chaise (droite)	0.847	0.7	**0.873**	0.837
Pendule d'échecs	0.908	0.952	**0.936**	0.996
Pile de jeux	0.946	1.0	**0.945**	1.0
Interrupteur	0.918	0.997	**0.935**	1.0
Manette de jeu	0.93	0.96	**0.941**	1.0

TABLE 2.2 – Comparaison de la précision obtenue des ellipses prédites pour un entraînement avec la fonction de coût *Multi-bin* et celle basée sur les ensembles de niveaux. Les objets sont ceux modélisés dans la scène *Chess*.

Comparaison des différentes fonctions de plongement

Comme évoqué dans la section 2.2.5, nous avons analysé différentes fonctions de plongement utilisées pour le calcul d'une distance entre ellipses. Ces expériences ont été réalisées sur l'objet *perceuse* de la base de données YCB (orange) et sur celle du jeu de données LINEMOD (verte). Les résultats, dans le tableau 2.3, montrent que la forme avec $[\alpha, \beta]$ sur la diagonale de la matrice centrale obtient de meilleures prédictions d'ellipse. La meilleure stabilité numérique de cette formule pour l'entraînement du réseau avait déjà été confirmée sur la figure 2.9.

Forme de la matrice centrale	Perceuse YCB		Perceuse LINEMOD	
	IoU moyen	Pourcentage d'IoU > 0.8	IoU moyen	Pourcentage d'IoU > 0.8
$\begin{bmatrix} \frac{1}{\alpha^2} & 0 \\ 0 & \frac{1}{\beta^2} \end{bmatrix}$	0.873	0.86	0.866	0.90
$\begin{bmatrix} \frac{1}{\alpha} & 0 \\ 0 & \frac{1}{\beta} \end{bmatrix}$	0.910	0.97	0.903	0.98
$\begin{bmatrix} \alpha^2 & 0 \\ 0 & \beta^2 \end{bmatrix}$	0.919	**0.98**	0.903	0.97
$\begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}$	**0.921**	**0.98**	**0.930**	**0.99**

TABLE 2.3 – Comparaison des résultats de prédiction d'ellipse obtenus avec les différentes versions de la fonction de plongement.

Influence de l'éloignement de la caméra

Nous avons ensuite évalué l'influence de l'éloignement de la caméra par rapport à l'objet. Pour cela, nous avons réutilisé la même procédure que celle décrite précédemment, dans la section 2.4.3, avec des rendus synthétiques d'un objet central et des caméras placées sur une demi-sphère autour de celui-ci. Les images d'entraînement ont ainsi été générées, à distance fixe de l'objet (50 cm), et nous avons fait varier l'éloignement de la caméra pour les images de test. Des rayons croissants,

de 50 cm à 200 cm, ont été utilisés pour placer la caméra, comme illustré sur la figure 2.25. La taille de l'objet diminuant avec l'éloignement de la caméra, les sous-images extraites de l'objet sont mises à l'échelle pour respecter la taille d'entrée requise par le réseau.

Les résultats de prédiction d'ellipses à différents niveaux d'éloignement sont disponibles sur les figures 2.27 et 2.26. Ils montrent une légère baisse dans la qualité de la prédiction au fur et à mesure que la caméra s'éloigne. Cependant, l'adaptation du réseau à une distance caméra-objet différente de celle d'entraînement reste satisfaisante.

Un éloignement plus important de la caméra et davantage de changements de points de vue entre les données d'entraînement et ceux de test sont évalués dans les expériences sur le jeu de données WatchPose et sur une scène synthétique dans la section 2.4.5.

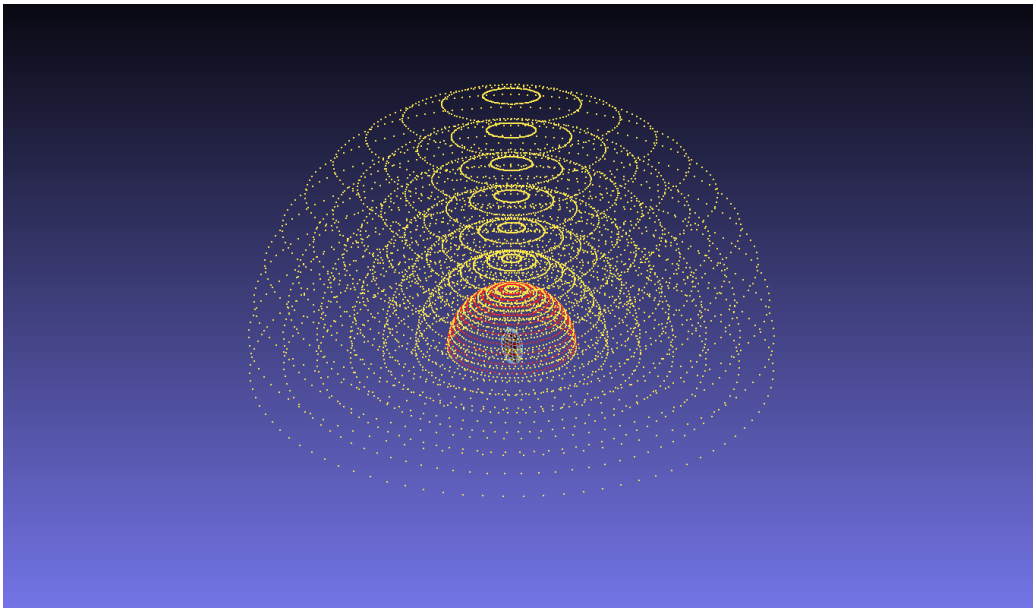


FIGURE 2.25 – Illustration des positions de la caméra utilisées pour générer les images de test pour un éloignement croissant par rapport à l'objet.

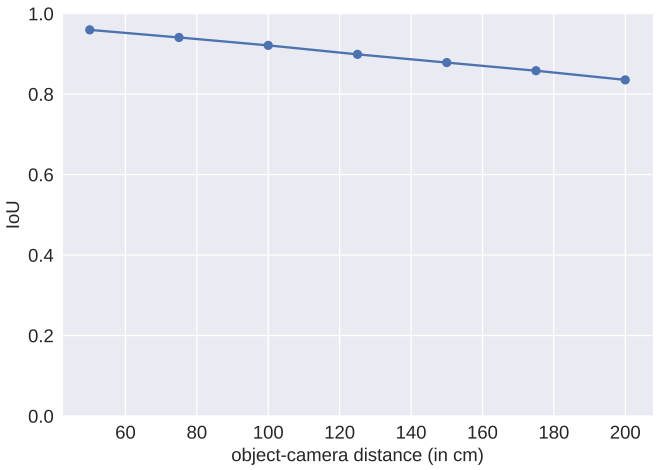


FIGURE 2.26 – Influence de l'éloignement de la caméra par rapport à l'objet dans les images de test, pour un entraînement à une distance fixe de 50 cm.

FIGURE 2.27 – Illustrations d'ellipses prédites dans le cas d'un éloignement croissant de la caméra par rapport à l'objet. Les ellipses prédites sont en vert et les ellipses de vérité terrain sont en rouge. On peut noter la baisse de qualité des images avec cet éloignement.

Influence de l'arrière-plan de l'image

Étant donné que le réseau de prédiction d'ellipse prend en entrée une image carrée, de taille fixe, l'arrière-plan de la scène est également visible dans certaines portions des images d'entrée. La quantité d'arrière-plan visible dépend évidemment de la forme de l'objet et du point de vue. Afin de mieux comprendre à quel point cet arrière-plan interfère ou contribue à la prédiction de l'ellipse, nous avons utilisé deux scènes synthétiques. Celles-ci sont composées du même objet central entouré d'autres objets dans des environnements différents (voir figure 2.28). Nous avons entraîné le réseau de prédiction sur des images de la première scène et ensuite évalué sa performance sur des images de test de chacune des scènes.

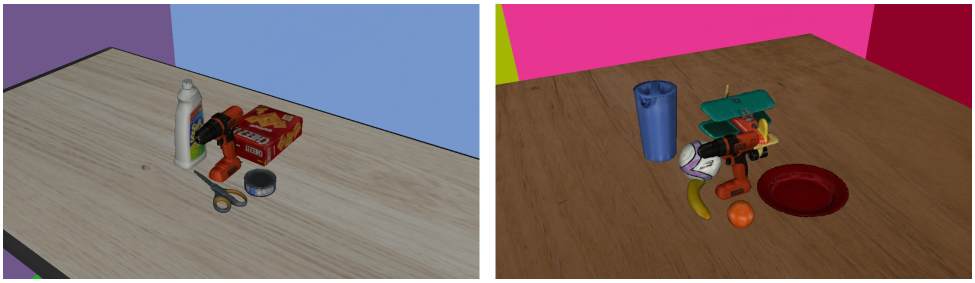


FIGURE 2.28 – Illustration des deux scènes virtuelles, avec le même objet central (la perceuse) dans des environnements différents.

Pendant l'entraînement, nous avons utilisé trois stratégies de masques différentes :

1. Pas de masque, les sous-images carrées contenant l'objet et l'arrière-plan sont utilisées.
2. Un masque elliptique est utilisé. Il est obtenu en projetant le modèle ellipsoïdal de l'objet et permet de modifier la partie extérieure (contenant principalement de l'arrière-plan) de manière aléatoire.
3. Un masque précis est utilisé pour remplacer tout l'arrière-plan par une image choisie aléatoirement. Ce masque peut être obtenu en utilisant un modèle 3D précis de l'objet.

Les modifications de l'arrière-plan consistent simplement à le remplacer par des images choisies aléatoirement dans la base de données COCO. Ces trois stratégies sont illustrées dans la figure 2.29. La qualité des ellipses prédites est à nouveau évaluée en termes d'IoU et nous avons répété les expériences pour deux objets centraux différents, une perceuse (nommée *driller*) et une boîte (nommée *cracker*).

(a) Pas de masque (b) Masque elliptique (c) Masque précis

FIGURE 2.29 – Les trois stratégies de masque évaluées, à savoir, sans masque, avec un masque elliptique et avec un masque précis de l'objet.

Les courbes sur la figure 2.30 montrent les IoU moyens obtenus pour les trois stratégies de masques sur les images de test pour différentes époques d'entraînement. Les résultats obtenus pour des images de test avec le même arrière-plan qu'à l'entraînement (colonne de gauche) montrent des performances relativement similaires pour toutes les stratégies et pour les deux objets. Seule la stratégie sans masque semble légèrement meilleure, indiquant que le réseau parvient à tirer un avantage de l'arrière-plan visible dans les sous-images.

La seconde évaluation, sur des images dans des environnements différents de ceux d'entraînement (colonne de droite), confirme ces impressions. En effet, c'est la stratégie du masque précis, permettant de complètement remplacer l'arrière-plan, qui donne les meilleurs résultats, alors que l'entraînement avec les sous-images originales, sans masque, performent le moins bien. Cette fois-ci, l'arrière-plan ne contribue plus à l'estimation de la pose de la caméra mais, au contraire, la perturbe. Les résultats obtenus avec les masques précis montrent bien que le fait de remplacer l'arrière-plan de manière aléatoire au cours de l'entraînement rend le réseau indépendant à l'arrière-plan. La stratégie de masque elliptique donne des résultats intermédiaires et semble tout de même fournir une certaine indépendance à l'arrière-plan.

Les performances de la prédiction d'ellipse sans utiliser de masque restent, malgré tout, relativement bonnes (autour de 0.85 d'IoU). Cela montre que le réseau utilise quand même principalement l'apparence de l'objet pour faire sa prédiction. Cela dépend évidemment de la forme de l'objet et de la proportion d'arrière-plan visible dans les sous-images.

Cette expérience montre que l'environnement autour des objets, et donc aussi l'arrière-plan visible autour des objets dans les images, peut être soit avantageux ou désavantageux en fonction

de l'application. Nous n'utilisons, par exemple, aucune stratégie de masques dans notre système car, dans nos cas d'applications, la scène reste fixe, et notamment les objets qui sont utilisés comme balises pour estimer la pose de la caméra. Dans le cas où une indépendance plus forte à l'arrière-plan est requise, la stratégie de masques elliptiques constitue un bon compromis car, au contraire des masques fins, ils ne nécessitent pas d'avoir les modèles 3D précis des objets et tire avantage de leurs modèles ellipsoïdaux.

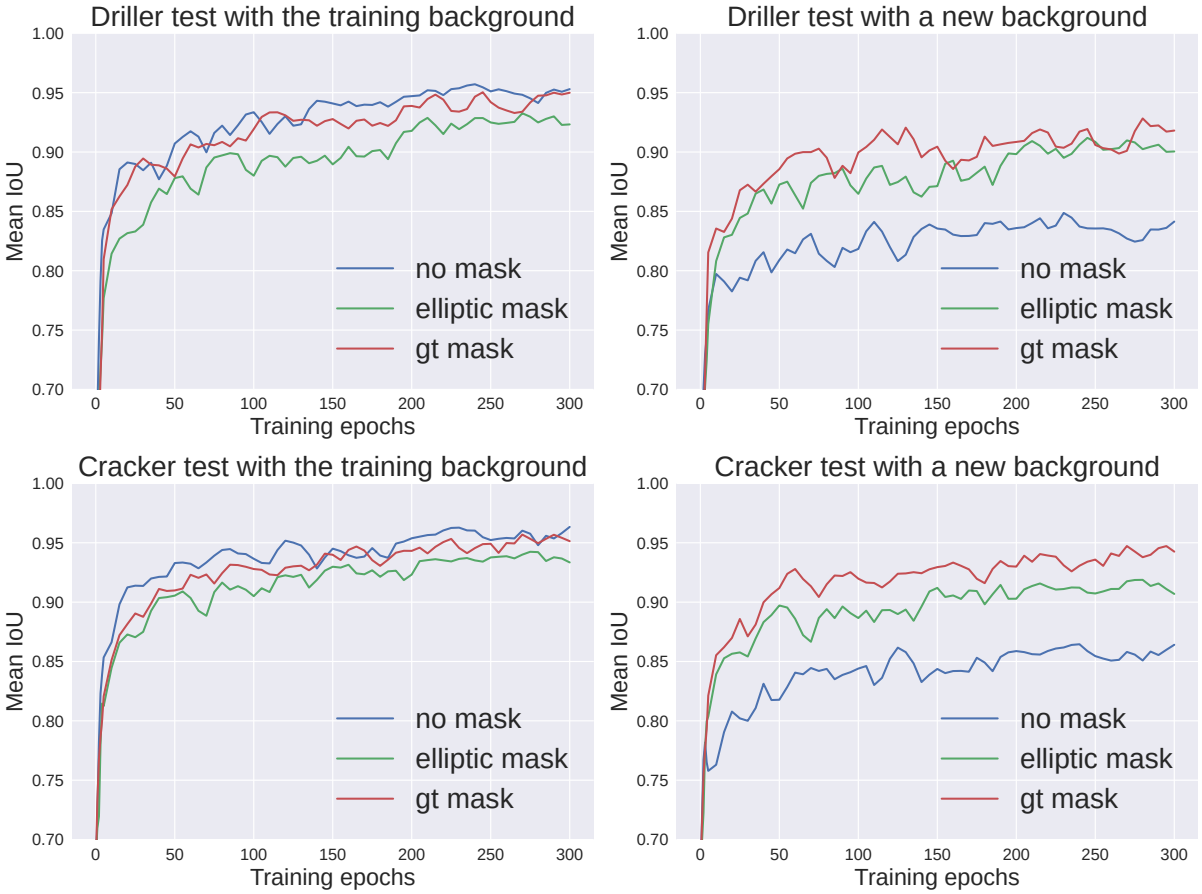


FIGURE 2.30 – IoUs moyens entre les ellipses prédites et la vérité terrain, obtenus sur les images de test tout au long de l'entraînement pour les différentes stratégies de masques. La colonne de gauche montre les résultats obtenus pour le même environnement que celui d'entraînement, donc sur des images de test ayant le même arrière-plan. La colonne de droite montre les résultats obtenus dans des environnement différents, et donc, avec des arrière-plans différents. La ligne du haut correspond à la perceuse orange montrée dans la figure 2.29 et celle du bas à la boîte de biscuits montrée dans la figure 2.27.

2.4.4 Évaluation du calcul de pose

Dans cette section, nous évaluons les apports des ellipses prédites par notre système au niveau du calcul de pose de la caméra. Nous présentons tout d'abord les deux autres méthodes d'estimation de pose de caméra auxquelles nous nous sommes comparés. Par la suite, l'estimation de la pose complète de la caméra est évaluée, et finalement, l'estimation de la position de la caméra à partir d'un seul objet est traitée.

Autres méthodes comparées

Nous avons comparé nos résultats à deux autres méthodes de localisation visuelle, une première qui utilise une approche classique basée structure (OpenVSLAM) et une seconde utilisant de l'apprentissage profond pour prédire la pose de la caméra (PoseLSTM).

OpenVLSAM [SSS19] est une réécriture d'ORB-SLAM2 [MT17], une méthode de cartographie et localisation visuelle simultanées basée sur des points. Ici, nous utilisons uniquement son module de relocalisation qui permet d'estimer la pose de la caméra à partir d'une image et d'une carte de la scène (un nuage de points). Cette méthode utilise des représentations globales de type *Bag-of-Words* [GT12b] pour trouver une image de référence proche, puis des points clés couplés à des descripteurs ORB pour le calcul de la pose de la caméra. Nous avons évalué cette méthode sur deux versions différentes de la carte, l'une obtenue avec la version monoculaire du SLAM et l'autre en utilisant la version RGB-D. Leurs résultats sont respectivement nommés *mono map* et *RGB-D map* dans les tableaux 2.4 et 2.5. Dans les deux cas, les trois séquences d'entraînements (1, 4, 6) ont été utilisées pour reconstruire la carte et la localisation est évaluée sur les séquences de test (2, 3, 5). Dans le cas monoculaire, le facteur d'échelle a été estimé manuellement. Nous avons choisi cette méthode car elle est considérée comme une référence dans le monde du SLAM, où notre approche pourrait justement être appliquée.

Nous avons également choisi de nous comparer à PoseLSTM [Wal+17], une méthode de régression de pose absolue de caméra, qui a largement amélioré les performances des premières approches de régression de pose telles que PoseNet [KGC15]. Ce réseau a été entraîné sur les 3000 images des séquences d'entraînement pendant 2000 époques. Nous avons utilisé le code publié à l'adresse suivante : `github.com/GrumpyZhou/visloc-apr`.

Afin d'évaluer l'intérêt des ellipses prédites de manière cohérente aux représentations 3D des objets, nous comparons également nos résultats à ceux obtenus en utilisant les ellipses inscrites dans les boîtes de détection. Dans ce cas, le détecteur d'objet est entraîné avec une supervision 2D, à partir d'annotations manuelles des objets dans les images d'entraînement.

Résultats du calcul de la pose complète

Nous avons évalué notre méthode de calcul de pose complète sur trois séquences de test de la scène *Chess* du jeu de donnée 7-Scenes. Le modèle de scène reconstruit et utilisé pour le calcul de la pose est illustré dans la figure 2.31. Le tableau 2.4 montre les erreurs de position et d'orientation médianes obtenues pour chaque séquence. Les erreurs médianes obtenues avec notre méthode sont mesurées uniquement sur les images avec au moins deux objets détectés. Il faut également noter que la méthode de relocalisation d'OpenVSLAM échoue de temps en temps et ne fournit aucun résultat de pose. Leurs erreurs de position et d'orientation médianes sont donc uniquement calculées sur les images pour lesquelles la relocalisation fonctionne. Les proportions d'images correctement localisées sont également indiquées. Elles correspondent à celles dont la caméra a été estimée avec une erreur de position inférieure à 20 cm et une erreur d'orientation inférieure à 20°. La répartition de ces erreurs pour différents nombres d'objets utilisés est disponible dans la figure 2.5. Elles montrent, pour chaque séquence de test, la proportion d'images localisées en fonction d'une certaine erreur de position ou d'orientation autorisée. Les colonnes correspondent à différents sous-ensembles d'images de test, respectivement à toutes les images, celles avec au moins deux objets détectés et celles avec au moins trois objets détectés.

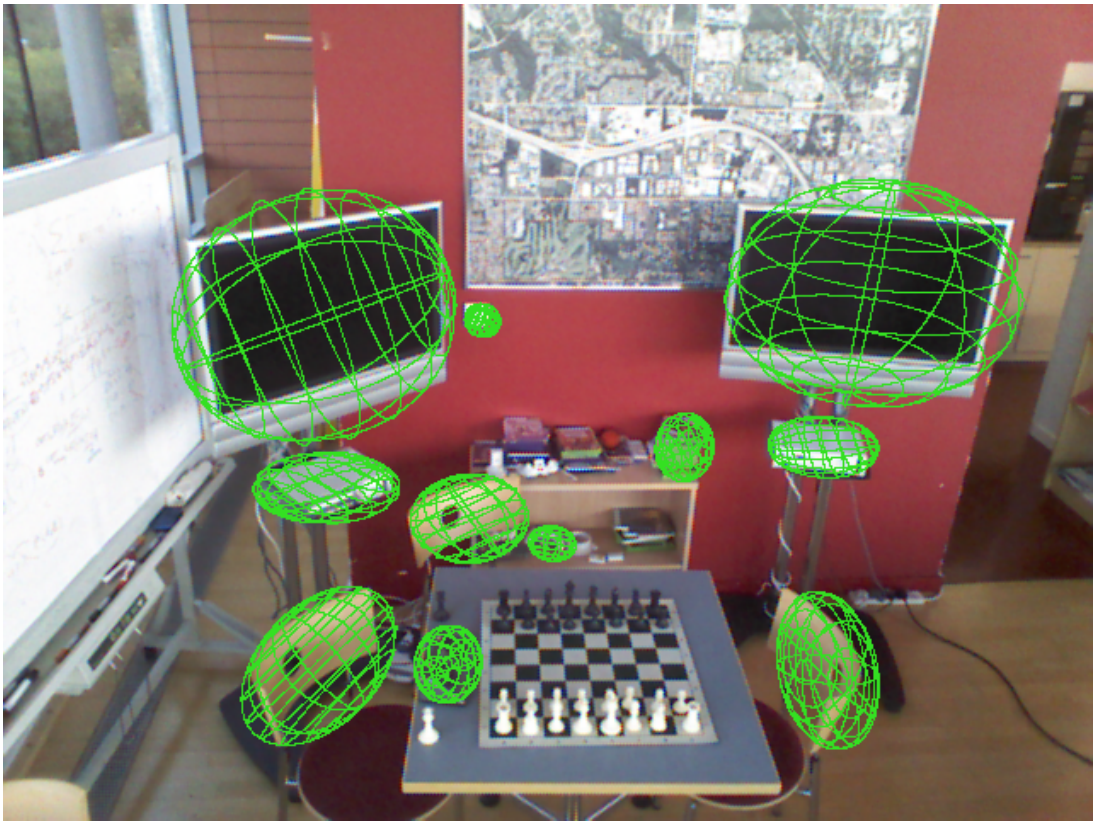


FIGURE 2.31 – Modèle de scène reconstruit de la scène *Chess* et utilisé pour l'estimation de la pose de la caméra dans les expériences.

Method	Sequence 2			Sequence 3			Sequence 5		
	pos. err.	rot. err.	% valid	pos. err.	rot. err.	% valid	pos. err.	rot. err.	% valid
Reloc OpenVSLAM (RGB-D map)	**5.14**	2.41	61.15	**5.05**	2.53	77.93	**4.57**	3.52	83.35
Reloc OpenVSLAM (mono map)	6.48	**2.04**	45.0	6.54	3.17	68.08	6.55	2.61	78.0
PoseLSTM	29.15	8.94	24.69	18.73	6.06	53.68	16.16	6.03	62.28
Inscribed (2D supervision)	11.62	3.69	69.69	10.56	3.12	70.78	10.20	3.24	75.33
Ours (3D-coherent ellipses)	6.46	2.20	**79.48**	7.03	**2.12**	**82.59**	6.42	**2.05**	**85.92**

TABLE 2.4 – Erreurs de position et d'orientation médianes obtenues sur les séquences de test de la scène *Chess* pour les différentes méthode de calcul de pose. Les pourcentages de poses valides sont également indiqués. Ici, une pose est considérée valide si son erreur de position est inférieure à 20 cm et son erreur d'orientation inférieure à 20°.

La figure 2.32 compare les erreurs de position obtenues pour chaque méthode sur les images de la séquence 2. Les erreurs supérieures à 1.75 m correspondent à des images pour lesquelles l'estimation de la pose de la caméra a échoué sans fournir de résultat. Cela arrive, en particulier, pour la méthode de localisation d'OpenVSLAM et pour notre méthode, par exemple, lorsque strictement moins de deux objets ont été détectés. C'est notamment le cas pour les images avec des indices entre 800 et 900 qui ont été prises avec la caméra très proche de la table, et donc, sur lesquelles un seul objet est visible.

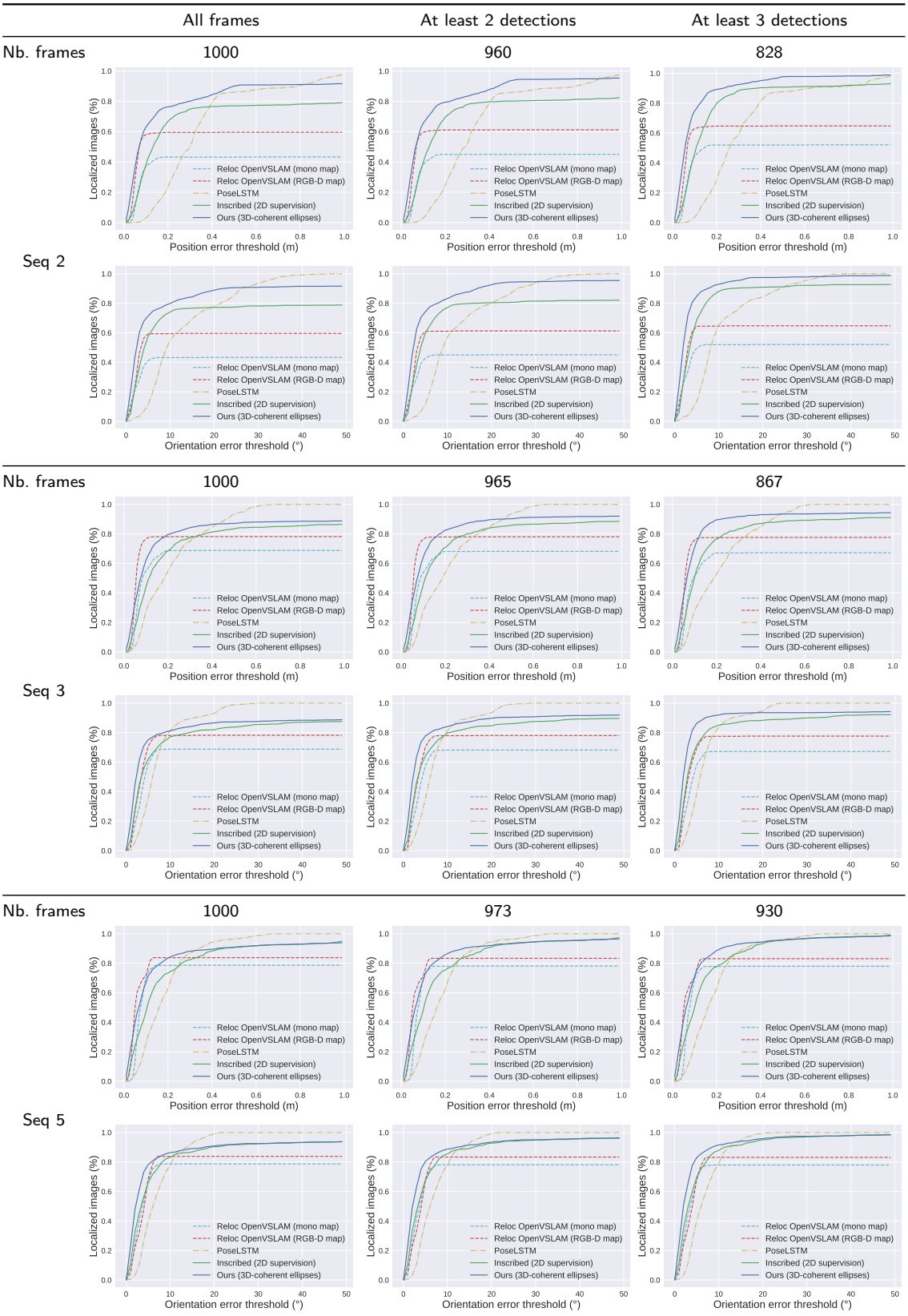


TABLE 2.5 – Proportion d'images correctement localisées en fonction d'une erreur de position ou d'orientation de la caméra sur les séquences de test de la scène *Chess*.

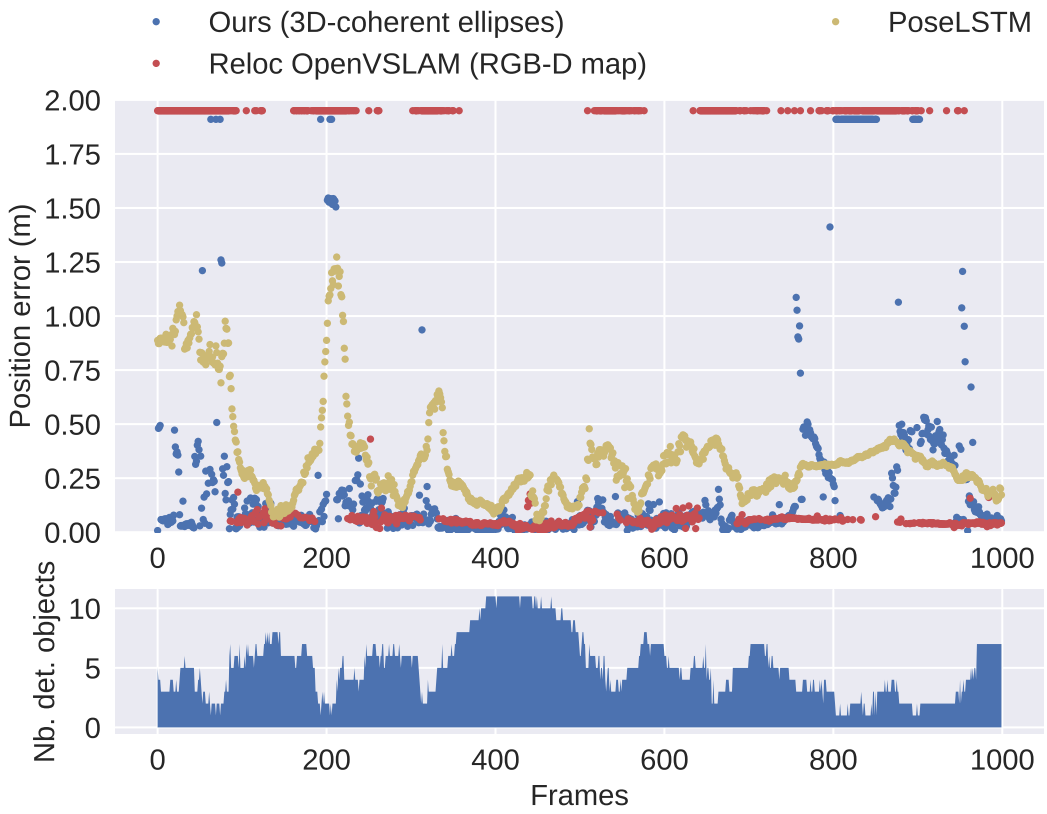


FIGURE 2.32 – Comparaison des erreurs de position de la caméra obtenues avec notre méthode par rapport à OpenVSLAM et PoseLSTM sur les images de la séquence de test 2 de la scène *Chess*. Les points situés à plus de 1.75 m signifient que l'estimation de la pose a échoué et n'a pas fourni de résultat. La courbe du bas montre le nombre d'objets détectés sur chaque image.

Ces résultats montrent clairement les avantages apportés par l'utilisation d'objets comme des balises de haut niveau pour le calcul de la pose de la caméra. Les méthodes basées points (avec la carte RGB-D) se montrent légèrement plus précises en position mais échouent bien plus fréquemment (notamment sur la séquence 2). PoseLSTM, de son côté, n'atteint pas le même niveau de précision mais a, cependant, l'avantage de toujours fournir un résultat. Enfin, l'utilisation des ellipses prédites, cohérentes aux modèles 3D, des objets surpasse nettement les performances obtenues avec les ellipses inscrites et la supervision 2D.

Enfin, la figure 2.33 montre les poses estimées pour quelques images. Il est intéressant de noter les prédictions d'ellipses multiples pour certains objets, par exemple les dossiers des chaises. Celles-ci sont dues aux multiples instances de cette classe d'objet dans la scène. Les ellipses en trait épais sont les projections des ellipsoïdes avec la pose de la caméra estimée. Les ellipses vertes sont celles utilisées dans le calcul de la pose (avec P3P ou P2E), les ellipses bleues sont considérées comme des *inliers* dans l'étape de validation par projection et les ellipses rouges ne sont pas utilisées. On peut noter que notre méthode reste robuste à des détections d'objets qui ne font pas partie du modèle de scène, telle que la détection de la chaise au fond à gauche sur la première image. L'image en bas à gauche montre la robustesse de notre méthode à un flou de mouvement. Enfin, la dernière image, en bas à droite, illustre un cas d'échec qui arrive principalement lorsque très peu d'objets sont détectés dans l'image (ici 2). En effet, le calcul de la pose à partir de deux objets seulement est particulièrement difficile car aucun autre objet ne

peut être utilisé dans la validation de la pose basée sur l'adéquation entre les détections et les reprojections.

Le temps de calcul nécessaire à notre système reste tout à fait compatible avec des tâches de localisation visuelle. Nous avons mesuré un temps de calcul d'environ 120 ms pour la détection des objets, puis 2 ms par prédiction d'ellipse et, enfin, un temps moyen autour de 100 ms pour le calcul de la pose de la caméra. Ce temps nécessaire au calcul de la pose dépend évidemment du nombre de combinaisons possibles entre les objets détectés dans l'image et ceux modélisés dans la scène, et donc, du nombre d'objets et de leur catégorie.

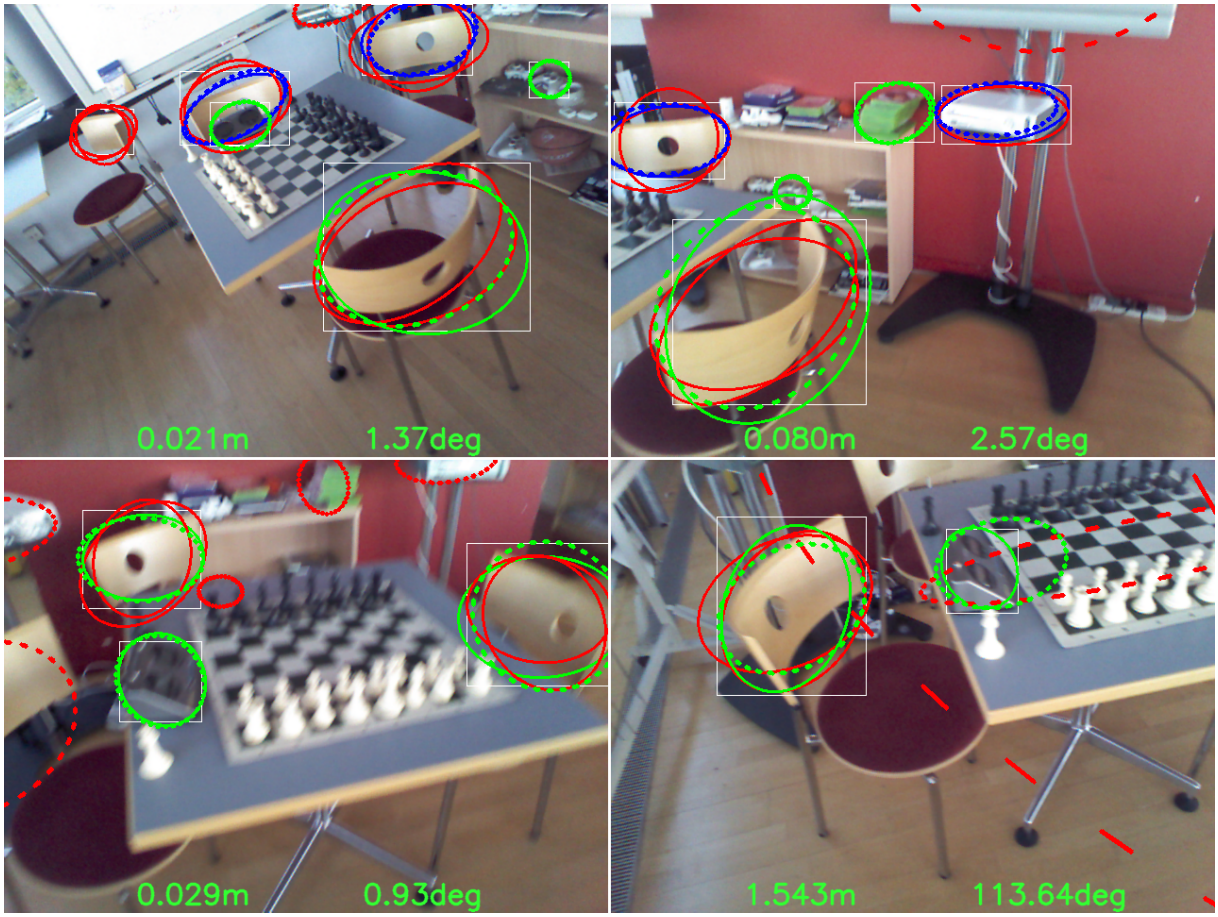


FIGURE 2.33 – Résultats obtenus sur des images des séquences de test de la scène *Chess*. Dans chacune, les erreurs de position et d'orientation de la pose estimées sont affichées en bas. Les boîtes de détection d'objets fournies par Faster R-CNN sont en blanc, les ellipses prédites sont en trait continu et les projections des ellipsoïdes avec la pose estimées sont discontinues. Les ellipses utilisées pour le calcul de la pose sont en vert, celles considérées comme cohérentes dans le processus de validation sont en bleu et les autres sont en rouge.

Résultats du calcul de la position uniquement

Dans certains cas, il est possible d'obtenir l'orientation de la caméra à partir d'un capteur externe de type gyroscope ou avec une méthode utilisant les points de fuite dans les images. Dans un tel scénario, notre système peut estimer la position de la caméra à partir d'un seul objet détecté.

Dans cette expérience, nous supposons donc que l'orientation de la caméra est connue. Cependant, plutôt que d'utiliser la valeur de vérité terrain, nous y avons ajouté un bruit aléatoire, uniforme entre $-2°$ et $2°$, sur chaque angle d'Euler de l'orientation. Cela nous permet de nous rapprocher d'un cas réel d'application et d'être plus en phase avec les valeurs bruitées que pourrait fournir un capteur de type gyroscope. Les tableaux 2.6 et 2.7 montrent, pour chaque objet, la proportion d'images dont la position a été correctement estimée par rapport à une erreur autorisée sur les deux métriques : erreur de reprojection et ADD. Ces deux métriques sont couramment utilisées pour évaluer des méthodes d'estimation de pose d'un objet. L'erreur de reprojection est calculée comme étant la distance, dans l'image, entre les points du modèle 3D de l'objet (fourni dans le jeu de données) projetés soit avec la pose estimée ou avec la pose vérité terrain. La métrique ADD se calcule en 3D et est définie comme étant la distance entre les points du modèle lorsque celui-ci est placé soit avec la pose estimée ou avec la pose vérité terrain. Des exemples d'ellipses prédites sont disponibles dans la figure 2.34. Les courbes sur la figure 2.35 montrent la proportion d'images pour lesquelles la position de la caméra a été correctement estimée en fonction d'une erreur autorisée. Ces résultats montrent un gain de précision significatif par rapport à l'utilisation des ellipses inscrites, alignées aux axes des images.

Méthodes	Ellipses inscrites [GSB19b]				Notre méthode		
Seuil	5 px	10 px	15 px	20 px	5 px	10 px	15 px
ape	95.39	100.0	100.0	100.0	**100.0**	100.0	100.0
cam	49.77	94.47	100.0	100.0	**100.0**	100.0	100.0
can	57.60	79.26	98.62	100.0	**100.0**	100.0	100.0
cat	68.20	98.62	100.0	100.0	**100.0**	100.0	100.0
driller	16.13	61.75	90.32	98.62	**96.31**	99.08	100.0
duck	89.40	100.0	100.0	100.0	**100.0**	100.0	100.0
eggbox	97.70	100.0	100.0	100.0	**100.0**	100.0	100.0
glue	54.38	88.02	95.85	99.54	**100.0**	100.0	100.0
holepunc	83.41	100.0	100.0	100.0	**100.0**	100.0	100.0
iron	17.05	51.15	78.34	93.09	**98.16**	99.54	100.0
lamp	18.43	60.37	84.79	97.24	**99.08**	100.0	100.0
phone	34.56	70.97	88.48	97.24	**99.54**	100.0	100.0

TABLE 2.6 – Proportion des positions de caméra correctement estimées pour différents seuils autorisés sur l'erreur de reprojection. Ces résultats sont obtenus sur les objets de LINEMOD et comparent notre méthode avec celle des travaux précédents utilisant des ellipses inscrites dans les boîtes de détection.

Méthodes	Ellipses inscrites [GSB19b]			Notre méthode		
Seuil (% du diamètre)	10%	15%	25%	10%	15%	25%
ape	18.43	35.94	56.68	**70.51**	85.25	92.63
cam	34.10	56.68	84.33	**91.24**	98.16	99.08
can	12.90	18.43	31.34	**93.09**	97.70	98.16
cat	26.27	37.33	55.30	**76.04**	92.63	96.77
driller	42.86	57.14	76.04	**85.71**	92.17	97.24
duck	31.34	47.00	67.28	**66.82**	83.87	92.17
eggbox	16.59	22.58	40.09	**88.48**	94.47	97.24
glue	11.98	23.04	32.72	**70.51**	82.95	92.63
holepunc	12.90	20.74	30.88	**86.64**	92.63	97.24
iron	16.59	25.81	40.55	**91.71**	98.62	99.54
lamp	23.04	35.48	58.99	**96.77**	100.0	100.0
phone	22.12	29.03	42.86	**92.63**	98.62	99.54

TABLE 2.7 – Proportion des positions de caméra correctement estimées pour différents seuils autorisés sur l'erreur ADD. Ces résultats sont obtenus sur les objets de LINEMOD et comparent notre méthode avec celle des travaux précédents utilisant des ellipses inscrites dans les boîtes de détection.

FIGURE 2.34 – Exemples d'ellipses prédites pour des images de test de LINEMOD. Les ellipses prédites sont en vert et celles de vérité terrain en rouge. Les erreurs de positions obtenues pour la caméra sont indiquées en haut à gauche.

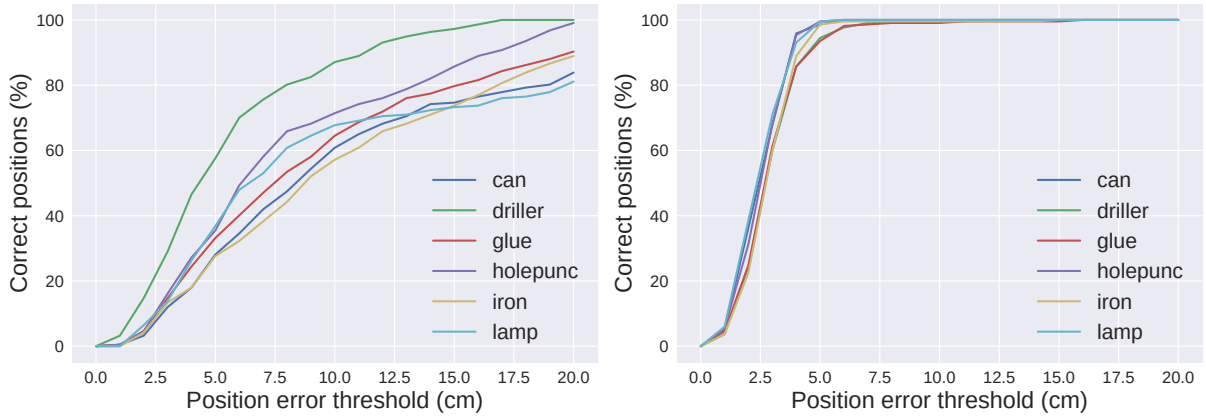


FIGURE 2.35 – Comparaison des proportions de positions de caméra correctement estimées en utilisant les ellipses inscrites dans les boîtes de détection (à gauche) par rapport à celles obtenues avec nos ellipses prédites (à droite). Les courbes correspondent à différents objets du jeu de données LINEMOD.

Enfin, nous avons également comparé nos résultats à d'autres méthodes existantes d'estimation de pose 6D d'un objet dans le tableau 2.8. Cette comparaison est donnée à titre indicatif car, contrairement à notre travail, ces méthodes estiment la pose complète (orientation et position) d'un seul objet mais requiert des modèles 3D précis des objets, au moins pour l'entraînement.

Objet	BB8 [RL17]	U-D 6D [Bra+16]	Tekin [TSF18]	Pix2Pose [PPV19]	Ell. inscrites [GSB19b]	Notre méthode
ape	27.9	33.2	21.6	58.1	18.43	**70.51**
cam	40.1	38.4	36.6	60.9	34.10	**91.24**
can	48.1	62.9	68.8	84.4	12.90	**93.09**
cat	45.2	42.7	41.8	65.0	26.27	**76.04**
driller	58.6	61.9	63.5	76.3	42.86	**85.71**
duck	32.8	30.2	27.2	43.58	31.34	**66.82**
eggbox	40.0	49.9	69.6	**96.8**	16.59	88.48
glue	27.0	31.2	80.0	**79.4**	11.98	70.51
holepunc	42.4	52.8	42.6	74.8	12.90	**86.64**
iron	67.0	80.0	75.0	83.4	16.59	**91.71**
lamp	39.9	67.0	71.1	82.0	23.04	**96.77**
phone	35.2	38.1	47.7	45.0	22.12	**92.63**

TABLE 2.8 – Comparaison des résultats d'estimation de la position de la caméra obtenus pour différentes méthodes. Les valeurs représentent le pourcentage de positions correctement estimées pour un seuil d'erreur ADD maximale de 10% du diamètre de l'objet.

2.4.5 Analyses

Dans cette section nous analysons différentes caractéristiques de notre méthode de calcul de pose de caméra basé objet.

Robustesse à de nouveaux points de vue

WatchPose Nous avons tout d'abord utilisé le jeu de données WatchPose, qui illustre bien un cas réel d'application et permet de varier la distance de la caméra entre l'apprentissage et le test. La scène utilisée montre un objet de type « valve » vu à des distances proches et éloignées. Nous avons évalué deux scénarios : un cas simple (nommé *Facile)* dans lequel des sous-ensembles d'images proches et éloignées de l'objet sont utilisés pour l'apprentissage et le test et un cas compliqué (nommé *Difficile*) pour lequel l'entraînement a été réalisé uniquement sur des vues proches de l'objet et le test sur des vues éloignées.

Des prédictions d'ellipse sont disponibles sur la figure 2.38 et le tableau 2.9 montre bien le bénéfice apporté par l'utilisation des ellipses prédites, cohérentes avec la 3D, plutôt que les ellipses inscrites dans les boîtes de détection. Une bonne généralisation à des points de vue avec un éloignement différent de ceux d'entraînement peut être notée dans le cas *Difficile*

Scénario	Méthode	Erreur médiane (mm)	Seuil d'erreur		
			5 cm	10 cm	15 cm
Facile	Ellipses inscrites	84.11	27.87	57.38	77.05
	Notre méthode	**26.63**	**77.27**	**97.72**	**100.0**
Difficile	Ellipses inscrites	120.31	14.61	38.20	66.29
	Notre méthode	**54.12**	**40.74**	**81.48**	**88.88**

TABLE 2.9 – Comparaison des erreurs de position médianes obtenues avec les ellipses inscrites et avec notre méthode, pour les deux scénarios testés : *Facile* (apprentissage et test à des distances mixtes) et *Difficile* (apprentissage uniquement sur des images proches et test sur des images plus éloignées). Les proportions de positions correctement estimées sont également indiquées.

Scène virtuelle Afin d'évaluer la robustesse de notre de calcul de pose complète à des changements de points de vue importants entre les images d'apprentissage et les images de test, nous avons créé une scène virtuelle à partir des objets de la base de données YCB. Les rendus ont été générés avec Blender [Com18], ce qui nous permet d'avoir un contrôle total sur les points de vues utilisés en entraînement et en test. La figure 2.36 montre la scène créée ainsi que les modèles ellipsoïdaux reconstruits pour les objets. Les images d'entraînement ont été générées à partir de trois trajectoires de caméra, situées à environ 4 m du centre de la scène, pour un total de 192 images. Les images de test ont été générées à partir de points de vue plus divers et plus distants de la scène (jusqu'à 8 m pour le cas 3). Les trajectoires utilisées en entraînement ainsi que celles de test sont montrées dans la figure 2.37.

Les résultats obtenus sont disponibles dans le tableau 2.10, les poses de caméra estimées sont visibles dans les figures 2.39, 2.41 et 2.43, et quelques exemples de prédiction d'ellipse sont illustrés dans la figure 2.45. Notre méthode atteint une très bonne précision sur les cas 1 et 2 avec une erreur de position d'environ 5 cm et une erreur d'orientation d'environ 5°. Les erreurs sont plus importantes pour le cas 3 mais restent acceptables étant donné la grande distance entre la caméra et la scène. Les cas 1 et 2 montrent également les avantages de l'utilisation des ellipses prédites. Dans le cas 3, les ellipses inscrites sont légèrement meilleures, ce qui peut s'expliquer

par le fait qu'à de telles distances, les objets apparaissent très petits dans l'image, ce qui réduit l'intérêt de la prédiction des ellipses cohérentes avec les modèles 3D des objets.

FIGURE 2.36 – Scène virtuelle créée avec les modèles ellipsoïdaux reconstruits des objets.

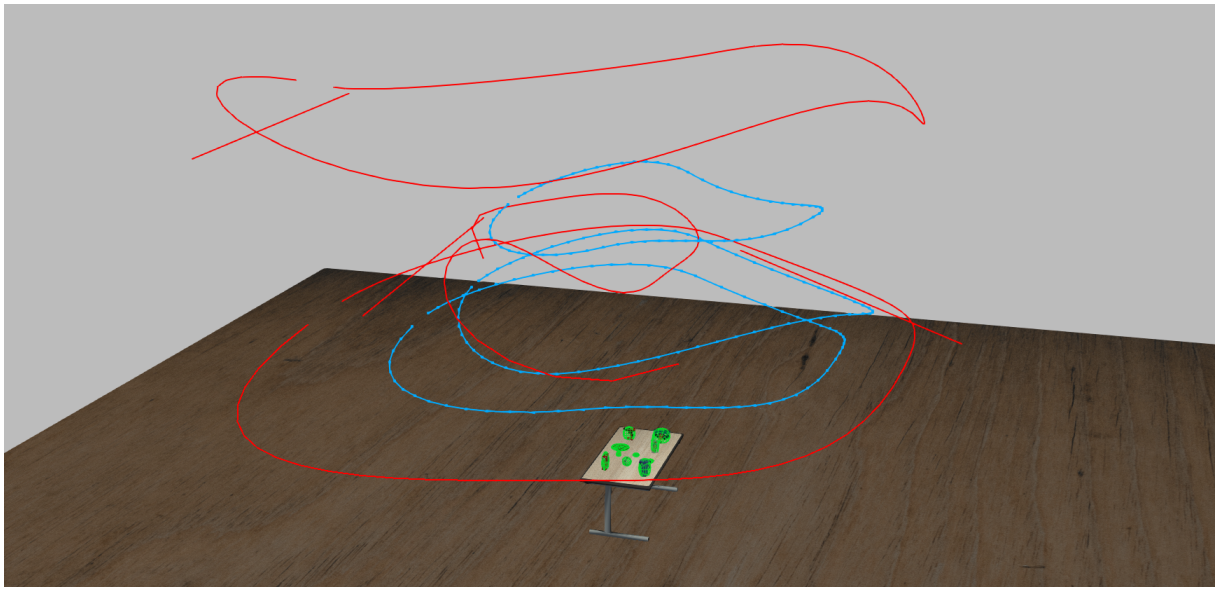


FIGURE 2.37 – Trajectoires de la caméra utilisées pour générer les images d'entraînement, en bleu, et les images de test, en rouge.

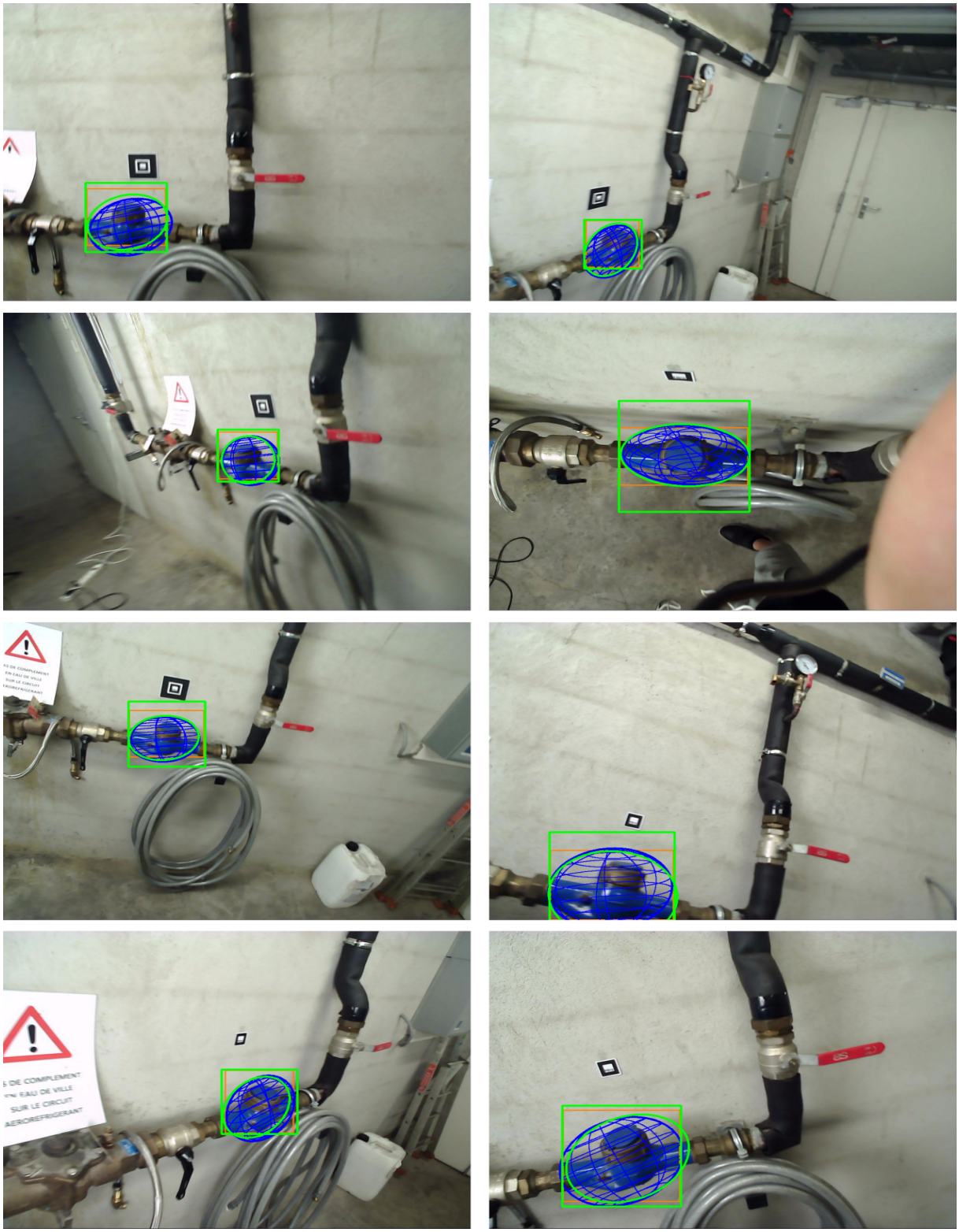


FIGURE 2.38 – Résultats obtenus sur les images de test du jeu de données WatchPose pour le calcul de la position de la caméra. La boîte de détection de l'objet, fournie par Faster R-CNN est en orange et sa version carrée, utilisée pour extraire la sous-image de l'objet qui est passée au réseau de prédiction d'ellipse, est en vert. L'ellipse prédite est également en vert. L'ellipsoïde bleu correspond à sa reprojection dans l'image avec la position estimée de la caméra.

Nous présentons également les résultats obtenus avec PoseLSTM, la méthode régression directe de pose, dans les figures 2.40, 2.42 et 2.44. Contrairement à notre module de prédiction d'ellipse, cette méthode a beaucoup plus de difficultés à généraliser aux nouveaux points de vue des images de test. Une différence notoire qui pourrait expliquer la meilleure généralisation de notre prédiction d'ellipse vient de la vision limitée qu'a le réseau sur l'image d'entrée. En effet, il ne prend en entrée qu'une sous-image locale autour de l'objet, ce qui limite beaucoup plus les changements d'apparence lorsque le point de vue varie.

D'autres exemples de localisation dans cette scène virtuelle sont disponibles à l'adresse : https://zinsmatt.github.io/these/#3DV_IJCV.

Scénario	Erreur	PoseLSTM [Wal+17]	Ellipses inscrites [GSB20]	Notre méthode
Test 1	position	0.645	0.057	**0.048**
	orientation	6.21	0.818	**0.592**
Test 2	position	1.93	0.064	**0.062**
	orientation	12.43	0.652	**0.538**
Test 3	position	3.53	**0.096**	0.118
	orientation	23.46	**0.884**	0.977

TABLE 2.10 – Comparaison des erreurs de position moyennes obtenues par notre méthode, les ellipses inscrites et PoseLSTM sur les images de test de la scène virtuelle.

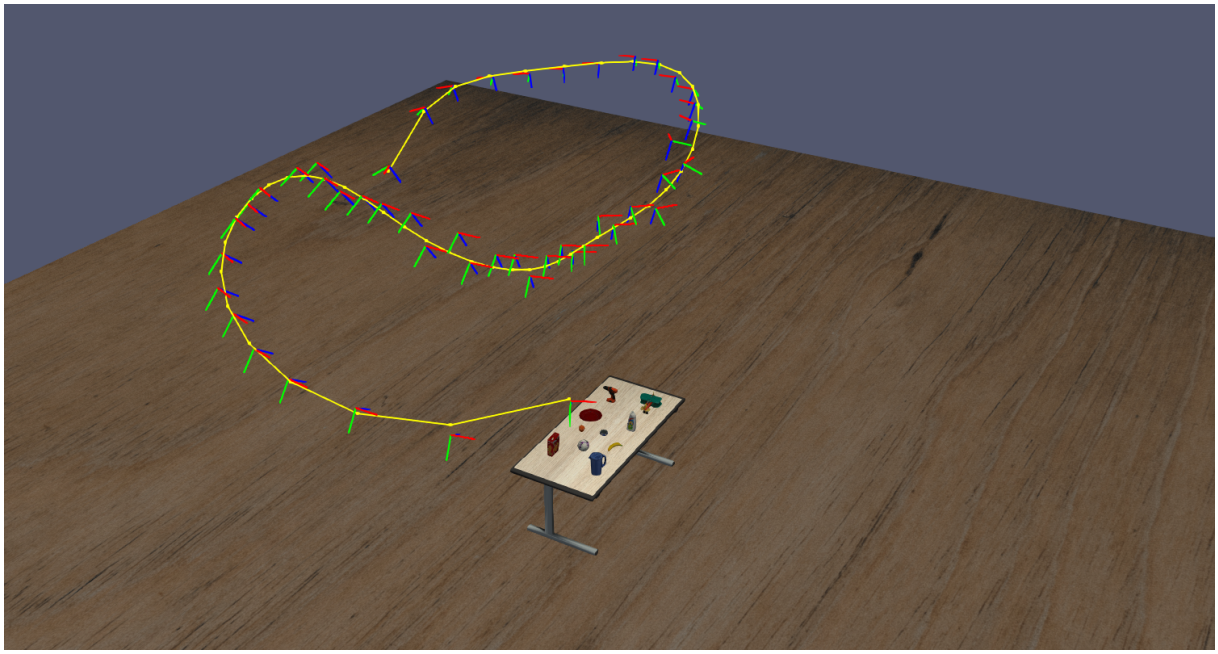


FIGURE 2.39 – Poses de la caméra estimées avec notre méthode et trajectoire de vérité terrain en jaune. Erreur de position médiane : 7.63 cm. Erreur d'orientation médiane : 0.79°.

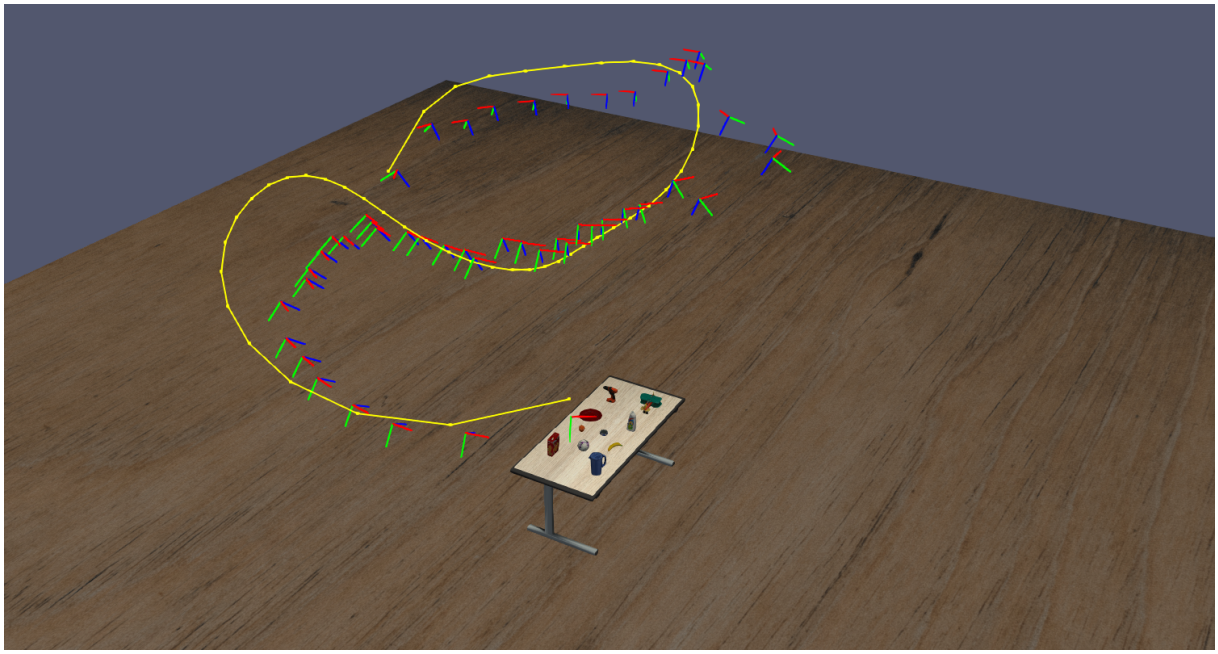


FIGURE 2.40 – Poses de la caméra estimées avec PoseLSTM et trajectoire de vérité terrain en jaune. Erreur de position médiane : 40.13 cm. Erreur d'orientation médiane : 3.04°.

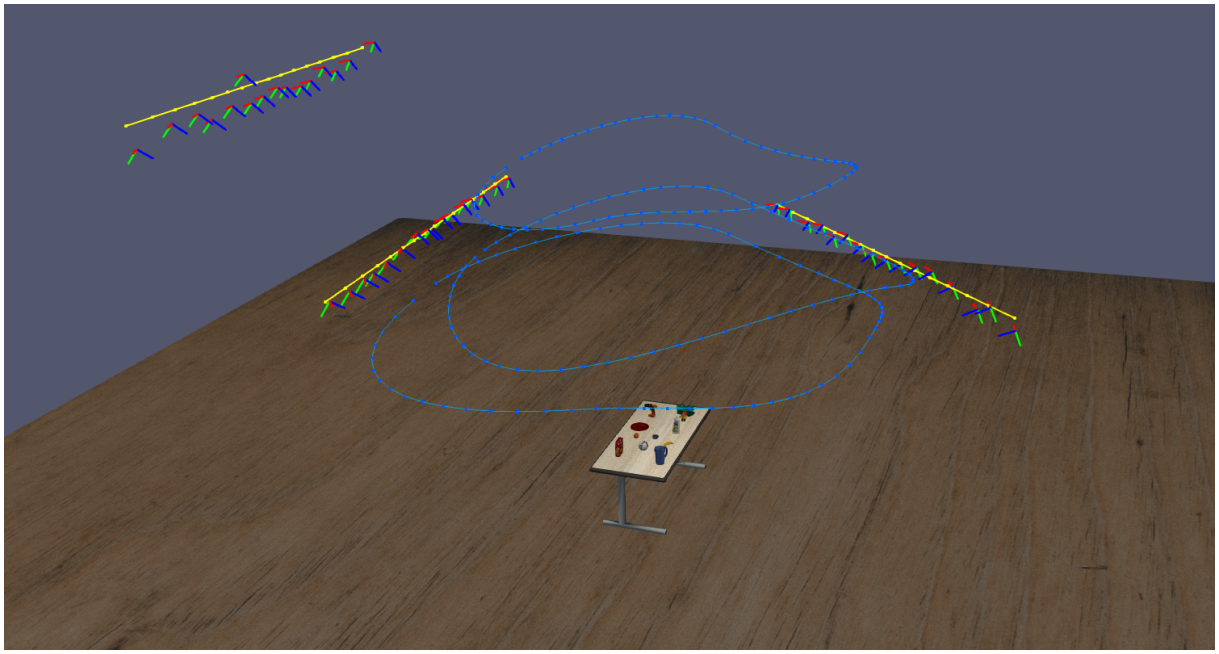


FIGURE 2.41 – Poses de la caméra estimées avec notre méthode et trajectoires de vérité terrain en jaune. Erreur de position médiane : 8.16 cm. Erreur d'orientation médiane : 0.75°.

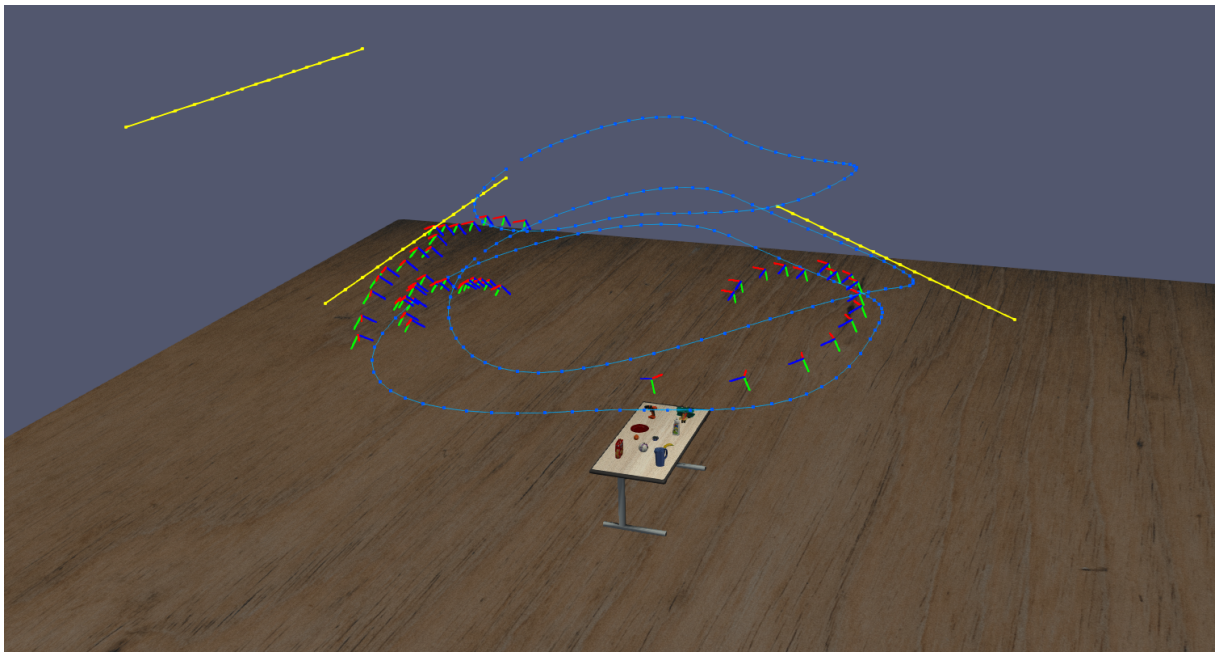


FIGURE 2.42 – Poses de la caméra estimées avec PoseLSTM et trajectoires de vérité terrain en jaune. Les trajectoires utilisées pour l'entraînement sont rappelées en bleu. Erreur de position médiane : 111.65 cm. Erreur d'orientation médiane : 6.72°.

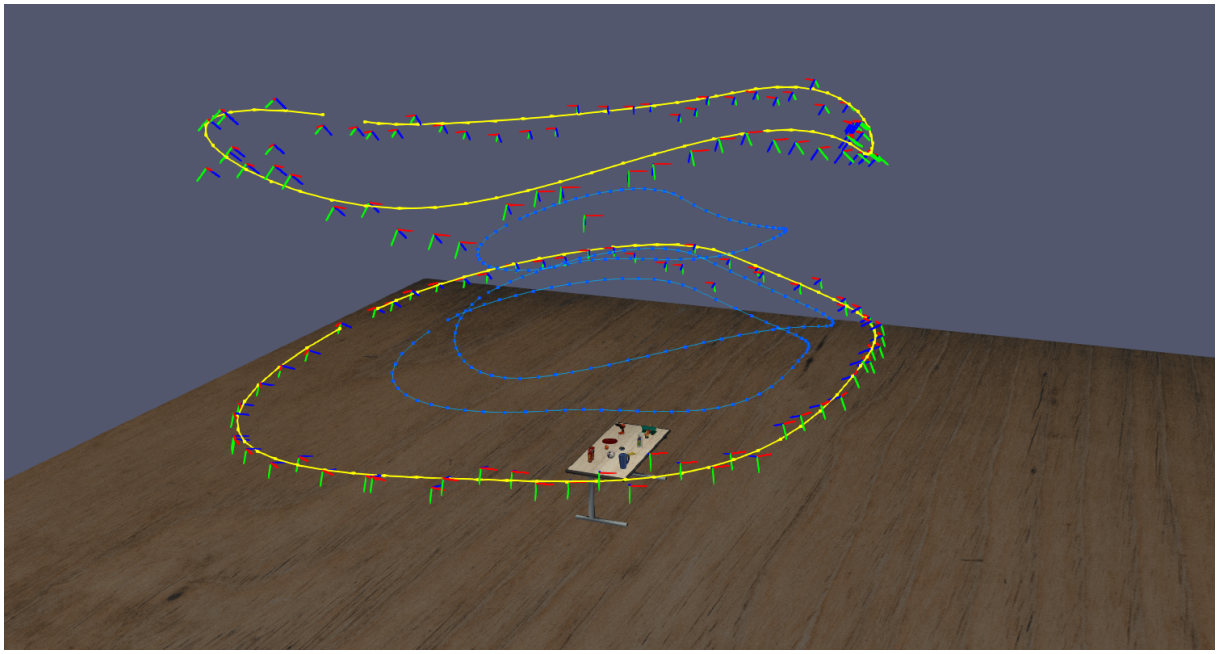


FIGURE 2.43 – Poses de la caméra estimées avec notre méthode et trajectoires de vérité terrain en jaune. Les trajectoires utilisées pour l'entraînement sont rappelées en bleu. Erreur de position médiane : 14.5 cm. Erreur d'orientation médiane : 1.09°.

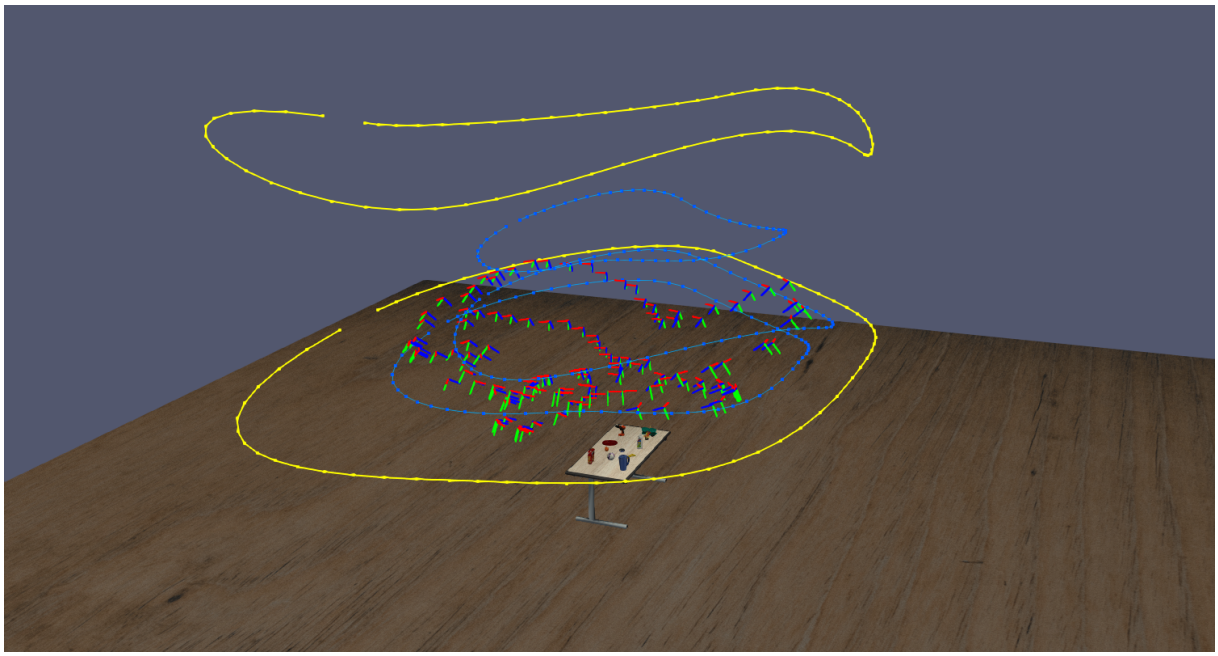


FIGURE 2.44 – Poses de la caméra estimées avec PoseLSTM et trajectoires de vérité terrain en jaune. Les trajectoires utilisées pour l'entraînement sont rappelées en bleu. Erreur de position médiane : 422.30 cm. Erreur d'orientation médiane : 15.82°.

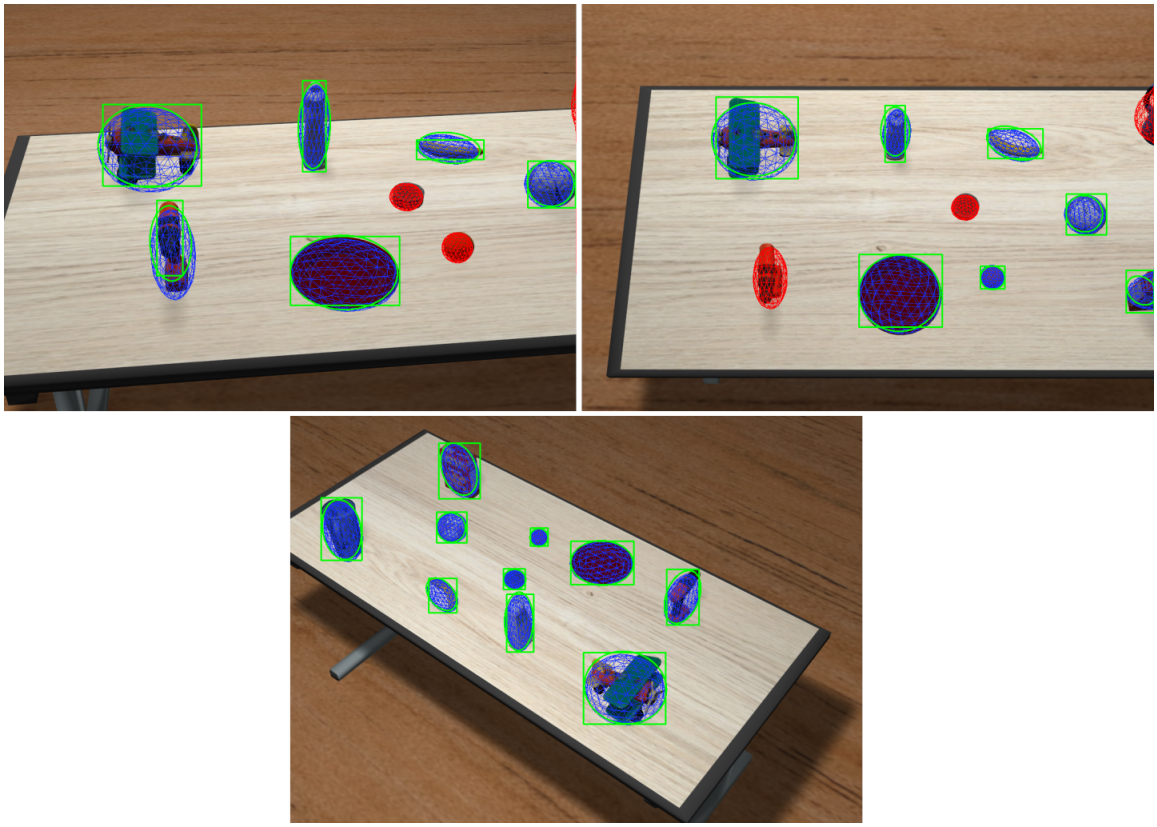


FIGURE 2.45 – Résultats d'estimation de pose de caméra pour la scène virtuelle. Les détections d'objets et les ellipses prédites sont en vert. Les ellipsoïdes sont projetés dans l'image en utilisant la pose estimée. Les ellipsoïdes bleus correspondent aux objets qui ont été associés à des détections d'objets dans l'image et les rouges à des objets non détectés.

Influence du modèle ellipsoïdal reconstruit

Une caractéristique très intéressante de notre méthode est qu'elle ne dépend pas des ellipsoïdes utilisés pour modéliser les objets. Cette indépendance est essentielle car la modélisation d'objet que l'on utilise est approximative et est généralement obtenue par simple reconstruction à partir de quelques points de vue. Un choix différent de points de vue génèrerait d'autres ellipsoïdes. Il est donc important que notre méthode soit capable d'apprendre la projection de plus ou moins n'importe quel ellipsoïde et pas seulement un ellipsoïde parfaitement ajusté aux contours de l'objet en 3D. Nous avons vérifié cette propriété en répétant nos expériences de calcul de la position ou de la pose complète de la caméra en utilisant trois modèles ellipsoïdaux différents des objets. Dans un premier temps, le calcul de la position a été évalué sur l'objet *perceuse* de LINEMOD en utilisant les trois modèles montrés sur la figure 2.46. Puis, le calcul de la pose complète a été évalué sur trois modélisations différentes des objets d'une scène de T-LESS (voir figure 2.47). Les nouvelles modélisations des objets ont été obtenues en déformant manuellement les ellipsoïdes reconstruits. Le réseau de prédiction d'ellipse a ensuite été ré-entraîné pour chaque modèle. Les résultats, disponibles dans les tableaux 2.11 et 2.12, montrent une précision similaire pour chacun des modèles de scène. Cela signifie bien que notre méthode n'est pas fortement liée à la précision de l'ajustement entre les modèles ellipsoïdaux et les objets réels en 3D. Ainsi, la variabilité de la reconstruction d'un ellipsoïde à partir de quelques vues n'est pas un problème.

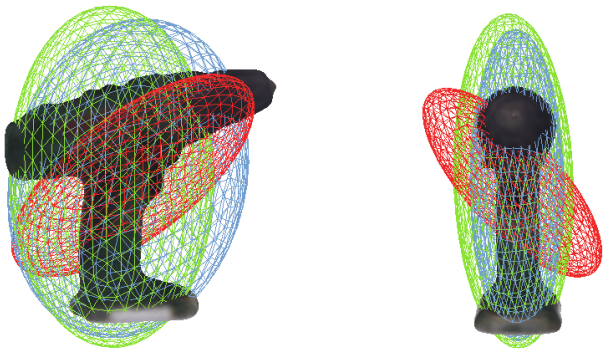


FIGURE 2.46 – Trois ellipsoides différents utilisés pour modéliser la perceuse. L'ellipsoïde 1 du tableau 2.11 est en bleu, l'ellipsoïde 2 en vert et l'ellipsoïde 3 en rouge.

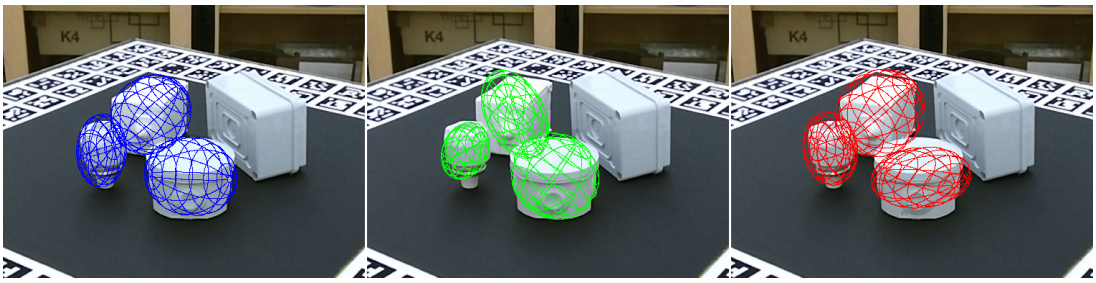


FIGURE 2.47 – Trois reconstructions ellipsoïdales différentes des objets de la scène de T-LESS. Les ellipsoïdes 1 dans le tableau 2.12 sont en bleu, les ellipsoïdes 2 en vert et les ellipsoïdes 3 en rouge.

Métrique	Erreur de reprojection	Erreur de position	ADD
Seuil	5 pixels	5 cm	10% du diam.
Ellipsoïde 1 (bleu)	96.31	95.85	85.71
Ellipsoïde 2 (vert)	96.31	96.31	85.25
Ellipsoïde 3 (rouge)	98.62	95.85	84.79

TABLE 2.11 – Résultats de l'estimation de la position de la caméra pour les trois différents modèles ellipsoïdaux de la perceuse, illustrés sur la figure 2.46.

	Erreur médiane de position (en cm)	Erreur médiane d'orientation (en °)
Ellipsoïdes 1 (bleu)	3.17	2.37
Ellipsoïdes 2 (vert)	3.10	2.44
Ellipsoïdes 3 (rouge)	2.87	2.15
Ellipses inscrites	4.81	3.58

TABLE 2.12 – Erreurs médianes de position et d'orientation de la pose estimée de la caméra pour les trois différents modèles de scène reconstruits et illustrés sur la figure 2.47.

Influence du bruit de détection

Etant donné que notre système est composé d'une étape de détection des objets puis de la prédiction des ellipses, nous avons voulu analyser l'influence d'une erreur de détection sur la prédiction de l'ellipse. Pour cela, nous avons simulé des détections d'objets imprécises sur une scène du jeu de données T-LESS, en ajoutant un bruit aléatoire de translation sur deux coins des boîtes de détections d'objets. Les sous-images extraites de l'objet et passées en entrée des réseaux de prédiction d'ellipses sont donc déformées et décalées. La scène est composée de six objets de type industriel. La figure 2.48 montre ces détections bruitées en vert et compare les ellipses inscrites (en vert sur la gauche) avec les ellipses prédites (en vert sur la droite). Les détections et les ellipses vérité terrain sont respectivement en bleu et rouge.

D'un côté, il est évident que des boîtes bruitées ont un impact direct sur les ellipses inscrites dans ces boîtes. De l'autre, les ellipses prédites semblent relativement robustes à ce bruit et sont capables de rester proches de la vérité terrain. Les courbes sur la figure 2.49 confirment ces résultats et montrent l'influence d'un bruit de détection croissant sur la pose estimée de la caméra. Les erreurs de pose augmentent bien plus rapidement en utilisant les ellipses inscrites, ce qui montre une certaine robustesse des ellipses prédites par rapport à un bruit sur les boîtes de détection. Cette robustesse est principalement obtenue grâce à notre stratégie d'augmentation des données pendant l'apprentissage, qui décale de manière aléatoire les sous-images de l'objet données en entrée du réseau (ce qui revient à décaler la boîte de détection d'un objet).

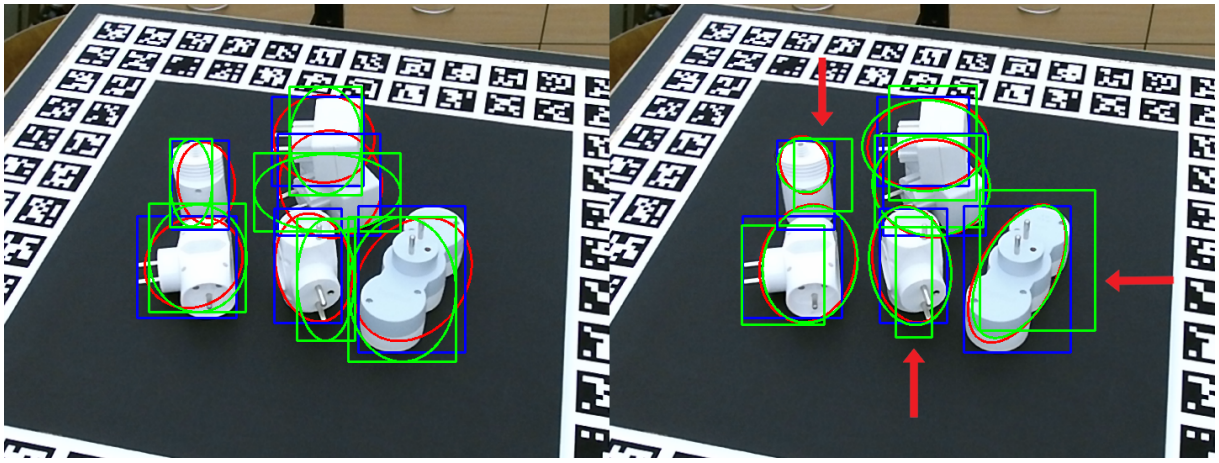


FIGURE 2.48 – Illustration de la robustesse de la prédiction d'ellipse au bruit de détection des objets. Les boîtes de détection bruitées en translation et taille sont en vert. Celles de vérité terrain sont en bleu. Les ellipses de vérité terrain sont en rouge. À gauche, les ellipses inscrites dans les boîtes de détection (en vert) sont utilisées. À droite, les ellipses prédites (en vert) sont utilisées.

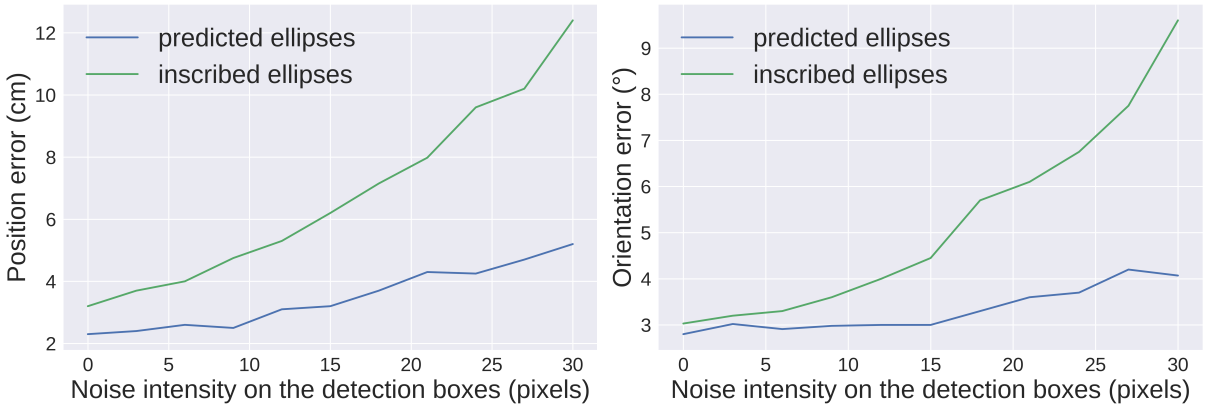


FIGURE 2.49 – Comparaison des erreurs de position (à gauche) et d'orientation (à droite), obtenues pour différents niveaux de bruit sur les détections des objets, en utilisant les ellipses prédites ou inscrites.

Comparaison entre P3P et P2E pour le calcul de pose

Nous avons également comparé notre méthode de calcul de la pose de la caméra à celle utilisée dans le travail précédent de Gaudillière et al. [GSB20]. D'un côté, notre méthode se base sur un algorithme P3P qui fournit quatre hypothèses de poses à partir de trois objets et l'utilise à l'intérieur d'une boucle considérant donc tous les triplets possibles d'associations ellipse-ellipsoïde. De l'autre, P2E utilise deux objets et est donc utilisé à l'intérieur d'une boucle qui passe en revue toutes les paires possibles d'associations ellipse-ellipsoïde. Contrairement à P3P qui fournit quatre hypothèses de poses, P2E nécessite une recherche exhaustive d'un paramètre angulaire et fournit donc un bien plus grand nombre d'hypothèses de poses, en fonction de la résolution de recherche.

Nous avons comparé leur performance pour l'estimation de la pose de la caméra sur les séquences de test de la scène *Chess*. Les résultats sont disponibles dans la figure 2.50, en fonction du nombre d'objets qui sont utilisés dans les images. Dans les deux cas les mêmes détections d'objets et prédictions d'ellipses sont utilisées. Les résultats montrent un avantage assez net de P3P à partir de quatre objets détectés. Dans le cas à trois objets, la différence est moins évidente mais P3P offre, tout de même, une meilleure précision. P2E se montre seulement légèrement plus précis en position pour quelques images avec des erreurs déjà supérieures à 30 cm. Dans le cas où seulement deux objets sont détectés dans l'image, seul P2E peut être utilisé. Par ailleurs, nous avons également mesuré un gain de vitesse de calcul considérable avec l'utilisation de P3P. La figure 2.51 montre les durées du calcul de pose mesurées pour les images de test avec plus de deux objets détectés.

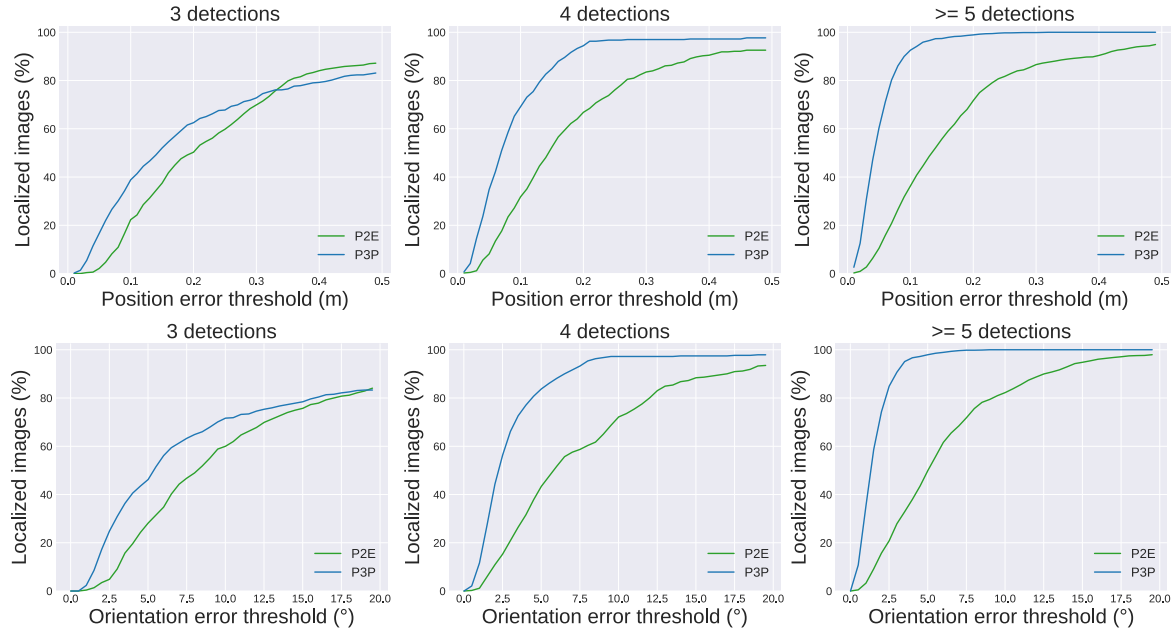


FIGURE 2.50 – Comparaison des pourcentages d'images correctement localisée en fonction d'une erreur de position (en haut) ou d'orientation (en bas), obtenus en utilisant P2E ou P3P, sur les images de test de la scène *Chess*. Les résultats sont séparés en fonction du nombre d'objets détectés dans les images.

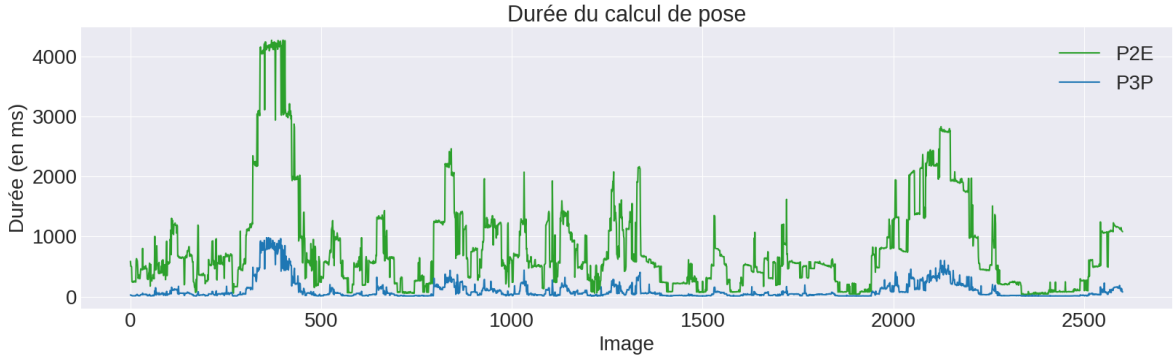


FIGURE 2.51 – Comparaison de la durée du calcul de pose réalisé avec P2E ou P3P sur les images de test de la scène *Chess* avec plus de deux objets détectés.

2.5 Approche jointe de détection d'objet et prédiction d'ellipse

Nous avons montré les bons résultats obtenus grâce à la prédiction d'ellipses cohérentes aux modèles ellipsoïdaux des objets. Cette méthode nécessite néanmoins l'entraînement et l'exécution de plusieurs réseaux de manière séparée, le réseau de détection des objets et les réseaux de prédiction des ellipses par objet, ce qui peut rendre son déploiement relativement lourd en pratique. En effet, comme expliqué dans la section 2.3.4, la détection des objets est faite au niveau de leur catégorie alors que la prédiction d'ellipse est faite par instance. Une détection d'un objet doit donc être couplée avec plusieurs prédictions d'ellipses lorsque des instances multiples de cette catégorie d'objet sont présentes dans la scène.

Nous proposons ici de détecter des instances d'objets et de fusionner leur détection avec la prédiction d'ellipses dans un seul réseau.

2.5.1 Prédiction d'ellipse dans YOLO

Nous avons modifié le réseau YOLO pour détecter les instances des objets de la scène et prédire des ellipses cohérentes avec les projections de leurs modèles ellipsoïdaux. Nous avons choisi YOLO car ses évolutions [RDGF16], [RF17], [RF18] et [BWL20] sont disponibles publiquement et semblent assez facilement adaptables. Nous nous sommes basés sur la cinquième version qui est disponible à l'adresse `github.com/ultralytics/yolov5`.

L'idée générale de YOLO est de diviser l'image d'entrée en cellules et de régresser, pour chacune d'entre elles, des coordonnées et tailles de boîte, des indices de confiance et des probabilités de classe. À noter que la prédiction d'une ellipse est équivalente à celle d'une boîte, car elles partagent les mêmes paramètres : centre, hauteur et largeur. Pour obtenir des ellipses orientées, nous avons ajouté une valeur d'angle pour chaque boîte en sortie du réseau.

La fonction de coût totale utilisée pour entraîner le réseau, combine différents termes pour prendre en compte à la fois la géométrie des boîtes prédites, les classes des objets détectés et la présence ou non d'un objet dans une cellule. La partie qui nous intéresse est celle liée à la géométrie des boîtes prédites. Initialement, ce coût est calculé par *Complete IoU* (CIoU) [Zhe+20], qui prend en compte trois facteurs géométriques : le recouvrement, la distance entre les centres et le rapport hauteur/largeur des boîtes. Ce coût est calculé par

$$CIoU(b, b_{gt}) = 1 - IoU(b, b_{gt}) + \frac{\|c - c_{gt}\|_2^2}{d^2} + \alpha v, \tag{2.16}$$

avec

$$v = \frac{4}{\pi^2}(\arctan\frac{w_{gt}}{h_{gt}} - \arctan\frac{w}{h})^2, \tag{2.17}$$

et

$$\alpha = \frac{v}{1 - IoU(b, b_{gt}) + v}, \tag{2.18}$$

où b et b_{gt} sont respectivement la boîte prédite et celle de vérité terrain, c et c_{gt} sont leurs centres respectifs, w et w_{gt} sont leurs largeurs, h et h_{gt} sont leurs hauteurs, d est la longueur de la diagonale de la plus petite boîte englobant b et b_{gt}, α est un terme de pondération et v mesure la cohérence du rapport hauteur/largeur des deux boîtes.

Nous avons créé deux versions différentes de la fonction de coût géométrique. Les autres termes de classification et de présence d'objet sont inchangés. YOLO prédit des boîtes de manière relative à un ensemble de boîtes d'ancrage prédéfinies qui sont alignées avec les axes de l'image. Dans une première version, nous avons donc conservé CIoU pour comparer les centres et les tailles des ellipses/boîtes alignées avec les axes de l'image et avons simplement ajouté un terme d'erreur quadratique sur l'angle. Notre fonction de coût entre deux ellipses $\mathcal{E}$ et $\mathcal{E}_{gt}$ devient

$$\mathcal{L}_{geo}(\mathcal{E}, \mathcal{E}_{gt}) = CIoU(b, b_{gt}) + \lambda(\sin\theta - \sin\theta_{gt})^2, \tag{2.19}$$

où b et b_{gt} sont les boîtes construites à partir des ellipses, comme décrit précédemment, et λ est un terme de pondération que nous avons fixé à 1.0 dans nos expériences. L'entraînement avec cette fonction de coût est nommé *Jointe* dans la suite.

Dans une seconde version, nous avons remplacé la partie géométrique de la fonction de coût par celle basée sur des ensembles de niveaux que nous avons proposée dans la section 2.2.5. L'entraînement avec cette fonction de coût est nommé *JointeLS* dans la suite.

2.5.2 Résultats de l'approche jointe de détection et prédiction

Nous avons évalué cette approche jointe de détection et prédiction d'ellipse sur la scène *Chess*. Nous avons initialisé notre réseau à partir de poids pré-entraînés sur la base de données COCO [Lin+14a] et nous l'avons ajusté, pendant 50 époques, pour détecter les instances d'objets présents dans la scène et prédire des ellipses cohérentes avec leurs modèles ellipsoïdaux. Les annotations elliptiques des instances d'objets sont obtenues par projection de leurs modèles ellipsoïdaux dans les images d'entraînement dont les poses sont connues. La procédure utilisée est la même que celle détaillée dans la section 2.3.3.

Des résultats de pose estimée avec cette approche sont illustrés sur la figure 2.52. Par construction, cette méthode ne prédit qu'une seule ellipse pour chaque instance d'un même type d'objet (les dossiers de chaise ou les consoles de jeu), contrairement à l'approche disjointe illustrée dans la figure 2.33. La figure 2.53 compare la pose calculée avec cette approche par rapport à la méthode disjointe initiale avec un réseau par objet. On peut noter que les résultats obtenus par *Jointe* se rapprochent fortement des résultats précédents. Ils sont seulement très légèrement inférieurs. On peut en déduire que le réseau est capable de distinguer les différentes instances d'un même type d'objet, en s'appuyant sur des différences visuelles et probablement aussi sur l'environnement proche autour de ces objets.

La seconde version *JointeLS* n'atteint pas les mêmes performances. Une explication pourrait venir du fait que la fonction de coût par ensembles de niveaux diffère plus fortement de celle utilisée lors du pré-entraînement et son équilibrage avec les autres termes de classification et de présence d'objet peut être moins adapté.

FIGURE 2.52 – Illustration de la pose calculée sur deux images de test de la scène *Chess* avec l'approche jointe de détection d'objets et prédiction d'ellipses. Les ellipses prédites sont en trait continu et les projections des ellipsoïdes avec la pose estimée sont discontinues. Les ellipses utilisées pour le calcul de la pose sont en vert, celles considérées comme cohérentes dans le processus de validation sont en bleu et les autres sont en rouge.

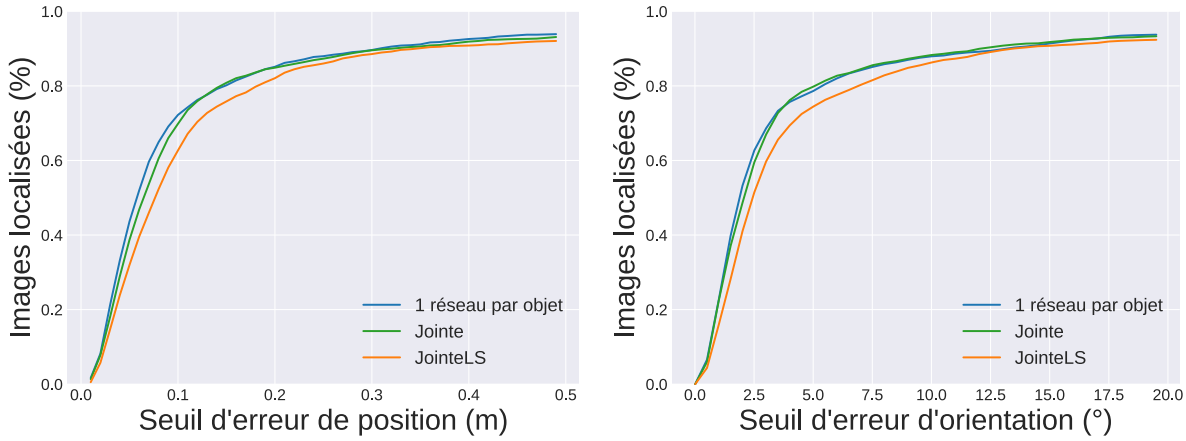


FIGURE 2.53 – Comparaison des pourcentages d'images correctement localisées en fonction d'une erreur de position (à gauche) ou d'orientation (à droite), obtenus avec la méthode à un réseau par objet décrite précédemment et avec les deux versions jointes. Ces résultats sont calculés sur l'ensemble des images de test de la scène *Chess* avec au moins trois objets détectés.

La figure 2.54 compare les temps nécessaires pour le calcul de la pose de la caméra avec la méthode à un réseau par objet et avec l'approche jointe. On peut observer un gain de vitesse considérable avec l'approche jointe qui s'explique par le fait qu'une détection des instances d'objets fournit directement des correspondances 2D-3D. Cela permet d'éviter un grand nombre de combinaisons entre les détections dans l'image et les objets dans la carte, nécessaires pour établir les correspondances 2D-3D avec l'approche disjointe et qui ne pouvaient être contraintes que par les catégories des objets.

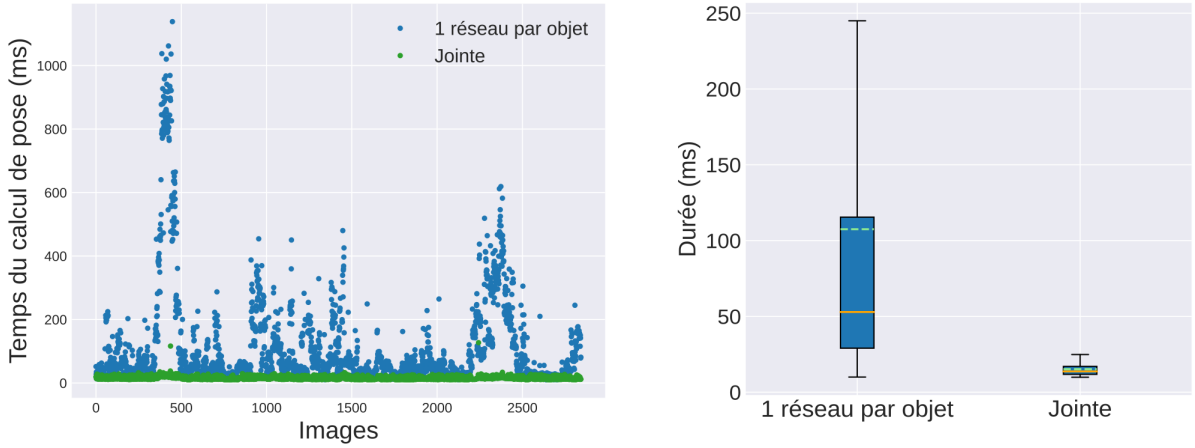


FIGURE 2.54 – Comparaison du temps de calcul nécessaire pour estimer la pose de la caméra entre la méthode initiale avec un réseau par objet et l'approche *Jointe*. Ces temps ne prennent en compte que le calcul de la pose, sans le temps de prédiction du ou des réseaux, et sont mesurés sur les images de test de la scène *Chess*.

Si la précision de pose obtenue avec cette approche jointe est proche de celle obtenue précédemment, ses deux intérêts principaux sont sa simplicité de déploiement, puisqu'un seul réseau de neurones est entraîné, et son temps de calcul de pose largement réduit. Il faut tout de même

noter que cette approche pourrait être limitée dans des cas très ambigus où il ne serait pas possible de distinguer des instances parfaitement identiques d'un même objet, placées dans un même environnement.

2.6 Conclusion du chapitre

Dans ce chapitre, nous avons présenté une méthode de calcul de pose de caméra basé objet. En particulier, cette approche ne nécessite pas de connaissances précises des objets de la scène de travail, telles que des modèles 3D. Les deux principales nouveautés par rapport aux méthodes existantes sont la détection améliorée des objets sous forme d'ellipses cohérentes avec leurs modèles ellipsoïdaux et l'utilisation d'un algorithme P3P pour le calcul de pose de la caméra à partir d'un ensemble minimal d'objets. Au travers de nos expériences, nous avons montré qu'en apprenant sur différents points de vue, notre réseau de prédiction d'ellipse est capable de prédire, à partir de l'apparence d'un objet, les paramètres d'une ellipse qui soit cohérente avec la projection de son modèle ellipsoïdal. Ces détections plus précises des objets, nous ont permis d'améliorer considérablement la précision du calcul de pose. Nous avons également démontré trois caractéristiques très importantes de notre méthode, qui sont son invariance au modèle ellipsoïdal choisi d'un objet, sa robustesse à un bruit sur la boîte de détection fournie par Faster R-CNN et le petit nombre d'annotations manuelles requises (le reste étant automatique).

Finalement, nous avons proposé une approche jointe de détection d'instances d'objets et de prédiction d'ellipses, à partir d'un seul réseau, qui fournit une précision de pose similaire à l'approche disjointe et offre un important gain de vitesse.

On peut également noter que, malgré le fait que cette méthode ait été développée afin de ne pas nécessiter de modèles 3D précis des objets, elle peut tout de même en tirer profit. Dans un cas où de tels modèles 3D seraient disponibles, notre système peut les utiliser pour générer des images d'entraînement synthétiques, offrant ainsi une meilleure couverture de points de vue de l'objet pour l'entraînement du réseau de prédiction d'ellipse.

Chapitre 3

Raffinement de la pose de la caméra et pondération par incertitude

Sommaire

Dans le chapitre 2, nous avons présenté une amélioration de la détection des objets, sous forme d'ellipses cohérentes à la 3D, afin d'atteindre un meilleur niveau de précision pour la pose estimée. Ce calcul de pose utilise un algorithme P3P et se base sur les centres de trois paires d'ellipse-ellipsoïde. Les différentes combinaisons de trois objets sont considérées et la pose finale est choisie comme étant celle avec la meilleure adéquation entre les objets détectés et les objets projetés de la scène dans l'image. Même si tous les objets détectés dans l'image interviennent

dans ce calcul d'adéquation, seulement trois d'entre eux sont réellement utilisés pour calculer la pose (voir figure 2.14). De plus, comme expliqué dans la section 2.3.6, le centre d'un ellipsoïde ne se projette pas parfaitement sur le centre de son ellipse de projection. Même si l'expérience dans la figure 2.15 a montré que l'erreur reste relativement faible, le calcul de pose utilisant les centres des ellipses et des ellipsoïdes est approximatif. Avec le modèle de caméra sténopé considéré dans ce travail, nous cherchons plutôt à aligner le cône de projection formé par un ellipsoïde avec le cône de rétroprojection issu de l'ellipse de détection de l'objet dans l'image.

Dans ce chapitre, nous proposons un raffinement de la pose de la caméra qui vise à aligner les ellipses de détection de tous les objets avec celles de projection de leurs modèles ellipsoïdaux. Nous formulons, tout d'abord, ce raffinement par la minimisation d'une erreur de reprojection dans la section 3.1, puis nous étudions différentes métriques entre ellipses, dans la section 3.2 permettant d'établir le coût que l'on cherche à minimiser. Nous comparons ces métriques et démontrons le gain de précision apporté par cette étape de raffinement de la pose dans la section 3.3. Enfin, dans la section 3.4, nous nous intéressons à la prédiction d'une incertitude pour les ellipses de détection des objets et à son utilisation, afin de mieux pondérer la contribution de chaque objet dans le calcul de la pose. Le contenu de ce chapitre a donné lieu à la publication [ZSB22a].

3.1 Raffinement par minimisation d'une erreur de reprojection

3.1.1 Optimisation non-linéaire

De manière similaire aux méthodes classiques utilisant des points 3D, nous avons donc cherché à raffiner la pose de la caméra, initialement estimée à partir d'un ensemble minimal d'objets. Ce raffinement vise à aligner les cônes de projection des ellipsoïdes avec les cônes de rétroprojection des ellipses de détection. L'opération est faite au niveau de l'image, en cherchant à minimiser l'erreur de reprojection entre les ellipses de détection et les ellipses de projection, comme illustré sur la figure 3.1. Cette minimisation s'exprime sous la forme d'une optimisation non-linéaire à six degrés de liberté :

$$\hat{R}, \hat{t} = \arg\min_{R,t} \sum_{j=1}^{N_{obj}} \Delta(\mathcal{E}_j, PQ_j^* P^T)^2, \tag{3.1}$$

où $P = K[R|t]$ est la matrice de projection de la caméra, avec K sa matrice de calibration et $[R|t]$ ses paramètres extrinsèques qui sont les variables recherchées. $\mathcal{E}_j$ est la j$^{\text{ème}}$ ellipse de détection et Q_j^* est la matrice de la forme duale de l'ellipsoïde correspondant. N_{obj} est le nombre d'objets considérés dans cette étape de raffinement, et enfin, $\Delta(\cdot)$ correspond à un coût entre deux ellipses que l'on cherche à minimiser.

La principale différence avec les méthodes basées sur les points vient justement de ce coût qui, classiquement, est choisi comme étant la distance euclidienne. Dans notre cas, cette distance n'a pas vraiment de sens et un coût entre deux ellipses est bien moins évident à établir. Ce choix de coût sera développé dans la section 3.2.

3.1.2 Pose initiale et association de données

Nous utilisons la pose estimée avec la méthode décrite dans le chapitre 2 comme estimation initiale. Celle-ci est calculée en utilisant P3P lorsqu'au moins trois objets sont détectés ou P2E dans le cas à deux objets uniquement. On peut noter que cette pose initiale pourrait également être obtenue par d'autres moyens, par exemple en utilisant des capteurs externes. L'estimation

initiale de la pose de la caméra est utilisée pour établir les associations entre les ellipses de détection et les objets de la scène. Pour cela, les ellipsoïdes sont projetés dans l'image et les paires d'ellipses ayant un recouvrement suffisant sont sélectionnées et sont prises en compte dans le raffinement de la pose. Nous avons utilisé un seuil de recouvrement minimal, en terme d'IoU, de 0.2 entre ellipses.

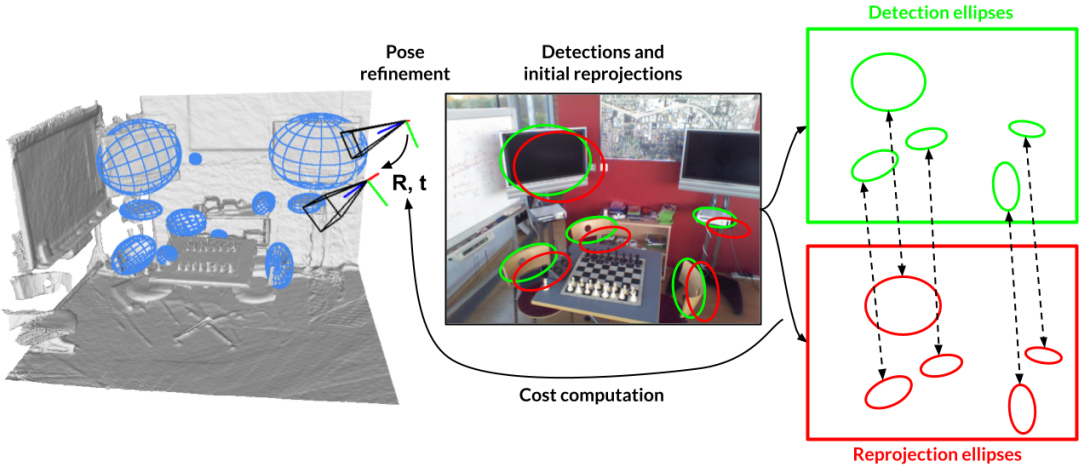


FIGURE 3.1 – Raffinement de la pose de la caméra par minimisation de l'erreur de reprojection entre les ellipses de détection dans l'image et les projections des modèles ellipsoïdaux des objets de la scène.

3.2 Comment établir un coût entre ellipses ?

Nous avons cherché à établir un coût entre ellipses qui a de bonnes propriétés de convergence pour notre problème de raffinement de pose de caméra. Par exemple, ce coût doit permettre de bien gérer des alignements entre une détection partielle d'un objet, qui peut être due à une sortie d'image ou à une occultation, avec la projection du modèle ellipsoïdal complet de cet objet. Dans cette section, nous passons en revue différentes métriques ellipse-ellipse existantes, puis nous proposons une nouvelle formulation basée sur des ensembles de niveaux.

3.2.1 Intersection-sur-Union standard et généralisée

L'intersection-sur-Union (IoU) est une métrique largement utilisée pour la comparaison de formes, avec des applications en segmentation sémantique [Cor+16], détection [Lin+14a] et suivi [Kri+21] d'objets. Elle permet de calculer le recouvrement relatif entre deux formes. Dans notre cas, nous avons cherché à minimiser la distance de Jaccard, définie par

$$\Delta_{IoU}(\mathcal{E}_1, \mathcal{E}_2) = 1 - \frac{|\mathcal{E}_1 \cap \mathcal{E}_2|}{|\mathcal{E}_1 \cup \mathcal{E}_2|}. \tag{3.2}$$

La difficulté majeure de l'utilisation de cette métrique vient du fait qu'elle reste constante quand les deux formes sont disjointes, c'est-à-dire lorsque leur recouvrement est nul. Elle quantifie donc

de la même manière deux ellipses disjointes proches et deux ellipses disjointes très éloignées. Ce comportement n'est pas désirable pour un problème d'optimisation. Rezatofighi *et al.* ont proposé une solution avec une version généralisée de cette métrique [Rez+19] (voir figure 3.2), notée GIoU et définie par

$$\Delta_{GIoU}(\mathcal{E}_1, \mathcal{E}_2) = 1 - \left(IoU - \frac{|C \setminus (\mathcal{E}_1 \cup \mathcal{E}_2)|}{|C|} \right), \tag{3.3}$$

où C correspond à la plus petite boîte alignée entourant les ellipses.

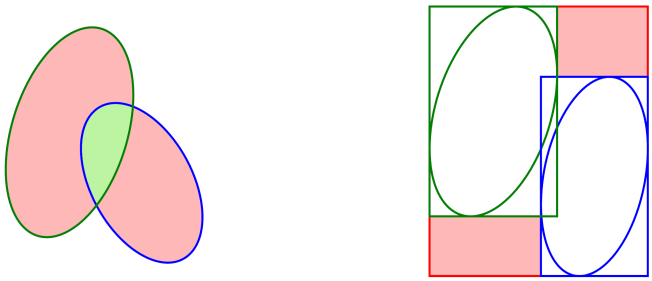


FIGURE 3.2 – L'Intersection-sur-Union (à gauche) mesure le ratio d'aire entre les surfaces vertes et rouges. La version généralisée de l'IoU (à droite) prend également en compte les zones en rouge, qu'il cherche à minimiser.

3.2.2 Distances entre boîtes

Il est également possible d'établir une métrique entre les boîtes englobantes des ellipses. On peut, par exemple, calculer l'erreur quadratique entre deux des coins de ces boîtes, telle que

$$\Delta_{bbox}(\mathcal{E}_1, \mathcal{E}_2) = \|\mathcal{B}_1 - \mathcal{B}_2\|_2^2, \tag{3.4}$$

où $\mathcal{B}_i$ contient les coordonnées des coins supérieur gauche et inférieur droit de la i$^{\text{ème}}$ boîte (min_x, min_y, max_x, max_y). La principale limite de cette métrique est qu'une boîte englobante, alignée aux axes de l'image, ne définit pas une ellipse unique. Au contraire, une infinité d'ellipses partagent la même boîte englobante, comme illustré dans la figure 3.3. Deux boîtes parfaitement alignées ne garantissent donc pas que les ellipses correspondantes le soient également. De plus, de telles boîtes sont définies par rapport aux axes de l'image, ce qui implique qu'une rotation appliquée à une paire d'ellipse peut modifier la valeur du coût (voir figure 3.4).

Une version améliorée de cette formulation de coût est proposée par Nicholson *et al.* dans QuadricSLAM [NMS19]. Abrégée *QBbox* dans les prochaines sections, elle prend en compte uniquement la partie des objets qui se trouve à l'intérieur de l'image (voir figure 3.5).

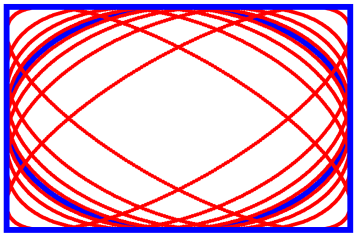


FIGURE 3.3 – Une boîte englobante peut définir une infinité d'ellipses.

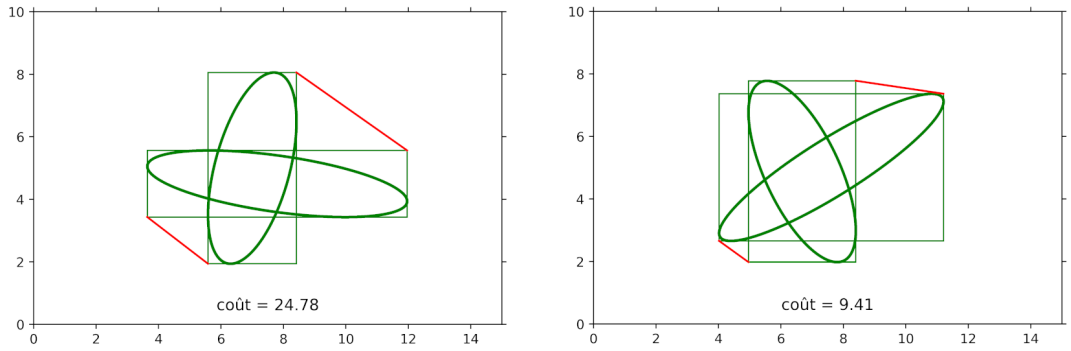


FIGURE 3.4 – Le coût calculé entre les boîtes englobantes de deux ellipses dépend de leur orientation dans l'image. Ici, une même paire d'ellipses n'obtient pas le même coût pour deux orientations différentes.

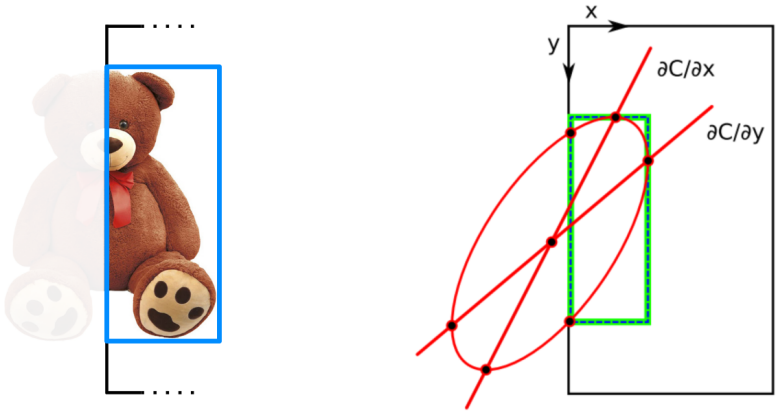


FIGURE 3.5 – À gauche : Illustration d'une détection partielle d'un objet. La partie de l'objet en dehors de l'image n'est pas détectée. À droite : QuadricSLAM [NMS19] restreint la projection d'un objet à la boîte englobante de sa partie qui est à l'intérieur de l'image. Cela permet de mieux prendre en compte le fait que seule cette partie est détectée.

3.2.3 Distances algébriques

Un coût peut également être établi entre les matrices des formes duales des ellipses. Pour rappel, une telle matrice définit une ellipse par une enveloppe de droites tangentes à son contour (voir figure 1.12). Dans le contexte de la reconstruction d'un ellipsoïde à partir d'observations d'ellipses dans plusieurs vues, Rubino *et al.* [RCB18] ont minimisé une erreur quadratique entre les formes vectorisées de ces matrices, donnée par

$$\Delta_{vec}(\mathcal{E}_1, \mathcal{E}_2) = \|vec(\mathbf{C}_1^*) - vec(\mathbf{C}_2^*)\|_2^2, \tag{3.5}$$

où $vec(\cdot)$ extrait les cinq éléments supérieurs d'une matrice symétrique. Hosseinzadeh *et al.* ont, quant à eux, calculé la norme de Frobenius de la différence de ces matrices [Hos+18] :

$$\Delta_{fro}(\mathcal{E}_1, \mathcal{E}_2) = \sqrt{Tr((\mathbf{C}_1^* - \mathbf{C}_2^*)(\mathbf{C}_1^* - \mathbf{C}_2^*)^T)}, \tag{3.6}$$

où $Tr(.)$ calcule la trace d'une matrice.

Ces deux métriques, *vectorisée* ou *Frobenius* comparent des matrices définissant les contours des ellipses. Bien que cela puisse fournir une bonne précision, le fait de considérer uniquement les contours peut limiter la robustesse lorsque les deux ellipses initiales diffèrent beaucoup. De plus, les valeurs de ces métriques changent si une translation globale est appliquée sur les deux ellipses, ce qui n'est pas souhaitable lorsque plusieurs paires d'ellipses, réparties dans l'ensemble de l'image, sont incluses conjointement dans l'optimisation. Enfin, il est difficile d'avoir une interprétation géométrique de ces distances basées sur les matrices des formes duales des ellipses.

3.2.4 Distances entre distributions de probabilité

Une ellipse, paramétrée par des axes (α, β), une orientation (θ) et un centre (c_x, c_y) peut également être vue comme une distribution gaussienne 2D, $\mathcal{N}(\mu, \Sigma)$, où

$$\mu = \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \qquad \Sigma^{-1} = R(\theta) \begin{bmatrix} \frac{1}{\alpha^2} & 0 \\ 0 & \frac{1}{\beta^2} \end{bmatrix} R(\theta)^T. \tag{3.7}$$

Il s'agit alors de calculer une distance entre des distributions de probabilité. Dans ce travail, nous avons considéré deux distances classiques entre distributions gaussiennes, à savoir, la distance de *Wasserstein* et celle de *Bhattacharyya*.

La distance de *Wasserstein*, également connue sous le nom de « distance du terrassier », permet de comparer des distributions de probabilité. Intuitivement, elle représente la quantité de travail minimum à fournir pour transformer un tas de terre en un autre. Arjovsky *et al.* l'ont utilisé pour l'apprentissage de distributions dans le cadre de réseaux antagonistes génératifs (GANs) [ACB17] et ont démontré ses bonnes propriétés de convergence. Cette distance a également été utilisée pour entraîner un réseau de détection de nœuds de bois sous la forme d'ellipses par Pan *et al.* [Pan+21]. La distance de *Wasserstein* peut être complexe à calculer dans le cas général mais une forme analytique existe dans le cas gaussien à deux dimensions et est donnée par

$$\Delta_{\mathcal{W}_2^2}(\mathcal{N}_1, \mathcal{N}_2) = \|\mu_1 - \mu_2\|_2^2 + Tr(\Sigma_1 + \Sigma_2 - 2(\Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}})^{\frac{1}{2}}). \tag{3.8}$$

La distance de *Bhattacharyya* exprime également une mesure de similarité entre deux distributions gaussiennes et est définie par

$$\Delta_{\mathcal{B}}(\mathcal{N}_1, \mathcal{N}_2) = \frac{1}{8}(\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2) + \frac{1}{2} \ln \left(\frac{\det \Sigma}{\sqrt{\det \Sigma_1 \det \Sigma_2}} \right), \tag{3.9}$$

avec $\Sigma = \frac{\Sigma_1 + \Sigma_2}{2}$. On peut noter que ces deux distances, *Wasserstein* et *Bhattacharyya*, combinent un terme de distance entre les centres des ellipses et un terme de différence des formes des ellipses.

3.2.5 Distance par ensembles de niveaux

Finalement, nous proposons également de réutiliser la métrique basée sur des ensembles de niveaux, que nous avons développée dans le chapitre 2.2.5 et utilisée pour entraîner le réseau de prédiction d'ellipse. Cette formulation basée sur une représentation implicite combine les avantages de considérer à la fois les contours des ellipses, pour la précision, mais également leurs surfaces, pour la robustesse. Comme expliqué précédemment, le coût se calcule entre les fonctions de plongement des ellipses. L'utilisation de l'équation quadratique d'une ellipse (équation 2.10) comme fonction de plongement avait posé des problèmes de stabilité numérique lors de l'entraînement du réseau de prédiction d'ellipse et nous avons proposé de modifier certaines parties (voir équation 2.12). C'était notamment le cas lorsque des valeurs très petites étaient prédites pour la

taille des axes, ce qui peut arriver dans les premières époques de l'entraînement. Ce n'est pas le cas ici, puisque nous cherchons à calculer un coût entre des ellipses de projection et des ellipses de détection d'objets de bonne qualité. Nous avons donc choisi d'utiliser l'équation quadratique originale d'une ellipse comme fonction de plongement.

Afin de garder le coût de calcul relativement bas, la distance entre les deux fonctions de plongement est à nouveau calculée de manière discrète par échantillonnage en des points fixes. Ces points sont placés régulièrement le long des rayons de l'ellipse de détection et restent fixes tout au long de l'optimisation. La figure 3.6 illustre ces points d'échantillonnage, ainsi que les fonctions de plongement des ellipses de détection et de projection. Les valeurs des fonctions de plongement sont calculées en ces points et le coût total est défini comme la somme des différences au carré, tel que

$$\Delta_{lvs}(\mathcal{E}_1, \mathcal{E}_2) = \sum_{i=1}^{N} (\Phi_1(\mathbf{x_i}) - \Phi_2(\mathbf{x_i}))^2, \tag{3.10}$$

où $\mathbf{x_i}$ est point d'échantillonnage et N est le nombre total de points. Le principal intérêt d'échantillonner les points le long de rayons de l'ellipse, plutôt que sur une grille alignée avec les axes de l'image, est que cela préserve l'invariance de la métrique à une rotation. Cette métrique est également notée *Level sets* dans les expériences qui suivent.

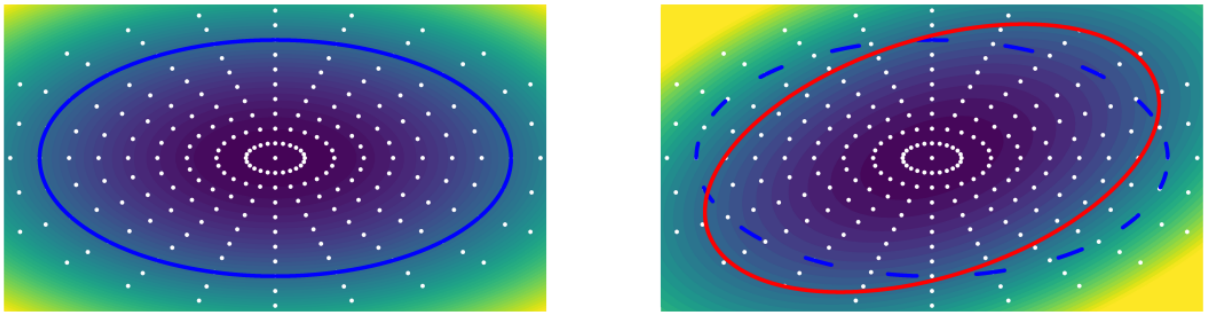


FIGURE 3.6 – À gauche : Ellipse de détection (en bleu) avec sa fonction de plongement décrivant des ensembles de niveaux et les points d'échantillonnage définis en blanc. À droite : Ellipse de détection (en rouge) avec sa fonction de plongement. Le coût est calculé en comparant les valeurs prises par chaque fonction de plongement au niveau des points d'échantillonnage.

3.3 Expériences et résultats

3.3.1 Implémentation

Nous avons utilisé l'algorithme Broyden-Fletcher-Goldfarb-Shanno [Avr03] (BFGS), disponible dans la bibliothèque SciPy, pour effectuer les optimisations. Les différentes métriques entre ellipses ont été implémentées en C++ et sont appelées à partir du code de raffinement, principalement développé en Python. Ce code est disponible à l'adresse suivante : `gitlab.inria.fr/tangram/level-set-based-camera-pose-estimation`.

3.3.2 Résultats pour l'alignement 2D d'ellipses

Nous avons tout d'abord évalué chacune de ces métriques pour un problème simplifié d'estimation de pose 2D. Celui-ci consiste à trouver la position et l'orientation d'une ellipse ($\mathcal{E}$) par rapport à une ellipse de référence ($\mathcal{E}_{ref}$). Pour générer des données synthétiques de test, nous avons suivi les étapes suivantes :

1. Une ellipse de référence $\mathcal{E}_{ref}$ est générée de manière aléatoire.
2. Une transformation rigide aléatoire (θ_{gt}, t_{gt}) est appliquée à $\mathcal{E}_{ref}$ pour obtenir $\mathcal{E}$. La rotation et la translation sont choisies de manière uniforme, respectivement entre $[-180°, 180°]$ et $[-60, 60]$ pixels.
3. Un bruit aléatoire d'échelle, anisotrope, est ajouté à l'ellipse $\mathcal{E}$ pour simuler les effets d'une détection imparfaite. Ce bruit est échantillonné de manière uniforme entre $\frac{1}{1.2}$ et 1.2.

L'objectif est ensuite d'estimer la transformation rigide (θ_{gt}, t_{gt}) en minimisant le coût d'alignement des deux ellipses, tel que

$$\hat{\theta}, \hat{t} = \arg\min_{\theta,t} \Delta(\mathcal{T}(\mathcal{E}, \theta, t), \mathcal{E}_{ref}), \tag{3.11}$$

où $\mathcal{T}(\mathcal{E}, \theta, t)$ effectue une rotation de l'ellipse $\mathcal{E}$ par θ et une translation par t. La transformation identité est choisie comme initialisation $(I_{3\times3}, 0_{3\times1})$.

Bien que cette expérience représente un problème simplifié par rapport à notre objectif de raffinement de pose de caméra, elle nous a permis d'identifier des comportements différents des métriques au cours de l'optimisation. Les erreurs moyennes de position et de rotation sont disponibles dans le tableau 3.1, avec et sans bruit de détection. On peut noter que l'IoU généralisée (*GIoU*), ainsi que les distances algébriques (*Vectorisée* et *Frobenius*) éprouvent des difficultés à converger. Par exemple, l'optimisation en utilisant l'IoU généralisée reste bloquée dans une zone de coût constant lorsqu'une ellipse se trouve entièrement à l'intérieur de l'autre. Au contraire, les distances entre distributions de probabilité (*Wasserstein* et *Bhattacharyya*), ainsi que notre métrique basée sur des ensembles de niveaux (*Level sets*) performent très bien. Il est important de noter que la métrique *Level sets* n'est pas sujette au problème de coût constant rencontré par *GIoU*, grâce à la forme de la fonction de plongement choisie.

Métrique	Sans bruit de détection		Avec bruit de détection	
	Err. pos (px)	Err. rot (°)	Err. pos (px)	Err. rot (°)
IoU généralisée (*GIoU*)	13.32	6.42	15.11	12.44
Boîte englobante (*QBbox*)	$1.0e^{-7}$	22.87	$1.2e^{-5}$	29.66
Algebrique Vectorisée	2.49	21.69	5.05	32.30
Algebrique Frobenius	2.59	22.08	5.04	32.30
Wasserstein	$9.8e^{-4}$	$7.8e^{-4}$	$7.9e^{-4}$	$4.6e^{-4}$
Bhattacharyya	$2.0e^{-2}$	$3.9e^{-3}$	$1.7e^{-2}$	$2.5e^{-3}$
Ensembles de niveaux (*Level sets*)	$1.0e^{-6}$	$1.0e^{-6}$	$2.9e^{-4}$	$2.7e^{-5}$

TABLE 3.1 – Erreurs moyennes de position (en pixels) et d'orientation (en degrés) obtenues pour la pose de l'ellipse pour 10000 paires d'ellipses générées aléatoirement.

3.3.3 Résultats du raffinement de la pose

Nous avons évalué les performances des différentes métriques pour notre problème de raffinement de pose de caméra sur la scène *Chess* du jeu de données 7-Scenes [GISC13], présenté dans la section 2.4.1. Nous avons employé la méthode décrite dans le chapitre 2 afin d'obtenir des détections d'objets améliorées sous forme d'ellipses cohérentes à la 3D. Le même modèle de scène a été utilisé (voir figure 2.31), ainsi que la même séparation entre données d'entraînement et de test.

La figure 3.7 montre les pourcentages d'images dont la position ou l'orientation a été correctement estimée, en fonction d'une erreur autorisée. Ces résultats sont distingués en fonction du nombre d'objets détectés dans l'image et utilisés pour le raffinement dans la figure 3.8. On peut, tout d'abord, noter que notre méthode de raffinement de la pose (*Level sets*) permet d'améliorer la précision de la pose initiale (*No refinement*) de manière significative. Le raffinement en utilisant cette métrique donne de meilleurs résultats dans tous les cas, à l'exception du cas à deux objets où le coût d'IoU généralisée semble apporter une robustesse légèrement supérieure. Il faut noter que ce cas à deux objets est particulièrement compliqué car très peu contraint. L'IoU généralisée semble être moins avantageuse dans le cas à trois objets. De son côté, le coût calculé sur les boîtes englobantes (*QBbox*) fournit de bonnes performances pour trois et quatre objets, mais est bien moins précis que *Level sets* et *GIoU* dans le cas à cinq objets et plus. On peut également noter que les coûts calculés par des distances entre distributions de probabilité donnent de moins bons résultats que lors de l'expérience simplifiée précédente pour l'estimation de la pose 2D d'une ellipse. Enfin, les métriques calculées sous forme de distances algébriques donnent de mauvais résultats et dégradent même globalement la précision de la pose initiale. De manière générale, on peut observer que l'étape de raffinement est particulièrement avantageuse lorsque relativement peu d'objets sont détectés dans l'image. On peut en effet noter que les gains sont plus faibles quand cinq objets ou plus sont utilisés, ce qui semble parfaitement normal. Des résultats de pose estimée, avant et après raffinement, sont donnés dans les figures 3.9 et 3.10. Les ellipses vertes et bleues sont utilisées dans le raffinement et on peut notamment observer un meilleur alignement entre les ellipses de détection (trait continu) et celles des projections des ellipsoïdes (trait discontinu). Enfin, la figure 3.11 illustre les erreurs de position obtenues avec les différentes métriques pour certaines portions des séquences de test.

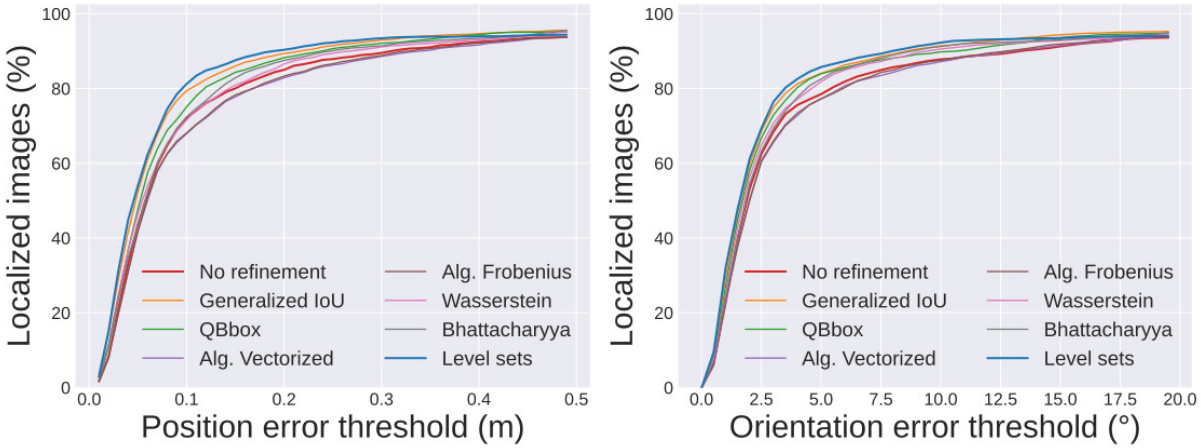


FIGURE 3.7 – Pourcentages d'images correctement localisées en fonction d'une erreur de position (à gauche) ou d'orientation (à droite). Ces résultats sont obtenus sur l'ensemble des images de test avec au moins deux objets détectés et utilisés dans le raffinement.

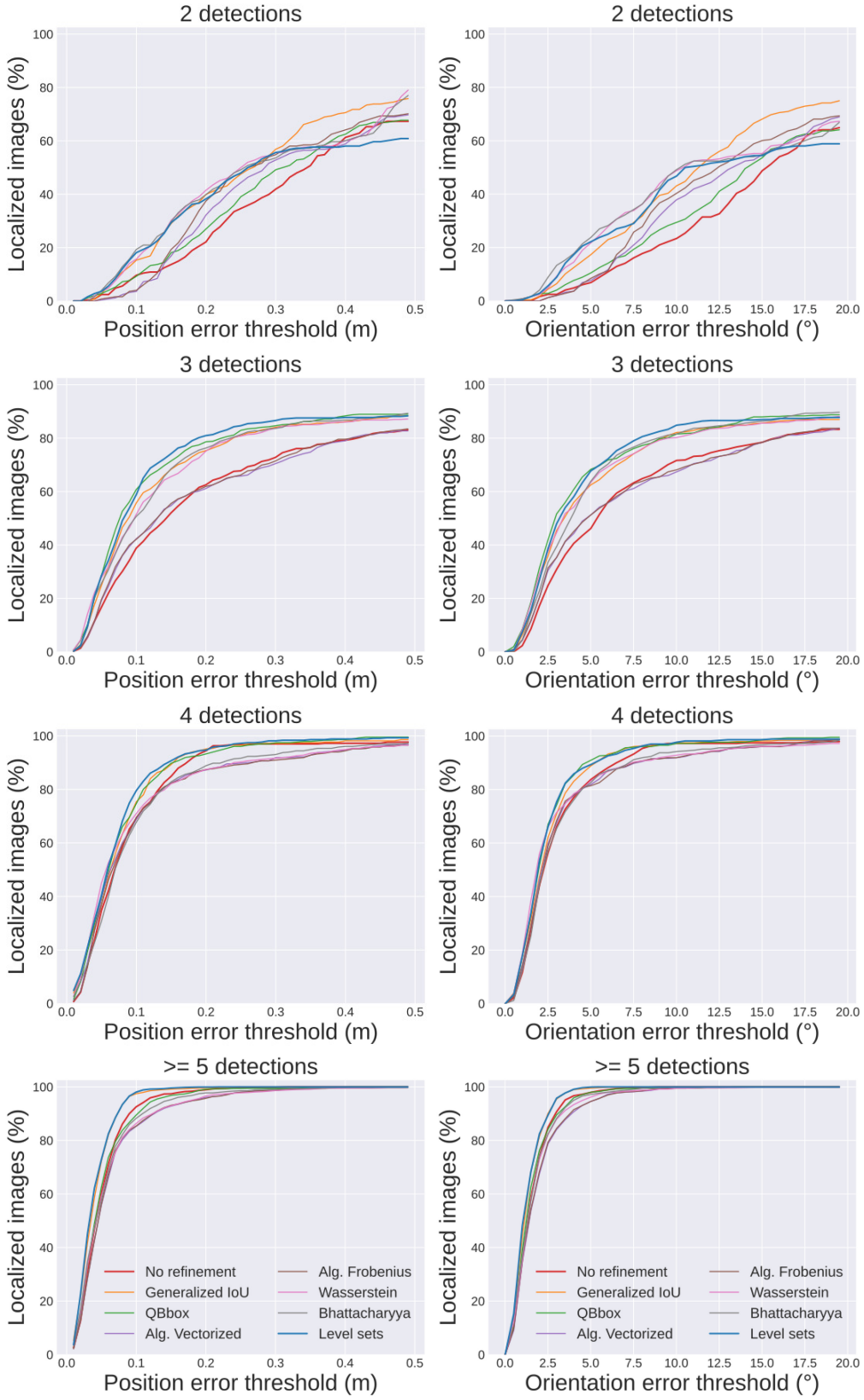


FIGURE 3.8 – Pourcentages d'images correctement localisées en fonction d'une erreur de position (à gauche) ou d'orientation (à droite). Les résultats sont divisés en fonction du nombre d'objets utilisés dans le raffinement.

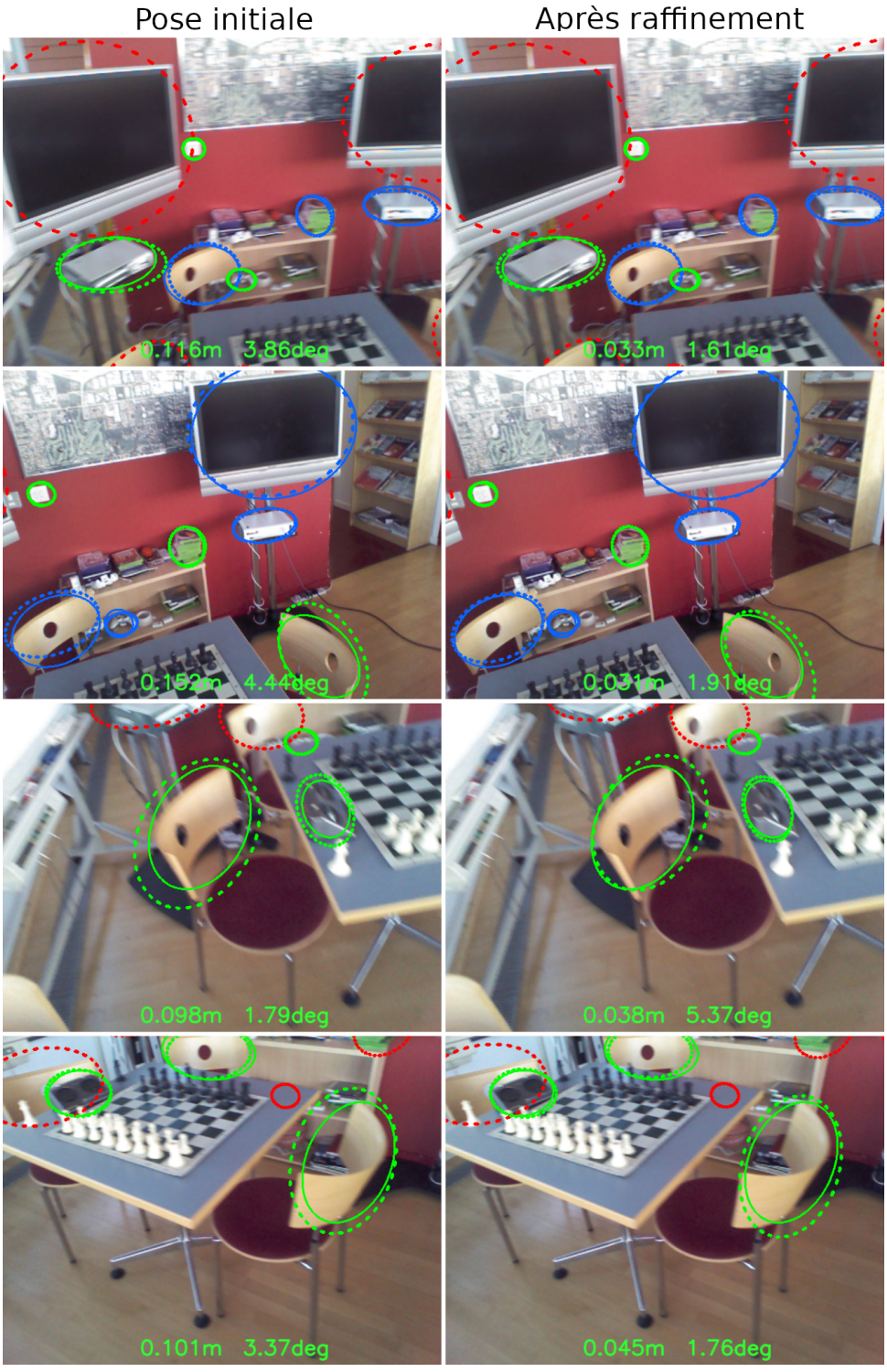


FIGURE 3.9 – Comparaison entre les poses de caméra obtenues avant raffinement (colonne de gauche) et celles après raffinement avec la métrique *Level sets* (colonne de droite). Les ellipses en trait continu sont les détections et celles discontinues correspondent aux projections des objets en utilisant la pose estimée. Seules les ellipses vertes sont utilisées pour le calcul de la pose initiale, les ellipses vertes et bleues sont utilisées dans le raffinement et les ellipses rouges sont des projections d'objets non détectés dans l'image. Les erreurs de position et d'orientation sont indiquées en bas.

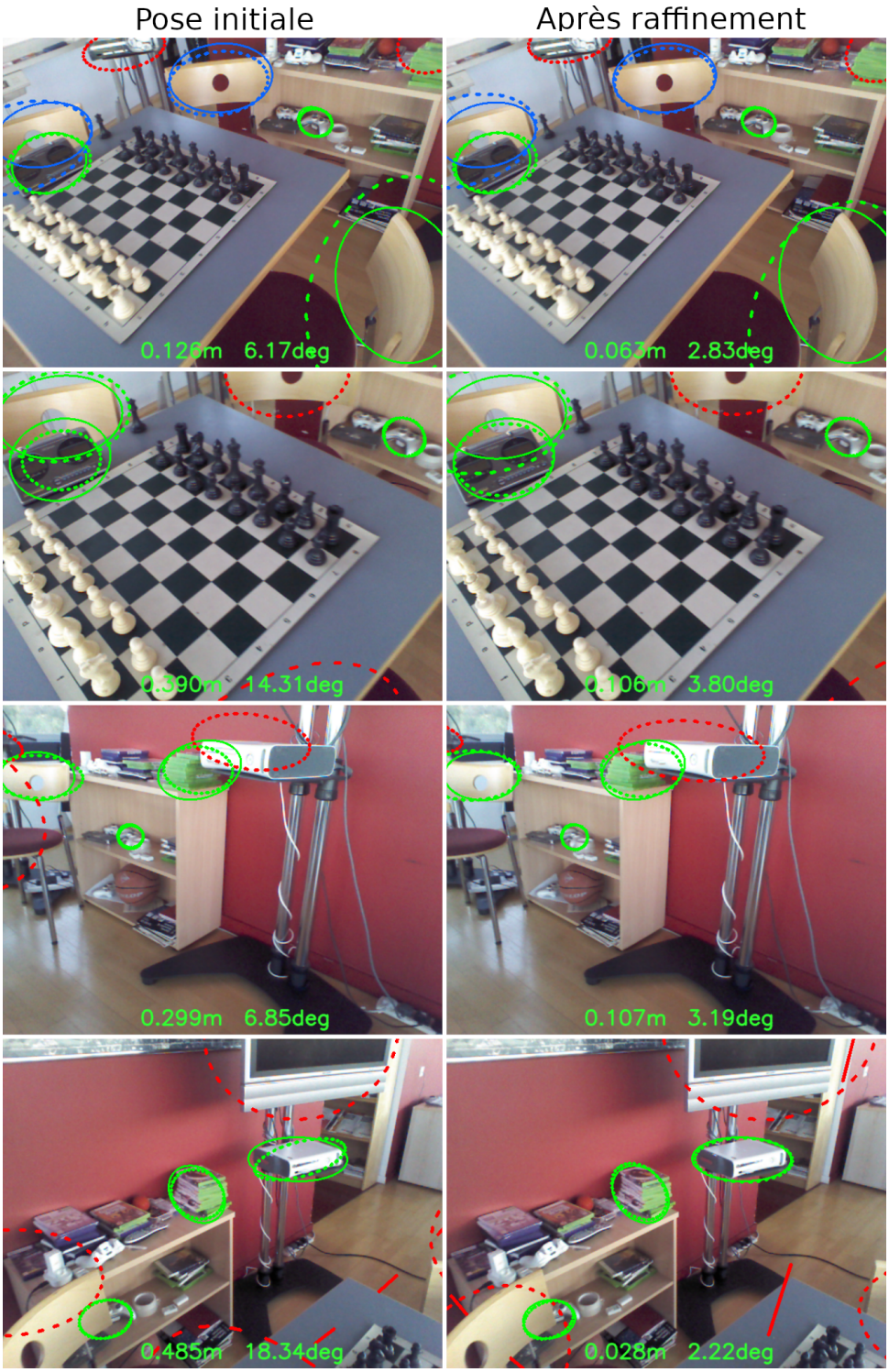


FIGURE 3.10 – Comparaison entre les poses de caméra obtenues avant raffinement (colonne de gauche) et celles après raffinement avec la métrique *Level sets* (colonne de droite). Les ellipses en trait continu sont les détections et celles discontinues correspondent aux projections des objets en utilisant la pose estimée. Seules les ellipses vertes sont utilisées pour le calcul de la pose initiale, les ellipses vertes et bleues sont utilisées dans le raffinement et les ellipses rouges sont des projections d'objets non détectés dans l'image. Les erreurs de position et d'orientation sont indiquées en bas.

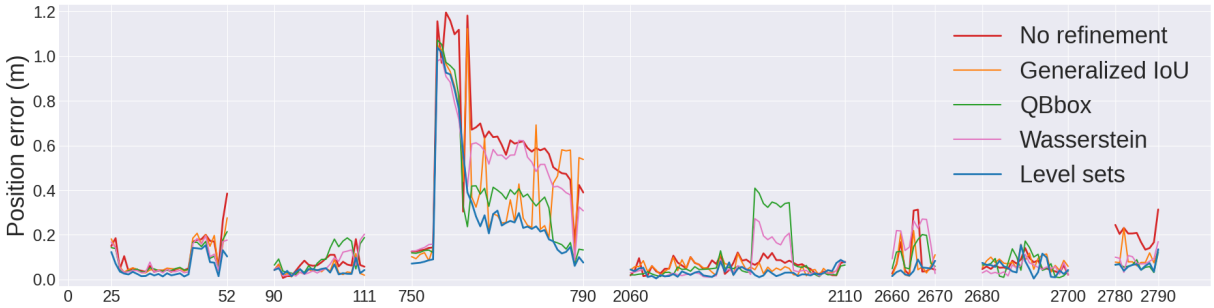


FIGURE 3.11 – Erreur de position estimée de la caméra sur certaines portions d'images des séquences de test de la scène *Chess*.

3.3.4 Analyse des caractéristiques de la métrique *Level sets*

Dans cette section, nous analysons le comportement de la métrique *Level sets* au cours de l'optimisation et le comparons aux autres métriques évaluées. Nous nous intéressons, en particulier, au problème de détection partielle d'objet, qui est un cas compliqué et assez fréquent en pratique. En effet, cela se produit lorsqu'un objet est seulement partiellement visible, par exemple à cause d'une occultation ou parce que celui-ci est en partie en dehors de l'image. Dans ce cas, essayer d'aligner la projection d'un modèle ellipsoïdal complet de l'objet avec une détection partielle n'a pas de sens. Au contraire, il est beaucoup plus souhaitable de chercher à aligner leurs contours.

La figure 3.12 illustre les valeurs des différents coûts, entre une ellipse de détection en vert et une ellipse de projection en rouge, dans deux configurations. La première, sur la ligne du haut, montre deux ellipses avec des tailles relativement proches. On peut observer que chaque coût augmente lorsque les ellipses se décentrent. La valeur minimale du coût est à chaque fois obtenue dans l'image de gauche, c'est-à-dire lorsque les deux ellipses sont parfaitement centrées l'une sur l'autre. Ce comportement correspondant bien à ce qui est recherché lors de notre optimisation de la pose de la caméra. Dans le second cas, illustré sur la ligne du bas, l'ellipse de détection (en vert) est bien plus petite que la projection (en rouge). Dans cette configuration, on peut voir que la majorité des métriques privilégient toujours l'alignement des centres des deux ellipses, à l'exception de *Generalized IoU* et *Level sets*. La valeur du coût *GIoU* reste, elle, constante lorsqu'une des ellipses est entièrement à l'intérieur de l'autre, ce qui n'est pas souhaitable lors de l'optimisation. Par contre, la métrique *Level sets* privilégie un alignement tangentiel entre les contours des deux ellipses.

Le comportement de notre métrique dépend donc de la taille, similaire ou non, des deux ellipses considérées. Le changement de comportement se produit de manière progressive. La figure 3.13 l'illustre et montre une carte du coût entre l'ellipse rouge et l'ellipse bleue. Chaque point donne la valeur du coût si l'ellipse bleue était centrée en ce point. L'ellipse bleue, dessinée sur chaque image, correspond à l'emplacement de coût minimal. À titre de comparaison, le comportement du coût *QBbox*, calculé sur les boîtes englobantes des ellipses, est donné dans la figure 3.14. On peut y constater que ce coût favorise toujours un alignement des centres des ellipses.

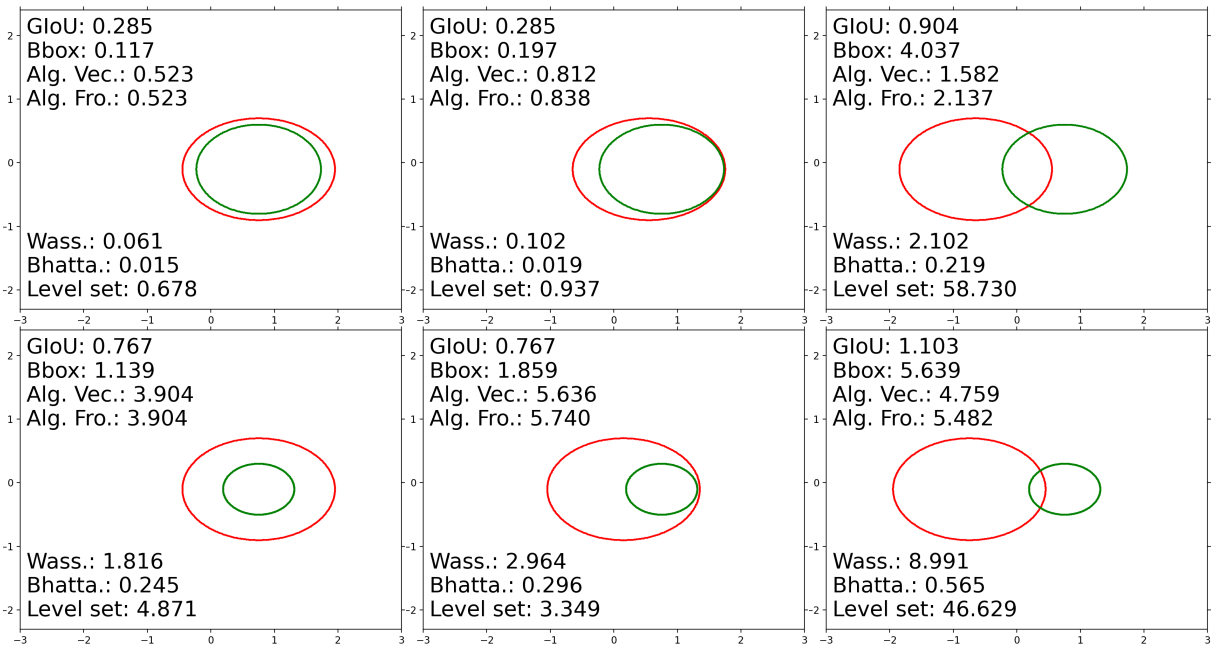


FIGURE 3.12 – Valeurs des différents coûts entre une ellipse de détection (en vert) et une ellipse de projection (en rouge). La ligne du haut montre le cas de deux ellipses de tailles relativement similaires et la ligne du bas montre une ellipse de détection bien plus petite que celle de projection.

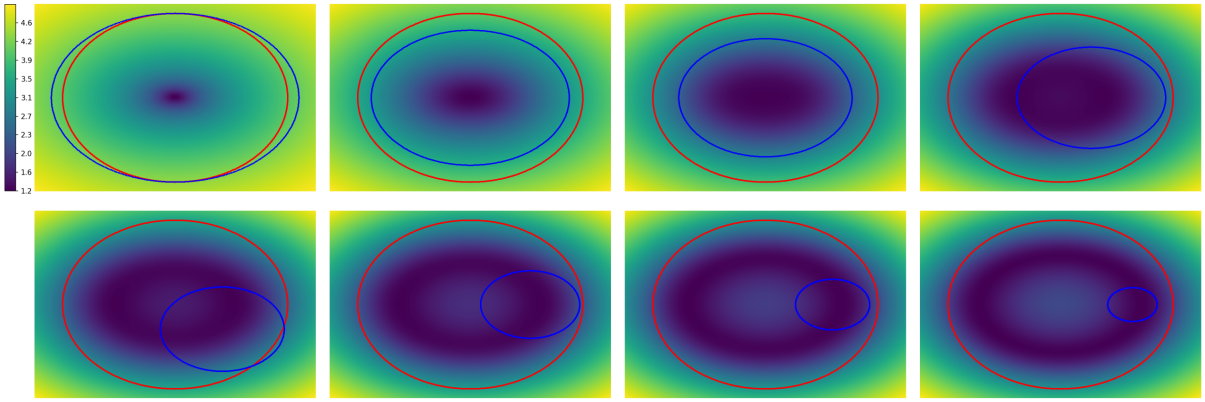


FIGURE 3.13 – Carte du coût (*Level sets*) entre les deux ellipses pour toutes les positions du centre de l'ellipse de détection (en bleu). L'ellipse bleue, affichée dans les images, correspond à celle de coût minimal (zone bleu foncée). Les différentes images montrent l'évolution de ce coût et de la localisation du minimum pour des ellipses de détection de plus en plus petites. Une échelle logarithmique est utilisée pour la carte du coût.

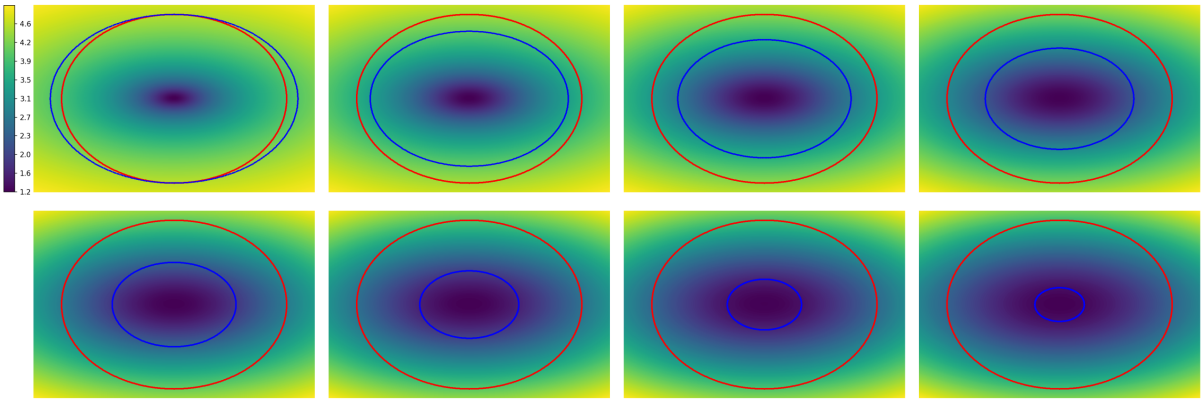


FIGURE 3.14 – Carte du coût (*QBbox*) entre les deux ellipses pour toutes les positions du centre de l'ellipse de détection (en bleu). L'ellipse bleue, affichée dans les images, correspond à celle de coût minimal (zone bleu foncée). Les différentes images montrent l'évolution de ce coût et de la localisation du minimum pour des ellipses de détection de plus en plus petites. Une échelle logarithmique est utilisée pour la carte du coût.

Objets partiellement visibles

La figure 3.15 illustre le calcul de pose de la caméra avec une détection partielle d'un objet. En effet, on peut voir que seule une partie du dossier de la chaise en bas de l'image est détectée. L'ellipse verte en trait continu montre les ellipses de détection. Les ellipses vertes discontinues correspondent aux projections des objets de la scène et les ellipses blanches continues sont les projections des objets avec la pose vérité terrain. Les erreurs de position et d'orientation de la pose estimée de la caméra sont indiquées en bas à gauche des images. La pose initiale, déterminée avant raffinement, présentait une erreur de position de 7.8 cm et une erreur d'orientation de 1.79°.

Les métriques suivantes sont comparées : *Level sets*, *Wasserstein* et *QBbox*. On peut, tout d'abord, noter que la distance de Wasserstein cherche bien à aligner le centre de la détection partielle du dossier de la chaise avec la projection de son modèle ellipsoïdal complet, ce qui dégrade la pose estimée pour la caméra. On voit également que les alignements, entre détections et projections, des quatre objets correctement détectés en haut de l'image, sont dégradés. En effet, l'optimisation cherche à minimiser le coût total de toutes les paires d'ellipses considérées dans l'image et la pose finale obtenue constitue un compromis entre les alignements des différents objets. L'optimisation a donc privilégié l'alignement des centres des ellipses du dossier de chaise, en bas de l'image, quitte à dégrader légèrement l'alignement des autres objets.

La métrique QBbox, utilisée dans QuadricSLAM [NMS19], est efficace pour gérer les détections partielles d'objet, mais uniquement dans le cas où l'objet est partiellement en dehors de l'image. En effet, cette métrique calcule une distance entre les boîtes englobantes des parties des ellipses qui se trouvent à l'intérieur de l'image (voir figure 3.5). Dans le cas traité ici, l'extrémité gauche de la chaise, non détectée, est tout de même visible dans l'image et sa mauvaise détection est due au point de vue peu représenté dans la base d'apprentissage. Cela, induit donc un coût élevé pour cet objet qui pousse l'optimisation à aligner les centres de ses ellipses de détection et projection. À nouveau, la pose de la caméra se retrouve dégradée.

Au contraire de ces deux métriques, le comportement de *Level sets* est bien plus favorable car il privilégie un alignement tangentiel entre les deux ellipses lorsqu'elles ont des tailles différentes. Cela résulte en une estimation bien meilleure, qui améliore la précision de la pose initiale.

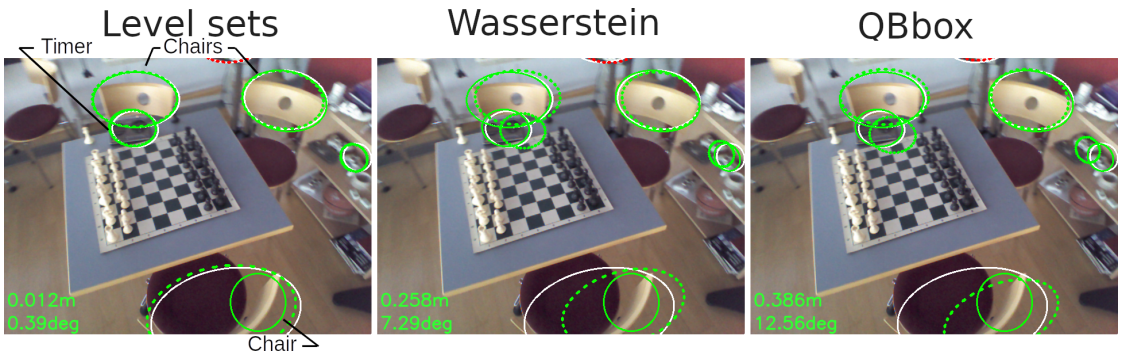


FIGURE 3.15 – Pose estimée avec les métriques *Level sets* (à gauche), *Wasserstein* (au centre) et *QBbox* (à droite). Les ellipses vertes continues sont les détections et celles discontinues correspondent aux projections des objets en utilisant la pose estimée de la caméra. Les ellipses vertes sont utilisées dans le raffinement de la caméra, les ellipses rouges sont des projections d'objets non détectés dans l'image et les ellipses blanches montrent les projections des objets avec la pose de vérité terrain. Les erreurs de position et d'orientation sont indiquées en bas à gauche.

Meilleure contribution des objets petits/lointains

La deuxième propriété importante de notre métrique vient de la meilleure contribution des objets petits et/ou lointains dans la scène, ce qui n'est pas toujours évident. Par exemple, l'absence de normalisation du coût par objet, pour la métrique *QBbox*, donne naturellement plus d'importance aux grands objets dans l'image, c'est-à-dire aux objets proches de la caméra et/ou grands dans la scène.

La figure 3.16 compare les résultats obtenus avec les métriques : *Level sets*, *GIoU* et *QBbox*. Bien que l'image contienne des objets partiellement visibles (le dossier de la chaise en bas et la console de jeu sur la gauche), ceux-ci ne sont pas problématiques pour les métriques comparées ici. En effet, *GIoU* ne force pas un alignement des centres lorsque l'ellipse de détection est bien plus petite que l'ellipse de projection car sa valeur reste constante. De plus, *QBbox* est capable de les gérer correctement car les parties non détectées des objets sont situées en dehors de l'image et donc ignorées par cette métrique. La difficulté du cas présenté est plutôt liée à l'ajustement de la contribution des objets, notamment entre les objets lointains et ceux proches. Nous pensons que les objets éloignés de la caméra constituent de très bons points d'ancrage pour le calcul de la pose de la caméra car une mauvaise estimation de cette pose a un impact plus fort sur l'alignement de leurs ellipses.

Contrairement à *QBbox* qui donne plus d'importance aux grands objets dans l'image, l'IoU généralisée est naturellement normalisée puisqu'elle correspond à un recouvrement relatif entre des ellipses et a donc tendance à égaliser les contributions des différents objets. Cependant, comme indiqué dans la dernière colonne de la figure 3.12, la valeur de ce coût augmente très lentement lors du décentrage des ellipses. Notre métrique *Level sets* est également normalisée, par la fonction de plongement choisie (la courbe de niveau 1 correspond toujours au contour de l'ellipse), mais produit, au contraire, un coût bien plus élevé pour des ellipses distantes (voir figure 3.12). Cette métrique permet donc de mieux tirer avantage des objets petits et/ou lointains détectés dans l'image. On peut effectivement observer un bien meilleur alignement d'ellipses pour le petit objet au centre de l'image (la manette de jeu) en utilisant la métrique *Level sets*. La pose initiale de la caméra était estimée avec une erreur de position de 30.9 cm et d'orientation de 8.61°. Le raffinement, avec chaque métrique, a permis d'améliorer la précision de la pose, mais les gains obtenus avec notre métrique sont bien plus importants.

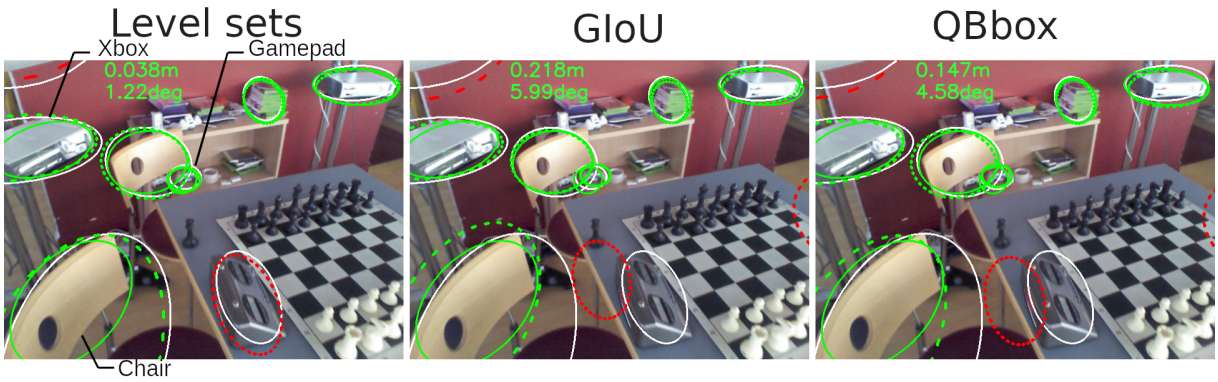


FIGURE 3.16 – Pose estimée avec les métriques *Level sets* (à gauche), *GIoU* (au centre) et *QBbox* (à droite). Les ellipses vertes continues sont les détections et celles discontinues correspondent aux projections des objets en utilisant la pose estimée de la caméra. Les ellipses vertes sont utilisées dans le raffinement de la caméra, les ellipses rouges sont des projections d'objets non détectés dans l'image et les ellipses blanches montrent les projections des objets avec la pose de vérité terrain. Les erreurs de position et d'orientation sont indiquées en haut.

3.3.5 Robustesse à la pose initiale et convergence

Afin d'évaluer les capacités de convergence des différentes métriques, ainsi que l'influence de l'estimation initiale de la pose, nous avons répété l'évaluation décrite dans la section 3.3.3 en ajoutant du bruit aux poses initiales de la caméra. Ces poses de caméra sont bruitées à la fois en rotation et en translation. Trois niveaux de bruits ont été utilisés. Le bruit de translation est généré de façon uniforme, sur chaque coordonnée, dans les plages suivantes : $[-30\,\text{cm}, 30\,\text{cm}]$, $[-40\,\text{cm}, 40\,\text{cm}]$, $[-50\,\text{cm}, 50\,\text{cm}]$. Le bruit de rotation est généré de manière uniforme dans les plages respectives $[-6°, 6°]$, $[-6°, 6°]$, $[-7°, 7°]$ et est appliqué sur chaque composante de la représentation en angles d'Euler de l'orientation de la caméra. Les résultats obtenus sont disponibles dans la figure 3.17 et montrent que la métrique proposée, *Level sets*, se montre particulièrement robuste. Les distances de *Wasserstein* et *Bhattacharyya* sont, elles aussi, seulement faiblement affectées par le bruit mais se montrent moins performantes que *Level sets*. Au contraire, les métriques *QBbox*, *GIoU* et les distances algébriques semblent plus sensibles à une pose initialement bruitée. Cela révèle les moins bonnes propriétés de convergence de ces métriques.

3.3.6 Analyse du temps de raffinement

Nous avons mesuré la durée de l'étape de raffinement de la pose de la caméra pour chaque métrique. Les résultats sont disponibles dans la figure 3.18 et montrent les temps moyens et médians de raffinement ainsi que leur dispersion sur les séquences de test de la scène *Chess*. On peut, tout d'abord, noter que la métrique proposée, *Level sets*, requiert un temps médian de 244 ms, ce qui est compatible avec l'application visée de localisation visuelle. Cette métrique se montre très légèrement plus rapide que les métriques *GIoU*, *QBbox* et *Wasserstein*. Les deux distances algébriques, *Alg. Vec.* et *Alg. Fro.*, sont beaucoup plus lentes. Enfin, seule la distance de *Bhattacharyya* offre un raffinement plus rapide mais, comme présenté dans la section 3.3.3, elle obtient de moins bon résultats en termes de précision de pose.

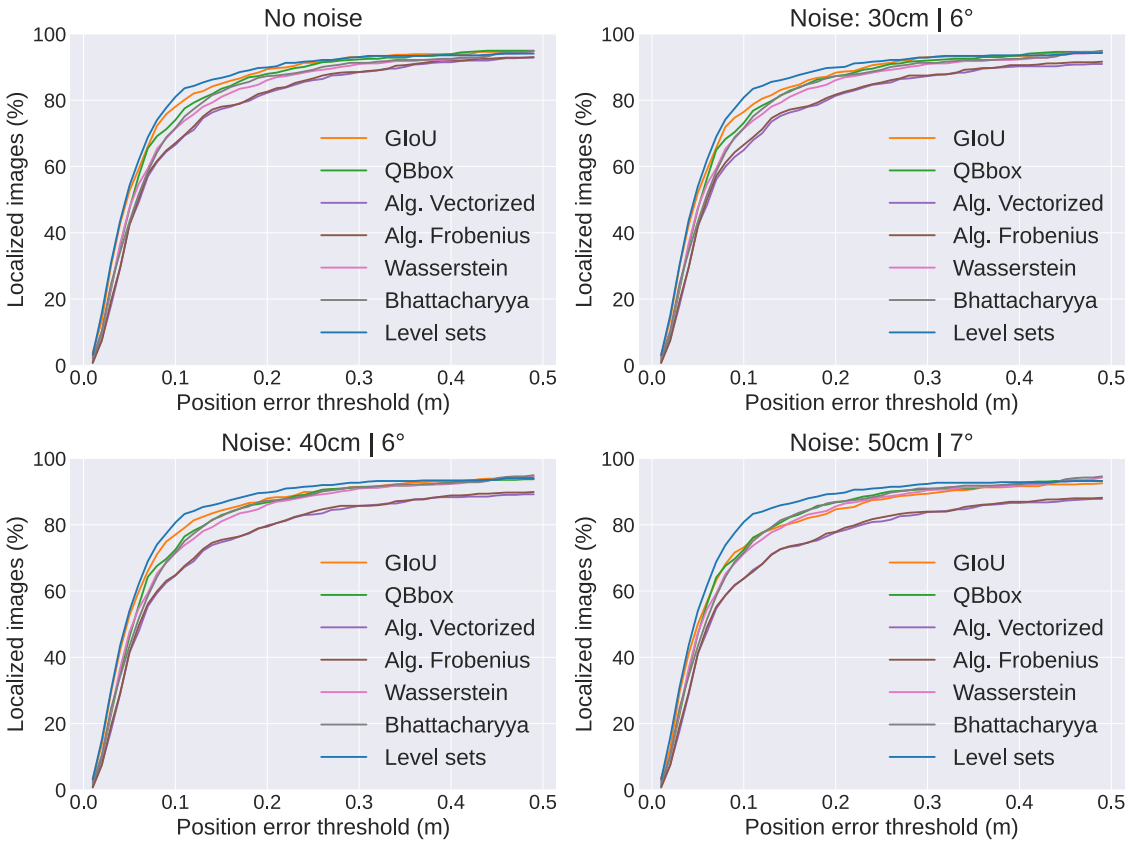


FIGURE 3.17 – Comparaison de la sensibilité des différentes métriques à une pose de caméra initiale bruitée. Les courbes montrent le pourcentage d'images correctement localisées en fonction d'une erreur de position pour différents niveaux de bruit.

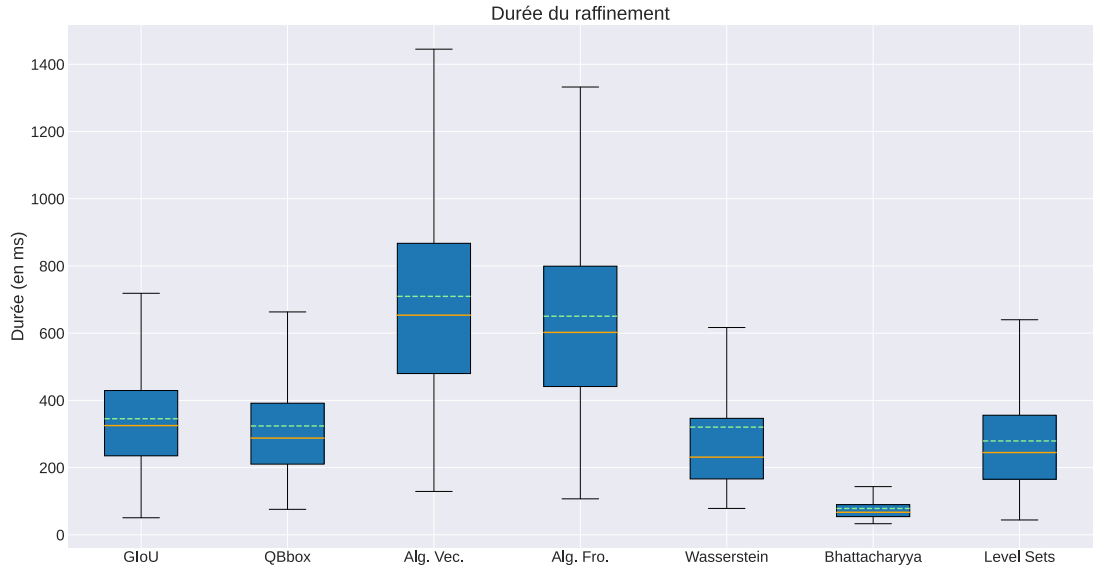
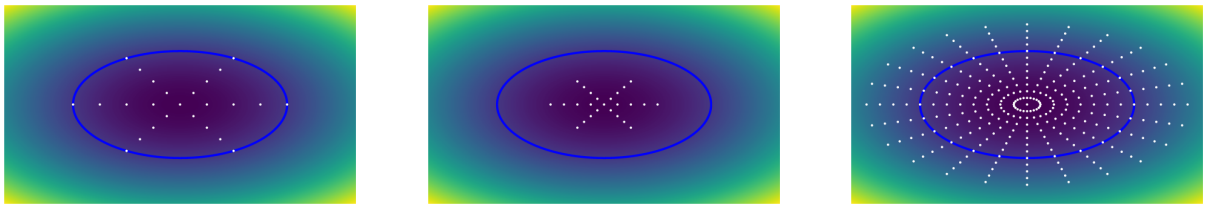


FIGURE 3.18 – Analyse du temps de calcul nécessaire à l'étape de raffinement pour les différentes métriques. Les durées médianes sont indiqués en orange et celles moyennes en vert.

3.3.7 Analyse de l'échelle d'échantillonnage de *Level sets*

Dans cette section, nous analysons différentes configurations de l'échantillonnage des points autour de l'ellipse de détection pour le calcul de la métrique *Level sets*. Nous nous intéressons en particulier à la taille de la zone considérée et au nombre de points utilisés.

Les points sont échantillonnés le long de segments (appelés azimuts) répartis de manière homogène tout autour du centre de l'ellipse. Le nombre de segments et le nombre de points par segment sont respectivement nommés *nb_ az* et *nb_ dists* dans la figure 3.20. La taille de l'échantillonnage définit l'échelle des segments par rapport au contour de l'ellipse. Une taille de 1.0 correspond à des segments allant du centre jusqu'au contour. Différentes configurations sont illustrées dans la figure 3.19.



(a) (b) (c)

FIGURE 3.19 – Différentes configurations d'échantillonnage. a) 6 azimuts, 4 points par segment et taille de 1.0. b) 6 azimuts, 4 points par segment et taille de 0.5. c) 24 azimuts, 12 points par segment et taille de 1.5.

Nous avons testé différents nombres de points d'échantillonage, en commençant à 1600 (40 azimuts et 40 points par segment). Nous avons pu réduire ce nombre jusqu'à 24, pour 6 valeurs d'azimut et 4 points par segment, sans observer de dégradation des performances mais, au contraire, avec un gain important en temps de calcul.

L'influence de différentes tailles d'échantillonnage sur la précision de la pose estimée est disponible dans la figure 3.20. Ces résultats ont été obtenus sur les images de test de la scène *Chess*. Pour les courbes de gauche, nous avons fait varier la taille de l'échantillonnage avec un nombre fixe de points (6 azimuts et 4 points par segment). Sur la droite, le nombre de points a été adapté en fonction de la taille d'échantillonage. Les courbes montrent qu'un échantillonage de taille 1.0 donne les meilleurs résultats. Élargir la zone dégrade les performances, même dans le cas où l'on augmente le nombre de points. Ce résultat est lié au fait que, dans notre cas, les ellipses optimisées ont déjà un certain recouvrement initial car l'association des données entre les objets et les détections est basée dessus. Dans d'autres cas d'application, où l'on chercherait, par exemple, à aligner des ellipses plus distantes, une zone d'échantillonnage élargie pourrait être plus utile.

3.3.8 Fonction d'optimisation *Trust-Region-Reflective*

Nous avons également utilisé l'algorithme *Trust-Region-Reflective* (TRF) [BCL99] pour le raffinement de la pose, qui est une méthode d'optimisation de moindres carrés non linéaires basée sur des régions de confiance. À nouveau, nous avons utilisé l'implémentation disponible dans la bibliothèque SciPy. La figure 3.21 compare les résultats obtenus avec BFGS et TRF sur la scène *Chess*. Les précisions obtenues sont très proches, avec un très léger avantage pour TRF. On peut cependant observer un gain de vitesse assez important avec l'utilisation de TRF. Par

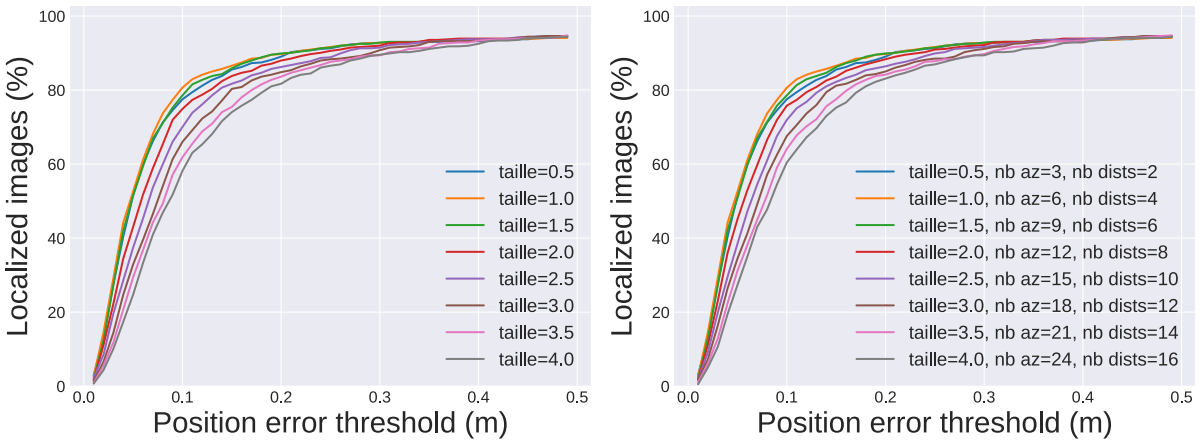


FIGURE 3.20 – Analyse de la taille de l'échantillonnage utilisé dans le calcul de la métrique *Level sets*. Les courbes montrent le pourcentage d'images correctement localisées en fonction d'une erreur de position. À gauche, différentes tailles d'échantillonnage ont été utilisées avec un nombre fixe de points (24 pour 6 azimuts et 4 points par segment). À droite, le nombre de points d'échantillonnage est adapté en fonction de la taille de la zone considérée.

exemple, la combinaison *Level sets* et TRF se montre même plus rapide que BFGS utilisé avec la distance de *Bhattacharyya*, qui était la méthode la plus rapide mesurée dans la figure 3.18.

Bien que cette méthode d'optimisation permette d'accélérer le raffinement avec la métrique *Level sets*, elle fournit des résultats inférieurs pour les autres métriques. Pour être équitable dans nos comparaisons, nous avons donc choisi d'utiliser BFGS pour toutes les métriques, dans les autres expériences du chapitre.

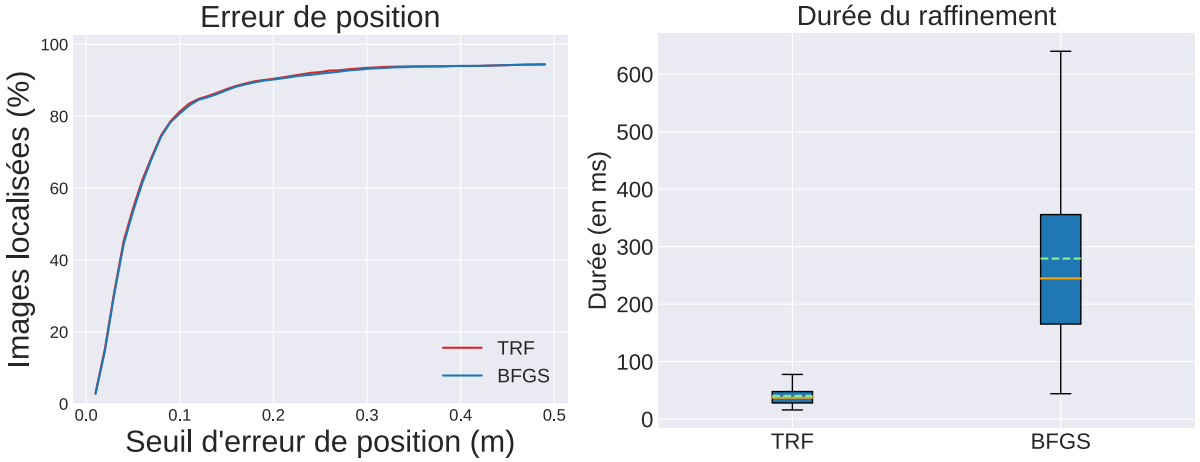


FIGURE 3.21 – Comparaison entre la méthode d'optimisation BFGS et une méthode par moindres carrés TRF pour l'étape de raffinement avec la métrique *Level sets*. À gauche : comparaison des pourcentages d'images correctement localisées en fonction d'une erreur de position. À droite : comparaison des temps de calcul.

Des résultats et explications supplémentaires sur le raffinement de la pose sont disponibles dans la vidéo à l'adresse : https://zinsmatt.github.io/these/#IROS.

3.4 Pondération des objets par incertitude

Dans ce chapitre, nous avons donc développé une méthode de raffinement de la pose de la caméra basée sur les objets et proposé une nouvelle métrique permettant d'établir un coût entre deux ellipses. Dans cette section, nous nous intéressons maintenant à la pondération des contributions des objets dans le raffinement de la pose. L'idée est de réduire l'influence des objets dont la détection serait de mauvaise qualité. Il peut s'agir d'objets partiellement en dehors de l'image ou occultés par d'autres éléments de la scène, ou encore, des détection imprécises du fait d'un point de vue peu commun sur l'objet.

Les objets utilisés dans l'étape de raffinement sont détectés dans l'image sous la forme d'ellipses. Pour rappel, celles-ci sont prédites par un réseau de neurone et sont censées être cohérentes avec les projections des modèles ellipsoïdaux des objets, afin d'améliorer la précision du calcul de pose (voir chapitre 2). Il s'agit donc ici d'être capable de caractériser l'incertitude des prédictions faites par ce réseau de neurones.

3.4.1 Estimation d'incertitude dans les réseaux de neurones

La quantification de l'incertitude dans les réseaux de neurones a récemment reçu un fort intérêt de la part de la communauté scientifique. En effet, être capable d'associer un indice de confiance à une prédiction est devenu très important, notamment pour des applications critiques, telles que la navigation autonome ou la chirurgie assistée par ordinateur. Skinner *et al.* ont, par exemple, proposé une compétition de détection probabiliste d'objets où le but est de détecter des objets sous forme de boîtes englobantes et de leur associer une incertitude spatiale exprimée par des distributions gaussiennes sur les coins de ces boîtes [Ski+19]. Une nouvelle métrique a d'ailleurs été développée pour évaluer ce genre de détections [Hal+20]. Les méthodes existantes de détection d'objets, telles que YOLO [RDGF16] ou Faster R-CNN [RHGS17], prédisent un score de confiance pour chaque détection mais ne sont pas discriminantes car ces valeurs sont généralement très élevées, même pour de mauvaises détections.

Deux types d'incertitude

La littérature distingue généralement deux types d'incertitude : l'incertitude de modèle (ou épistémique) et l'incertitude des données (ou aléatorique) [KG17].

L'incertitude de modèle représente le manque de connaissance d'un modèle par rapport à une donnée d'entrée et permet de représenter la confiance qu'un réseau a dans sa prédiction. Elle est généralement liée à un manque de données d'apprentissage, en nombre ou en diversité, ou à un nombre d'époques d'entraînement insuffisant. Elle peut donc être réduite en ajoutant des données d'apprentissage.

L'incertitude de données est, elle, liée au bruit ou aux ambiguïtés dans les données d'entrée du réseau. Il peut s'agir, par exemple, d'un bruit lié au capteur ayant acquis la donnée. Ce type d'incertitude peut encore être catégorisé en deux versions : homoscédastique et hétéroscédastique. L'incertitude homoscédastique reste constante pour différentes entrées du réseau, alors que l'incertitude hétéroscédastique varie en fonction des entrées, avec certaines données causant potentiellement des sorties plus bruitées que d'autres. Cette incertitude hétéroscédastique est particulièrement importante pour les applications de vision par ordinateur. Par exemple, pour la régression de profondeur, les images d'entrée très texturées avec de fortes lignes de fuite devraient donner lieu à des prédictions fiables, alors qu'une image d'un mur sans caractéristiques devrait présenter une très grande incertitude.

Les méthode actuelles proposent de mesurer l'incertitude en modélisant les paramètres ou les sorties d'un modèle par des distributions de probabilité. L'incertitude épistémique est modélisée en plaçant une distribution préalable sur les poids d'un modèle, puis en essayant d'analyser comment ces poids varient en fonction de certaines données. L'incertitude aléatorique, quant à elle, est modélisée en plaçant une distribution sur la sortie du modèle.

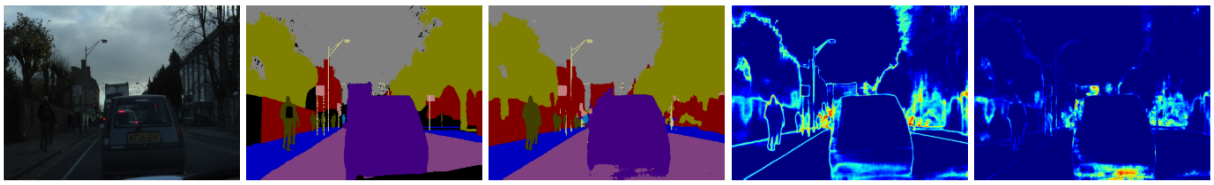


FIGURE 3.22 – Prédiction d'incertitude pour une tâche de segmentation sémantique. Les images montrent, de gauche à droite, l'image d'entrée, la segmentation de vérité terrain, la segmentation obtenue, la carte d'incertitude aléatorique obtenue et la carte d'incertitude épistémique obtenue [KG17].

Distribution de probabilité sur les sorties d'un réseau de neurones

Kendall *et al.* ont étudié la prédiction d'incertitude dans les réseaux de neurones dans [KG17] et ont notamment proposé une méthode pour estimer l'incertitude aléatorique. Dans le cas d'une régression, ils proposent de remplacer les sorties scalaires du réseau par des distributions de probabilité gaussiennes. Pour cela, les sorties sont dupliquées et permettent de prédire, à la fois, les valeurs de sortie moyennes, $\hat{\mathbf{y}} \in \mathbb{R}$, et leurs variances respectives $\hat{\sigma}_i^2$. Ils appliquent leur méthode à une tâche de segmentation sémantique d'image (voir figure 3.22) et la fonction objectif à minimiser pour entraîner le réseau devient alors

$$\mathcal{L}_{BNN} = \frac{1}{D} \sum_i \frac{1}{2} \hat{\sigma}_i^{-2} \left\| \mathbf{y}_i - \hat{\mathbf{y}}_i \right\|^2 + \frac{1}{2} \log \hat{\sigma}_i^2, \tag{3.12}$$

où D est le nombre de pixels de sortie, $\hat{\mathbf{y}}$ est la valeur moyenne de sortie pour le i$^{\text{ème}}$ pixel et $\hat{\sigma}_i^2$ est la covariance prédite correspondante. Ce coût se compose de deux éléments : un terme de résidu (à gauche) et un terme de régularisation (à droite). Le résidu est calculé sous la forme d'un erreur quadratique entre les valeurs moyennes prédites et celles de vérité terrain annotées et est pondéré par les variances prédites. Le point important de cette méthode est qu'elle ne nécessite pas de données annotées d'incertitude. Au contraire, seules les valeurs régressées nécessitent une supervision et l'apprentissage des covariances $\hat{\sigma}_i^2$ est fait de manière implicite à travers la fonction de coût. Le terme de régularisation $\log \hat{\sigma}_i^2$ empêche le réseau de prédire une incertitude infinie pour toutes les données d'entrées. En pratique, le réseau est entraîné à prédire la log variance, $s_i = \log \sigma_i$:

$$\mathcal{L}_{BNN} = \frac{1}{D} \sum_i \frac{1}{2} \exp(-s_i) \left\| \mathbf{y}_i - \hat{\mathbf{y}}_i \right\|^2 + \frac{1}{2} s_i. \tag{3.13}$$

Ce changement de variable permet d'éviter le cas d'une division par zéro et rend la prédiction plus stable numériquement. De même, le passage à l'exponentiel permet de régresser des valeurs scalaires non contraintes qui sont ensuite passées dans le domaine positif, donnant ainsi des valeurs de variance valides. Kendall *et al.* notent que l'incertitude aléatorique dans des tâches de régression peut être interprétée comme une atténuation du coût, rendant le modèle plus robuste à des données bruitées. Cette atténuation agit de la même manière qu'une fonction de

régression robuste adaptative et permet au réseau de pondérer les résidus. Pour des données d'entrée complexes, pour lesquelles le réseau ne parvient pas à prédire des résidus faibles, celui-ci peut tout de même diminuer le coût total en choisissant d'augmenter l'incertitude prédite.

Yang *et al.* ont appliqué cette technique pour prédire une incertitude photométrique dans le contexte d'odométrie visuelle monoculaire [YSWC20]. Truong-Le *et al.* ont repris cette idée d'atténuation de la fonction de coût pour estimer une incertitude de détection d'objet avec le réseau *Single Shot MultiBox Detector*[Liu+16] (SSD) pour des applications critiques [TDBK18]. Enfin, Dong *et al.* ont utilisé cette approche pour prédire des incertitudes d'ellipses dans le contexte d'estimation 3D multi-vue d'objets [DI21].

Incertitude de modèle

Les réseaux de neurones bayésiens ont été développés afin de mesurer l'incertitude de modèle. Dans ce type de réseau, les valeurs scalaires des poids sont remplacées par des variables aléatoires suivant une certaine distribution de probabilité (voir figure 3.23), choisie a priori, par exemple une distribution gaussienne.

L'entraînement d'un tel réseau, sur un ensemble de données $D = \{X, Y\}$ où X sont les données d'entrées et Y les valeurs de sortie annotées, revient à optimiser les paramètres ω d'une fonction $y = f^{\omega}(x)$ afin qu'elle produise les sorties désirées. La distribution de probabilité a posteriori des paramètres ω s'obtient en appliquant la formule de Bayes, telle que

$$p(\omega|X,Y) = \frac{p(Y|X,\omega)p(\omega)}{p(Y|X)} = \frac{p(Y|X,\omega)p(\omega)}{\int p(Y|X,\omega)p(\omega)d\omega}0 \tag{3.14}$$

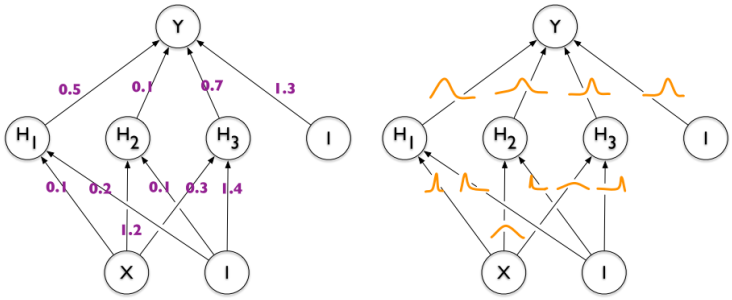


FIGURE 3.23 – Gauche : réseau de neurones classique où chaque poids est une valeur scalaire. Droite : réseau de neurones bayésien où les poids sont remplacés par des distributions de probabilité [BCKW15].

Pour des tâches de régression, une vraisemblance gaussienne est généralement choisie, donnée par

$$p(y|x,\omega) = \mathcal{N}(y; f^{\omega}(x), \tau^{-1}I), \tag{3.15}$$

où τ représente la précision du modèle et une vraisemblance sous la forme d'un *Softmax* est utilisée pour de la classification [Abd+21], telle que

$$p(y = c|x,\omega) = \frac{\exp(f_c^{\omega}(x))}{\sum_{c'} \exp(f_{c'}^{\omega}(x))}. \tag{3.16}$$

Pour une entrée x^*, la distribution de la prédiction du réseau est donnée par

$$\begin{aligned} p(y^*|x^*, X, Y) &= \mathbb{E}_{p(\omega|X,Y)}[p(y^*|x^*, \omega)] \\ &= \int p(y^*|x^*, \omega)p(\omega|X, Y)d\omega. \end{aligned} \tag{3.17}$$

Le principal défaut de cette formulation bayésienne d'un réseau de neurone vient du fait que la distribution a posteriori (équation 3.14) et la distribution prédictive (équation 3.17) ne sont pas calculables de manière analytique, mais doivent être approximées.

Inférence Variationnelle Une méthode pour résoudre ce problème est d'utiliser l'inférence variationnelle, qui permet d'approximer la distribution a posteriori $p(\omega|X, Y)$ par une distribution paramétrique $q_\theta(\omega)$. Les paramètres θ de cette distribution sont ajustés pour la rendre la plus proche possible de la vraie distribution p. La divergence de Kullback-Leibler permet de mesurer le degré de similarité entre deux distributions de probabilités et est définie par

$$KL[q_\theta(\omega)||p(\omega|X, Y)] = \int q_\theta(\omega) \log \frac{q_\theta(\omega)}{p(\omega|X, Y)} d\omega. \tag{3.18}$$

Les paramètres θ de la distribution $q_\theta(w)$ peuvent donc être estimés en minimisant la divergence de Kullback-Leibler entre les deux distributions, tel que

$$\begin{aligned} \theta^* &= \arg\min_\theta KL[q_\theta(\omega)||p(\omega X, Y)] \\ &= \arg\min_\theta \int q_\theta(\omega) \log \frac{q_\theta(\omega)}{p(\omega)p(Y|X, \omega)} d\omega \\ &= \arg\min_\theta KL[q_\theta(\omega)||p(\omega)] - \int q_\theta(\omega) \log p(Y|X, \omega)d\omega \\ &= \arg\min_\theta KL[q_\theta(\omega)||p(\omega)] - \mathbb{E}_{q_\theta(\omega)}[\log p(Y|X, \omega)]. \end{aligned} \tag{3.19}$$

Cette minimisation de la divergence de Kullback-Leibler est également connue sous la forme de la maximisation de l'*evidence lower bound* (ELBO). On peut noter que la fonction de coût contient à la fois un terme visant à respecter la distribution a priori choisie pour les poids du réseau $p(\omega)$ (le terme de gauche) et un terme cherchant à satisfaire la complexité des données $\{X, Y\}$ (le terme de droite). La distribution prédictive approximée devient alors

$$p(y^*|x^*, X, Y) \approx \int p(y^*|x^*, \omega)q_\theta(\omega)d\omega. \tag{3.20}$$

Blundell *et al.* ont developpé une méthode d'optimisation de ce coût ELBO, appelée *Bayes by Backprop* [BCKW15], qui est compatible avec la rétropropagation du gradient utilisée pour l'entraînement des réseaux de neurones. Ils montrent que l'incertitude apprise dans les poids d'un réseau permet d'améliorer sa généralisation pour des problèmes de régression non linéaires.

Monte Carlo Dropout Une autre technique permettant d'estimer l'incertitude de modèle d'un réseau est le *Monte Carlo dropout* [GG16]. Pour rappel, le *dropout* consiste à désactiver temporairement certains neurones d'un réseau, ainsi que toutes ses connexions entrantes ou sortantes. Ce choix de désactiver un neurone est répété, de manière aléatoire, à chaque époque

avec une certaine probabilité. La configuration des neurones est donc différente à chaque passage dans le réseau. C'est une technique bien connue qui permet d'éviter le sur-ajustement des poids du réseau sur les données d'entraînement. Gal *et al.* [GG16] ont montré que l'utilisation du *dropout* à l'inférence pouvait être interprétée comme de l'inférence variationnelle. Cela signifie que l'exécution du *dropout* peut être considérée comme un échantillonnage des poids à partir de la distribution variationnelle $q_\theta(\omega)$. Ils ont également montré qu'entraîner un réseau avec du *dropout* et une régularisation de type *L2 weight decay* est équivalent à maximiser l'*evidence lower bound* (ELBO).

Le principe est donc, à l'inférence, de réaliser plusieurs passages dans le réseau en activant le *dropout*, afin d'obtenir des prédictions multiples et estimer leur variance. L'utilisation du *Monte Carlo Dropout* pour estimer l'incertitude de la prédiction d'un réseau de neurone est avantageuse car elle ne nécessite pas de modifier l'architecture du réseau. De plus, Loquercio *et al.* [LSS20] ont montré qu'il peut être appliqué à un réseau déjà entraîné sans besoin de ré-entraînement.

Cette méthode a été utilisée dans plusieurs travaux. Une des premières applications, *Bayesian SegNet* [KBC17], a été proposée par Kendall *et al.* et permet de prédire une incertitude pour chaque pixel d'une segmentation sémantique d'image. Miller *et al.* ont étudié son utilisation pour la détection probabiliste d'objets [MNDS18] avec le réseau SSD. Dans le même esprit, Morrison *et al.* l'ont intégré dans le réseau de segmentation d'instances Mask R-CNN [HGDG17] pour obtenir des incertitudes sémantiques et spatiales [Mor19]. Wang *et al.* l'ont utilisé pour estimer l'incertitude de réseaux de neurones convolutifs dans le contexte de segmentation d'images médicales [Wan+19a]. Liu *et al.* ont proposé un modèle unifié permettant d'approximer les incertitudes épistémiques et aléatoriques de réseaux de neurones, en présence de perturbations antagonistes [Liu+19]. En particulier, l'incertitude épistémique a été estimée en utilisant du *Monte Carlo dropout*. Dans le cadre de la navigation autonome, Amini *et al.* ont proposé de l'utiliser pour calculer l'incertitude d'un modèle de contrôle du véhicule. Dans le même contexte, Loquercio *et al.* ont développé un système permettant de propager l'incertitude liée au bruit du capteur dans un réseau bayésien et le combine avec du *Monte Carlo dropout* pour estimer, à la fois, l'incertitude épistémique et l'incertitude aléatorique [LSS20]. Ils appliquent ce système à la prédiction de l'angle de direction d'un véhicule.

Le *Monte Carlo dropout* a également été couplé avec la méthode d'atténuation du coût, présentée précédemment, pour permettre d'estimer les deux types d'incertitudes. Par exemple, Phan *et al.* les ont combinés pour prédire des incertitudes de localisation d'un objet [Pha+18] et Kraus *et al.* les ont intégrés dans le réseau de détection d'objets YOLOv3 [KD19]. Enfin, Feng *et al.* les ont appliqués à de la détection de véhicules dans des données Lidar [FRD18]. Ils ont utilisé le jeu de données KITTI [GLSU13] et une architecture de réseau basée sur le détecteur à deux étapes Faster R-CNN [RHGS17]. Le *dropout* a été placé sur les couches entièrement connectées, après le réseau de proposition de régions (RPN). Les auteurs ont noté que la modélisation de l'incertitude aléatorique avait permis un gain de performances de 1 à 5%, mais que l'utilisation du *dropout* à l'inférence les ont légèrement dégradées.

Outre cette baisse de performances, l'utilisation du *dropout* à l'inférence présente quelques inconvénients :

— Les passages multiples dans le réseau rendent le coût de calcul relativement élevé.

— Les couches du réseau où placer le *dropout* ainsi que la probabilité à utiliser pour la désactivation d'un neurone ne sont pas toujours clairement établies dans la littérature. Pourtant, ces choix ont un impact important sur les résultats obtenus.

— Dans le cadre de la détection probabiliste d'objets, les différents passages dans le réseau produisent de nombreuses boîtes de détection qui sont ensuite regroupées afin d'estimer

leurs variances (voir figure 3.24). Ce groupement est en soi un véritable problème. Miller *et al.* ont étudié plusieurs méthodes [MDMS19] : une approche séquentielle, l'algorithme Hongrois [Kuh10] et HDBSCAN [MHA17]. Ces approches reposent sur le calcul d'un recouvrement entre les détections et imposent un recouvrement minimal entre deux détections pour appartenir au même groupe. L'incertitude spatiale estimée des boîtes de détection est donc fortement impactée par le choix de ce seuil de recouvrement minimal. Miller *et al.* proposent d'utiliser un seuil de 90%, mais qui semble contraindre fortement la variance spatiale des boîtes de détection.

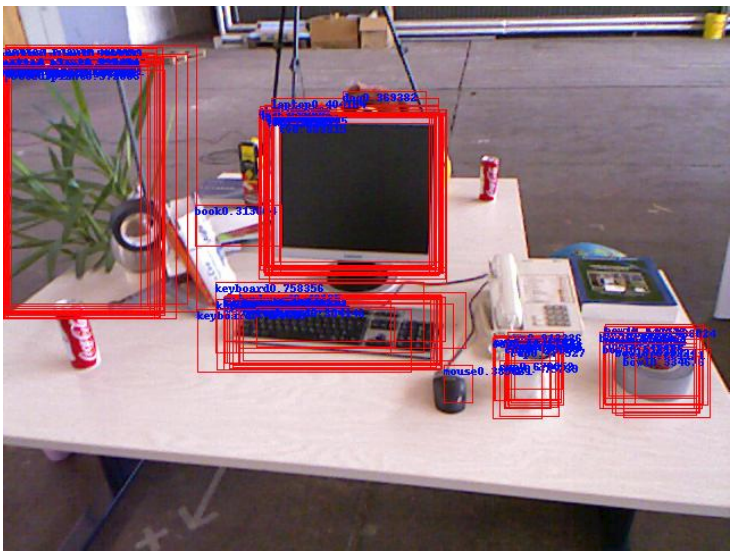


FIGURE 3.24 – Illustration de l'utilisation du *Monte Carlo dropout* à l'inférence avec le réseau détection d'objets SSD.

Méthodes par ensembles Lakshminarayanan *et al.* ont proposé une méthode alternative, non-bayésienne, permettant de quantifier l'incertitude de prédiction d'un réseau de neurones [LPB17]. Ils décrivent leur approche comme simple à implémenter, parallélisable et nécessitant très peu d'ajustements d'hyperparamètres. Plutôt que d'annuler de manière aléatoire certains neurones, ils proposent d'apprendre plusieurs valeurs de poids $\{\omega^{(m)}\}_{m=1}^{M}$ pour le réseau. Pour cela, celui-ci est entraîné plusieurs fois, à partir d'initialisations différentes des poids, générées aléatoirement. Ils proposent de moyenner les différents modèles pour obtenir la distribution prédictive finale suivante :

$$p(y^*|x^*) = \frac{1}{M}\sum_{m=1}^{M} p(y^*|x^*, \omega^{(m)}), \tag{3.21}$$

où M désigne le nombre total de réseaux entraînés. Ils ont évalué leur approche sur des problèmes de régression et classification et ont montré qu'elle surpasse la méthode de *Monte Carlo dropout*.

Gustafsson *et al.* [GDS20] interprètent les différentes valeurs des poids comme des échantillons d'une distribution q_θ, qui serait similaire à celle utilisée pour approximer la distribution a posteriori dans les méthodes d'inférence variationnelle présentées précédemment. Ils considèrent donc tout de même cette méthode comme une approche bayésienne. Ils proposent une méthode d'évaluation de l'incertitude épistémique et comparent les approches par ensembles avec le *Monte Carlo dropout*. Leurs résultats montrent que les méthodes par ensembles fournissent des valeurs

d'incertitude plus fiables en pratique et ils en concluent que cette approche devrait être considérée comme la nouvelle méthode à privilégier pour l'estimation de l'incertitude épistémique. Le principal défaut de cette approche vient, à nouveau, de son coût de calcul élevé. En effet, elle nécessite plusieurs entraînements du réseau et des passages multiples dans le réseau à l'inférence, ce qui limite ses applications. Les entraînements et les inférences peuvent être réalisés en parallèle mais nécessitent une infrastructure de calcul conséquente.

3.4.2 Prédiction d'une incertitude globale par ellipse

Dans notre cas, nous nous intéressons en particulier à l'incertitude aléatorique hétéroscédastique, qui nous permettrait d'identifier les objets détectés les plus sûrs à utiliser pour estimer la pose de la caméra. Plutôt que de modéliser chaque paramètre de l'ellipse prédite par une distribution gaussienne univariée comme dans [DI21], nous cherchons à prédire une incertitude globale qui représente la qualité géométrique de cette ellipse. Pour cela, nous modélisons la distance entre ellipses $\mathcal{D}^2(\mathcal{E}_{pred}, \mathcal{E}_{gt})$, décrite dans la section 2.2.5 et qui est utilisée comme fonction de coût dans l'entraînement du réseau de prédiction, par une distribution gaussienne. La fonction de coût à minimiser devient

$$\mathcal{L}_{unc} = \frac{1}{2}\sigma^{-2}\mathcal{D}^2(\mathcal{E}_{pred}, \mathcal{E}_{gt}) + \frac{1}{2}\log\sigma^2. \tag{3.22}$$

Comme expliqué précédemment, nous entraînons le réseau à prédire la log-variance $\alpha = \log\sigma^2$ afin d'éviter des problèmes d'instabilité numérique lorsque σ est petit. La fonction de coût devient alors

$$\mathcal{L}_{unc} = \frac{1}{2}\exp(-\alpha)\mathcal{D}^2(\mathcal{E}_{pred}, \mathcal{E}_{gt}) + \frac{1}{2}\alpha, \tag{3.23}$$

où α correspond au paramètre de sortie du réseau correspondant à l'information d'incertitude. Afin de prédire ce paramètre d'incertitude supplémentaire, nous avons ajouté une couche de sortie entièrement connectée à la fin du perceptron multicouche (MLP) du réseau de prédiction d'ellipse. Cette nouvelle architecture est illustrée dans la figure 3.25.

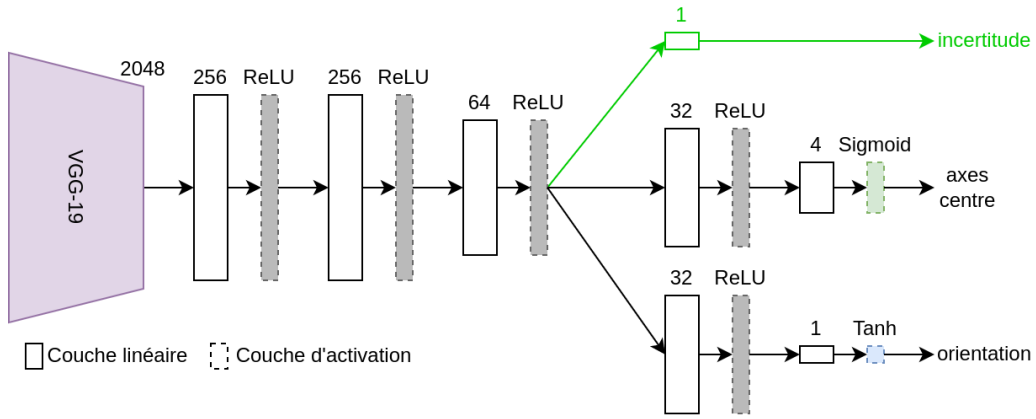


FIGURE 3.25 – Architecture du réseau de prédiction d'ellipse modifiée pour prédire une valeur d'incertitude. La branche verte a été ajoutée par rapport à l'architecture de base présentée dans la figure 2.10.

3.4.3 Calcul de pose pondéré par objet

Une incertitude est donc associée à chaque ellipse prédite et peut être utilisée pour pondérer la contribution de chaque objet dans l'étape de raffinement. Plus précisément, nous utilisons l'inverse de la variance prédite, σ^{-2}, comme facteur de pondération du coût de chaque objet. Cela permet de donner plus d'importance aux objets dont l'incertitude est faible et, inversement, de diminuer le poids des objets avec une incertitude élevée. L'opération de minimisation devient

$$\hat{R}, \hat{t} = \arg\min_{R,t} \sum_{j=1}^{N_{obj}} \sigma_j^{-2} \Delta(\mathcal{E}_j, P Q_j^* P^T), \tag{3.24}$$

où σ_j^{-2} correspond à l'inverse de la valeur d'incertitude prédite pour le j$^{\text{ème}}$ objet, Q_j est la matrice de la forme duale de son ellipsoïde et $\mathcal{E}_j$ est l'ellipse de détection qui lui a été associée. La matrice $P = K[R|t]$ est la matrice de projection de la caméra, avec K sa matrice de calibration et $[R|t]$ ses paramètres extrinsèques, qui sont les variables recherchées. N_{obj} est le nombre d'objets considérés dans le raffinement et $\Delta(\cdot)$ correspond au coût entre ellipses que l'on cherche à minimiser, typiquement à la métrique *Level sets*.

3.4.4 Expériences et résultats

Corrélation entre l'incertitude prédite et qualité de l'ellipse

Nous avons, tout d'abord, cherché à vérifier la fiabilité et le sens des valeurs d'incertitude prédites. Nous avons donc cherché à analyser la valeur de l'incertitude lorsque l'ellipse prédite est de mauvaise qualité. Pour obtenir de mauvaises prédictions d'ellipses, nous avons bruité les pixels de certaines zones des images d'entrée du réseau. La figure 3.26 illustre cette expérience. L'ellipse prédite est en vert et l'ellipse de vérité terrain en rouge. La valeur indiquée en haut à gauche de chaque image correspond à la valeur d'incertitude prédite par le réseau. On peut observer une certaine corrélation entre la qualité de l'ellipse prédite et la valeur d'incertitude associée. En effet, la valeur de l'incertitude augmente avec la dégradation de l'ellipse prédite.

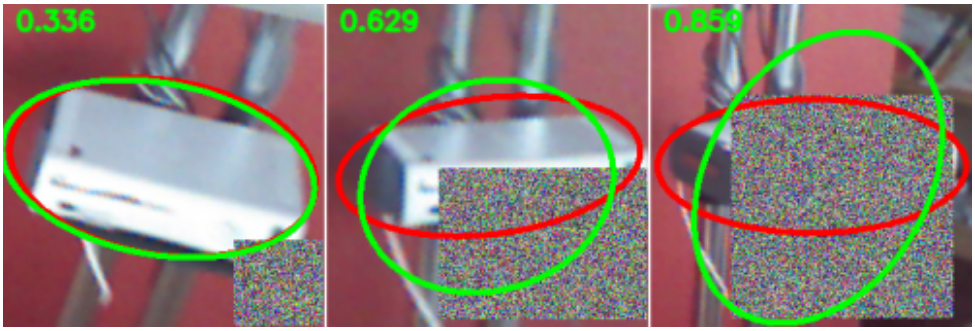


FIGURE 3.26 – Corrélation entre la qualité de l'ellipse prédite et l'incertitude σ^2 fournie par le réseau (en haut à gauche). L'ellipse prédite est en vert et la vérité terrain en rouge.

Résultats sur Chess

Nous avons ensuite évalué le gain apporté par la pondération des contributions des objets basée sur les incertitudes prédites. La figure 3.27 compare les résultats obtenus avec et sans pondération pour les métriques *Level sets*, *Wasserstein*, *QBbox* et *Alg. Vectorized*.

On peut, tout d'abord, remarquer que la pondération basée sur l'incertitude des ellipses permet d'obtenir de meilleurs résultats pour chaque métrique. Il est intéressant de noter que le bénéfice est légèrement plus important pour la métrique *Wasserstein*, plutôt que *Level sets* ou *QBbox*. Cela peut s'expliquer par le fait que, au contraire de *Wasserstein*, ces deux dernières traitent naturellement mieux les objets partiellement visibles (dans toute l'image pour *Level sets* et au niveau des sorties d'image pour *QBbox*).

Les courbes sur la figure 3.28 montrent les erreurs de position obtenues dans des portions des séquences de test, pour lesquelles la pondération par incertitude des objets apporte un bénéfice important. Des comparaisons entre l'utilisation ou non de cette pondération sont montrées dans les figures 3.29, 3.30, 3.31 et 3.32. Les ellipses vertes, en trait continu, sont celle utilisées pour le raffinement, les ellipses discontinues correspondent aux projections des objets et les ellipses blanches sont les projections avec la pose de vérité terrain. Les valeurs indiquées en orange sur les images de droite sont les facteurs de pondération de chaque objet, c'est-à-dire à l'inverse de l'incertitude prédite. Les erreurs de position et d'orientation de la caméra estimée sont indiquées en vert.

Sur la figure 3.29, on peut tout d'abord noter que, sans incertitude, la métrique *Level sets* surpasse celle de *Wasserstein*. Cela confirme les résultats obtenus dans la section 3.3 et peut s'expliquer par le fait que le dossier de chaise sur la droite, se trouve en partie en dehors de l'image et que son ellipse de détection est seulement partielle. On note ensuite que la pondération des coûts des objets par l'inverse de l'incertitude apporte un gain de précision pour les deux métriques. L'erreur obtenue sur la position passe en effet de 15.10 cm à 9 cm et celle sur l'orientation de 6.04° à 3.70° avec la métrique *Level sets*. On peut voir que le réseau a beaucoup moins confiance dans la détection du dossier de la chaise sur la droite de l'image et lui a donc attribué une contribution plus faible dans l'optimisation. On peut effectivement observer que sa projection est moins contrainte par sa détection (qui est d'ailleurs de mauvaise qualité). Au contraire, les autres objets, avec des pondérations plus fortes, obtiennent un meilleur alignement entre détection et projection. Le constat est similaire dans les autres images, figures 3.30, 3.31 et 3.32, c'est-à-dire que donner davantage de poids aux objets avec une confiance élevée dans leur détection permet d'atteindre une meilleure précision de pose.

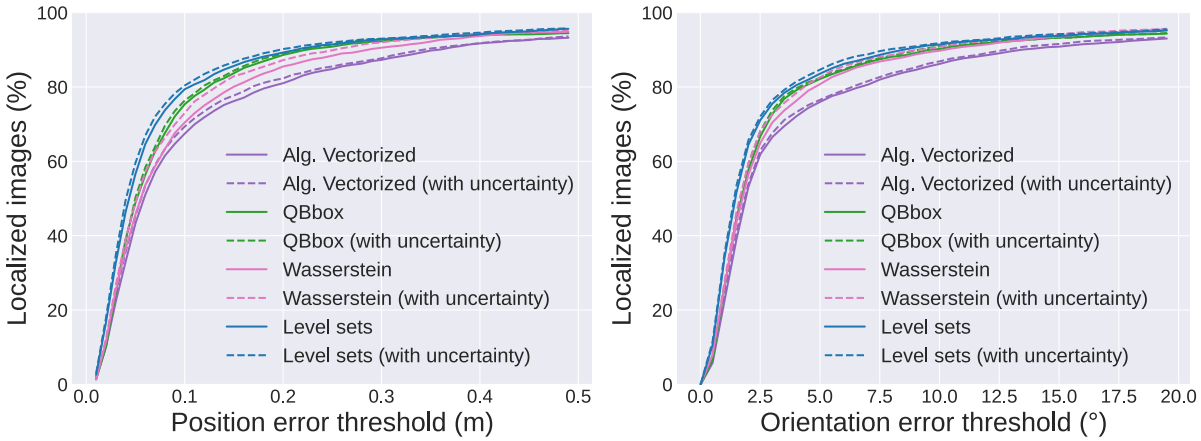


FIGURE 3.27 – Analyse de l'apport de l'incertitude dans le raffinement de la caméra. Les courbes montrent les pourcentages d'images correctement localisées en fonction d'une erreur de position (à gauche) ou d'orientation (à droite), pour différentes métriques, avec ou sans la pondération par l'incertitude.

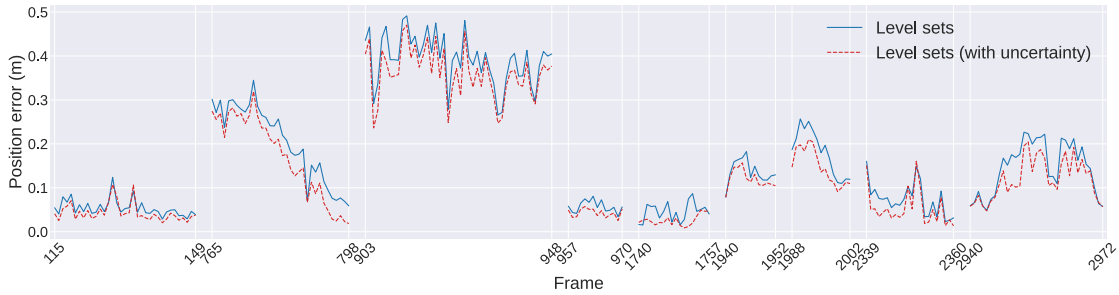


FIGURE 3.28 – Erreurs de position obtenues en utilisant la métrique *Level sets*, avec et sans incertitude, sur des portions des séquences de test de la scène *Chess*.

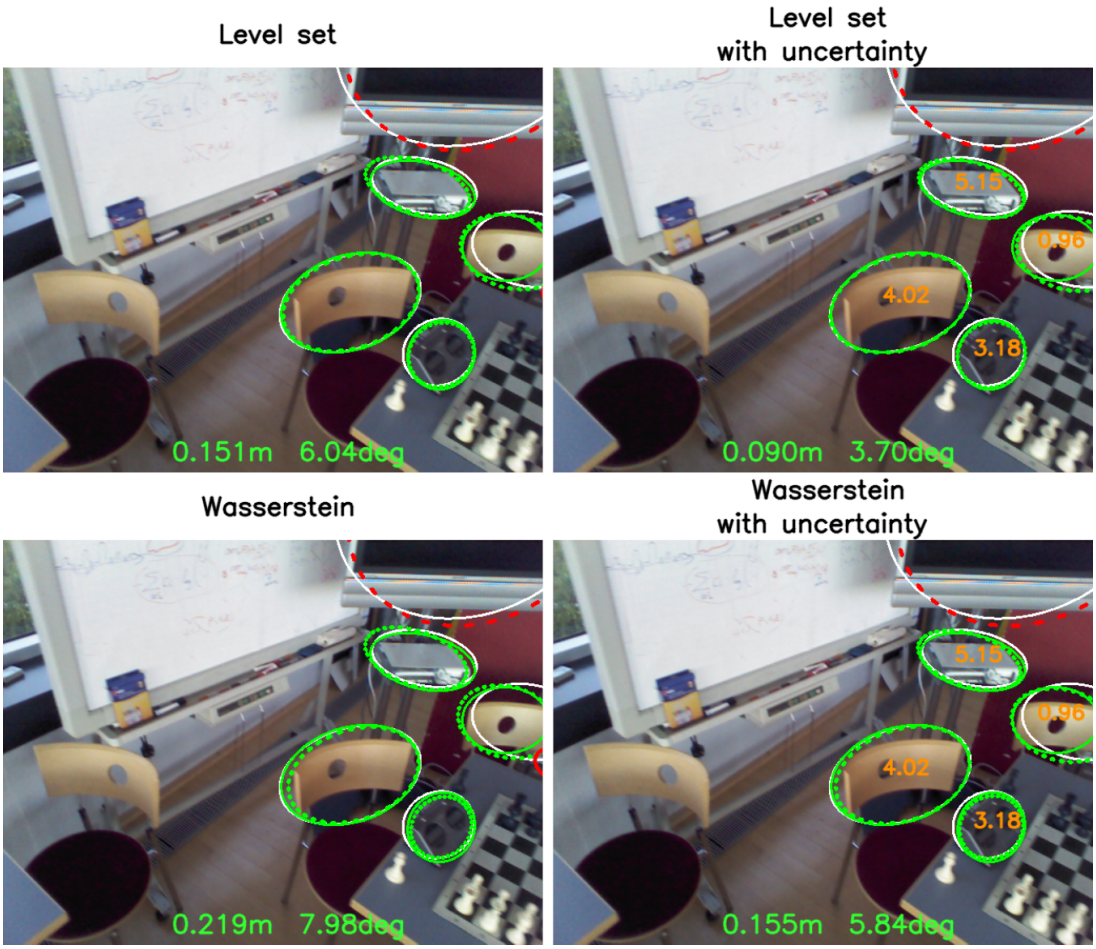


FIGURE 3.29 – Pose estimée de la caméra avec les métriques *Level sets* et *Wasserstein*, avec et sans utilisation de l'incertitude. Les ellipses vertes continues sont les détections et celles discontinues correspondent aux projections des objets en utilisant la pose estimée de la caméra. Les ellipses vertes sont utilisées dans le raffinement de la caméra, les ellipses rouges sont des projections d'objets non détectés dans l'image et les ellipses blanches montrent les projections des objets avec la pose de vérité terrain. Dans le cas utilisant l'incertitude (colonne de droite), les coefficients de pondération (σ^{-2}) des objets sont indiqués en orange. Les erreurs de position et d'orientation sont indiquées en bas.

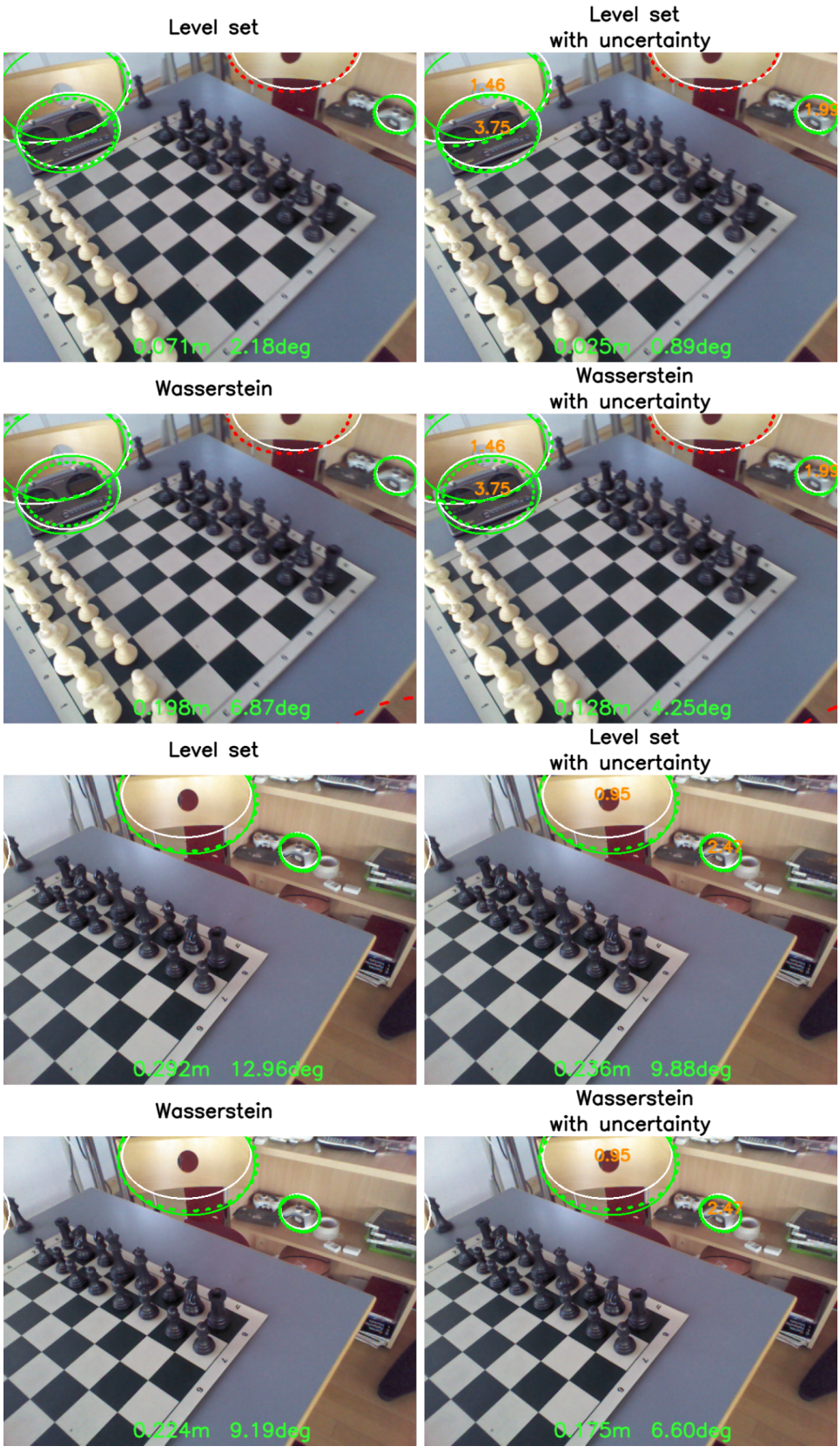


FIGURE 3.30 – Pose estimée de la caméra avec les métriques *Level sets* et *Wasserstein*, avec et sans utilisation de l'incertitude. La légende est identique à la figure 3.29.

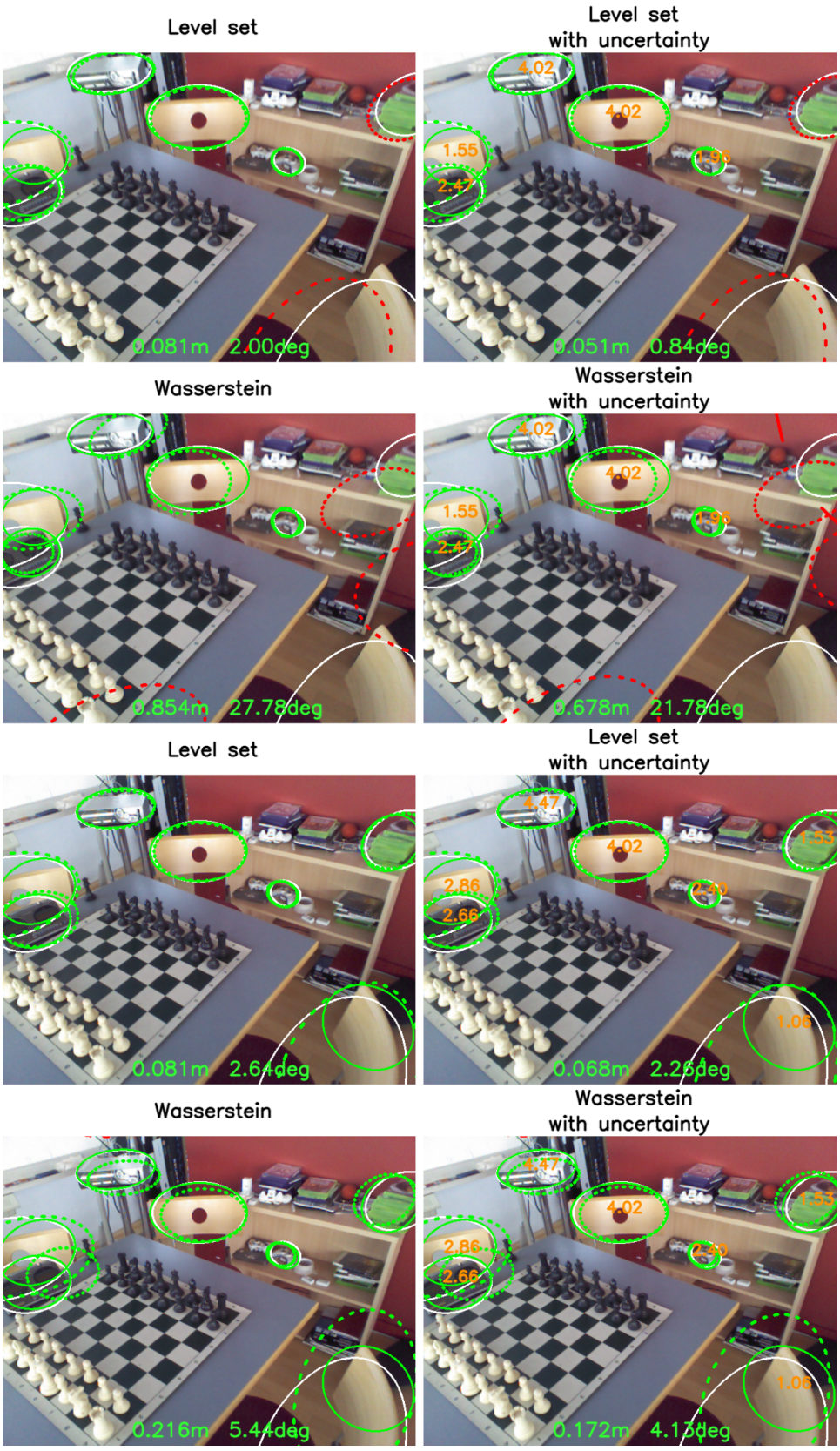


FIGURE 3.31 – Pose estimée de la caméra avec les métriques *Level sets* et *Wasserstein*, avec et sans utilisation de l'incertitude. La légende est identique à la figure 3.29.

FIGURE 3.32 – Pose estimée de la caméra avec les métriques *Level sets* et *Wasserstein*, avec et sans utilisation de l'incertitude. La légende est identique à la figure 3.29.

3.5 Conclusion

Dans ce chapitre, nous avons donc proposé une étape de raffinement de la pose de caméra, qui permet de prendre en considération tous les objets de la scène détectés dans l'image. Cette étape cherche effectivement à aligner les cônes de projection et rétroprojection lors du calcul de la pose de la caméra et permet de relâcher la supposition, théoriquement fausse, de l'alignement par projection centrale des centres des ellipsoïdes avec les centres des ellipses. Elle s'exprime sous la forme d'une minimisation de l'erreur entre les ellipses détectées dans l'image et celles obtenues par la projection des modèles ellipsoïdaux des objets. Nous avons comparé différentes métriques permettant d'établir un coût entre deux ellipses et nos expériences ont montré que la métrique *Level sets*, que nous proposons, donne les meilleurs résultats et permet d'améliorer de manière significative la précision de la pose estimée initialement. Nous avons montré ses bonnes propriétés de convergence ainsi que sa meilleure gestion des objets partiellement détectés dans l'image. Enfin, nous avons proposé une méthode de quantification de l'incertitude des ellipses de détection prédites par notre réseau de neurones, proposé au chapitre 2, et nous avons montré comment son utilisation pour pondérer la contribution de chaque objet a permis d'améliorer la précision du calcul de pose.

Chapitre 4

SLAM basé objet

Sommaire

Dans les deux chapitres précédents, nous nous sommes intéressés au calcul de pose de caméra basé sur les objets uniquement et nous avons cherché, en particulier, à améliorer sa précision. Dans un premier temps, nous avons proposé un réseau de prédiction d'ellipse afin d'obtenir de

meilleures détections des objets dans les images, qui soient cohérentes avec les modélisations ellipsoïdales des objets. Dans un second temps, nous avons ajouté une étape de raffinement de la pose, sous la forme d'une minimisation d'erreurs de reprojection, permettant ainsi de prendre en compte tous les objets détectés dans l'image et d'améliorer la précision de la pose estimée. Ces méthodes nécessitent cependant une étape d'apprentissage spécifique à la scène traitée et sont donc particulièrement utiles pour des scénarios dans lesquels une première prise de vue de la scène est possible, afin d'obtenir des données d'apprentissage et d'entraîner les réseaux. Dans ce dernier chapitre, nous nous intéressons à des applications plus générales, dans des environnements inconnus, ne nécessitant pas de phases d'apprentissage spécifiques. Notre objectif est d'intégrer la notion d'objet dans un système de SLAM, permettant une cartographie automatique des objets de la scène et d'en tirer avantage pour améliorer la robustesse de sa relocalisation. Pour cela, nous étudions la collaboration des points et des objets, afin de combiner leurs avantages respectifs de robustesse et précision.

Nous débutons ce chapitre par des rappels généraux sur le SLAM dans la section 4.1. Nous y présentons des approches classiques considérées comme des références mais également des méthodes qui intègrent de l'information sémantique. Nous décrivons ensuite notre système de SLAM aidé par les objets dans la section 4.3. Enfin, nous analysons et comparons les performances de notre système sur différents jeux de données dans la section 4.4 et démontrons son intérêt pour des applications de réalité augmentée dans la section 4.5. Les travaux décrits dans ce chapitre ont fait l'objet de la publication [ZSB22b].

4.1 Rappels sur le SLAM

4.1.1 Méthodes par filtrage

Historiquement, les premières approches de localisation et cartographie simultanées reposaient sur des filtres, tels que les filtres de Kalman [DRMS07 ; CDM08] ou les filtres particulaires [ED06]. Ces méthodes estiment la distribution de probabilité jointe entre la pose de la caméra et les positions de points 3D de la scène et les mettent continuellement à jour. Elles alternent généralement deux étapes. Une étape de prédiction du prochain état du système en propageant les distributions et une étape de correction qui corrige l'état courant du système à partir des mesures faites par les capteurs. L'historique des poses de la caméra n'est généralement pas conservé et l'estimation de la pose courante se fait à partir de la pose précédente, de la carte et des mesures. MonoSLAM [DRMS07], proposé par Davison *et al.*, est l'une des premières méthodes de SLAM utilisant un filtre de Kalman étendu (EKF). Elle est notamment la première à obtenir des performances de localisation et cartographie en temps réel à partir d'une caméra monoculaire. Elle repose principalement sur une forme probabiliste de carte qui représente, à chaque instant, l'estimation courante de l'état du système avec la pose de la caméra, les points 3D reconstruits et les incertitudes associées. Ils supposent que chaque point 3D de la carte se trouve sur une surface localement plate, orientée orthogonalement au rayon entre le point 3D et le centre de la caméra. Cette texture 3D orientée est projetée dans les nouvelles images pour permettre le suivi du point.

L'inconvénient majeur des systèmes de SLAM basés sur un EKF est son coût de calcul qui augmente de manière quadratique avec la taille de la carte. Strasdat *et al.* [SMD10] ont montré que pour améliorer la précision d'un système de SLAM monoculaire, il était préférable d'augmenter le nombre de balises 3D considérées. Ils concluent donc que les approches basées sur des filtres peuvent être intéressants dans des environnements restreints et avec des capacités de calcul limitées, mais que des approches par optimisation sont préférables dans les autres cas.

4.1.2 Méthodes par optimisation

Les approches par optimisation sont souvent privilégiées pour traiter ce problème de localisation et cartographie simultanées dans le cas général. Celles-ci sont composées de deux parties : un module de localisation (ou *frontend*) et un module de cartographie (ou *backend*).

— Le *frontend* récupère les images du capteur, y détecte et extrait des points d'intérêt et les associe avec des points existants de la carte locale. Une première estimation de la pose est obtenue à partir de ces associations 2D-3D, généralement en utilisant un algorithme PnP [LMF09].

— Le *backend* traite la cartographie locale et l'ajustement de faisceaux global permettant de raffiner les poses de la caméra, les positions des points 3D de la carte, de prendre en considération des contraintes globales et de corriger la dérive causée par l'accumulation d'imprécisions (*drift*). Ce module est plus coûteux en temps de calcul et est donc généralement exécuté de manière asynchrone dans un thread séparé.

Afin de garder une taille d'optimisation raisonnable, ces méthodes ont introduit la notion d'image clé. En effet, toutes les poses de la caméra ne sont pas retenues, mais seulement celles des images clés. Celles-ci sont choisies suivant certains critères de champ de vision et de déplacement par rapport aux images précédentes. De même, seules ces images clés sont utilisées pour reconstruire de nouveaux points dans la carte.

Parmi les méthodes de SLAM par optimisation, on peut distinguer deux approches : les méthodes directes et les méthodes basées sur des points d'intérêt. Elles se différencient par la manière dont elles utilisent les images d'entrée. Leurs différences sont illustrées sur la figure 4.1.

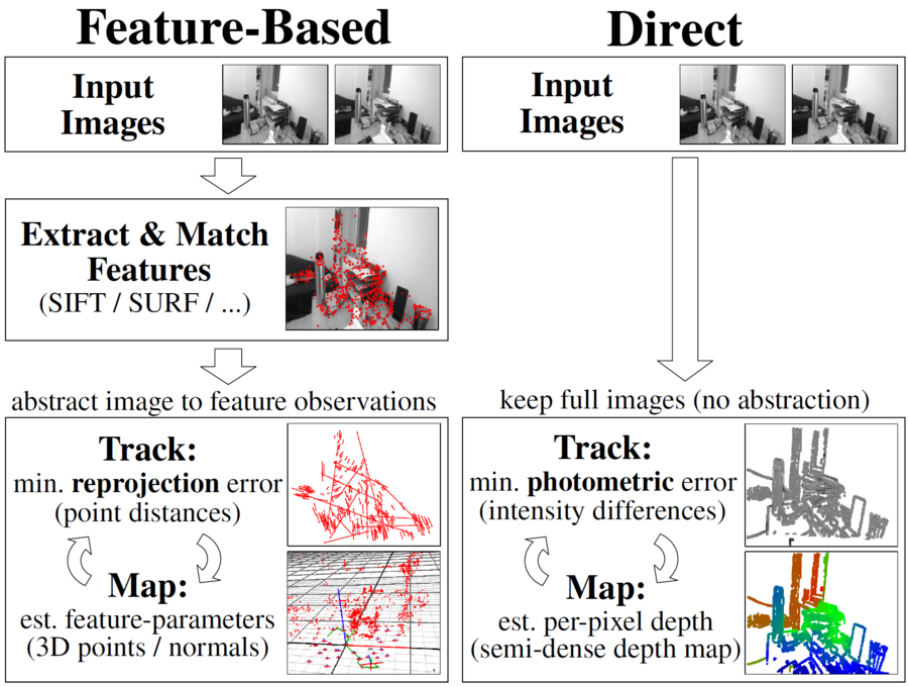


FIGURE 4.1 – Comparaison entre les approches de SLAM basées sur des points d'intérêt (ou indirectes) et les approches directes [ESC14].

4.1.3 Méthodes par points d'intérêt (indirectes)

L'approche par points d'intérêt est la façon plus courante de résoudre ce problème. De manière similaire aux méthodes de localisation basées sur la structure 3D de la scène, présentée dans la section 1.1, cette approche extrait des points d'intérêt dans les images et les associe à des points 3D existants, pour permettre l'estimation de la pose courante de la caméra. De nouveaux points 3D peuvent également être créés. Une mise en correspondance efficace et robuste des points est primordiale. Elle repose généralement sur des descripteurs locaux, tels que ceux présentés dans la section 1.1, par exemple SIFT [Low04], SURF [BTG06] ou ORB [RRKB11]. Ces méthodes utilisent ensuite un ajustement de faisceaux, illustré dans la figure 4.2, pour raffiner les poses de la caméra (courante et passées) et la structure de la scène (points 3D). Celui-ci minimise l'erreur géométrique de reprojection des points 3D par rapport à leurs observations dans les images et s'exprime sous la forme :

$$\underset{R_j,t_j,\mathbf{X}_i}{\arg\min} \sum_{i}^{N} \sum_{j \in O_i} \|\mathbf{x}_{ij} - \pi(R_j \mathbf{X}_i + t_j)\|^2_{\Sigma_{ij}} \tag{4.1}$$

où $\mathcal{O}_i$ contient les observations du i$^{\text{ème}}$ point 3D, $\mathbf{x}_{ij}$ correspond à l'observation du point 3D $\mathbf{X}_i$ dans l'image j et π est l'opération de projection. R_j et t_j sont les paramètres extrinsèques (rotation et translation) de la caméra à l'image i et Σ_{ij} est la matrice d'information de l'observation $\mathbf{x}_{ij}$. Cette optimisation prend la forme d'un problème de moindres carrés non linéaires et peut être résolue, par exemple, avec l'algorithme de Levenberg-Marquardt (LM). En pratique, elle est généralement représentée sous la forme d'un graphe, combinant les poses de la caméra et les points 3D de la carte. Différentes bibliothèques existent et permettent de résoudre ce type de problème, telles que : g2o [Küm+11], GTSAM [Del+22], iSAM [KRD08] ou Ceres [AMT22].

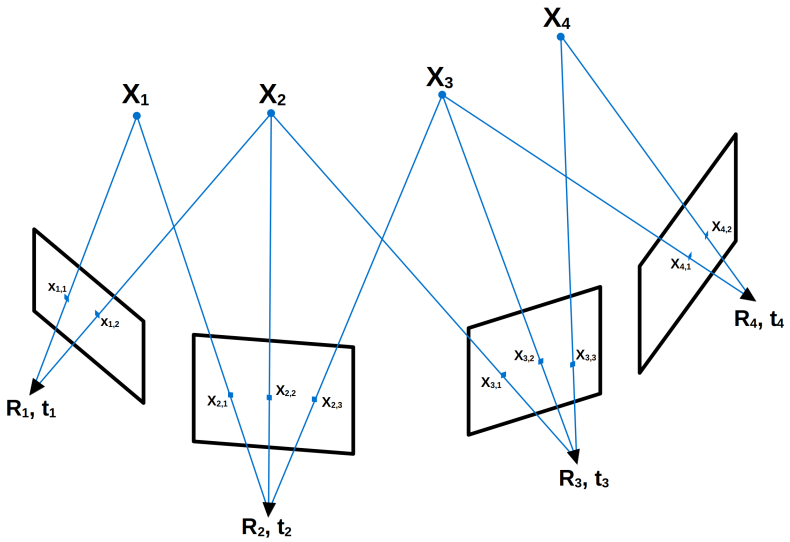


FIGURE 4.2 – Ajustement de faisceaux entre 4 points et 4 poses de caméra.

Parallel Tracking and Mapping (PTAM) [KM07] a été proposé en 2008 par Klein *et al.* . Il est l'un des premiers systèmes à proposer un découplage entre le suivi de la caméra, peu coûteux en temps de calcul, et la cartographie. Ces deux tâches sont exécutées en parallèle dans des threads séparés et à des fréquences différentes. Les auteurs démontrent la possibilité d'atteindre une exécution en temps réel du système, tout en incluant un ajustement de faisceaux qui, jusque-là,

était plutôt utilisé hors ligne dans une étape de post-traitement. Le suivi de la caméra repose sur le détecteur de points d'intérêt FAST [RD06]. Pour la cartographie, PTAM utilise la notion d'images clés qui permettent, à la fois, de garantir un déplacement suffisant entre les images utilisées pour trianguler de nouveaux points 3D et de réduire le coût de calcul. Le système introduit un ajustement de faisceaux local qui, au contraire de l'ajustement global, s'exécute seulement sur les images clés voisines de l'image courante. Le système est initialisé à partir de l'algorithme à 5 points [Nis03]. Son fonctionnement est illustré dans la figure 4.3.

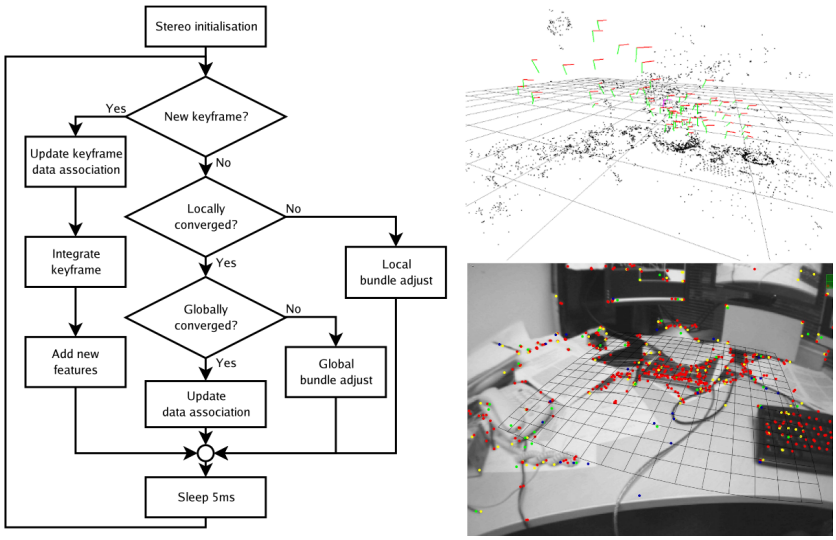


FIGURE 4.3 – Architecture du système Parallel Tracking and Mapping (PTAM) [KM07], à gauche, et illustration du suivi de la caméra et de la carte reconstruite à droite.

ORB-SLAM [MMT15] a constitué une évolution importante de PTAM, qui améliore ses performances et sa robustesse. Le détecteur de points d'intérêt FAST et la description par patchs locaux de l'image ont été remplacés par ORB [RRKB11], une combinaison entre *Oriented FAST* and *rotated BRIEF*. Par rapport à PTAM qui optimise des poses de caméra à six degrés de liberté, ORB-SLAM estime des transformations de type similitude qui combinent une orientation, une position et une échelle. Cela lui permet de corriger des accumulations d'erreurs d'échelle, dues à l'ambiguïté inhérente aux données d'entrée monoculaires. Son architecture, détaillée dans la figure 4.4, est composée de trois modules qui sont exécutés en parallèle dans des threads séparés :

— Le module de suivi (*Tracking*), qui estime les poses succéssives de la caméra, à partir des images précédentes, d'un éventuel modèle de mouvement, des points d'intérêt détectés et de la carte courante reconstuite. C'est également lui qui décide si une image doit être considérée comme une nouvelle image clé et insérée dans le module de cartographie locale. Pour cela, une image doit être suffisamment différente de l'image précédente et avoir assez de points clés détectés et associés.

— Le module de cartographie locale (*Local Mapping*), qui met à jour la carte en introduisant de nouveaux points 3D, triangulés à partir d'observations 2D, lorsqu'une nouvelle image clé est intégrée. Une stratégie assez stricte est utilisée pour ne garder que des points de très bonne qualité et éviter d'introduire un *outlier* dans la carte. Un ajustement de faisceaux local est réalisé pour raffiner la carte et les poses des images clés voisines de l'image courante. Cette notion de voisinage est maintenue dans un graphe de covisibilité qui regroupe les images clés observant les mêmes points 3D que l'image courante.

— Le module de fermeture de boucle, qui cherche à détecter lorsque la caméra observe un lieu déjà visité. DBoW2 [GT12a] est utilisé pour trouver une image clé candidate dans la trajectoire passée à partir de descripteurs globaux de type *Bag-of-Words* (BoW). Lorsqu'une image clé candidate est identifiée, un ajustement de faisceaux global de la carte est exécuté pour corriger les erreurs d'accumulation.

ORB-SLAM peut être considéré aujourd'hui comme le système de référence, grâce à sa très bonne précision, son coût de calcul relativement bas, son implémentation open-source et ses extensions à différents types de caméras, RGB-D dans ORB-SLAM2 [MT17], et à des capteurs inertiels, IMU dans ORB-SLAM3 [Cam+21].

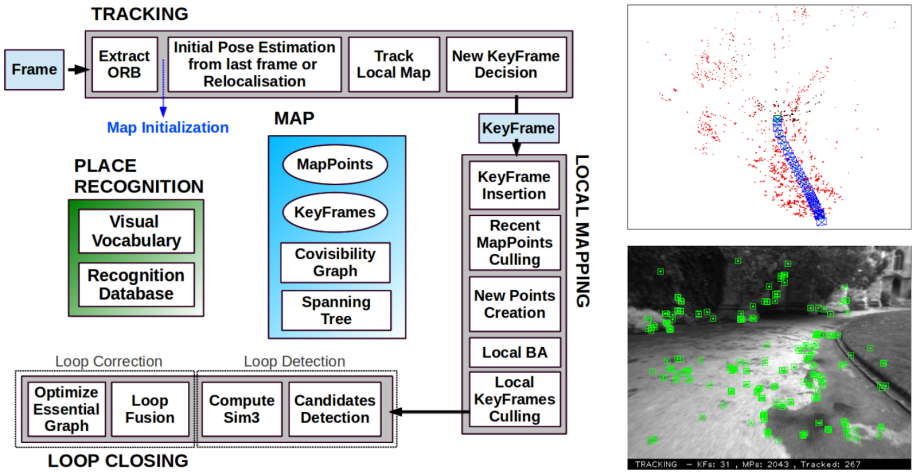


FIGURE 4.4 – Architecture du système ORB-SLAM [MMT15], à gauche, et illustration du suivi des points d'intérêt et de la carte reconstruite, à droite.

4.1.4 Méthodes directes

L'approche directe repose sur les intensités des pixels et estime une pose relative entre deux images en minimisant une erreur photométrique. Au contraire des méthodes présentées précédemment, qui se basent sur des points épars dans l'image, cette approche directe utilise tous les pixels. Elle élimine donc les étapes d'extraction et description des points d'intérêt et de leur association robuste qui sont coûteuses en temps de calcul. Un système de SLAM direct comporte généralement trois étapes :

1. Le suivi de la caméra, qui permet de calculer la pose relative de la caméra courante par rapport à la dernière image clé. Cette estimation se fait par l'alignement des pixels de l'image courante (I_k) avec ceux de l'image clé précédente (I_{k-t}), sous la forme d'une minimisation d'une erreur photométrique, comme illustrée dans la figure 4.5 et définie par

$$\mathcal{E}_{photo}(\mathbf{T}_{k,k-t}) = \sum_i (I_{k-1}(\mathbf{u}_i) - I_k(\omega(\mathbf{u}_i, D_{k-1}(\mathbf{u}_i), \mathbf{T}_{k,k-1})))^2, \tag{4.2}$$

où $\mathbf{T}_{k,k-1}$ est la pose relative recherchée et ω correspond au déplacement du pixel $\mathbf{u}_i$, de profondeur $D_{k-1}(\mathbf{u}_i)$ dans l'image $k-1$ par la pose. La pose de la dernière image traitée est utilisée comme valeur initiale. Cette étape repose sur l'hypothèse de surfaces lambertiennes, c'est-à-dire que l'intensité en niveaux de gris d'un point 3D dans différentes images est constante, mais qui n'est pas toujours le cas en pratique.

2. L'estimation de la profondeur qui utilise la pose relative estimée pour raffiner la carte de profondeur de la dernière image clé. Si le déplacement de la caméra est trop important, une nouvelle image clé est créée et sa carte de profondeur est initialisée en projetant les points des images clés proches, déjà existantes.
3. L'optimisation globale de la carte qui raffine les poses de toutes les images clés et permet de corriger les erreurs de dérive.

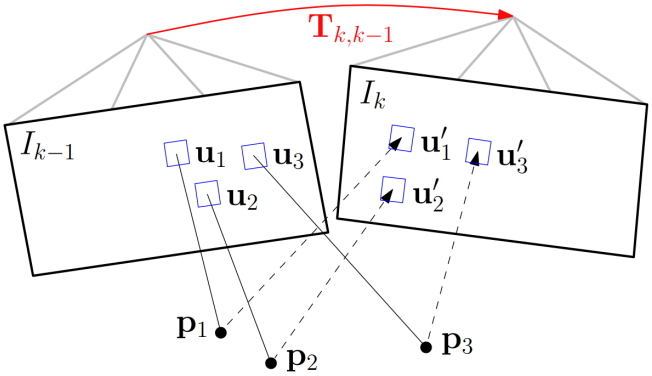


FIGURE 4.5 – Estimation de la pose relative des images I_k et I_{k-1} par alignement des pixels, en minimisant leur erreur photométrique [FPS14].

Elle permet également une cartographie bien plus dense de la scène. Cependant, en comparaison de l'erreur de reprojection de points, l'erreur photométrique est hautement non linéaire et non convexe. Son optimisation requiert donc une bonne initialisation de la caméra et de la carte. Des approches multi-échelles, avec des pyramides d'images, sont souvent utilisées.

DTAM [NLD11] a été un des premiers systèmes monoculaires, temps réel, capable de créer un modèle de surface 3D dense et de l'utiliser pour le suivi de la caméra. Ses performances en temps réel nécessitent toutefois l'utilisation d'une carte graphique. Ces méthodes sont en effet plus coûteuses en temps de calcul que les méthodes par points d'intérêt. Afin de réduire les ressources de calcul nécessaires, des évolutions semi-denses ont été développées [CC15 ; ESC14]. Par exemple, DPPTAM [CC15] utilise des superpixels pour réduire le coût de calcul de la cartographie. On retrouve également LSD-SLAM [ESC14], un autre système de référence de SLAM direct, qui adopte une approche semi-dense en se limitant aux pixels les plus pertinents dans les images, c'est-à-dire ceux avec de forts gradients. Son fonctionnement est illustré dans la figure 4.6. Il permet de reconstruire, en temps réel, des cartes à grande échelle. Les auteurs ont démontré qu'il était suffisant d'initialiser ce système par des valeurs aléatoires avec une grande variance. Cette méthode se distingue également par les transformations de type similitude estimées entre les images, ce qui permet de corriger les accumulations d'erreur d'échelle.

Semi-Direct Monocular Visual Odometry (SVO) [FPS14] est une méthode hybride qui combine les approches directe et indirecte. Elle extrait des points clés, de type FAST, dans les images, utilise l'approche directe pour estimer la pose relative de la caméra et exécute un ajustement de faisceaux pour optimiser la trajectoire de la caméra et la structure 3D en minimisant une erreur de reprojection géométrique.

Direct Sparse Odometry (DSO) [EKC18] est une autre méthode hybride. Elle est capable de calculer des poses précises de la caméra dans des situations où les détecteurs de points sont peu performants, améliorant ainsi la robustesse des méthodes indirectes dans des zones faiblement texturées ou pour des images floues. Elle introduit un ajustement de faisceaux photométrique

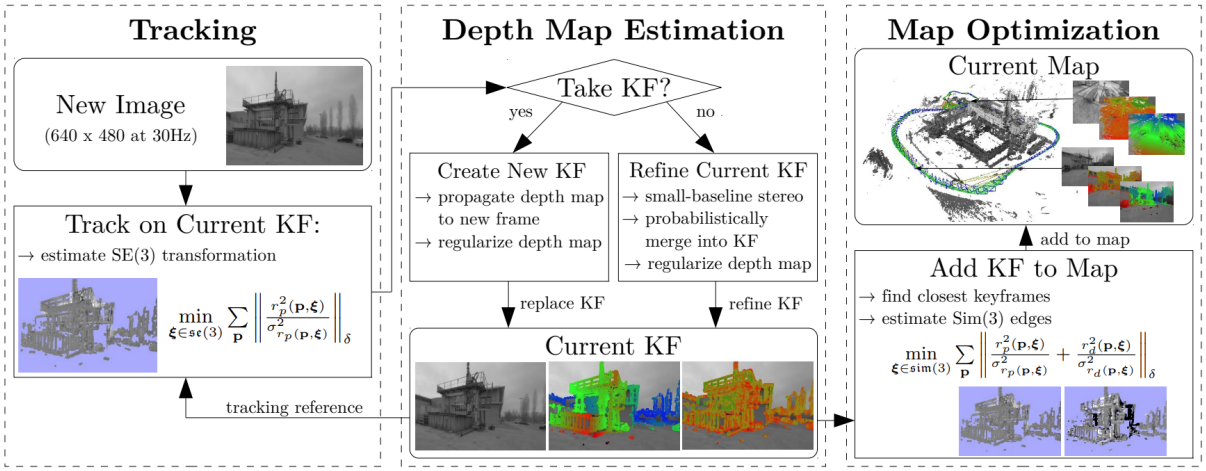


FIGURE 4.6 – Aperçu global du système de SLAM direct LSD-SLAM [ESC14].

local qui optimise simultanément une fenêtre glissante d'images clés récentes et la profondeur inverse des points. Cependant, ces deux méthodes, SVO et DSO, traitent uniquement de la partie d'odométrie visuelle d'un SLAM et ne comportent pas, par exemple, de fermeture de boucle ou d'optimisation globale de la carte.

4.1.5 Fermeture de boucle et relocalisation

La fermeture de boucle et la relocalisation sont deux techniques importantes dans un système de SLAM robuste. La première consiste à détecter lorsque la caméra retourne dans une zone déjà explorée et est généralement suivie d'une optimisation globale des poses successives de la caméra pour corriger l'accumulation d'erreurs d'estimation. Cette détection requiert une très bonne robustesse car une fausse détection de fermeture de boucle peut fortement dégrader le résultat du SLAM. De son côté, la relocalisation consiste à calculer la position et l'orientation de la caméra dans la carte reconstruite, à partir de seulement l'image courante. Cette procédure de relocalisation est primordiale dans un SLAM et est notamment utilisée lorsque le système est perdu. Cela peut être causé par une occultation temporaire de la caméra ou par des mouvements brusques, comme illustré sur la figure 4.7. La relocalisation permet également de réinitialiser un système à partir d'une carte pré-construite de l'environnement de travail, en calculant un alignement initial. Cela est notamment utile dans le contexte de cartographie continue, quand on cherche à étendre ou améliorer une carte déjà existante. De même, dans des applications de réalité augmentée, il est nécessaire de pouvoir se localiser par rapport à une carte existante et augmentée d'un environnement.

Les premières approches de reprise de SLAM travaillaient à un niveau local, en essayant de combler de courtes défaillances du suivi de la caméra. Pupilli *et al.* ont notamment proposé d'utiliser des hypothèses multiples avec un filtre particulaire [CPMC06]. Dans le contexte de suivi d'un modèle de scène connu, Reitmayr *et al.* ont proposé un module de relocalisation basé sur la mise en correspondance de points d'intérêt de l'image courante avec des images précédentes. Ces approches sont cependant locales et ne traitent que des pertes brèves du suivi de la caméra.

Williams *et al.* ont proposé une approche de relocalisation plus globale qui utilise la carte reconstruite de la scène plutôt que seulement les images clés précédentes [WKR07]. Ils utilisent un classifieur de patchs d'image pour établir des correspondances entre les points d'intérêt détectés dans l'image et les points de la carte, puis utilisent P3P combiné avec l'algorithme RANSAC

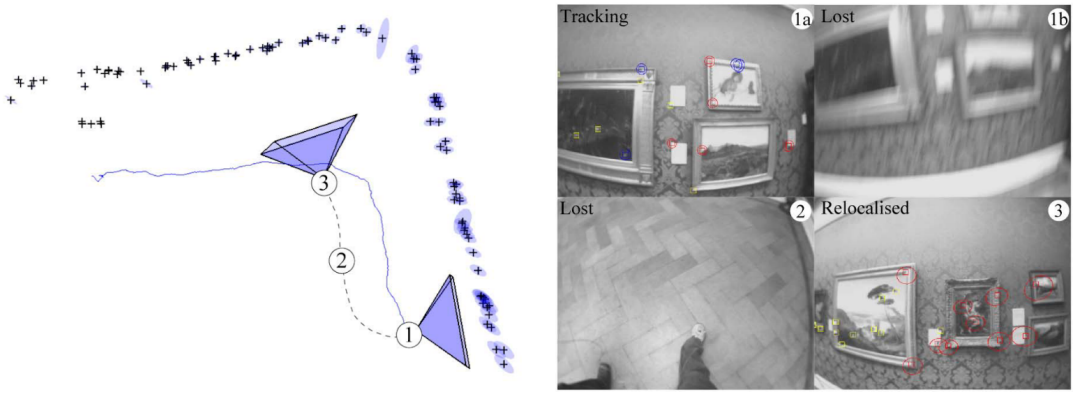


FIGURE 4.7 – Illustration d'une reprise de SLAM [WKR07]. 1a) Fonctionnement normal du système avec le suivi de la caméra et la construction de la carte. 1b et 2) Perte du suivi et déplacement de l'utilisateur. 3) Le système se relocalise par rapport à la carte créée.

pour calculer la pose de la caméra. Cette méthode a notamment été utilisée dans PTAM [KM07]. Les auteurs ont ensuite étendu leur approche à la détection de fermeture de boucle [WKR11]. Eade *et al.* ont proposé un système unifié de fermeture de boucle et relocalisation [ED08]. Ils adoptent une approche basée sur l'apparence et construisent un dictionnaire de mots visuels (*Bag-of-Words*) à partir de descripteurs SIFT. Ce modèle permet de détecter un lieu déjà visité ou de trouver une image clé candidate de relocalisation. Un modèle d'apparence local est ensuite utilisé pour mettre en correspondance les points clés de cette image avec des points 3D de la carte et calculer la pose. Cummins *et al.* ont proposé FAB-MAP [CN08 ; CN10], un modèle probabiliste basé sur l'apparence des images permettant de traiter les problèmes de fermeture de boucle et de relocalisation dans un SLAM. Il repose sur un dictionnaire de mots visuels calculés sur des descripteurs SIFT [Low04] ou SURF [BTG06]. Les auteurs montrent que ce système obtient de bonnes performances pour détecter des fermetures de boucle à grande échelle, dans des environnements extérieurs complexes. Cette approche a été utilisée dans CD-SLAM [PRB11], qui lui permet de détecter des lieux déjà visités sur de longues périodes de temps et dans des environnements dynamiques. Dans la version améliorée de PTAM [CC15], Klein *et al.* effectuent la relocalisation en comparant l'image courante avec les images clés précédentes par une somme de différences quadratiques entre des images lissées et de résolution réduite. Cette technique ne fonctionne cependant que pour des points de vue très similaires. Strasdat *et al.* [SDMK11] distinguent deux types de fermeture de boucle, certaines locales pouvant être détectées géométriquement à partir de la structure observée de la scène et d'autres, à plus grande échelle, qui nécessitent une approche basée sur l'apparence. Pour celles-ci, ils utilisent un dictionnaire de mots visuels calculés par des descripteurs SURF [BTG06].

ORB-SLAM [MMT15] repose également sur une approche basée sur l'apparence pour la recherche d'images clés candidates lors d'une fermeture de boucle ou d'une relocalisation. La recherche d'images similaires utilise DBoW2 [GT12a] combiné à des descripteurs ORB. Ce système permet de créer un dictionnaire de mots visuels, sous forme d'arbre hiérarchique, optimisé pour des descripteurs binaires et permettant une recherche très rapide. Cela permet d'établir une liste d'images clés candidates. Lors d'une relocalisation, un appariement de points d'intérêt est réalisé entre ces images et l'image courante et des hypothèses de poses sont calculées avec un algorithme PnP. Lorsqu'une pose avec suffisamment d'associations 2D-3D valides est trouvée, différentes stratégies d'optimisation et de recherche guidée de correspondances supplémentaires de points sont exécutées.

Mur-Artal *et al.* ont montré que l'utilisation de DBoW2 obtient de meilleurs résultats que FAB-MAP [MT14]. L'utilisation de descripteurs binaires, plutôt que SIFT ou SURF dans FAB-MAP, permet également de créer un dictionnaire plus compact et offre un gain de vitesse conséquent. Le module de relocalisation d'ORB-SLAM, basé sur DBoW2, est considéré aujourd'hui comme le système de référence, obtenant la meilleure précision.

Les méthodes de SLAM direct reposent généralement sur les mêmes méthodes que les approches indirectes pour la fermeture de boucle ou la relocalisation. LSD-SLAM [ESC14] utilise, par exemple, le système FAB-MAP pour la recherche de fermeture de boucle. Un module de fermeture de boucle, basé sur des mots visuels a également été intégré à DSO dans LDSO [GWDC18]. Enfin, Gladkova *et al.* ont ajouté une capacité de relocalisation à DSO, au travers d'un couplage étroit avec les méthodes utilisées dans les systèmes de SLAM indirect [GWZC21].

Le tableau 4.1 regroupe les principaux systèmes de SLAM classiques, considérés comme des références et représentatifs des approches par points d'intérêt, directes ou par filtrage.

	SLAM ou VO	**Détecteur**	**Association de données**	**Estimation**	**Relocalisation**	**Fermeture de boucle**	**Monoculaire**	**Stéréo**	**Fisheye**
Mono-SLAM [DRMS07]	SLAM	Shi Tomasi	Corrélation	EKF	-	-	√	-	-
PTAM [KM07]	SLAM	Fast	Pyramide SSD	BA	Patchs d'image	-	√	-	-
LSD-SLAM [ESC14]	SLAM	Forts gradients	Direct	PG	-	FAB-MAP PG	√	√	√
SVO [FPS14]	VO	FAST + gradients	Direct	Local BA	-	-	√	√	√
DSO [EKC18]	VO	Forts gradients	Direct	Local BA	-	-	√	√	√
ORB-SLAM2 [MT17]	SLAM	ORB	Detector	Local BA	DBoW2	DBoW2 PG+BA	√	√	-
ORB-SLAM3 [Cam+21]	SLAM	ORB	Detector	Local BA	DBoW2	DBoW2 PG+BA	√	√	√

TABLE 4.1 – Résumé des principaux systèmes de SLAM ou d'odométrie visuelle (VO), considérés comme des références et représentatifs des approches par points d'intérêt, directes ou par filtrage. (BA = Ajustement de faisceaux et PG = graphe de poses).

4.1.6 SLAM et apprentissage profond

L'apprentissage profond a également été appliqué au problème de SLAM. Certains travaux se sont attachés à améliorer des sous-parties des approches classiques, telles que l'extraction et la description de points d'intérêt [DMR18 ; Luo+18 ; MMRM17 ; OTFY18], leur association [SDMR20] ou encore la localisation [Sar+21 ; SWYC20]. D'autres travaux, ont proposé des systèmes entraînables de bout en bout et capables de prédire les poses relatives de la caméra. Zhou *et al.* ont proposé DeepTAM [ZUB18], un système de suivi de caméra dense et d'estimation de cartes de profondeur entièrement appris. DeepVO [WCWT17], une méthode d'odométrie visuelle, entraînée de bout-en-bout et basée sur des réseaux de neurones convolutifs récurrents a également été développée. Ce système se limite à l'estimation de la trajectoire de la caméra,

sans module de cartographie, et ne constitue pas un SLAM complet. Des méthodes utilisant de l'apprentissage non supervisé ont également été proposées [ZBSL17 ; Zha+18 ; LWLG18].

CNN-SLAM [TTLN17] intègre un réseau de neurones de prédiction de profondeur monoculaire dans LSD-SLAM [ESC14]. Ils proposent une méthode de fusion permettant d'utiliser la profondeur prédite par le réseau lorsque l'approche directe du SLAM échoue. DVSO [YWSC18] propose d'halluciner des images stéréo virtuelles pour entraîner un réseau de prédiction de profondeur semi-supervisé. Cela lui permet d'atteindre des performances comparables avec celles des meilleurs systèmes classiques d'odométrie visuelle stéréo. D3VO [YSWC20] étend cette approche et exploite des réseaux de neurones pour prédire la profondeur, les poses et les incertitudes associées. Murthy *et al.* ont développé ∇SLAM [JIP19], qui fournit des versions différentiables des composants d'un système de SLAM dense classique. DeepFactors [CLCD20], basé sur CodeSLAM [Blo+18], est un système de SLAM capable de fournir une reconstruction dense à partir d'une optimisation multi-vue et de connaissances a priori apprises. Récemment, DROID-SLAM [TD21] a été proposé et repose sur une architecture de réseau de neurones avec une couche d'ajustement de faisceaux dense qui permet des mises à jours itératives des poses de la caméra et des profondeurs prédites.

4.1.7 SLAM sémantique

Alors que les approches classiques utilisent principalement des points d'intérêt, pour la localisation et la cartographie, des systèmes utilisant d'autres primitives géométriques, telles que des lignes [Gom+19] ou des plans [HLLR19 ; Lia+22 ; Hos+18] ainsi que des informations sémantiques [MHDL17 ; Sün+17 ; Sal+13 ; HXHK19 ; NMS19 ; Hos+18 ; Ok+19 ; FPM18 ; YS19 ; Wu+20 ; Lia+22 ; WRA21] sont apparus.

Au niveau de la cartographie des objets, Crocco *et al.* ont développé une solution analytique au problème de reconstruction d'objet sous la forme d'un ellipsoïde à partir d'observations dans plusieurs vues [CRB16], en utilisant un modèle de caméra affine. Rubino *et al.* l'ont ensuite étendu au modèle sténopé [RCB18]. Chen *et al.* ont adressé le problème d'une estimation 3D initiale d'un objet, dans le contexte spécifique d'un mouvement de translation avant de la caméra, difficile car seulement certains côtés des objets sont visibles, mais courant dans des applications de navigation autonome de véhicules [Che+21]. Plus récemment, Li *et al.* ont développé FroDo [Rün+20] et obtiennent des reconstructions 3D précises d'instances d'objets à partir d'une séquence vidéo. Leur méthode repose sur l'apprentissage d'un espace latent des formes d'objets pouvant générer à la fois une représentation éparse ou dense. Les objets sont détectés dans les images en 2D, des boîtes englobantes 3D sont instantiées et un encodage de la forme de l'objet est prédit puis optimisé par des contraintes multi-vues géométriques, photométriques et de silhouettes. Les associations de données nécessaires pour distinguer les instances d'objets sont obtenues par *clustering* de segments 3D issus des rayons passant par le centre de la caméra et le centre des détections des objets. Ces travaux se focalisent sur la partie cartographie d'un SLAM et permettent d'enrichir la carte reconstruite avec des informations sur les objets présents. Ils ne considérèrent pas le problème de localisation et supposent que les poses de la caméra sont connues.

Un travail pionnier dans l'intégration d'information sémantique d'objets dans un système de localisation et cartographie a été proposé par Bao *et al.* [BBCS12]. Ils utilisent des méthodes de détection d'objet et de segmentation d'image pour reconnaître des éléments sémantiques et les localiser en 3D. Ils modélisent notamment les interactions entre les points 3D, les objets et les régions. Ils montrent que l'incorporation de données sémantiques fournit à leur système une meilleure robustesse dans l'estimation des poses de caméra. Ce travail concerne cependant un système de Structure-from-Motion (SfM), qui ne dispose pas des mêmes contraintes de rapidité et

de traitement séquentiel qu'un SLAM. McCormac *et al.* [MHDL17] et Sünderhauf *et al.* [Sün+17] ont proposé de combiner un système de SLAM RGB-D avec de la segmentation sémantique et des détection d'objets pour obtenir une carte annotée sémantiquement. Dans ces travaux, l'information sémantique est ajoutée à la carte après sa création mais n'est pas utilisée pour améliorer la localisation. De plus, les cartes construites ne distinguent pas des instances d'objets, mais prennent la forme de nuages de points denses, dans lesquels chaque point porte une étiquette sémantique.

SLAM++ [Sal+13] est une méthode de SLAM RGB-D permettant de cartographier les instances d'objets d'une scène. Ils combinent des méthodes de reconnaissance et d'estimation de pose 6D d'objets. Ce système nécessite cependant d'avoir accès à des modèles 3D précis des objets de la scène, ce qui rend son déploiement complexe et limite ses applications. Deep-SLAM++ [HXHK19] étend ce travail à des environnements inconnus, avec des objets non modélisés, en essayant d'apprendre des connaissances a prioi de la forme des objets pour chaque catégorie.

Des modèles plus légers, et surtout génériques, des objets ont également été utilisés et sont davantage compatibles avec un déploiement dans des environnements inconnus et non modélisés.

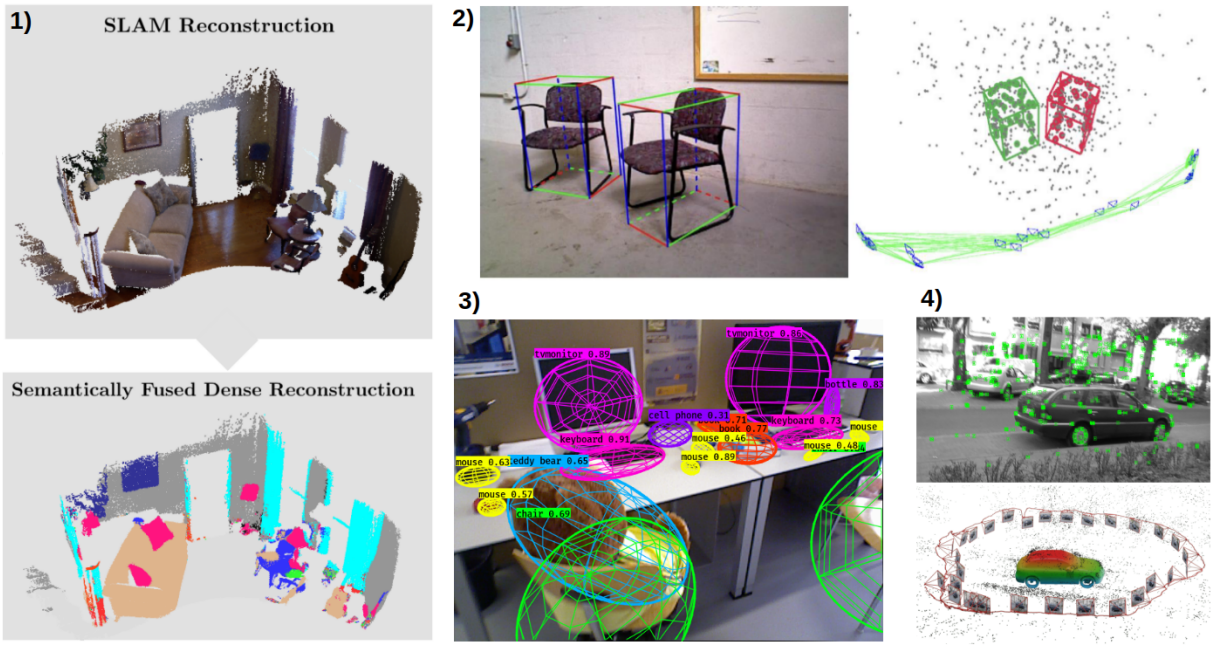


FIGURE 4.8 – 1) SemanticFusion [MHDL17] : ajout d'information sémantique dans le nuage de points reconstruit. 2) CubeSLAM [YS19] : modélisation des objets par des cuboïdes. 3) QuadricSLAM [NMS19] : modélisation des objets par des ellipsoïdes. 4) DSP-SLAM [WRA21] : modélisation précise des objets par un encodage appris.

Nicholson *et al.* ont développé QuadricSLAM [NMS19], un système de SLAM basé sur des objets représentés par des ellipsoïdes. Ils estiment conjointement la pose de la caméra et les paramètres des ellipsoïdes en combinant les objets avec de l'odométrie visuelle dans une formulation de SLAM sous forme de graphe. Les données d'odométrie sont obtenues par un système classique basé sur des points d'intérêt. Par exemple, ORB-SLAM2 peut être utilisé. Les auteurs supposent, cependant, une mise en correspondance parfaitement connue des objets de la carte avec leurs détections dans les images. Dans leurs expériences, ces associations sont établies manuellement, ce qui limite considérablement les applications du système. EAO-SLAM [Wu+20] est un système

de SLAM semi-dense intégrant des objets. Les auteurs exploitent différentes statistiques pour améliorer la robustesse de l'association de données automatique. Les objets sont représentés sous forme de cuboïdes ou d'ellipsoïdes, en fonction de leur nature régulière ou non. Ils sont cependant supposés être placés parallèlement au sol, ce qui nécessite de connaître la direction verticale de la scène. Hosseinzadeh *et al.* ont combiné des points, des plans et des ellipsoïdes dans un système de SLAM basé sur un graphe [Hos+18 ; HLLR19]. Ils couplent un détecteur d'objets avec un réseau de neurones convolutif pour prédire la profondeur, les normales des surfaces et une segmentation sémantique des images. Cela leur permet de capturer la structure de la scène et de définir des contraintes point-plan, plan-plan et objet-plan. Leurs expériences ont montré que l'incorporation des objets et des plans dans un SLAM permet de produire des cartes plus informatives de l'environnement et d'estimer des trajectoires plus précises de la caméra. Dans SO-SLAM [Lia+22], Liao *et al.* ont défini des échelles a priori des objets en fonction de leur sémantique, ainsi que des contraintes de symétrie. Ils ont également utilisé des plans extraits manuellement pour ajouter des contraintes de support aux objets. Cependant, comme pour QuadricSLAM, la mise en correspondance des objets n'est pas traitée et repose sur des annotations manuelles.

Un système de SLAM basé objet et spécialisé pour la navigation autonome a été proposé par Ok *et al.* [Ok+19]. Ils exploitent les détections d'objets par boîtes englobantes, la texture de l'image et des connaissances sémantiques a priori sur la forme des objets pour inférer leurs modèles ellipsoïdaux et surmonter les difficultés d'observabilité liées au mouvement de translation avant des véhicules. Cependant, l'influence des objets sur l'estimation de la trajectoire de la caméra n'est pas discutée et seule la partie cartographie du système est évaluée.

Yang *et al.* ont proposé CubeSLAM [YS19], un système de SLAM qui représente les objets par des cuboïdes. Leur méthode permet de générer des propositions d'objets à partir d'une seule image en utilisant des boîtes de détection 2D et des points de fuite. Les cuboïdes sont ensuite optimisés conjointement avec les poses de la caméra et les points 3D de la carte.

Forst *et al.* ont développé un système qui modélise les objets par des sphères et les utilise pour résoudre l'ambiguïté d'échelle et réduire l'erreur de dérive du SLAM [FPM18]. Récemment, des informations a priori sur la forme des objets (DeepSDF [Par+19]), obtenues par apprentissage profond, pour chaque catégorie d'objet, ont été exploitées pour obtenir des reconstructions 3D précises des objets sous la forme de maillages et intégrées dans un système de SLAM visuel [WRA21]. La figure 4.8 résume les principales modélisations choisies pour intégrer les objets dans un SLAM.

Relocalisation basée objet

Les travaux mentionnés ci-dessus intègrent de l'information sémantique dans des systèmes de SLAM, notamment au travers d'objets, mais n'en tirent pas vraiment avantage dans l'étape de relocalisation. La plupart de ces systèmes sont basés sur ORB-SLAM2 et reposent sur sa méthode de relocalisation classique par points. Seul, Dudek *et al.* ont proposé d'utiliser l'information sémantique de la carte pour améliorer la relocalisation [LMD19]. Cependant, leur méthode consiste plutôt en une étape de post-traitement permettant d'estimer une transformation entre deux cartes construites d'une scène. Très récemment, Mahattansin *et al.* ont utilisé le nombre et les catégories des objets détectés dans les images pour filtrer les images clés candidates à la relocalisation [MSBI22]. Cependant, la pose est toujours calculée en recherchant des correspondances de points dans l'image clé la plus similaire.

4.2 Motivations pour un système de SLAM visuel combinant points et objets

L'ajout d'information sémantique dans un système de SLAM, au travers des objets, a un double intérêt : améliorer la robustesse de sa relocalisation et permettre une meilleure compréhension de la scène. Cependant, les méthodes existantes de SLAM basé objet ont surtout tenté d'intégrer les objets dans une optimisation jointe caméra-point-objet mais ne s'intéressent pas vraiment au problème de relocalisation. De plus, la majorité de ces systèmes nécessitent une intervention manuelle pour l'association de données [NMS19 ; Lia+22 ; Ok+19] ou requièrent des informations sur la scène, par exemple la direction de la verticale dans EAO-SLAM [Wu+20]. Nous visons donc à développer un système de SLAM complet, combinant points et objets, qui permet la reconstruction d'une carte d'objets, à la volée, de manière automatique et qui en tire avantage pour se relocaliser.

4.3 Système de SLAM point-objets

4.3.1 Aperçu global

Notre système OA-SLAM, pour *Object-Aided SLAM*, est détaillé dans la figure 4.9. Il se base sur ORB-SLAM2 [MT17] pour le suivi de la caméra (*tracking*), la cartographie locale des points (*local mapping*) et la fermeture de boucle (*loop closure*). Des modules additionnels ont été ajoutés pour la gestion des objets. Nous utilisons la même représentation que celle des chapitres précédents, c'est-à-dire sous la forme d'une ellipse dans l'image et d'un ellipsoïde dans la carte en 3D, et qui a l'avantage d'être particulièrement légère. Notre système traite les objets de manière similaire aux points. Ils sont suivis au fil des images, estimés en 3D, insérés dans la carte, puis continuellement raffinés. Le suivi des objets et leur reconstruction initiale ont été ajoutés au thread principal du SLAM, tandis que leur raffinement (*Local Object Mapping*) s'exécute dans un thread séparé. Ils sont décrits dans la section 4.3.2. Finalement, le module de relocalisation est amélioré par l'utilisation des objets, ce qui accroît nettement sa robustesse, et est décrit dans la section 4.3.3,

4.3.2 Cartographie des objets

Dans cette section, nous présentons le module de cartographie des objets qui permet à notre système de reconstruire les modèles 3D des objets (ellipsoïdes) et de les intégrer à la carte. L'idée générale consiste à suivre les objets dans les images, les reconstruire en 3D sous forme d'ellipsoïdes et à les intégrer dans la carte.

Détection des objets

Nous avons utilisé le détecteur d'objets YOLO [BWL20] pour obtenir les détections d'objets dans les images. Chaque détection inclut une boîte alignée aux axes de l'image, une catégorie d'objet et un score de détection. Nous avons appliqué un seuil de 0.5 sur ce score afin d'éliminer les détections faussement positives. Nous avons choisi d'utiliser le détecteur d'objets YOLO plutôt que Faster R-CNN [RHGS17], qui a été utilisé dans les chapitres 2 et 3, pour des raisons de vitesse. En effet, son architecture à une étape lui permet d'être plus rapide que des méthodes à deux étapes telles que Faster R-CNN, qui nécessite d'abord une proposition de régions puis une classification. Dans le cas d'un système de SLAM basé objet, la vitesse du détecteur est

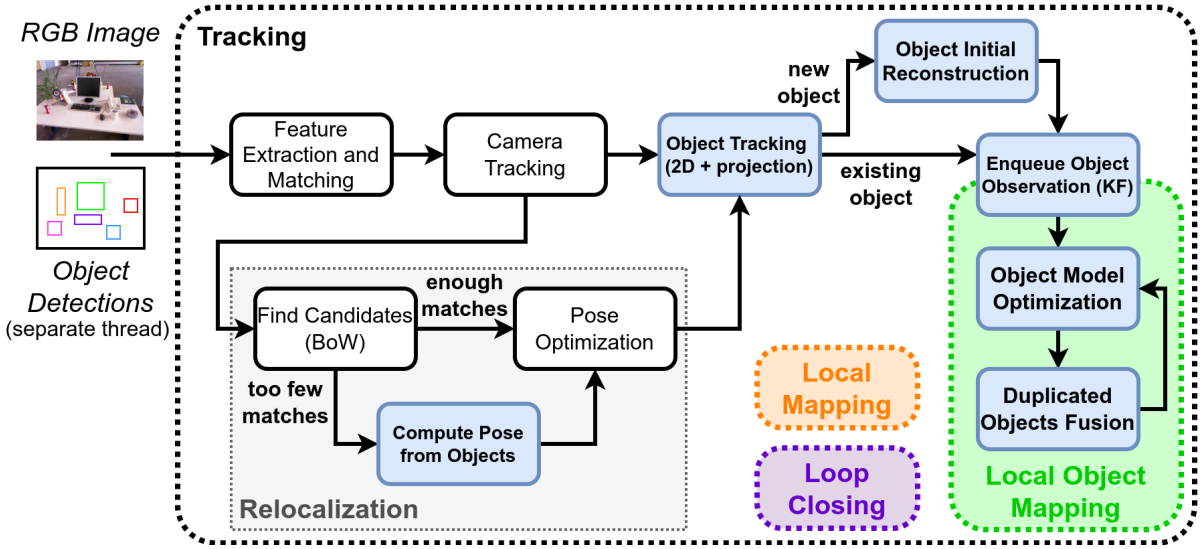


FIGURE 4.9 – Architecture de OA-SLAM. Les éléments bleus sont nouveaux et ont été ajoutés à l'architecture basée sur ORB-SLAM2. À noter que les différents modules (*Tracking*, *Local Mapping*, *Loop Closing* and *Local Object Mapping*) sont exécutés dans des threads séparés.

importante car celui-ci est exécuté sur chaque image. Par ailleurs, la modularité de notre méthode lui permet également de s'adapter à d'autres détecteurs, par exemple avec des architectures plus légères, tels que MobileNet-SSD v2 [Chi+20], YOLO-LITE [HPC18] ou CSL-YOLO [ZLHF21], pour des cas d'utilisation avec des ressources de calcul limitées.

En fonction de l'environnement dans lequel notre système est utilisé, le détecteur d'objets peut être pré-entraîné sur la base de données COCO [Lin+14a], pour des objets assez classiques (livre, chaise, tasse, ...) ou ajusté pour des objets plus spécifiques, par exemple des objets de musées comme illustrés dans la figure 4.23. Cet ajustement du détecteur nécessite simplement des annotations manuelles des objets à détecter sous la forme de boîtes dans quelques images.

Dans ce chapitre, nous revenons à un détecteur d'objets standard qui prédit des boîtes alignées aux axes de l'image, contrairement au chapitre 2 dans lequel un réseau de prédiction d'ellipses a été proposé pour améliorer la précision du calcul de pose. La raison principale est que nous visons ici un cas d'application très général dans des environnements inconnus pour lesquels des annotations de pose d'entraînement ne sont pas toujours disponibles. De plus, nous montrons dans la section 4.4.4 que la précision de la pose estimée dans l'étape de relocalisation est principalement apportée par l'utilisation des points, les objets apportant plutôt de la robustesse. Dans des cas particuliers, où il serait tout de même possible d'entraîner le réseau de prédiction d'ellipse proposé dans le chapitre 2, les détections sous forme d'ellipses peuvent également être utilisées.

Suivi des objets par boîtes de détection

Le suivi d'objet au cours du temps est un élément crucial de notre système. Étant donné un ensemble de détections d'objets dans l'image courante, l'objectif est de les mettre en correspondance avec des objets 3D existants ou d'en créer de nouveaux. Les associations entre les détections sont contraintes par les catégories des objets. Le suivi d'objet est réalisé à deux niveaux, un premier à court terme dans des images temporellement proches et un second à plus long terme qui permet, par exemple, le suivi d'un objet après que celui-ci soit sorti du champ de vision de la caméra :

1. Avant d'être reconstruits, les objets sont suivis en 2D, au fil des images, en se basant sur le chevauchement de leurs boîtes englobantes et la cohérence de leurs catégories. Ce suivi est possible à court terme, lorsque le mouvement entre deux images est relativement faible. Cependant, il est sujet à des erreurs lorsque des détections sont manquantes dans certaines images, lorsqu'un objet sort du champ de vision de la caméra ou en cas de mouvement brusque de la caméra.
2. Une fois que le suivi 2D d'un objet a été réalisé sur une séquence d'images avec un changement de point de vue suffisant, celui-ci peut être estimé en 3D et un suivi à plus long terme est possible. Pour cela, son modèle ellipsoïdal est projeté dans l'image actuelle et le recouvrement avec les détections d'objets dans cette image est utilisé pour établir les associations.

Dans les deux cas, les associations optimales sont établies en utilisant l'algorithme Hongrois [Kuh10], également appelé algorithme de Kuhn-Munkres, une méthode bien connue pour résoudre le problème d'affectation. Cet algorithme maximise un score total de correspondance afin de trouver les meilleures affectations possibles entre N détections et M objets. Nous définissons sa matrice de score, à l'image t, par

$$\mathbf{S}^t = [s_{ij}^t]_{N\times M}$$
$$\text{avec } s_{ij}^t = \max(\text{IoU}(\mathtt{D}_i^t, \mathtt{B}_j^{t-k}), \underbrace{\text{IoU}(\mathtt{D}_i^t, \text{box}(\text{proj}(\mathtt{O}_j^t)))}_{\text{ou 0 si reconstruction non disponible}}), \tag{4.3}$$

où $\mathtt{D}_i^t$ est la i$^{\text{ème}}$ détection dans l'image courante, $\mathtt{B}_j^{t-k}$ correspond à la dernière boîte englobante associée au j$^{\text{ème}}$ objet et $\mathtt{O}_j^t$ est l'ellipsoïde reconstruit de cet objet. Les opérations *IoU*, *box* et *proj* représentent respectivement le calcul de l'Intersection-sur-Union, le calcul de la boîte englobante d'une ellipse et la projection dans l'image courante. Le terme d'IoU de gauche fait référence au suivi à court terme des boîtes englobantes réalisé en 2D, alors que le terme de droite exprime le suivi à long terme par projection, possible uniquement lorsqu'une reconstruction initiale de l'objet est disponible.

Suivi des objets par points

La méthode de suivi d'objet à long terme présentée précédemment repose uniquement sur la géométrie de l'ellipsoïde reconstruit, ce qui est très intéressant pour les objets de petite taille ou sans texture mais qui peut échouer en cas de détections peu précises ou partielles. En effet, le suivi est basé sur le modèle global d'un objet (ellipsoïde), dont la détection dans l'image peut présenter une certaine variance. Par conséquent, nous avons choisi de mettre également à contribution des points d'intérêt qui fournissent une localisation 2D plus précise. Lors de l'estimation de la pose de la caméra, les points d'intérêt de l'image sont mis en correspondance de manière robuste avec les points 3D de la carte. Ces correspondances sont utilisées pour établir des liens entre les boîtes de détection 2D et les ellipsoïdes des objets dans la carte, en utilisant les conditions suivantes :

— Dans l'image, un point d'intérêt est associé à une boîte de détection s'il est situé à l'intérieur de celle-ci.

— Dans la carte, un point 3D est associé à un objet s'il est situé à l'intérieur de son ellipsoïde.

Si au moins τ correspondances de points sont établies entre une boîte de détection et l'ellipsoïde d'un objet, l'association est prise en compte. Dans nos expériences, nous avons utilisé $\tau = 10$. Cette approche de suivi n'est réalisée qu'une fois que l'objet a été estimé en 3D et se montre particulièrement utile pour des objets hautement texturés et dans les cas où le suivi basé sur la géométrie échoue.

Reconstruction initiale

L'idée principale est de générer des reconstructions initiales des objets rapidement après leur apparition dans les images, afin que le suivi basé sur la projection des ellipsoïdes puisse être effectué, en plus du suivi à court terme par boîtes 2D. La validation et l'intégration d'un objet dans la carte se fait plus tard, si le suivi de l'objet dans les images successives est cohérent avec l'estimation de son modèle 3D. Sinon, la reconstruction de l'objet est rejetée.

Avant d'estimer l'ellipsoïde d'un objet, nous nous assurons que l'angle de vue, calculé entre les rayons passants par le centre de la caméra et les centres des boîtes de détection, varie suffisamment. Nous avons utilisé un seuil de 10°. Rubino *et al.* [RCB18] ont proposé une méthode de reconstruction d'un ellipsoïde à partir de détections elliptiques dans les images, mais nous avons observé une instabilité numérique lorsque les angles de vue des images utilisées sont proches. Notre objectif étant de reconstruire une estimation initiale d'un objet à partir d'un petit nombre d'images, les points de vue utilisés sont généralement peu variés. Nous avons donc opté pour une approche différente, plus fiable, dans laquelle un objet est initialement reconstruit comme une sphère, puis raffiné sous la forme d'un ellipsoïde, au fur et à mesure de l'arrivée de nouvelles vues. La position de cette sphère est obtenue par triangulation à partir des centres des boîtes de détection de l'objet et son rayon est déterminé par la taille moyenne des boîtes, mise à l'échelle à la position 3D estimée, telle que

$$rayon = \frac{1}{2n} \sum_{i=1}^{n} \frac{z_i}{2} \left(\frac{w_i}{f_x} + \frac{h_i}{f_y} \right), \tag{4.4}$$

où z_i est la coordonnée sur l'axe z du centre de l'objet dans le système de coordonnées de la i$^{\text{ème}}$ caméra, c'est-à-dire sa profondeur, w_i et h_i sont la largeur et la hauteur de la boîte de détection dans la i$^{\text{ème}}$ image, f_x et f_y sont les paramètres intrinsèques de la caméra et n désigne le nombre d'images dans lesquelles l'objet a été détecté.

Cette sphère est ensuite raffinée sous la forme d'un ellipsoïde lorsque l'objet est détecté dans de nouvelles images. Seules la taille de ses axes ainsi que sa position sont mises à jour. Son orientation est, dans un premier temps, maintenue fixe, à l'identité. Ce premier raffinement est exprimé sous la forme d'une minimisation d'erreurs de reprojection, mesurées par la distance de Wasserstein entre les projections de l'ellipsoïde et les ellipses inscrites dans les boîtes de détection dans les images.

Il faut noter qu'à partir de ce moment, le suivi d'un objet peut exploiter sa première estimation 3D afin de rechercher des correspondances possibles par projection dans les images.

Intégration dans la carte

Une fois qu'un objet a été reconstruit et raffiné dans un nombre suffisant d'images (typiquement 40 images), il peut être intégré dans la carte. Nous vérifions tout de même que son modèle ellipsoïdal reconstruit est cohérent avec ses détections dans les images. Nous avons fixé un seuil d'IoU minimal à 0.3 entre ses projections et ses détections.

Notre méthode de reconstruction est robuste, à la fois, à de fausses détections d'objet et à de mauvaises associations. En effet, les détections faussement positives que YOLO peut fournir ne sont généralement pas stables au cours du temps et ne sont donc pas suivies. De plus, la vérification d'un recouvrement minimal entre la projection de l'ellipsoïde reconstruit et les détections dans les images permet de détecter les erreurs d'association éventuellement commises lors du suivi d'un objet.

Raffinement des objets

Une fois intégré dans la carte, le modèle d'un objet est régulièrement raffiné par rapport à ses observations dans les images clés. Chaque fois qu'une nouvelle image clé observe un objet présent dans la carte, celui-ci est mis à jour par la minimisation d'une erreur de reprojection. La distance de Wasserstein est à nouveau utilisée entre l'ellipse inscrite dans la boîte de détection et la projection de l'ellipsoïde estimé. Contrairement au problème de localisation d'une image, traité dans le chapitre 3, les sorties d'image partielles d'un objet ne sont pas problématiques ici. Les images où la détection est adjacente à un bord ne sont simplement pas prises en compte pour la reconstruction. La distance de *Wasserstein* a été décrite précédemment dans la section 3.2.4. Le modèle ellipsoïdal du i$^{\text{ème}}$ objet est donc raffiné en minimisant

$$\sum_{j=0}^{N} \sigma_j^{-1} \mathcal{W}_2^2(E_{ij}, P_j Q_i^* P_j^T), \tag{4.5}$$

où E_{ij} est l'ellipse inscrite dans sa j$^{\text{ème}}$ détection, Q_i^* est la matrice de forme duale de son modèle ellipsoïdal, σ_j est son score de détection, P_j est la matrice de projection de la j$^{\text{ème}}$ image clé et N est le nombre total d'observations de l'objet dans des images clés.

Fusion d'objets

Dans certains cas, un objet peut être dupliqué dans la carte. Cela peut se produire lorsque de nouvelles détections d'un objet ne parviennent pas être associées à sa reconstruction existante et créent une nouvelle estimation 3D. Pour éviter de tels cas, notre système vérifie régulièrement les objets dupliqués. Deux objets de la même catégorie sont considérés comme un objet unique dans trois cas :

1. Si leur recouvrement 3D excède un certain seuil η, fixé à 0.2 dans nos expériences.
2. Si le centre d'un ellipsoïde se trouve à l'intérieur de l'autre.
3. Si les deux ellipsoïdes partagent plus de $\tau = 10$ points 3D communs.

Nous calculons le recouvrement par l'Intersection-sur-Union de leurs boîtes englobantes alignées aux axes en 3D, car cela demande très peu de temps de calcul. Dans le cas d'une fusion, les boîtes de détection suivies, pour les deux objets dans les images clés, sont combinées et un nouvel ellipsoïde est initialisé en suivant la procédure décrite dans la section 4.3.2, mais uniquement sur les images clés. Nous nous assurons de la validité de la fusion de deux objets en imposant un seuil de recouvrement, fixé à 0.3, entre les projections du nouvel ellipsoïde reconstruit et toutes les observations des deux objets dans les images clés. Si ce seuil n'est pas atteint, la fusion est annulée et les deux objets sont conservés deux manière séparée dans la carte.

4.3.3 Relocalisation par objets et points

La méthode de relocalisation originale d'ORB-SLAM2 offre une bonne précision, mais échoue souvent lorsque l'image requête est éloignée de la trajectoire passée de la caméra. En effet, elle utilise des mots visuels comme descripteurs globaux pour trouver des images clés candidates et cherche ensuite à établir des correspondances de points entre les images à partir de descripteurs locaux. Cette mise en correspondance de points est complexe et particulièrement sensible à des changements de point de vue. Comme décrit dans la figure 4.9, nous améliorons cette méthode de relocalisation en utilisant les objets comme balises, plus robustes aux changements de point

de vue. Ce calcul de pose basé sur les objets est utilisé lorsque trop peu de correspondances entre points ont pu être établies. En effet, les objets appris sur une grande base de données ont l'avantage de pouvoir être détectés à partir d'une large variété de points de vue (de face, de côté ou même de derrière), ouvrant ainsi la voie à une relocalisation plus robuste.

Sachant que les poses de caméra calculées à partir de points, avec un algorithme PnP, sont généralement plus précises que celles obtenues à partir des objets, mais que l'association des points est un problème complexe (changement de point de vue et d'échelle ou motifs répétitifs), notre idée est d'utiliser les objets pour guider cette mise en correspondance. La précision de la pose obtenue à partir des objets est généralement suffisante pour que les projections des points 3D soit proches des points d'intérêt détectés dans l'image, permettant ainsi d'établir plus facilement des correspondances entre points et d'utiliser ensuite PnP pour calculer une pose plus précise de la caméra.

Notre approche de localisation basée sur les objets est une version modifiée de l'algorithme présenté dans la section 2.3.6 du chapitre 2, qui détermine conjointement les correspondances d'objets entre l'image et la carte et estime la pose de la caméra. Les paires ellipse-ellipsoïde sont établies en fonction de leurs catégories. À chaque itération, un ensemble minimal de trois paires est sélectionné et la pose de la caméra est calculée avec un algorithme Perspective-3-Point (P3P) sur les centres des ellipses et des ellipsoïdes. P3P fournit quatre solutions potentielles pour chaque combinaison de trois objets. Pour chacune de ces hypothèses de pose, les ellipsoïdes des objets sont projetés, associés aux détections dans l'image en fonction de leur recouvrement et un coût de cohérence entre les détections et les projections est calculé. Une étape optionnelle de raffinement de la pose de la caméra, comme celle proposée dans le chapitre 3 peut également être employée.

Les différentes hypothèses de poses, obtenues à partir des objets, sont ensuite utilisées pour guider la mise en correspondance des points 3D de la carte avec les points d'intérêt détectés dans l'image. Finalement, l'hypothèse de pose avec le coût minimal de reprojection des objets et avec au moins 30 correspondances de points est sélectionnée et raffinée en minimisant l'erreur de reprojection des points. C'est cette collaboration entre points et objets que l'on nomme *objects+points* par la suite.

Le système de SLAM peut alors reprendre le suivi. Si trop peu de correspondances de points ont pu être établies, par exemple dans un environnement faiblement texturé, l'hypothèse de pose avec le coût minimal de reprojection des objets est sélectionnée, sans raffinement par points. Dans ce cas, la pose courante de la caméra est estimée uniquement à partir des objets, mais le suivi du SLAM ne peut pas reprendre et le système déclenche une nouvelle procédure de relocalisation à l'arrivée de l'image suivante.

4.3.4 Utilisation des objets pour le suivi de la caméra

Nous explorons également la question de l'utilisation des objets dans le suivi de la caméra, hors du contexte de relocalisation. Notre système, OA-SLAM, repose uniquement sur les points pour effectuer ce suivi, mais nous avons créé deux autres versions impliquant les objets dans l'ajustement de faisceaux. Dans une première version, appelée *Obj_ dets*, les objets interviennent sous la forme de coûts additionnels dans l'ajustement de faisceaux, sans mettre à jour leurs modèles ellipsoïdaux. Dans une seconde, appelée *Full_ BA*, les modèles des objets sont entièrement intégrés, créant un ajustement de faisceaux joint entre les poses de la caméra, les points 3D et les objets. Dans les deux versions, les coûts associés aux objets sont calculés par la distance de Wasserstein entre leurs projections et leurs détections dans les images. Une difficulté avec la seconde formulation vient du fait que les termes de coût calculés sur les points et ceux calculés sur

les objets doivent être équilibrés. Nous avons donc manuellement appliqué un facteur d'échelle sur les résidus des objets afin d'obtenir des valeurs du même ordre de grandeur que les résidus des points.

Notre objectif avec ces deux autres versions est de voir si l'utilisation d'objets modélisés de manière approximative, par des ellipsoïdes, est bénéfique pour l'estimation de la trajectoire de la caméra dans un SLAM. Ces deux versions sont illustrées dans la figure 4.10 et sont comparées à notre système dans la section 4.4.5.

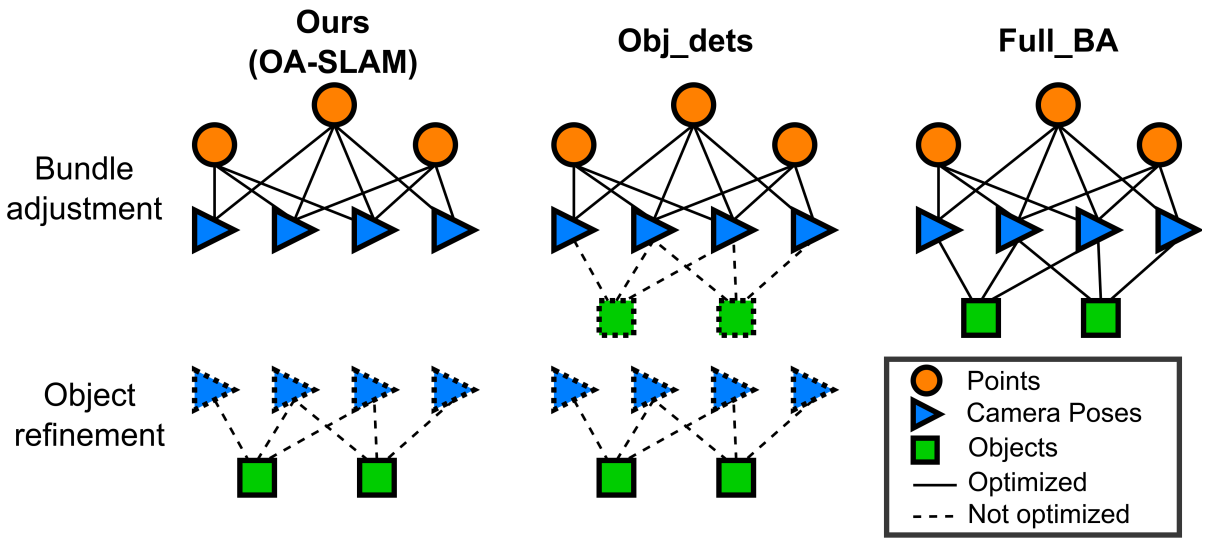


FIGURE 4.10 – Trois configurations différentes pour l'intégration des objets dans l'ajustement de faisceaux du SLAM. À gauche (notre méthode) : objets et points séparés. Au milieu : objets comme nouveaux termes de coût. À droite : optimisation jointe des poses de caméra, points et objets.

4.4 Expériences et résultats

4.4.1 Jeux de données

Nous présentons ici les jeux de données utilisés pour évaluer notre système de SLAM.

TUM RGB-D

TUM RGB-D [Stu+12] est un ensemble de données bien connu pour l'évaluation de méthodes d'odométrie visuelle et de SLAM. Il fournit des images couleur et de profondeur enregistrées avec un capteur de type Kinect, ainsi que la trajectoire de vérité terrain de la caméra. Cette trajectoire de référence a été obtenue à partir d'un système de capture de mouvement de haute précision avec huit caméras de suivi à haute vitesse. Nous avons, en particulier, utilisé les séquences *fr2/desk* et *fr3/long_office_household*. Des échantillons de ces séquences sont disponibles dans la figure 4.11. Ce jeu de données se limite néanmoins à une seule trajectoire de caméra, essentiellement orbitale, par scène.

FIGURE 4.11 – Échantillons d'images des séquences *fr2/desk* et *fr3/long_office_household* du jeu de données TUM RGB-D.

Notre propre jeu de données

Afin d'avoir un meilleur contrôle sur la caméra et d'obtenir plusieurs trajectoires d'une même scène, nous avons également enregistré nos propres jeux de données. Cela nous permet, par exemple, d'évaluer notre méthode dans des scénarios comportant d'importants changements de points de vue, d'échelle et des cas de perte du suivi de la caméra, qui sont assez fréquents en réalité augmentée.

Les images ont été acquises avec la caméra d'un smartphone standard. Nous avons utilisé COLMAP [SF16], un système de Structure-from-Motion (SfM) pour obtenir les poses de vérité terrain de la caméra ainsi que ses paramètres intrinsèques. Ces scènes sont illustrées dans la figure 4.12 et contiennent, à la fois, des objets généraux (livre, chaise, tasse, ...), mais aussi des objets plus spécifiques (statue, amphore, ...). Les scènes de l'évier et du bureau, illustrées respectivement dans les figures 4.21 et 4.22 montrent, par exemple, comment notre système peut être utilisé dans un environnement de la vie quotidienne, en utilisant un détecteur d'objets pré-entraîné. Les scène illustrées dans les figures 4.17 et 4.23 sont plus spécifiques, avec des objets de musées. Le détecteur d'objets a dû être affiné sur quelques images annotées manuellement (environ 50). Cela donne un aperçu de l'utilisation de notre système pour des applications de réalité augmentée dans un musée.

FIGURE 4.12 – Échantillons d'images des séquences de test que nous avons enregistrées.

4.4.2 Implémentation

OA-SLAM a été implémenté en C++, en se basant sur ORB-SLAM2 (`github.com/raulmur/ORB_SLAM2`). Les bibliothèques suivantes ont été utilisées :

— *Pangolin*, utilisée pour la visualisation de la carte reconstruite ainsi que les parties de réalité augmentée.

— *OpenCV*, utilisée pour manipuler les images, extraire les points d'intérêt et calculer les descripteurs locaux.

— *DBoW2*, utilisée pour le calcul des descripteurs globaux *Bag-of-Words* et la recherche de l'image la plus proche.

— *Eigen3*, utilisée pour la manipulation d'objets mathématiques et les opérations d'algèbre linéaire.

— *g2o*, utilisée résoudre les optimisations non linéaires sous forme de graphe.

— *Dlib*, utilisée pour résoudre le problème d'affectation, avec l'algorithme hongrois.

— *RPG Monocular Pose Estimator*, utilisée pour l'implémentation d'un algorithme P3P, permettant le calcul de pose à partir de trois points.

Enfin, YOLOv5 (`github.com/ultralytics/yolov5`), implémenté avec la bibliothèque *PyTorch* a été utilisé pour la partie détection d'objets. Le code d'OA-SLAM est disponible à l'adresse suivante : `gitlab.inria.fr/tangram/oa-slam`.

4.4.3 Cartographie des objets de la scène

Dans cette section, nous évaluons la cartographie d'objets de notre système, qui reconstruit des modèles ellipsoïdaux des objets de la scène, à la volée, en utilisant les poses calculées et les détections des objets. La figure 4.13 montre la carte reconstruite sur les séquences *fr2/desk* et *fr3/long_office_household*. On peut y voir les ellipsoïdes et les points 3D estimés (1 et 2), ainsi que leurs associations (3). Les reconstructions denses (4) sont données uniquement pour illustration et permettent de s'assurer que les modèles d'objets reconstruits sont globalement bien positionnés par rapport aux objets.

La figure 4.14 montre une scène de petite taille mais compliquée en termes de reconstruction car trois objets identiques sont placés côte à côte. Dans ce cas, les classes d'objets ne peuvent pas être utilisées pour contraindre l'association de données. De plus, deux d'entre eux sont occultés lorsqu'ils sont vus de côté. Notre système parvient tout de même à reconstruire trois modèles précis des objets.

Enfin, nous comparons la capacité de cartographie de notre système avec celle d'EAO-SLAM [Wu+20] dans la figure 4.15. Nous avons utilisé le code publié à l'adresse `github.com/yanmin-wu/EAO-SLAM`. Le scénario est simple, avec des objets placés sur une table et une caméra décrivant une trajectoire orbitale tout autour. Les mêmes détections d'objets sont utilisées dans les deux méthodes. Pour EAO-SLAM, nous avons dû, en plus, régler manuellement la direction verticale de la carte, afin que les objets puissent être estimés de manière parallèle au sol. Les deux méthodes sont basées sur ORB-SLAM2 et reconstruisent des nuages de points similaires. En ce qui concerne les objets, notre méthode reconstruit des modèles ellipsoïdaux ajustés aux objets présents sur la table, qu'ils soient relativement grands (l'ordinateur portable), fins (les bouteilles) ou petits (la souris). EAO-SLAM, de son côté, fait une distinction entre les objets de forme régulière ou non et les représente respectivement par des cuboïdes et des ellipsoïdes. La carte reconstruite avec EAO-SLAM présente quelques erreurs. La souris et le livre noir (l'ellipsoïde rouge à droite dans la carte) sont dupliqués. Ils ont été reconstruits à la fois par un ellipsoïde et un cuboïde. Les bouteilles sont représentées par des ellipsoïdes plus larges qui épousent moins bien leur forme. Enfin, la tasse placée derrière l'ordinateur portable a une reconstruction ellipsoïdale bien trop grande (le gros ellipsoïde vert au milieu de leur carte en comparaison du petit ellipsoïde cyan dans notre carte).

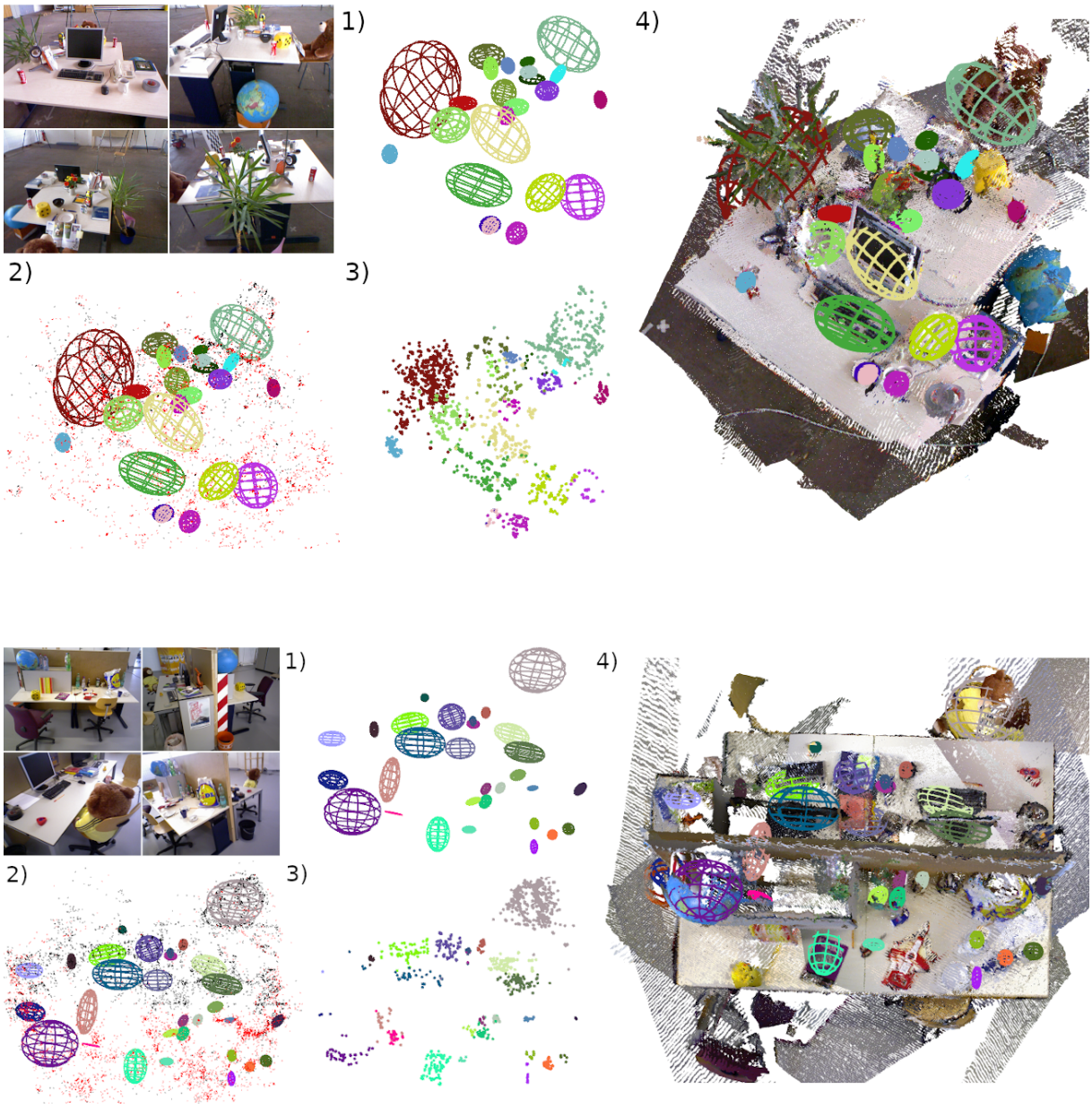


FIGURE 4.13 – Cartes reconstruites avec notre système sur les séquences *fr2/desk* (en haut) et *fr3/long_office_household* (en bas). Les images en haut à gauche donnent un aperçu de chaque séquence et les résultats correspondent à : 1) la carte d'objets reconstruite ; 2) la carte combinant objets et points ; 3) les points 3D associés aux objets ; 4) la carte des objets reconstruits affichée sur une reconstruction dense de la scène (uniquement pour la visualisation).

FIGURE 4.14 – La carte reconstruite avec notre système pour trois objets identiques placées côte à côte. Les images sur la gauche donnent un aperçu de la séquence.

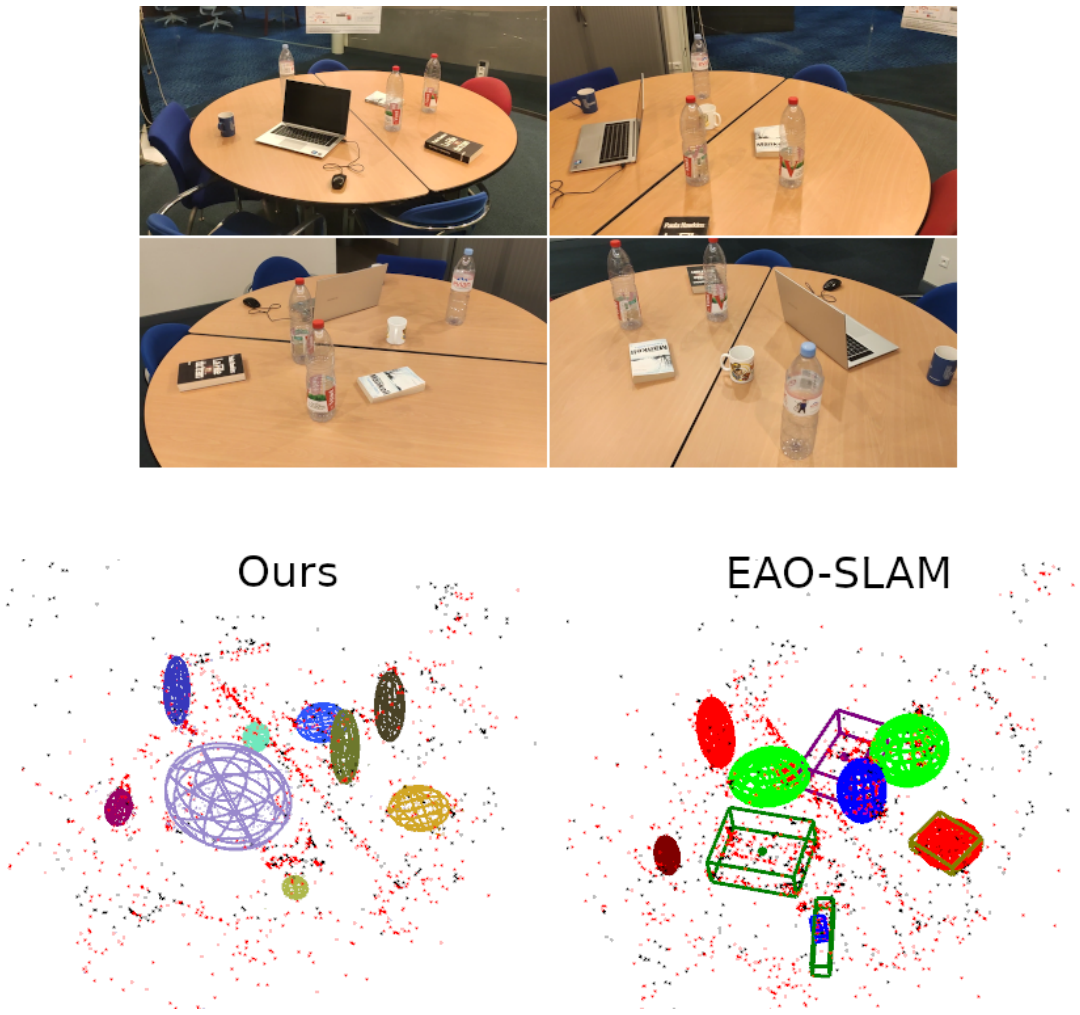


FIGURE 4.15 – Comparaison entre la carte obtenue avec notre système et celle obtenue par EAO-SLAM [Wu+20]. La carte semi-dense reconstruite par EAO-SLAM n'est pas montrée ici car elle n'est utilisée que pour de la visualisation dans leur travail.

4.4.4 Résultats de relocalisation

Dans cette section, nous analysons les avantages de l'utilisation jointe des objets et des points dans la procédure de relocalisation d'un SLAM. Le scénario de l'expérience est le suivant : nous cartographions d'abord la scène à partir de points de vue limités, principalement d'un côté, avec notre système SLAM, puis nous exécutons la procédure de relocalisation dans la scène reconstruite sur des images de test provenant de différents points de vue. Étant donné qu'une seule séquence d'images est disponible dans TUM RGB-D pour la scène *fr2/desk*, nous l'avons divisée en deux parties : les 700 premières images ont été utilisées pour la cartographie tandis que les 2266 images restantes sont utilisées évaluer la relocalisation. Nous avons également utilisé nos propres séquences d'images offrant des points de vue plus variés, en termes d'angle et d'éloignement.

Les figures 4.16 et 4.17 comparent les positions estimées de la caméra par notre méthode avec celles obtenues en utilisant la méthode d'ORB-SLAM2. On voit clairement que notre méthode est capable de localiser la caméra avec une bonne précision pour des points de vues très variés, tout autour de la scène. Les résultats sur la figure 4.16, montrent même que notre méthode est capable de se relocaliser pour des points de vue à l'opposé de ceux utilisés pour la cartographie de la scène. La séquence *high* de la figure 4.17 montre les mêmes résultats pour des changements d'élévation de la caméra par rapport à la trajectoire utilisée pour la cartographie. Enfin, les séquences *tour* et *near-far* font varier de manière significative l'éloignement de la caméra par rapport à la scène. La précision des poses estimées par notre méthode ne diminue que légèrement pour des images prises très loin ou très près de la scène. Ces résultats sont obtenus grâce à la robustesse du détecteur d'objets qui est capable de détecter des objets sous pratiquement n'importe quel point de vue. Au contraire, la méthode de relocalisation basée point d'ORB-SLAM2 ne fonctionne que sur des petites portions de la trajectoire, proches des points de vue utilisés pour la cartographie.

Ces résultats sont confirmés par les courbes des figures 4.18 et 4.19, qui montrent qu'une proportion beaucoup plus grande d'images peut être localisée en introduisant les objets dans la relocalisation. On peut noter que l'utilisation des objets seuls dégrade légèrement la précision des poses estimées par rapport à la méthode basée sur les points (voir la partie inférieure gauche de la courbe). Cela s'explique principalement par la représentation assez grossière des objets sous forme d'ellipsoïdes, mais élargit considérablement sa zone de fonctionnement. L'utilisation conjointe des points et des objets permet de combiner les avantages de chacun, à savoir la précision des points et la robustesse des objets. Le plateau à environ 70% d'images localisées avec notre méthode sur la séquence *fr2/desk* peut être expliqué par le fait que très peu d'objets (1 voire 0) sont visibles dans une partie des images.

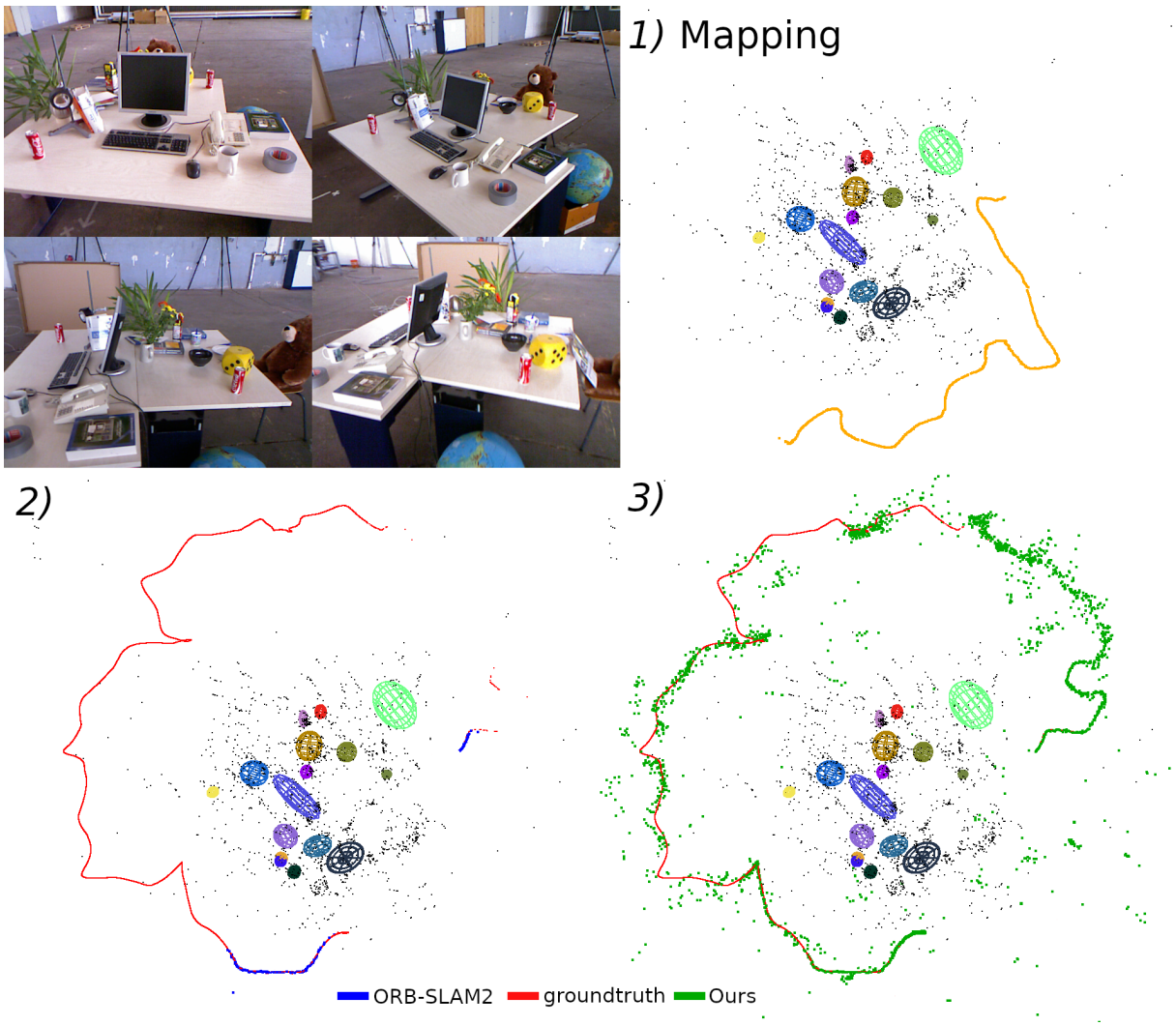


FIGURE 4.16 – Caméras estimées par notre module de relocalisation (image par image) sur la séquence *fr2/desk*. Les images en haut à gauche donnent un aperçu de la séquence utilisée pour la cartographie. 1) La trajectoire orange a été utilisée pour la reconstruction. 2) Les positions de caméra estimées par ORB-SLAM2 sont en bleu. 3) Les positions de caméra estimées par notre système sont en vert. La trajectoire rouge correspond à la vérité terrain des positions de la caméra.

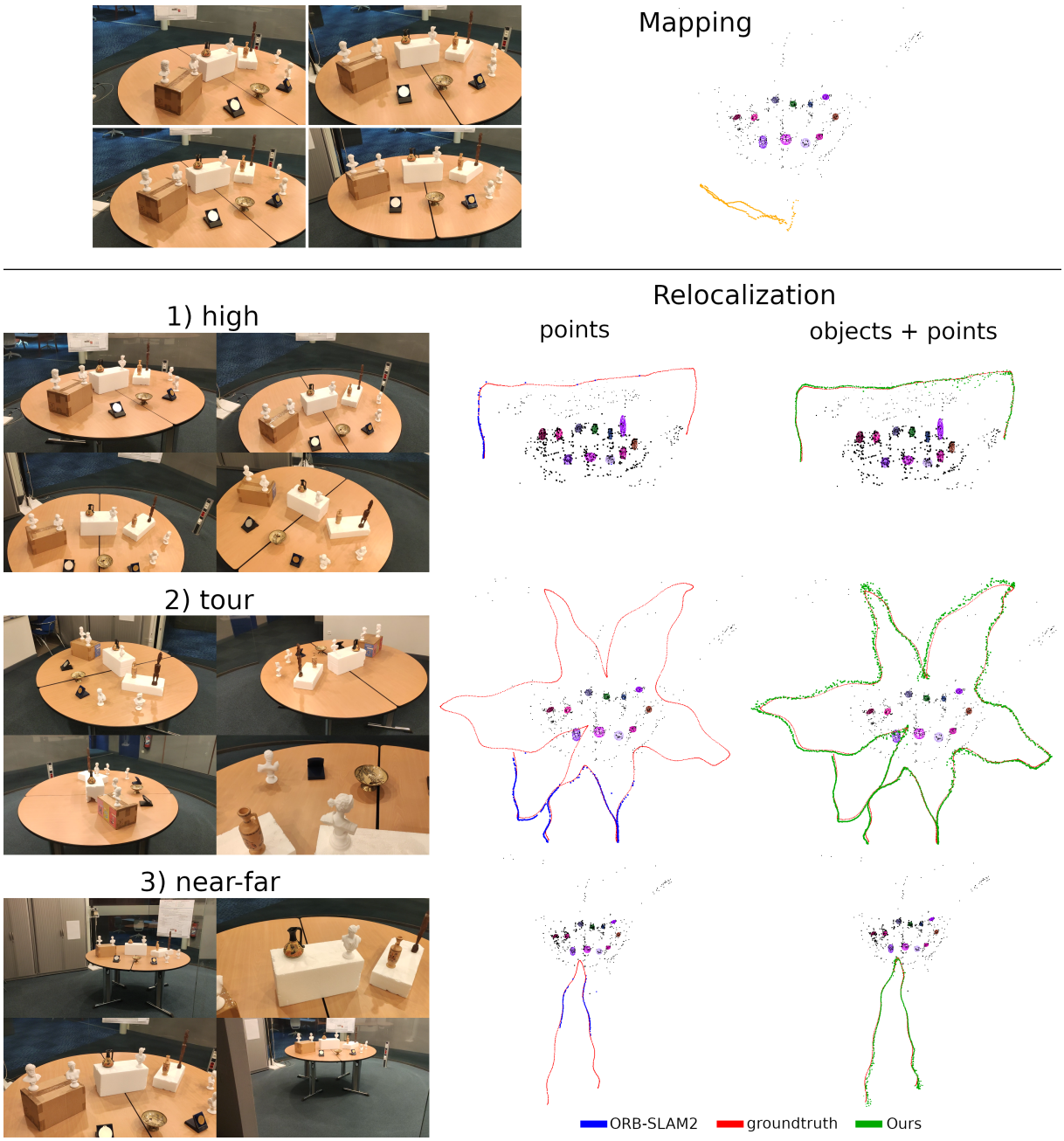


FIGURE 4.17 – Caméras estimées par notre module de relocalisation (image par image) sur les séquences *high*, *tour* et *near-far*, avec de larges changements de points de vue. En haut : l'étape de cartographie de la scène (images, trajectoire de caméra et carte reconstruite). En-dessous : les résultats de relocalisation pour les 3 séquences de test. Les trajectoires de caméra estimées par notre système sont en vert, celles obtenues avec ORB-SLAM2 sont en bleu et la vérité terrain en rouge. Les images dans la colonne de gauche donnent un aperçu des points de relocalisation.

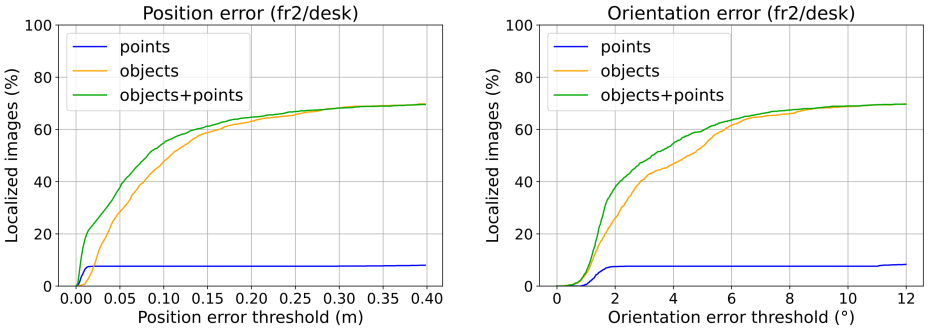


FIGURE 4.18 – Pourcentage des images correctement localisées en fonction d'une erreur de position (à gauche) ou d'orientation (à droite) sur la séquence *fr2/desk*. Les courbes nommées *points* correspondent à ORB-SLAM2, *objects+points* à notre système OA-SLAM et *objects* à une approche intermédiaire utilisant uniquement les objets.

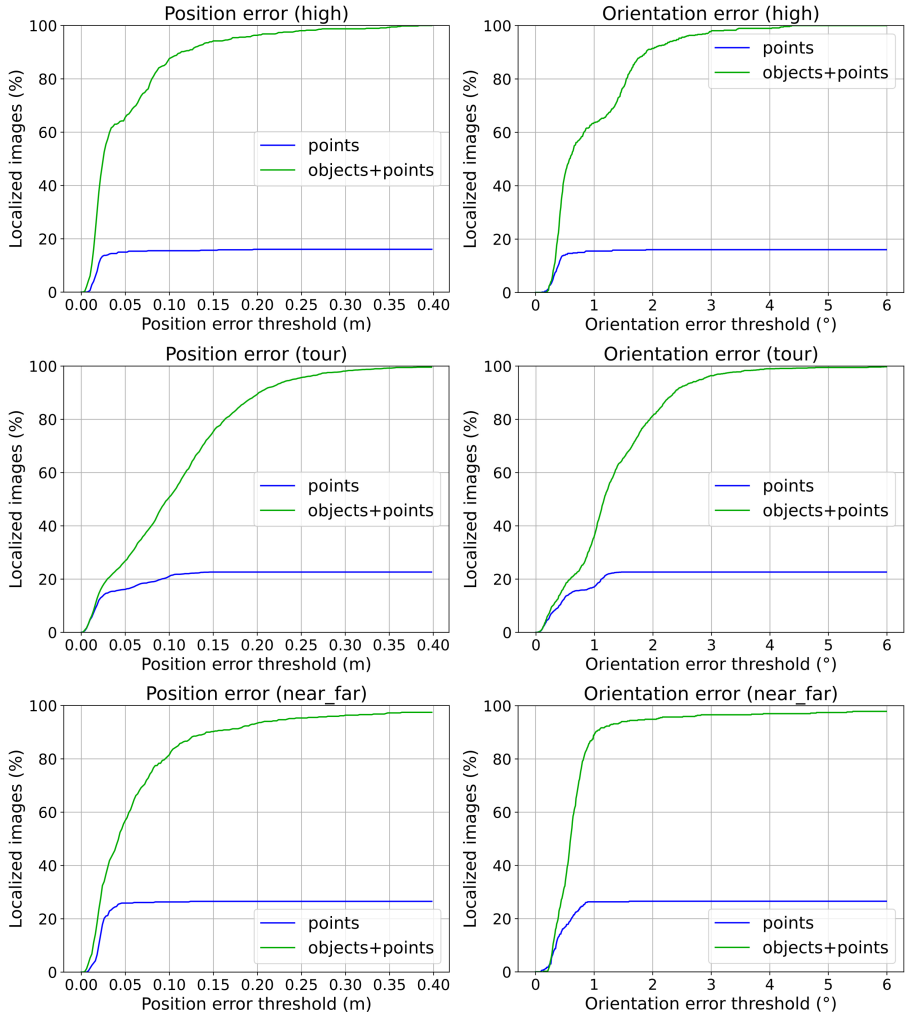


FIGURE 4.19 – Pourcentage des images correctement localisées en fonction d'une erreur de position (à gauche) ou d'orientation (à droite) sur les séquences *high*, *tour* et *near-far*. Les courbes nommées *points* correspondent à ORB-SLAM2 et *objects+points* à OA-SLAM.

4.4.5 Intégration des objets dans l'ajustement de faisceaux

Nous avons comparé les performances du suivi de la caméra entre notre système et les deux versions intégrant les objets dans l'ajustement de faisceaux (*Obj_dets* et *Full_BA*). Pour cela, nous avons mesuré les erreurs de position obtenues pour les images clés sur les séquences *fr2/desk* et *fr3/long_office_household*. Les résultats, disponibles dans le tableau 4.2, montrent que l'intégration des objets dans l'ajustement de faisceaux local diminue légèrement la précision de la pose estimée. À noter que les résultats d'ORB-SLAM2 sont les mêmes que ceux de notre méthode, OA-SLAM, car son suivi de la caméra, sans relocalisation, est équivalent à notre système.

Ces résultats sont similaires à ceux obtenus par QuadricSLAM [NMS19] et SO-SLAM [Lia+22], pour lesquels les auteurs ont également observé une baisse de la précision du suivi de la caméra avec la prise en compte des objets. Seuls Hosseinzadeh *et al.* avaient observé une amélioration [HLLR19]. Nous expliquons ces résultats par la modélisation très approximative des objets par des ellipsoïdes et par leur détection sous forme de boîtes englobantes alignées aux axes des images. Ils ne semblent pas être en mesure de fournir un meilleur niveau de précision au suivi de caméra que l'approche basée point d'ORB-SLAM2, dans des zones suffisamment texturées.

Sequence	OA-SLAM (ours)	Obj_dets	Full_BA
fr2/desk	**0.808**	0.901	0.860
fr3/long_office_household	**1.180**	1.978	2.143

TABLE 4.2 – RMSE (en cm) des erreurs de trajectoire absolue (ATE) des images clés, obtenues pour les trois configurations de l'intégration des objets dans l'ajustement de faisceaux du SLAM. Les valeurs données sont les valeurs médianes obtenues pour 20 exécutions.

4.4.6 Analyse du temps de calcul

Nous avons mesuré les temps médians suivants, par image, dans les expériences décrites sur la figure 4.17 :

— 13.9 ms pour la détection des objets.

— 27 ms pour le suivi de la caméra et des objets.

— 22 ms pour une éventuelle procédure de relocalisation.

Ces valeurs ont été mesurées avec un CPU Intel Xeon 3.6GHz et une carte graphique NVIDIA Quadro P4000 utilisée pour la détection des objets. La rapidité de notre système dépend évidemment du nombre d'objets présents dans la scène et de leur diversité de catégorie. On peut noter que la scène utilisée dans cette expérience comporte un nombre relativement élevé d'objets (11), avec, en particulier, cinq objets de même type (les statues), ce qui augmente largement le nombre d'associations potentielles d'objets. À titre de comparaison, le suivi de la caméra d'ORB-SLAM2 nécessite un temps médian de 25 ms, à peine moins que notre méthode, ce qui montre que le suivi des objets est très rapide. Nous avons également mesuré un temps médian de 8.75 ms pour la procédure de relocalisation d'ORB-SLAM2. C'est effectivement plus rapide que notre approche mais les expériences ont montré qu'elle échoue beaucoup plus fréquemment lors de variations importantes de point de vue.

4.5 Application à la réalité augmentée

Nous présentons ici deux applications de notre système dans des scènes de la vie quotidienne, telles qu'une cuisine ou une salle à manger. Ces scènes illustrent également bien la notion assez large du terme « objet » et contiennent des éléments fixes, comme un évier, qui constituent des balises intéressantes pour la relocalisation visuelle. À noter que, comme ORB-SLAM2, notre système peut être utilisé dans deux modes différents :

— Le mode « SLAM » qui est le mode par défaut et qui réalise, en parallèle, le suivi de la caméra et des objets, la reconstruction de la carte et la fermeture de boucle. Ce mode est illustré dans la section 4.5.2.

— Le mode « Localisation » qui réalise uniquement le positionnement visuel de la caméra et le suivi des objets et des points. Les modules de reconstruction de la carte et de fermeture de boucle sont désactivés. Ce mode est principalement utilisé en réalité augmentée lorsqu'une bonne reconstruction de la carte de l'environnement de travail existe déjà (voir section 4.5.1).

4.5.1 Suivi de caméra à partir d'une scène reconstruite

La capacité de relocalisation améliorée de notre système est particulièrement intéressante pour initialiser le suivi 3D de la caméra dans des applications de réalité augmentée. Une fois qu'une carte de la zone de travail a été reconstruite, et potentiellement augmentée avec des éléments virtuels, une pose initiale de la caméra par rapport à cette carte est nécessaire pour initier une nouvelle séquence de localisation. Nous montrons, dans les figures 4.20, 4.21, 4.22 et 4.23, comment notre méthode peut être utilisée pour des scénarios complexes, où la scène est vue principalement d'un côté, à une distance constante, pendant la cartographie, puis la localisation est effectuée à partir de points de vue opposés, à des distances variables. En comparaison, ORB-SLAM2 rencontre de grosses difficultés pour initialiser la localisation.

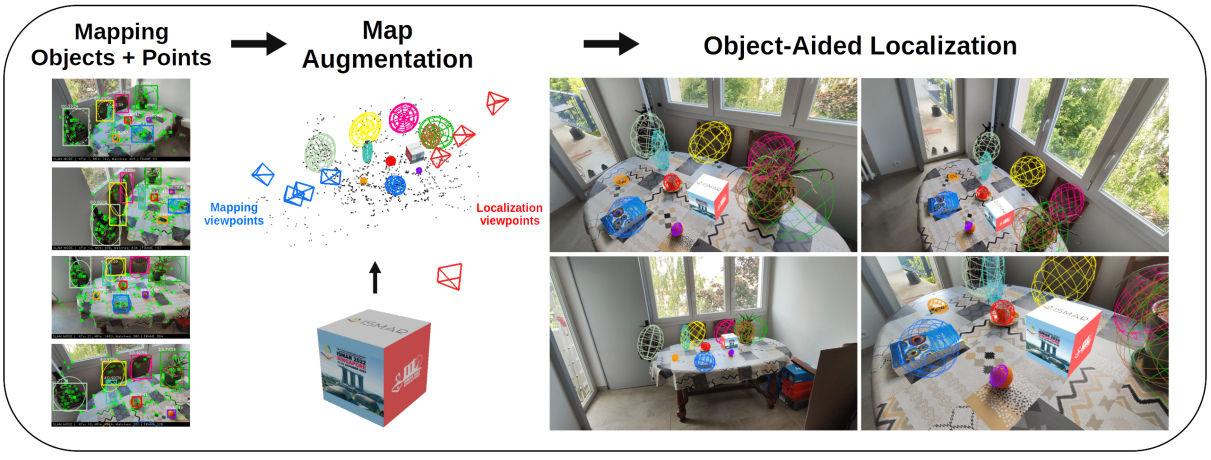


FIGURE 4.20 – Illustration du mode « Localisation » de OA-SLAM et son initialisation par rapport à une carte existante. La scène de travail a été cartographiée, dans un premier temps, pour obtenir une carte de points et d'objets. Des éléments virtuels peuvent ensuite être ajoutés à cette carte. Enfin, OA-SLAM est utilisé pour le suivi de la caméra par rapport à cette scène et permet d'obtenir des images augmentées de la scène.

FIGURE 4.21 – Illustration de l'initialisation d'une séquence de localisation à partir d'une carte existante. La carte a été créée à partir de la séquence illustrée par les images en haut à gauche. Les images en dessous montrent les résultats obtenus par notre système en mode « Localisation » par rapport à la carte existante. La bonne adéquation entre les objets dans l'image et les projections de leurs modèles ellipsoïdaux démontre une localisation précise. Des échantillons de points de vue utilisés pour la cartographie sont affichés en bleu dans la carte et ceux de localisation sont affichés en rouge.

FIGURE 4.22 – Illustration de l'initialisation d'une séquence de localisation à partir d'une carte existante. La carte a été créée à partir de la séquence illustrée par les images en haut à gauche. Les images en dessous montrent les résultats obtenus par notre système en mode « Localisation » par rapport à la carte existante. La bonne adéquation entre les objets dans l'image et les projections de leurs modèles ellipsoïdaux démontre une localisation précise. Des échantillons de points de vue utilisés pour la cartographie sont affichés en bleu dans la carte et ceux de localisation sont affichés en rouge.

FIGURE 4.23 – Illustration de l'initialisation d'une séquence de localisation à partir d'une carte existante. La carte a été créée à partir de la séquence illustrée par les images en haut à gauche. Les images en dessous montrent les résultats obtenus par notre système en mode « Localisation » par rapport à la carte existante. La bonne adéquation entre les objets dans l'image et les projections de leurs modèles ellipsoïdaux démontre une localisation précise. Des échantillons de points de vue utilisés pour la cartographie sont affichés en bleu dans la carte et ceux de localisation sont affichés en rouge.

4.5.2 Reprise de SLAM après une perte de suivi

L'amélioration de la capacité de relocalisation de notre système se montre également utile dans des cas de perte de suivi de la caméra, ce qui peut arriver fréquemment dans des applications de réalité augmentée où le mouvement de la caméra est laissé totalement libre à l'utilisateur (mouvements brusques, caméra filmant le sol ou le plafond, ...). Nous présentons, dans la figure 4.24, la gestion d'une perte du suivi de la caméra par notre système :

— (1-4) Le système démarre le suivi de la caméra et construit une carte de points et d'objets.

— (5-6) Le suivi se perd à cause d'un mouvement brusque de la caméra. Celle-ci est orientée vers le sol pendant un certain temps.

— (7-9) Lorsque la scène reconstruite est à nouveau visible, le module de relocalisation estime la pose de la caméra à partir des objets, établit des correspondances entre points et permet au suivi visuel de la caméra et à la cartographie de reprendre.

Dans un tel scénario, notre approche basée sur les objets étend considérablement la capacité de reprise du SLAM par rapport aux approches classiques basées uniquement sur des points. En particulier, ORB-SLAM2 ne parvient pas à reprendre le suivi de la caméra sur cette séquence.

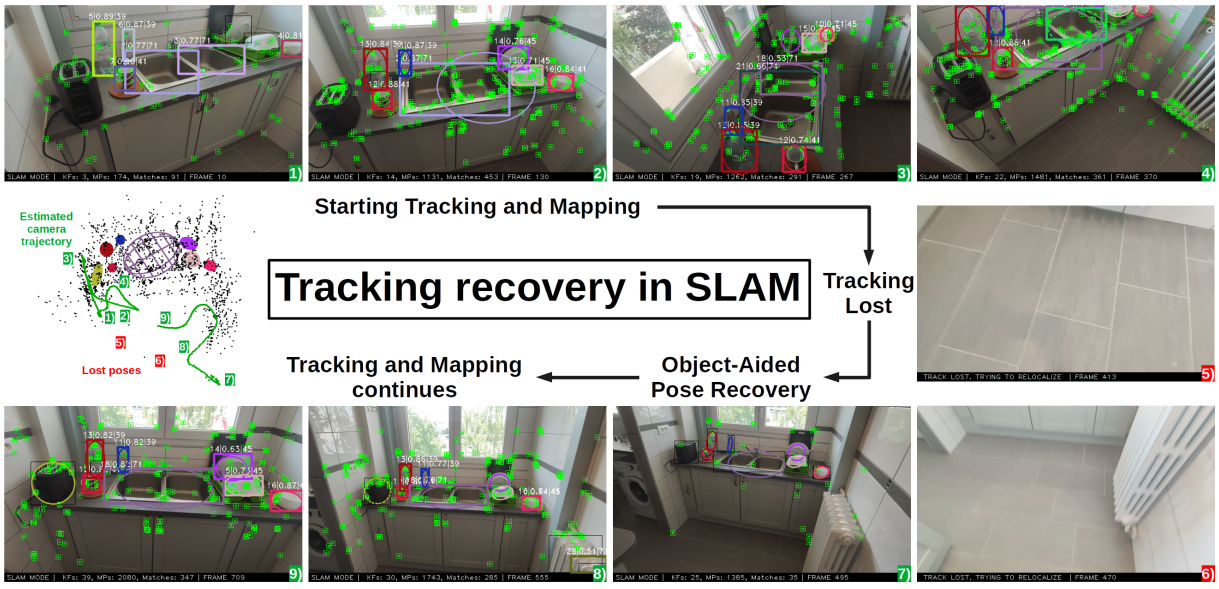


FIGURE 4.24 – Illustration de la reprise du SLAM après une perte de suivi de la caméra. Les ellipses visibles dans les images correspondent à la projection des modèles ellipsoïdaux des objets avec la pose estimée. Les boîtes visibles dans les images sont les détections d'objets fournies par YOLO et les valeurs inscrites au-dessus correspondent respectivement à l'identifiant de l'objet associé, son score de détection et sa classe.

Une vidéo illustrant la cartographie et la relocalisation d'OA-SLAM est disponible à l'adresse : https://zinsmatt.github.io/these/#ISMAR

4.5.3 Modélisation des objets par parties

Dans cette dernière partie, nous montrons, qu'en fonction du contexte, il peut être intéressant d'adapter le niveau de détail des objets considérés dans la scène (objets complets ou parties d'objets). Par exemple, lorsque la caméra observe la scène de relativement loin, les objets apparaissent

petits dans l'image mais sont généralement en nombre suffisant pour permettre la relocalisation (au moins trois). Dans ce cas, la modélisation de chaque objet par un ellipsoïde global est suffisante. Au contraire, lorsque la caméra se rapproche, seul un ou deux objets peuvent être visibles et une modélisation par parties devient alors souhaitable, afin d'augmenter le nombre de balises potentielles pour la localisation. Cela peut également permettre une meilleure gestion des objets partiellement occultés.

Notre système permet de gérer cela de manière flexible, puisque le réseau de détection peut être ajusté pour détecter des parties distinctes d'un objet. Cela nécessite simplement des annotations manuelles de ces parties sous la forme de boîtes englobantes dans quelques images d'entraînement. La figure 4.25 illustre un résultat de relocalisation obtenu avec notre méthode après un réglage fin de YOLO pour détecter, soit la statue dans sa globalité, soit des parties de statue (tête, épaules et pied). Dans la ligne du haut, avec une caméra proche de la scène, la tête, les épaules et le pied de la statue sont utilisés pour estimer la pose de la caméra. À l'inverse, dans les deux images du bas, où la caméra est plus éloignée par rapport à la scène, la statue complète est utilisée, conjointement avec trois autres objets.

FIGURE 4.25 – Illustration de la modélisation d'un objet par parties. À gauche : la scène reconstruite. À droite : des résultats de relocalisation en utilisant soit un modèle complet de la statue, en bas, soit des modélisations de ses parties (tête, épaules, pied), en haut.

4.6 Conclusion

Dans ce chapitre, nous avons développé un système de SLAM monoculaire alliant objets et points. Ce système est capable de reconstruire, à la volée, une carte de points 3D et d'objets de la scène, modélisés par des ellipsoïdes. Nous y avons intégré la méthode d'estimation de pose de caméra à partir d'un ensemble minimal d'objets présentée dans le chapitre 2, ainsi que la procédure de raffinement de la pose, proposée dans le chapitre 3, permettant d'utiliser tous les objets détectés. Les expériences réalisées ont montré l'apport des objets dans le cas où le suivi de la caméra est perdu et où une relocalisation est nécessaire. Contrairement à ORB-SLAM2, qui ne permet de se relocaliser qu'à partir de points de vue relativement proches de la trajectoire passée de la caméra, notre système est capable de se relocaliser de manière plus robuste pour des points de vues très variés. Enfin, nous avons montré l'intérêt de notre système pour des applications de réalité augmentée, dans des cas de perte de suivi de la caméra ou pour initialiser une séquence de localisation par rapport à une carte existante de l'environnement de travail.

Conclusion

1 Résumé des contributions

Nous avons, dans un premier temps, cherché à améliorer la précision de l'estimation de la pose basée objet, sans nécessiter de modèles 3D précis des objets. En particulier, nous avons proposé une détection améliorée des objets par des ellipses orientées, qui sont cohérentes avec la projection de leurs modèles ellipsoïdaux. Au travers de nos expériences, nous avons montré qu'en apprenant à partir de différents points de vue, notre réseau de prédiction d'ellipse est capable de prédire, à partir de l'apparence d'un objet, les paramètres d'une ellipse qui soit cohérente avec la projection de son modèle ellipsoïdal. Ces détections plus précises des objets, nous ont permis d'améliorer considérablement la précision du calcul de pose. Nous avons également démontré trois caractéristiques très importantes de notre méthode, qui sont son invariance au modèle ellipsoïdal choisi d'un objet, sa robustesse à un bruit sur la boîte de détection et le petit nombre d'annotations manuelles requises pour l'apprentissage (le reste étant automatique).

Nous avons également proposé une approche jointe de détection d'instances d'objets et de prédiction d'ellipses, qui offre un gain de vitesse important au calcul de la pose et rend son déploiement plus facile, avec un seul réseau.

Toujours dans le but d'améliorer la précision de la pose de la caméra, nous avons développé une étape de raffinement de cette pose, qui permet de prendre en considération tous les objets de la scène détectés dans l'image. Cette étape cherche à aligner les cônes de projection et rétroprojection lors du calcul de la pose de la caméra, ce qui permet de relâcher la supposition, théoriquement fausse, de l'alignement par projection centrale des centres des ellipsoïdes avec les centres des ellipses. Elle s'exprime sous la forme d'une minimisation de l'erreur entre les ellipses détectées dans l'image et celles obtenues par la projection des modèles ellipsoïdaux des objets. Nous avons proposé une métrique ellipse-ellipse basée sur des ensembles de niveaux et nous l'avons comparée avec différentes autres formulations de coût entre deux ellipses. Nos expériences ont montré que cette métrique donne les meilleurs résultats et permet d'améliorer de manière significative la précision de la pose estimée initialement. Nous avons montré ses bonnes propriétés de convergence ainsi que sa meilleure gestion des objets partiellement détectés. Enfin, nous avons proposé une méthode de quantification de l'incertitude des ellipses de détection prédites et nous avons montré que son utilisation pour pondérer la contribution de chaque objet permet d'améliorer la précision du calcul de pose.

Ces travaux ont montré que, malgré l'utilisation de modèles approximatifs des objets, leur détection ajustée spécifiquement à la scène de travail permet d'obtenir une précision de pose qui se rapproche de celle des méthodes basées sur les points.

Dans un second temps, nous avons exploré l'utilisation des objets dans un SLAM monoculaire, notamment dans sa procédure de relocalisation. Pour cela, nous avons dû repasser à des détections plus génériques des objets sous forme de boîtes englobantes, alignées avec les axes de l'image. Bien que cela diminue la précision de la pose calculée uniquement sur les objets, l'idée est de

tirer avantage de leur robustesse pour obtenir une première estimation de la pose. Une meilleure précision est ensuite apportée par les points.

Nous avons développé OA-SLAM, un système de SLAM monoculaire capable de reconstruire, à la volée, une carte sémantique de points et d'objets. Nous y avons intégré la méthode de calcul de la pose de la caméra à partir de trois paires ellipse-ellipsoïde présentée dans le chapitre 2, ainsi que la procédure de raffinement de la pose, proposée dans le chapitre 3 et permettant de prendre en compte tous les objets détectés.

Nos expériences ont montré que la collaboration entre les points et les objets dans la procédure de relocalisation permet de bénéficier de leurs avantages respectifs de précision et de robustesse. Contrairement à ORB-SLAM2 qui ne permet de se relocaliser qu'à partir de points de vue relativement proches de la trajectoire passée de la caméra, OA-SLAM est capable de se relocaliser de manière plus robuste avec des points de vues très variés. Nous avons également montré l'intérêt de notre système pour des applications de réalité augmentée avec des cas de perte de suivi de la caméra ou pour initialiser une séquence de localisation par rapport à une carte existante de l'environnement de travail.

2 Perspectives

La précision du calcul de la pose de la caméra à partir d'objets repose essentiellement sur l'adéquation entre leurs modélisations 3D et leurs observations 2D dans les images. Malgré le fait que la détection d'un objet sous la forme d'une boîte soit imprécise, nous avons choisi, dans le chapitre 2, de la rendre cohérente avec la projection des modèles ellipsoïdaux des objets. Cela nous a permis d'obtenir des poses très précises pour une instance d'objet donnée, malgré une modélisation approximative. De nos jours, certains types d'objets sont présents en très grande quantité dans des bases de données. Pour ces objets, un raisonnement par catégorie avec un espace de coordonnées normalisé, similaire à celui proposé dans NOCS [Wan+19b] et capable de représenter toutes les instances d'un type d'objet, pourrait permettre d'étendre la généralisation de notre méthode et constitue une première direction intéressante à explorer.

Outre l'adéquation entre détections et modèles, une représentation plus précise des objets serait également bénéfique à la localisation visuelle basée objet. Par exemple, dans le cas des ellipsoïdes, deux voire trois objets sont nécessaires pour calculer la pose de la caméra. Au contraire, avec des modèles très précis, la pose peut être retrouvée à partir d'un seul objet. De plus, une modélisation précise offre également une information plus riche, mieux localisée dans la carte et plus utile pour d'autres applications, telles que la saisie d'un objet.

De nombreuses méthodes d'estimation de pose se sont appuyées sur des modèles 3D précis d'objets mais sont limitées à certains objets dont les modèles 3D sont connus. Au contraire, dans ce travail, nous avons utilisé une modélisation approximative et générique pour tous les objets sous la forme d'ellipsoïdes. Une approche hybride pourrait être envisagée, dans laquelle certains objets, bien connus et dont les modèles 3D sont disponibles, seraient modélisés de manière précise alors que des modélisations plus génériques seraient adoptées pour les autres objets de la scène.

L'utilisation de superquadriques comme primitives géométriques pourrait également être explorée. Celles-ci permettent de conserver la généricité des modèles virtuels ellipsoïdaux, avec également une représentation continue, tout en modélisant les objets de manière plus précise et mieux ajustée en 3D. La figure 1 illustre des modélisations d'un objet par des superquadriques qui se rapprochent d'un cuboïde, d'un cylindre, d'un ellipsoïde ou d'un octaèdre. L'utilisation de superquadriques pour modéliser les objets offre des perspectives intéressantes, mais également un défi, car aucune solution analytique n'existe pour le calcul de la pose à partir de superquadriques.

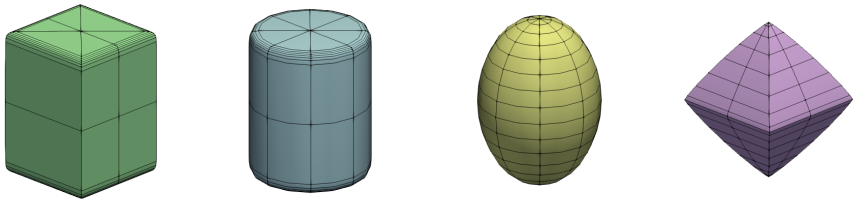


FIGURE 1 – Différentes forme de superquadriques.

La gestion des gros objets, notamment les éléments fixes du mobilier (table, bureau, canapé, lit, armoire, ...) dans les scène intérieures, est particulièrement importante, mais pose parfois problème car ces objets sont rarement entièrement visibles dans l'image. D'une part, la métrique *Level sets*, développée dans le chapitre 3, nous a permis de mieux traiter des sorties partielles des objets mais suppose tout de même qu'une partie importante de l'objet soit visible dans l'image, notamment pour pouvoir être détectée. D'autre part, notre approche permet une modélisation des objets par parties, qui peuvent être détectées avec un ajustement du réseau de détection. Le choix des parties reste cependant manuel et des approches automatiques et répétables de décomposition d'un objet pourraient être imaginées.

Du point de vue du SLAM, les objets ont été utilisés pour améliorer la procédure de relocalisation, qui est exécutée quand le suivi de la caméra échoue. Nous avons également tenté de les intégrer au suivi de la caméra, mais sans parvenir à l'améliorer pour l'instant. Les objets modélisés de façon approximative ne semblent pas apporter de gain de précision dans des zones suffisamment texturées, lorsqu'un grand nombre de points sont utilisés. Des situations dans lesquelles peu de points sont disponibles seraient également intéressantes à analyser, par exemple dans des environnements faiblement texturés ou avec un fort niveau de zoom.

D'autres utilisations des objets dans un SLAM pourraient également être envisagées, notamment pour réduire les erreurs de dérive. Par exemple, ils peuvent aider à détecter une fermeture de boucle. Dans les systèmes existants, comme ORB-SLAM2, celle-ci est faite en se basant sur l'apparence de l'image courante et en recherchant une image similaire parmi les images clés passées de la séquence.

Enfin, notre système OA-SLAM a principalement été évalué dans des environnements intérieurs de la taille d'une pièce. Le passage à plus grande échelle constitue une perspective intéressante, qui amènerait de nouvelles problématiques, notamment de gestion de la taille de la carte avec une combinatoire plus importante due au grand nombre d'objets présents et aux configurations ambiguës qui peuvent apparaître à cause d'objets répétés. Des critères de visibilité et un partitionnement de la carte, par pièce d'un bâtiment par exemple, pourraient être envisagés. Finalement, une évaluation à grande échelle, en extérieur, serait également intéressante.

Bibliographie

[Abd+21] Moloud Abdar et al. "A review of uncertainty quantification in deep learning : Techniques, applications and challenges". In : *Inf. Fusion* 76 (2021), p. 243-297.

[AMT22] Sameer Agarwal, Keir Mierle et The Ceres Solver Team. *Ceres Solver*. Version 2.1. 2022.

[Ahm+21] Adel Ahmadyan, Liangkai Zhang, Artsiom Ablavatski, Jianing Wei et Matthias Grundmann. "Objectron : A Large Scale Dataset of Object-Centric Videos in the Wild With Pose Annotations". In : *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. 2021, p. 7822-7831.

[Ara+16] Relja Arandjelovic, Petr Gronát, Akihiko Torii, Tomás Pajdla et Josef Sivic. "NetVLAD : CNN Architecture for Weakly Supervised Place Recognition". In : *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, p. 5297-5307.

[AZ14] Relja Arandjelović et Andrew Zisserman. "DisLocation : Scalable descriptor distinctiveness for location recognition". In : *Asian conference on computer vision*. Springer. 2014, p. 188-204.

[AZ12] Relja Arandjelović et Andrew Zisserman. "Three things everyone should know to improve object retrieval". In : *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, p. 2911-2918.

[ACB17] Martin Arjovsky, Soumith Chintala et Léon Bottou. "Wasserstein Generative Adversarial Networks". In : *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. T. 70. Proceedings of Machine Learning Research. 2017, p. 214-223.

[Avr03] M. Avriel. *Nonlinear Programming : Analysis and Methods*. 2003.

[BSCL14] Artem Babenko, Anton Slesarev, Alexander Chigorin et Victor S. Lempitsky. "Neural Codes for Image Retrieval". In : *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*. T. 8689. Lecture Notes in Computer Science. 2014, p. 584-599.

[BLP18] Vassileios Balntas, Shuda Li et Victor Prisacariu. "Relocnet : Continuous metric learning relocalisation using neural nets". In : *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, p. 751-767.

[BRPM16] Vassileios Balntas, Edgar Riba, Daniel Ponsa et Krystian Mikolajczyk. "Learning local feature descriptors with triplets and shallow convolutional neural networks". In : *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. 2016.

[BBCS12] Sid Ying-Ze Bao, Mohit Bagra, Yu-Wei Chao et Silvio Savarese. "Semantic structure from motion with points, regions, and objects". In : *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*. 2012, p. 2703-2710.

[BTG06] Herbert Bay, Tinne Tuytelaars et Luc Van Gool. "SURF : Speeded Up Robust Features". In : *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part I*. T. 3951. Lecture Notes in Computer Science. 2006, p. 404-417.

[Blo+18] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger et Andrew J. Davison. "CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM". In : *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, p. 2560-2568.

[BCKW15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu et Daan Wierstra. "Weight Uncertainty in Neural Networks". In : *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France, 2015, p. 1613-1622.

[BWL20] Alexey Bochkovskiy, Chien-Yao Wang et Hong-Yuan Mark Liao. "YOLOv4 : Optimal Speed and Accuracy of Object Detection". In : *CoRR* (2020).

[Bra+17] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold et Carsten Rother. "DSAC - Differentiable RANSAC for Camera Localization". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, p. 2492-2500.

[Bra+16] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold et Carsten Rother. "Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image". In : *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, p. 3364-3372.

[BR19] Eric Brachmann et Carsten Rother. "Expert Sample Consensus Applied to Camera Re-Localization". In : *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 2019, p. 7524-7533.

[BR18] Eric Brachmann et Carsten Rother. "Learning Less Is More - 6D Camera Localization via 3D Surface Regression". In : *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, p. 4654-4662.

[BR22] Eric Brachmann et Carsten Rother. "Visual Camera Re-Localization From RGB and RGB-D Images Using DSAC". In : *IEEE Trans. Pattern Anal. Mach. Intell.* 44.9 (2022), p. 5847-5865.

[Bra00] Gary Bradski. "The openCV library." In : *Dr. Dobb's Journal : Software Tools for the Professional Programmer* 25.11 (2000), p. 120-123.

[Bra+18] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays et Jan Kautz. "Geometry-Aware Learning of Maps for Camera Localization". In : *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, p. 2616-2625.

[BCL99] Mary Ann Branch, Thomas F Coleman et Yuying Li. "A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems". In : *SIAM Journal on Scientific Computing* 21.1 (1999), p. 1-23.

[BAIN18] Mai Bui, Shadi Albarqouni, Slobodan Ilic et Nassir Navab. "Scene Coordinate and Correspondence Learning for Image-Based Localization". In : *British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, September 3-6, 2018*. 2018, p. 3.

[Çal+15] Berk Çalli, Arjun Singh, Aaron Walsman, Siddhartha S. Srinivasa, Pieter Abbeel et Aaron M. Dollar. "The YCB object and Model set : Towards common benchmarks for manipulation research". In : *International Conference on Advanced Robotics, ICAR 2015, Istanbul, Turkey, July 27-31, 2015*. 2015, p. 510-517.

[CLSF10] Michael Calonder, Vincent Lepetit, Christoph Strecha et Pascal Fua. "BRIEF : Binary Robust Independent Elementary Features". In : *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV*. T. 6314. Lecture Notes in Computer Science. 2010, p. 778-792.

[Cam+21] Carlos Campos, Richard Elvira, Juan J. Gómez Rodriguez, José M. M. Montiel et Juan D. Tardós. "ORB-SLAM3 : An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM". In : *IEEE Trans. Robotics* 37.6 (2021), p. 1874-1890.

[CCPS18] Federico Camposeco, Andrea Cohen, Marc Pollefeys et Torsten Sattler. "Hybrid camera pose estimation". In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, p. 136-144.

[CS13] Song Cao et Noah Snavely. "Graph-based discriminative learning for location recognition". In : *Proceedings of the ieee conference on computer vision and pattern recognition*. 2013, p. 700-707.

[Cav+19] Tommaso Cavallari, Luca Bertinetto, Jishnu Mukhoti, Philip H. S. Torr et Stuart Golodetz. "Let's Take This Online : Adapting Scene Coordinate Regression Network Predictions for Online RGB-D Camera Relocalisation". In : *2019 International Conference on 3D Vision, 3DV 2019, Québec City, QC, Canada, September 16-19, 2019*. 2019, p. 564-573.

[Cav+17] Tommaso Cavallari, Stuart Golodetz, Nicholas A. Lord, Julien P. C. Valentin, Luigi di Stefano et Philip H. S. Torr. "On-the-Fly Adaptation of Regression Forests for Online Camera Relocalisation". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, p. 218-227.

[CPMC06] Denis Chekhlov, Mark Pupilli, Walterio W. Mayol-Cuevas et Andrew Calway. "Real-Time and Robust Monocular SLAM Using Predictive Multi-resolution Descriptors". In : *Advances in Visual Computing, Second International Symposium, ISVC 2006 Lake Tahoe, NV, USA, November 6-8, 2006. Proceedings, Part II*. T. 4292. Lecture Notes in Computer Science. 2006, p. 276-285.

[Che+21] Shujia Chen, Shuangfu Song, Junqiao Zhao, Tiantian Feng, Chen Ye, Lu Xiong et Deyi Li. "Robust Dual Quadric Initialization for Forward-Translating Camera Movements". In : *IEEE Robotics Autom. Lett.* 6.3 (2021), p. 4712-4719.

[Che+17] Zetao Chen, Adam Jacobson, Niko Sünderhauf, Ben Upcroft, Lingqiao Liu, Chunhua Shen, Ian Reid et Michael Milford. "Deep learning features at scale for visual place recognition". In : *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, p. 3223-3230.

[Chi+20] Yu-Chen Chiu, Chi-Yi Tsai, Mind-Da Ruan, Guan-Yu Shen et Tsu-Tian Lee. "Mobilenet-SSDv2 : An Improved Object Detection Model for Embedded Systems". In : *International Conference on System Science and Engineering, ICSSE 2020, Kagawa, Japan, August 31 - September 3, 2020*. 2020, p. 1-5.

[CDM08] Javier Civera, Andrew J. Davison et J. M. M. Montiel. "Inverse Depth Parametrization for Monocular SLAM". In : *IEEE Trans. Robotics* 24.5 (2008), p. 932-945.

[Cla+17] Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni et Hongkai Wen. "VidLoc : A Deep Spatio-Temporal Model for 6-DoF Video-Clip Relocalization". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, p. 2652-2660.

[Com18] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018.

[CC15] Alejo Concha et Javier Civera. "DPPTAM : Dense piecewise planar tracking and mapping from a monocular sequence". In : *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*. 2015, p. 5686-5693.

[Cor+16] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In : *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, p. 3213-3223.

[Cri+15] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua et Vincent Lepetit. "A novel representation of parts for accurate 3D object detection and tracking in monocular images". In : *Proceedings of the IEEE international conference on computer vision*. 2015, p. 4391-4399.

[Cri+17] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua et Vincent Lepetit. "Robust 3D object tracking from monocular images using stable parts". In : *IEEE transactions on pattern analysis and machine intelligence* 40.6 (2017), p. 1465-1479.

[CRB16] Marco Crocco, Cosimo Rubino et Alessio Del Bue. "Structure from Motion with Objects". In : *CVPR*. 2016.

[CN10] Mark Joseph Cummins et Paul M. Newman. "FAB-MAP : Appearance-Based Place Recognition and Mapping using a Learned Visual Vocabulary Model". In : *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. 2010, p. 3-10.

[CN08] Mark Joseph Cummins et Paul M. Newman. "FAB-MAP : Probabilistic Localization and Mapping in the Space of Appearance". In : *Int. J. Robotics Res.* 27.6 (2008), p. 647-665.

[CLCD20] Jan Czarnowski, Tristan Laidlow, Ronald Clark et Andrew J. Davison. "DeepFactors : Real-Time Probabilistic Dense Monocular SLAM". In : *IEEE Robotics Autom. Lett.* 5.2 (2020), p. 721-728.

[DRMS07] Andrew J. Davison, Ian D. Reid, Nicholas Molton et Olivier Stasse. "MonoSLAM : Real-Time Single Camera SLAM". In : *IEEE Trans. Pattern Anal. Mach. Intell.* 29.6 (2007), p. 1052-1067.

[DGJP13] Jonathan Delhumeau, Philippe Henri Gosselin, Hervé Jégou et Patrick Pérez. "Revisiting the VLAD image representation". In : *ACM Multimedia Conference, MM '13, Barcelona, Spain, October 21-25, 2013*. 2013, p. 653-656.

[Del+22] Frank Dellaert et al. *borglab/gtsam*. Version 4.2a7. 2022.

[Den+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li et Li Fei-Fei. "Imagenet : A large-scale hierarchical image database". In : *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, p. 248-255.

[DMR18] Daniel DeTone, Tomasz Malisiewicz et Andrew Rabinovich. "SuperPoint : Self-Supervised Interest Point Detection and Description". In : *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, p. 224-236.

[Din+19] Mingyu Ding, Zhe Wang, Jiankai Sun, Jianping Shi et Ping Luo. "CamNet : Coarse-to-fine retrieval for camera re-localization". In : *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, p. 2871-2880.

[DS15] Jingming Dong et Stefano Soatto. "Domain-size pooling in local descriptors : DSP-SIFT". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, p. 5097-5106.

[DI21] Wenbo Dong et Volkan Isler. "Ellipse Regression with Predicted Uncertainties for Accurate Multi-View 3D Object Estimation". In : *CoRR* abs/2101.05212 (2021).

[DRPI21] Wenbo Dong, Pravakar Roy, Cheng Peng et Volkan Isler. "Ellipse R-CNN : Learning to Infer Elliptical Object From Clustering and Occlusion". In : *IEEE Trans. Image Process.* 30 (2021), p. 2193-2206.

[Dus+19] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii et Torsten Sattler. "D2-net : A trainable cnn for joint description and detection of local features". In : *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*. 2019, p. 8092-8101.

[ED06] Ethan Eade et Tom Drummond. "Scalable Monocular SLAM". In : *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*. 2006, p. 469-476.

[ED08] Ethan Eade et Tom Drummond. "Unified Loop Closing and Recovery for Real Time Monocular SLAM". In : *Proceedings of the British Machine Vision Conference 2008, Leeds, UK, September 2008*. 2008, p. 1-10.

[EKC18] J. Engel, V. Koltun et D. Cremers. "Direct Sparse Odometry". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

[ESC14] Jakob Engel, Thomas Schöps et Daniel Cremers. "LSD-SLAM : Large-Scale Direct Monocular SLAM". In : *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II*. T. 8690. Lecture Notes in Computer Science. 2014, p. 834-849.

[FRD18] Di Feng, Lars Rosenbaum et Klaus Dietmayer. "Towards Safe Autonomous Driving : Capture Uncertainty in the Deep Neural Network For Lidar 3D Vehicle Detection". In : *21st International Conference on Intelligent Transportation Systems, ITSC 2018, Maui, HI, USA, November 4-7, 2018*. 2018, p. 3266-3273.

[FFW15] Youji Feng, Lixin Fan et Yihong Wu. "Fast localization in large-scale environments using supervised indexing of binary features". In : *IEEE Transactions on Image Processing* 25.1 (2015), p. 343-358.

[FB81] Martin A Fischler et Robert C Bolles. "Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography". In : *Communications of the ACM* 24.6 (1981), p. 381-395.

[FOBD18] Avelino Forechi, Thiago Oliveira-Santos, Claudine Badue et Alberto Ferreira De Souza. "Visual global localization with a hybrid WNN-CNN approach". In : *2018 international joint conference on neural networks (IJCNN)*. IEEE. 2018, p. 1-9.

[FPS14] Christian Forster, Matia Pizzoli et Davide Scaramuzza. "SVO : Fast semi-direct monocular visual odometry". In : *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*. 2014, p. 15-22.

[FPM18] Duncan P. Frost, Victor Adrian Prisacariu et David William Murray. "Recovering Stable Scale in Monocular SLAM Using Object-Supplemented Bundle Adjustment". In : *IEEE Trans. Robotics* 34.3 (2018), p. 736-747.

[GG16] Yarin Gal et Zoubin Ghahramani. "Dropout as a Bayesian Approximation : Representing Model Uncertainty in Deep Learning". In : *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. T. 48. JMLR Workshop and Conference Proceedings. 2016, p. 1050-1059.

[GT12a] Dorian Gálvez-López et J. D. Tardós. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In : *IEEE Transactions on Robotics* 28.5 (2012), p. 1188-1197.

[GT12b] Dorian Gálvez-López et Juan D. Tardós. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In : *IEEE Trans. Robotics* 28.5 (2012), p. 1188-1197.

[GWDC18] Xiang Gao, Rui Wang, Nikolaus Demmel et Daniel Cremers. "LDSO : Direct Sparse Odometry with Loop Closure". In : *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*. 2018, p. 2198-2204.

[GHTC03] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang et Hang-Fei Cheng. "Complete solution classification for the perspective-three-point problem". In : *IEEE transactions on pattern analysis and machine intelligence* 25.8 (2003), p. 930-943.

[GSB19a] Vincent Gaudillière, Gilles Simon et Marie-Odile Berger. "Camera Pose Estimation with Semantic 3D Model". In : *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019*. 2019, p. 4569-4576.

[GSB19b] Vincent Gaudillière, Gilles Simon et Marie-Odile Berger. "Camera Relocalization with Ellipsoidal Abstraction of Objects". In : *2019 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2019, Beijing, China, October 14-18, 2019*. 2019, p. 8-18.

[GSB20] Vincent Gaudillière, Gilles Simon et Marie-Odile Berger. "Perspective-2-Ellipsoid : Bridging the Gap Between Object Detections and 6-DoF Camera Pose". In : *IEEE Robotics Automation Letters* (2020).

[Gau09] Carl Friedrich Gauss. "Theoria motus corporum coelestum". In : *Werke* (1809).

[GRBD17] Paul Gay, Cosimo Rubino, Vaibhav Bansal et Alessio Del Bue. "Probabilistic structure from motion with objects (psfmo)". In : *Proceedings of the IEEE International Conference on Computer Vision*. 2017, p. 3075-3084.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller et Raquel Urtasun. "Vision meets robotics : The KITTI dataset". In : *Int. J. Robotics Res.* 32.11 (2013), p. 1231-1237.

[GLB21] Hugo Germain, Vincent Lepetit et Guillaume Bourmaud. "Neural reprojection error : Merging feature learning and camera pose estimation". In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, p. 414-423.

[GWZC21] Mariia Gladkova, Rui Wang, Niclas Zeller et Daniel Cremers. "Tight Integration of Feature-based Relocalization in Monocular Direct Visual Odometry". In : *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*. 2021, p. 9608-9614.

[GISC13] Ben Glocker, Shahram Izadi, Jamie Shotton et Antonio Criminisi. "Real-time RGB-D camera relocalization". In : *IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013, Adelaide, Australia, October 1-4, 2013*. 2013, p. 173-179.

[Gom+19] Ruben Gomez-Ojeda, Francisco Angel Moreno, David Zuñiga-Noël, Davide Scaramuzza et Javier González Jiménez. "PL-SLAM : A Stereo SLAM System Through the Combination of Points and Line Segments". In : *IEEE Trans. Robotics* 35.3 (2019), p. 734-746.

[GDS20] Fredrik K. Gustafsson, Martin Danelljan et Thomas B. Schön. "Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision". In : *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*. 2020, p. 1289-1298.

[Hal+20] David Hall et al. "Probabilistic Object Detection : Definition and Evaluation". In : *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*. 2020, p. 1020-1029.

[HS88] Christopher G. Harris et Mike Stephens. "A Combined Corner and Edge Detector". In : *Proceedings of the Alvey Vision Conference, AVC 1988, Manchester, UK, September, 1988*. 1988, p. 1-6.

[HZ03] Richard Hartley et Andrew Zisserman. *Multiple view geometry in computer vision*. 2003.

[Hau+21] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford et Tobias Fischer. "Patch-netvlad : Multi-scale fusion of locally-global descriptors for place recognition". In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, p. 14141-14152.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár et Ross B. Girshick. "Mask R-CNN". In : *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, p. 2980-2988.

[Hod+17] Tomas Hodan, Pavel Haluza, Stepán Obdrzálek, Jiri Matas, Manolis I. A. Lourakis et Xenophon Zabulis. "T-LESS : An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects". In : *2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, Santa Rosa, CA, USA, March 24-31, 2017*. 2017, p. 880-888.

[HA15] Elad Hoffer et Nir Ailon. "Deep metric learning using Triplet network". In : *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. 2015.

[Hos+18] Mehdi Hosseinzadeh, Yasir Latif, Trung Pham, Niko Sünderhauf et Ian D. Reid. "Structure Aware SLAM Using Quadrics and Planes". In : *Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, Perth, Australia, December 2-6, 2018, Revised Selected Papers, Part III*. T. 11363. Lecture Notes in Computer Science. 2018, p. 410-426.

[HLLR19] Mehdi Hosseinzadeh, Kejie Li, Yasir Latif et Ian D. Reid. "Real-Time Monocular Object-Model Aware Sparse SLAM". In : *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. 2019, p. 7123-7129.

[Hou59] Paul VC Hough. "Machine analysis of bubble chamber pictures". In : *Proc. of the International Conference on High Energy Accelerators and Instrumentation, Sept. 1959*. 1959, p. 554-556.

[HXHK19] Lan Hu, Wanting Xu, Kun Huang et Laurent Kneip. "Deep-SLAM++ : Object-level RGBD SLAM based on class-specific deep shape priors". In : *CoRR* abs/1907.09691 (2019).

[HHFS19] Yinlin Hu, Joachim Hugonot, Pascal Fua et Mathieu Salzmann. "Segmentation-driven 6d object pose estimation". In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, p. 3385-3394.

[HPC18] Rachel Huang, Jonathan Pedoeem et Cuixian Chen. "YOLO-LITE : A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers". In : *IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10-13, 2018*. 2018, p. 2503-2510.

[IZFB09] Arnold Irschara, Christopher Zach, Jan-Michael Frahm et Horst Bischof. "From structure-from-motion point clouds to fast location recognition". In : *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, p. 2599-2606.

[JIP19] Krishna Murthy Jatavallabhula, Ganesh Iyer et Liam Paull. "gradSLAM : Dense SLAM meets Automatic Differentiation". In : *CoRR* abs/1910.10672 (2019).

[JDSP10] Hervé Jégou, Matthijs Douze, Cordelia Schmid et Patrick Pérez. "Aggregating local descriptors into a compact image representation". In : *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*. 2010, p. 3304-3311.

[JDF17] Hyo Jin Kim, Enrique Dunn et Jan-Michael Frahm. "Learned contextual feature reweighting for image geo-localization". In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, p. 2136-2145.

[KRD08] Michael Kaess, Ananth Ranganathan et Frank Dellaert. "iSAM : Incremental Smoothing and Mapping". In : *IEEE Trans. Robotics* 24.6 (2008), p. 1365-1378.

[Keh+17] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic et Nassir Navab. "Ssd-6d : Making rgb-based 3d detection and 6d pose estimation great again". In : *Proceedings of the IEEE international conference on computer vision*. 2017, p. 1521-1529.

[KBC17] Alex Kendall, Vijay Badrinarayanan et Roberto Cipolla. "Bayesian SegNet : Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding". In : *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*. 2017.

[KC17] Alex Kendall et Roberto Cipolla. "Geometric Loss Functions for Camera Pose Regression with Deep Learning". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, p. 6555-6564.

[KC16] Alex Kendall et Roberto Cipolla. "Modelling uncertainty in deep learning for camera relocalization". In : *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*. 2016, p. 4762-4769.

[KG17] Alex Kendall et Yarin Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision ?" In : *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, p. 5574-5584.

[KGC15] Alex Kendall, Matthew Grimes et Roberto Cipolla. "PoseNet : A Convolutional Network for Real-Time 6-DOF Camera Relocalization". In : *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 2015, p. 2938-2946.

[KB15] Diederik P. Kingma et Jimmy Ba. "Adam : A Method for Stochastic Optimization". In : *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[KM07] Georg Klein et David William Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In : *Sixth IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR 2007, 13-16 November 2007, Nara, Japan*. 2007, p. 225-234.

[KSS11] Laurent Kneip, Davide Scaramuzza et Roland Siegwart. "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation". In : *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*. 2011, p. 2969-2976.

[KD19] Florian Kraus et Klaus Dietmayer. "Uncertainty Estimation in One-Stage Object Detection". In : *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019, Auckland, New Zealand, October 27-30, 2019*. 2019, p. 53-60.

[Kri+21] Matej Kristan et al. "The Ninth Visual Object Tracking VOT2021 Challenge Results". In : *IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, BC, Canada, October 11-17, 2021*. 2021, p. 2711-2738.

[Kuh10] Harold W. Kuhn. "The Hungarian Method for the Assignment Problem". In : *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*. 2010, p. 29-47.

[Küm+11] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige et Wolfram Burgard. "G^2o : A general framework for graph optimization". In : *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*. 2011, p. 3607-3613.

[LCAS20] Yann Labbé, Justin Carpentier, Mathieu Aubry et Josef Sivic. "Cosypose : Consistent multi-view multi-object 6d pose estimation". In : *European Conference on Computer Vision*. Springer. 2020, p. 574-591.

[LPB17] Balaji Lakshminarayanan, Alexander Pritzel et Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In : *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, p. 6402-6413.

[LMKK17] Zakaria Laskar, Iaroslav Melekhov, Surya Kalia et Juho Kannala. "Camera relocalization by computing pairwise relative poses using convolutional neural network". In : *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, p. 929-938.

[LF+05] Vincent Lepetit, Pascal Fua et al. "Monocular model-based 3d tracking of rigid objects : A survey". In : *Foundations and Trends® in Computer Graphics and Vision* 1.1 (2005), p. 1-89.

[LMF09] Vincent Lepetit, Francesc Moreno-Noguer et Pascal Fua. "EPnP : An Accurate $O(n)$ Solution to the PnP Problem". In : *Int. J. Comput. Vis.* 81.2 (2009), p. 155-166.

[LMD19] Jimmy Li, David Meger et Gregory Dudek. "Semantic Mapping for View-Invariant Relocalization". In : *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. 2019, p. 7108-7115.

[LWLG18] Ruihao Li, Sen Wang, Zhiqiang Long et Dongbing Gu. "UnDeepVO : Monocular Visual Odometry Through Unsupervised Deep Learning". In : *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. 2018, p. 7286-7291.

[LSHF12] Yunpeng Li, Noah Snavely, Dan Huttenlocher et Pascal Fua. "Worldwide pose estimation using 3d point clouds". In : *European conference on computer vision*. Springer. 2012, p. 15-29.

[LSH10] Yunpeng Li, Noah Snavely et Daniel P Huttenlocher. "Location recognition using prioritized feature matching". In : *European conference on computer vision*. Springer. 2010, p. 791-804.

[Lia+22] Ziwei Liao, Yutong Hu, Jiadong Zhang, Xianyu Qi, Xiaoyu Zhang et Wei Wang. "SO-SLAM : Semantic Object SLAM With Scale Proportional and Symmetrical Texture Constraints". In : *IEEE Robotics Autom. Lett.* 7.2 (2022), p. 4008-4015.

[Lin+14a] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár et C. Lawrence Zitnick. "Microsoft COCO : Common Objects in Context". In : *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*. T. 8693. Lecture Notes in Computer Science. 2014, p. 740-755.

[Lin+14b] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár et C. Lawrence Zitnick. "Microsoft COCO : Common Objects in Context". In : *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*. T. 8693. Lecture Notes in Computer Science. 2014, p. 740-755.

[Liu+19] Hong Liu, Rongrong Ji, Jie Li, Baochang Zhang, Yue Gao, Yongjian Wu et Feiyue Huang. "Universal Adversarial Perturbation via Prior Driven Uncertainty Approximation". In : *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 2019, p. 2941-2949.

[LLD17] Liu Liu, Hongdong Li et Yuchao Dai. "Efficient global 2d-3d matching for camera localization in a large-scale 3d map". In : *Proceedings of the IEEE International Conference on Computer Vision*. 2017, p. 2372-2381.

[Liu+16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu et Alexander C. Berg. "SSD : Single Shot MultiBox Detector". In : *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*. T. 9905. Lecture Notes in Computer Science. 2016, p. 21-37.

[LSS20] Antonio Loquercio, Mattia Segù et Davide Scaramuzza. "A General Framework for Uncertainty Estimation in Deep Learning". In : *IEEE Robotics Autom. Lett.* 5.2 (2020), p. 3153-3160.

[Low04] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In : *Int. J. Comput. Vis.* 60.2 (2004), p. 91-110.

[Low+15] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke et Michael J Milford. "Visual place recognition : A survey". In : *IEEE Transactions on Robotics* 32.1 (2015), p. 1-19.

[Lu+15] Guoyu Lu, Yan Yan, Li Ren, Jingkuan Song, Nicu Sebe et Chandra Kambhamettu. "Localize me anywhere, anytime : a multi-task point-retrieval approach". In : *Proceedings of the IEEE International Conference on Computer Vision*. 2015, p. 2434-2442.

[Luo+18] Zixin Luo, Tianwei Shen, Lei Zhou, Siyu Zhu, Runze Zhang, Yao Yao, Tian Fang et Long Quan. "GeoDesc : Learning Local Descriptors by Integrating Geometry Constraints". In : *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IX*. T. 11213. Lecture Notes in Computer Science. 2018, p. 170-185.

[Lyn+15] Simon Lynen, Torsten Sattler, Michael Bosse, Joel A Hesch, Marc Pollefeys et Roland Siegwart. "Get out of my lab : Large-scale, real-time visual-inertial localization." In : *Robotics : Science and Systems*. T. 1. 2015, p. 1.

[MSBI22] Nithid Mahattansin, Kanjanapan Sukvichai, Pished Bunnun et Tsuyoshi Isshiki. "Improving Relocalization in Visual SLAM by using Object Detection". In : *2022 19th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. 2022, p. 1-4.

[MUS15] Eric Marchand, Hideaki Uchiyama et Fabien Spindler. "Pose estimation for augmented reality : a hands-on survey". In : *IEEE transactions on visualization and computer graphics* 22.12 (2015), p. 2633-2651.

[Mar63] Donald W Marquardt. "An algorithm for least-squares estimation of nonlinear parameters". In : *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), p. 431-441.

[MCUP02] Jiri Matas, Ondrej Chum, Martin Urban et Tomás Pajdla. "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions". In : *Proceedings of the British Machine Vision Conference 2002, BMVC 2002, Cardiff, UK, 2-5 September 2002*. 2002, p. 1-10.

[MHDL17] John McCormac, Ankur Handa, Andrew J. Davison et Stefan Leutenegger. "SemanticFusion : Dense 3D semantic mapping with convolutional neural networks". In : *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*. 2017, p. 4628-4635.

[MHA17] Leland McInnes, John Healy et Steve Astels. "hdbscan : Hierarchical density based clustering". In : *J. Open Source Softw.* 2.11 (2017), p. 205.

[MSUK14] Sven Middelberg, Torsten Sattler, Ole Untzelmann et Leif Kobbelt. "Scalable 6-dof localization on mobile devices". In : *European conference on computer vision*. Springer. 2014, p. 268-283.

[MS04] Krystian Mikolajczyk et Cordelia Schmid. "Scale & Affine Invariant Interest Point Detectors". In : *Int. J. Comput. Vis.* 60.1 (2004), p. 63-86.

[MDMS19] Dimity Miller, Feras Dayoub, Michael Milford et Niko Sünderhauf. "Evaluating Merging Strategies for Sampling-based Uncertainty Techniques in Object Detection". In : *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. 2019, p. 2348-2354.

[MNDS18] Dimity Miller, Lachlan Nicholson, Feras Dayoub et Niko Sünderhauf. "Dropout Sampling for Robust Object Detection in Open-Set Conditions". In : *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. 2018, p. 1-7.

[MMRM17] Anastasya Mishchuk, Dmytro Mishkin, Filip Radenovic et Jiri Matas. "Working hard to know your neighbor's margins : Local descriptor learning loss". In : *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, p. 4826-4837.

[Mor80] Hans P. Moravec. "Obstacle avoidance and navigation in the real world by a seeing robot rover". Thèse de doct. Stanford University, USA, 1980.

[Mor19] Doug Morrison. "Uncertainty-aware Instance Segmentation using Dropout Sampling". In : 2019.

[MAFK17] Arsalan Mousavian, Dragomir Anguelov, John Flynn et Jana Kosecka. "3D Bounding Box Estimation Using Deep Learning and Geometry". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, p. 5632-5640.

[MMT15] Raul Mur-Artal, J. M. M. Montiel et Juan D. Tardós. "ORB-SLAM : A Versatile and Accurate Monocular SLAM System". In : *IEEE Trans. Robotics* 31.5 (2015), p. 1147-1163.

[MT14] Raul Mur-Artal et Juan D. Tardós. "Fast relocalisation and loop closing in keyframe-based SLAM". In : *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*. 2014, p. 846-853.

[MT17] Raul Mur-Artal et Juan D. Tardós. "ORB-SLAM2 : An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In : *IEEE Trans. Robotics* 33.5 (2017), p. 1255-1262.

[NB17] Tayyab Naseer et Wolfram Burgard. "Deep regression for monocular camera-based 6-DoF global localization in outdoor environments". In : *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*. 2017, p. 1525-1530.

[NLD11] Richard A. Newcombe, Steven Lovegrove et Andrew J. Davison. "DTAM : Dense tracking and mapping in real-time". In : *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. 2011, p. 2320-2327.

[New+11] Richard A. Newcombe et al. "KinectFusion : Real-time dense surface mapping and tracking". In : *10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011, Basel, Switzerland, October 26-29, 2011*. 2011, p. 127-136.

[NMS19] Lachlan Nicholson, Michael Milford et Niko Sünderhauf. "QuadricSLAM : Dual Quadrics From Object Detections as Landmarks in Object-Oriented SLAM". In : *IEEE Robotics Autom. Lett.* 4.1 (2019), p. 1-8.

[Nis03] David Nistér. "An Efficient Solution to the Five-Point Relative Pose Problem". In : *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), 16-22 June 2003, Madison, WI, USA*. 2003, p. 195-202.

[NS06] David Nistér et Henrik Stewénius. "Scalable Recognition with a Vocabulary Tree". In : *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*. 2006, p. 2161-2168.

[ORL18] Markus Oberweger, Mahdi Rad et Vincent Lepetit. "Making deep heatmaps robust to partial occlusions for 3d object pose estimation". In : *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, p. 119-134.

[Ok+19] Kyel Ok, Katherine Liu, Kristoffer M. Frey, Jonathan P. How et Nicholas Roy. "Robust Object-based SLAM for High-speed Autonomous Navigation". In : *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. 2019, p. 669-675.

[OT01] Aude Oliva et Antonio Torralba. "Modeling the Shape of the Scene : A Holistic Representation of the Spatial Envelope". In : *Int. J. Comput. Vis.* 42.3 (2001), p. 145-175.

[OTFY18] Yuki Ono, Eduard Trulls, Pascal Fua et Kwang Moo Yi. "LF-Net : Learning Local Features from Images". In : *Advances in Neural Information Processing Systems 31 : Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. 2018, p. 6237-6247.

[Pan+21] Shenyi Pan, Shuxian Fan, Samuel W. K. Wong, James V. Zidek et Helge Rhodin. "Ellipse Detection and Localization with Applications to Knots in Sawn Lumber Images". In : *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*. 2021, p. 3891-3900.

[Par+19] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe et Steven Lovegrove. "DeepSDF : Learning Continuous Signed Distance Functions for Shape Representation". In : *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, p. 165-174.

[PPV19] Kiru Park, Timothy Patten et Markus Vincze. "Pix2Pose : Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation". In : *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 2019, p. 7667-7676.

[PUG19] Despoina Paschalidou, Ali Osman Ulusoy et Andreas Geiger. "Superquadrics revisited : Learning 3d shape parsing beyond cuboids". In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, p. 10344-10353.

[Pav+17] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis et Kostas Daniilidis. "6-dof object pose from semantic keypoints". In : *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, p. 2011-2018.

[Pen+19] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou et Hujun Bao. "Pvnet : Pixel-wise voting network for 6dof pose estimation". In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, p. 4561-4570.

[PLSP10] Florent Perronnin, Yan Liu, Jorge Sánchez et Hervé Poirier. "Large-scale image retrieval with compressed Fisher vectors". In : *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*. 2010, p. 3384-3391.

[Pha+18] Buu Phan, Rick Salay, Krzysztof Czarnecki, Vahdat Abdelzad, Taylor Denouden et Sachin Vernekar. "Calibrating Uncertainties in Object Localization Task". In : *CoRR* abs/1811.11210 (2018).

[Phi+07] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic et Andrew Zisserman. "Object retrieval with large vocabularies and fast spatial matching". In : *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*. 2007.

[PSDG19] Nathan Piasco, Désiré Sidibé, Cédric Demonceaux et Valérie Gouet-Brunet. "Perspective-n-learned-point : Pose estimation from relative depth". In : *British machine vision conference (BMVC)*. 2019.

[PRB11] Katrin Pirker, Matthias Rüther et Horst Bischof. "CD SLAM - Continuous localization and mapping in a dynamic world". In : *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*. 2011, p. 3990-3997.

[PIL19] Giorgia Pitteri, Slobodan Ilic et Vincent Lepetit. "CorNet : generic 3D corners for 6D pose estimation of new objects without retraining". In : *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019.

[QSHP16] Xiaozhi Qu, Bahman Soheilian, Emmanuel Habets et Nicolas Paparoditis. "EVALUATION OF SIFT AND SURF FOR VISION BASED LOCALIZATION". In : *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLI-B3 (2016), p. 685-692.

[RL17] Mahdi Rad et Vincent Lepetit. "BB8 : A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth". In : *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, p. 3848-3856.

[RTC16] Filip Radenovic, Giorgos Tolias et Ondrej Chum. "CNN Image Retrieval Learns from BoW : Unsupervised Fine-Tuning with Hard Examples". In : *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*. T. 9905. Lecture Notes in Computer Science. 2016, p. 3-20.

[RVB18] Noha Radwan, Abhinav Valada et Wolfram Burgard. "VLocNet++ : Deep Multitask Learning for Semantic Visual Localization and Odometry". In : *IEEE Robotics Autom. Lett.* 3.4 (2018), p. 4407-4414.

[RDGF16] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick et Ali Farhadi. "You Only Look Once : Unified, Real-Time Object Detection". In : *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, p. 779-788.

[RF17] Joseph Redmon et Ali Farhadi. "YOLO9000 : better, faster, stronger". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 7263-7271.

[RF18] Joseph Redmon et Ali Farhadi. "Yolov3 : An incremental improvement". In : *arXiv preprint arXiv :1804.02767* (2018).

[RHGS17] Shaoqing Ren, Kaiming He, Ross B. Girshick et Jian Sun. "Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks". In : *IEEE Trans. Pattern Anal. Mach. Intell.* (2017).

[RDHW19] Jerome Revaud, Cesar De Souza, Martin Humenberger et Philippe Weinzaepfel. "R2d2 : Reliable and repeatable detector and descriptor". In : *Advances in neural information processing systems* 32 (2019).

[Rez+19] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid et Silvio Savarese. "Generalized Intersection Over Union : A Metric and a Loss for Bounding Box Regression". In : *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, p. 658-666.

[RFB15] Olaf Ronneberger, Philipp Fischer et Thomas Brox. "U-Net : Convolutional Networks for Biomedical Image Segmentation". In : *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*. T. 9351. Lecture Notes in Computer Science. 2015, p. 234-241.

[RBCS06] Bodo Rosenhahn, Thomas Brox, Daniel Cremers et Hans-Peter Seidel. "A Comparison of Shape Matching Methods for Contour Based Pose Estimation". In : *Combinatorial Image Analysis, 11th International Workshop, IWCIA 2006, Berlin, Germany, June 19-21, 2006, Proceedings*. T. 4040. Lecture Notes in Computer Science. 2006, p. 263-276.

[RD06] Edward Rosten et Tom Drummond. "Machine Learning for High-Speed Corner Detection". In : *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part I*. T. 3951. Lecture Notes in Computer Science. 2006, p. 430-443.

[RCB18] Cosimo Rubino, Marco Crocco et Alessio Del Bue. "3D Object Localisation from Multi-View Image Detections". In : *IEEE TPAMI* (2018).

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige et Gary R. Bradski. "ORB : An efficient alternative to SIFT or SURF". In : *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. 2011, p. 2564-2571.

[Rün+20] Martin Rünz et al. "FroDO : From Detections to 3D Objects". In : *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. 2020, p. 14708-14717.

[Sal+13] Renato F. Salas-Moreno, Richard A. Newcombe, Hauke Strasdat, Paul H. J. Kelly et Andrew J. Davison. "SLAM++ : Simultaneous Localisation and Mapping at the Level of Objects". In : *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*. 2013, p. 1352-1359.

[SDMR20] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz et Andrew Rabinovich. "SuperGlue : Learning Feature Matching With Graph Neural Networks". In : *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. 2020, p. 4937-4946.

[Sar+21] Paul-Edouard Sarlin et al. "Back to the Feature : Learning Robust Camera Localization From Pixels To Pose". In : *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. 2021, p. 3247-3257.

[Sat+15] Torsten Sattler, Michal Havlena, Filip Radenovic, Konrad Schindler et Marc Pollefeys. "Hyperpoints and fine vocabularies for large-scale location recognition". In : *Proceedings of the IEEE International Conference on Computer Vision*. 2015, p. 2102-2110.

[SHSP16] Torsten Sattler, Michal Havlena, Konrad Schindler et Marc Pollefeys. "Large-scale location recognition and the geometric burstiness problem". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 1582-1590.

[SLK11] Torsten Sattler, Bastian Leibe et Leif Kobbelt. "Fast image-based localization using direct 2d-to-3d matching". In : *2011 International Conference on Computer Vision*. IEEE. 2011, p. 667-674.

[SLK12] Torsten Sattler, Bastian Leibe et Leif Kobbelt. "Improving image-based localization by active correspondence search". In : *European conference on computer vision*. Springer. 2012, p. 752-765.

[Sat+17] Torsten Sattler, Akihiko Torii, Josef Sivic, Marc Pollefeys, Hajime Taira, Masatoshi Okutomi et Tomas Pajdla. "Are large-scale 3d models really necessary for accurate visual localization ?" In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, p. 1637-1646.

[SWLK12] Torsten Sattler, Tobias Weyand, Bastian Leibe et Leif Kobbelt. "Image Retrieval for Image-Based Localization Revisited." In : *BMVC*. T. 1. 2. 2012, p. 4.

[SZPL19] Torsten Sattler, Qunjie Zhou, Marc Pollefeys et Laura Leal-Taixé. "Understanding the Limitations of CNN-Based Absolute Camera Pose Regression". In : *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, p. 3302-3312.

[Sav+17] Nikolay Savinov, Akihito Seki, Lubor Ladicky, Torsten Sattler et Marc Pollefeys. "Quad-Networks : Unsupervised Learning to Rank for Interest Point Detection". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, p. 3929-3937.

[SHSP17] Johannes L Schonberger, Hans Hardmeier, Torsten Sattler et Marc Pollefeys. "Comparative evaluation of hand-crafted and learned local features". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 1482-1491.

[SF16] Johannes Lutz Schönberger et Jan-Michael Frahm. "Structure-from-Motion Revisited". In : *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[SZPF16] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys et Jan-Michael Frahm. "Pixelwise View Selection for Unstructured Multi-View Stereo". In : *European Conference on Computer Vision (ECCV)*. 2016.

[SMLK06] W. Schroeder, K. Martin, B. Lorensen et Inc Kitware. *The Visualization Toolkit : An Object-oriented Approach to 3D Graphics*. 2006.

[Sho+13] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi et Andrew W. Fitzgibbon. "Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images". In : *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*. 2013, p. 2930-2937.

[Sim+15] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua et Francesc Moreno-Noguer. "Discriminative Learning of Deep Convolutional Feature Point Descriptors". In : *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 2015, p. 118-126.

[SFB18] Gilles Simon, Antoine Fond et Marie-Odile Berger. "A-Contrario Horizon-First Vanishing Point Detection Using Second-Order Grouping Laws". In : *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X*. T. 11214. Lecture Notes in Computer Science. 2018, p. 323-338.

[SVZ14] Karen Simonyan, Andrea Vedaldi et Andrew Zisserman. "Learning Local Feature Descriptors Using Convex Optimisation". In : *IEEE Trans. Pattern Anal. Mach. Intell.* 36.8 (2014), p. 1573-1585.

[SZ15] Karen Simonyan et Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In : *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[Ski+19] John Skinner, David Hall, Haoyang Zhang, Feras Dayoub et Niko Sünderhauf. "The Probabilistic Object Detection Challenge". In : *CoRR* abs/1903.07840 (2019).

[SSS08] Noah Snavely, Steven M Seitz et Richard Szeliski. "Modeling the world from internet photo collections". In : *International journal of computer vision* 80.2 (2008), p. 189-210.

[SDMK11] Hauke Strasdat, Andrew J. Davison, J. M. M. Montiel et Kurt Konolige. "Double window optimisation for constant time visual SLAM". In : *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. 2011, p. 2352-2359.

[SMD10] Hauke Strasdat, J. M. M. Montiel et Andrew J. Davison. "Real-time monocular SLAM : Why filter ?" In : *IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 3-7 May 2010*. 2010, p. 2657-2664.

[SWYC20] Lukas von Stumberg, Patrick Wenzel, Nan Yang et Daniel Cremers. "LM-Reloc : Levenberg-Marquardt Based Direct Visual Relocalization". In : *8th International Conference on 3D Vision, 3DV 2020, Virtual Event, Japan, November 25-28, 2020*. 2020, p. 968-977.

[Stu+16] Elena Stumm, Christopher Mei, Simon Lacroix, Juan I. Nieto, Marco Hutter et Roland Siegwart. "Robust Visual Place Recognition with Graph Kernels". In : *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, p. 4535-4544.

[Stu+12] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard et Daniel Cremers. "A benchmark for the evaluation of RGB-D SLAM systems". In : *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*. 2012, p. 573-580.

[SSS19] Shinya Sumikura, Mikiya Shibuya et Ken Sakurada. "OpenVSLAM : A Versatile Visual SLAM Framework". In : *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*. 2019, p. 2292-2295.

[Sün+17] Niko Sünderhauf, Trung T. Pham, Yasir Latif, Michael Milford et Ian D. Reid. "Meaningful maps with object-oriented semantic mapping". In : *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*. 2017, p. 5079-5085.

[Sün+15] Niko Sünderhauf, Sareh Shirazi, Adam Jacobson, Feras Dayoub, Edward Pepperell, Ben Upcroft et Michael Milford. "Place Recognition with ConvNet Landmarks : Viewpoint-Robust, Condition-Robust, Training-Free". In : *Robotics : Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*. 2015.

[Sun+18] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker et Rudolph Triebel. "Implicit 3d orientation learning for 6d object detection from rgb images". In : *Proceedings of the european conference on computer vision (ECCV)*. 2018, p. 699-715.

[SMDT20] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner et Rudolph Triebel. "Augmented autoencoders : Implicit 3d orientation learning for 6d object detection". In : *International Journal of Computer Vision* 128.3 (2020), p. 714-729.

[SEOK14] Linus Svarm, Olof Enqvist, Magnus Oskarsson et Fredrik Kahl. "Accurate localization and pose estimation for large 3d models". In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, p. 532-539.

[SEKO16] Linus Svärm, Olof Enqvist, Fredrik Kahl et Magnus Oskarsson. "City-scale localization for cameras with known vertical direction". In : *IEEE transactions on pattern analysis and machine intelligence* 39.7 (2016), p. 1455-1461.

[Tai+18] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla et Akihiko Torii. "InLoc : Indoor visual localization with dense matching and view synthesis". In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, p. 7199-7209.

[TTLN17] Keisuke Tateno, Federico Tombari, Iro Laina et Nassir Navab. "CNN-SLAM : Real-Time Dense Monocular SLAM with Learned Depth Prediction". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, p. 6565-6574.

[TD21] Zachary Teed et Jia Deng. "DROID-SLAM : Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras". In : *Advances in Neural Information Processing Systems 34 : Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 2021, p. 16558-16569.

[TSF18] Bugra Tekin, Sudipta N. Sinha et Pascal Fua. "Real-Time Seamless Single Shot 6D Object Pose Prediction". In : *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, p. 292-301.

[Tof+18] Carl Toft, Erik Stenborg, Lars Hammarstrand, Lucas Brynte, Marc Pollefeys, Torsten Sattler et Fredrik Kahl. "Semantic match consistency for long-term visual localization". In : *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, p. 383-399.

[Tof+20] Carl Toft et al. "Long-term visual localization revisited". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[Tor+15] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi et Tomas Pajdla. "24/7 place recognition by view synthesis". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, p. 1808-1817.

[TSPO13] Akihiko Torii, Josef Sivic, Tomas Pajdla et Masatoshi Okutomi. "Visual place recognition with repetitive structures". In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, p. 883-890.

[TZ00] Philip H. S. Torr et Andrew Zisserman. "MLESAC : A New Robust Estimator with Application to Estimating Image Geometry". In : *Comput. Vis. Image Underst.* 78.1 (2000), p. 138-156.

[TDBK18] Michael Truong-Le, Frederik Diehl, Thomas Brunner et Alois C. Knoll. "Uncertainty Estimation for Deep Neural Object Detectors in Safety-Critical Applications". In : *21st International Conference on Intelligent Transportation Systems, ITSC 2018, Maui, HI, USA, November 4-7, 2018*. 2018, p. 3873-3878.

[Wal+17] Florian Walch, Caner Hazirbas, Laura Leal-Taixé, Torsten Sattler, Sebastian Hilsenbeck et Daniel Cremers. "Image-Based Localization Using LSTMs for Structured Feature Correlation". In : *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, p. 627-637.

[Wan+19a] Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin et Tom Vercauteren. "Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks". In : *Neurocomputing* 338 (2019), p. 34-45.

[Wan+19b] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song et Leonidas J Guibas. "Normalized object coordinate space for category-level 6d object pose and size estimation". In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, p. 2642-2651.

[Wan+14] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen et Ying Wu. "Learning Fine-Grained Image Similarity with Deep Ranking". In : *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. 2014, p. 1386-1393.

[WRA21] Jingwen Wang, Martin Rünz et Lourdes Agapito. "DSP-SLAM : Object Oriented SLAM with Deep Shape Priors". In : *International Conference on 3D Vision, 3DV 2021, London, United Kingdom, December 1-3, 2021*. 2021, p. 1362-1371.

[WCWT17] Sen Wang, Ronald Clark, Hongkai Wen et Niki Trigoni. "DeepVO : Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks". In : *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*. 2017, p. 2043-2050.

[WCCH19] Philippe Weinzaepfel, Gabriela Csurka, Yohann Cabon et Martin Humenberger. "Visual localization by learning objects-of-interest dense match regression". In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, p. 5634-5643.

[WKP16] Tobias Weyand, Ilya Kostrikov et James Philbin. "Planet-photo geolocation with convolutional neural networks". In : *European Conference on Computer Vision*. Springer. 2016, p. 37-55.

[WKR11] Brian Patrick Williams, Georg Klein et Ian Reid. "Automatic Relocalization and Loop Closing for Real-Time Monocular SLAM". In : *IEEE Trans. Pattern Anal. Mach. Intell.* 33.9 (2011), p. 1699-1712.

[WKR07] Brian Patrick Williams, Georg Klein et Ian D. Reid. "Real-Time SLAM Relocalisation". In : *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*. 2007, p. 1-8.

[Wu+20] Yanmin Wu, Yunzhou Zhang, Delong Zhu, Yonghui Feng, Sonya Coleman et Dermot Kerr. "EAO-SLAM : Monocular Semi-Dense Object SLAM Based on Ensemble Data Association". In : *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*. 2020, p. 4966-4973.

[XSNF18] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan et Dieter Fox. "PoseCNN : A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes". In : *Robotics : Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*. 2018.

[Xu+22] Meng Xu, Youchen Wang, Bin Xu, Jun Zhang, Jian Ren, Stefan Poslad et Pengfei Xu. "A Critical Analysis of Image-based Camera Pose Estimation Techniques". In : *arXiv preprint arXiv :2201.05816* (2022).

[Yan+20] Cong Yang, Gilles Simon, John See, Marie-Odile Berger et Wenyong Wang. "WatchPose : A View-Aware Approach for Camera Pose Data Collection in Industrial Environments". In : *Sensors* 20.11 (2020).

[Yan+19] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa et Ping Tan. "SANet : Scene Agnostic Network for Camera Localization". In : *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 2019, p. 42-51.

[YSWC20] Nan Yang, Lukas von Stumberg, Rui Wang et Daniel Cremers. "D3VO : Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry". In : *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. 2020, p. 1278-1289.

[YWSC18] Nan Yang, Rui Wang, Jörg Stückler et Daniel Cremers. "Deep Virtual Stereo Odometry : Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry". In : *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*. T. 11212. Lecture Notes in Computer Science. 2018, p. 835-852.

[YS19] Shichao Yang et Sebastian A. Scherer. "CubeSLAM : Monocular 3-D Object SLAM". In : *IEEE Trans. Robotics* 35.4 (2019), p. 925-938.

[YTLF16] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit et Pascal Fua. "LIFT : Learned Invariant Feature Transform". In : *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*. T. 9910. Lecture Notes in Computer Science. 2016, p. 467-483.

[ZSI19] Sergey Zakharov, Ivan Shugurov et Slobodan Ilic. "Dpod : 6d pose object detector and refiner". In : *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, p. 1941-1950.

[ZSP15] Bernhard Zeisl, Torsten Sattler et Marc Pollefeys. "Camera pose voting for large-scale image-based localization". In : *Proceedings of the IEEE International Conference on Computer Vision*. 2015, p. 2704-2712.

[Zha+18] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal et Ian D. Reid. "Unsupervised Learning of Monocular Depth Estimation and Visual Odometry With Deep Feature Reconstruction". In : *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, p. 340-349.

[ZR18] Linguang Zhang et Szymon Rusinkiewicz. "Learning to Detect Features in Texture Images". In : *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, p. 6325-6333.

[ZLHF21] Yu-Ming Zhang, Chun-Chieh Lee, Jun-Wei Hsieh et Kuo-Chin Fan. "CSL-YOLO : A New Lightweight Object Detection System for Edge Computing". In : *CoRR* abs/2107.04829 (2021).

[Zhe+20] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye et Dongwei Ren. "Distance-IoU loss : Faster and better learning for bounding box regression". In : *Proceedings of the AAAI conference on artificial intelligence*. T. 34. 07. 2020, p. 12993-13000.

[ZUB18] Huizhong Zhou, Benjamin Ummenhofer et Thomas Brox. "DeepTAM : Deep Tracking and Mapping". In : *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI*. T. 11220. Lecture Notes in Computer Science. 2018, p. 851-868.

[ZSPL20] Qunjie Zhou, Torsten Sattler, Marc Pollefeys et Laura Leal-Taixe. "To learn or not to learn : Visual localization from essential matrices". In : *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, p. 3319-3326.

[ZBSL17] Tinghui Zhou, Matthew Brown, Noah Snavely et David G. Lowe. "Unsupervised Learning of Depth and Ego-Motion from Video". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, p. 6612-6619.

[ZSB20] Matthieu Zins, Gilles Simon et Marie-Odile Berger. "3D-aware ellipse prediction for object-based camera pose estimation". In : *2020 International Conference on 3D Vision (3DV)*. IEEE. 2020, p. 281-290.

[ZSB22a] Matthieu Zins, Gilles Simon et Marie-Odile Berger. "Level Set-Based Camera Pose Estimation From Multiple 2D/3D Ellipse-Ellipsoid Correspondences". In : *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. 2022.

[ZSB22b] Matthieu Zins, Gilles Simon et Marie-Odile Berger. "OA-SLAM : Leveraging Objects for Camera Relocalization in Visual SLAM". In : *2022 IEEE International Symposium on Mixed and Augmented Reality, ISMAR*. 2022.

[ZSB22c] Matthieu Zins, Gilles Simon et Marie-Odile Berger. "Object-Based Visual Camera Pose Estimation From Ellipsoidal Model and 3D-Aware Ellipse Prediction". In : *International Journal of Computer Vision* 130.4 (2022), p. 1107-1126.

Résumé

La localisation visuelle est un problème bien connu en vision par ordinateur, qui a de nombreuses applications, par exemple, en robotique pour la navigation de systèmes autonomes (robots, drones, véhicules) ou en réalité augmentée. Elle consiste à estimer la position et l'orientation de la caméra dans une scène. Les approches classiques reposent généralement sur la structure géométrique de la scène et cherchent à mettre en correspondance des points d'intérêt 2D, détectés dans les images, avec des points 3D de la scène. Cet appariement est cependant un problème complexe en pratique, notamment parce qu'il repose sur de l'information locale, extraite dans un voisinage autour des points d'intérêt. Selon la taille de la scène, ces méthodes peuvent être très coûteuses en calcul. Elles sont également sensibles à des changements importants de points de vue, à des conditions visuelles dégradées et échouent dans des zones faiblement texturées.

Dans ce travail de thèse, nous nous sommes intéressés à l'utilisation des objets comme balises sémantiques pour le positionnement visuel. Grâce aux avancées récentes, notamment avec l'apprentissage profond, il est possible de détecter des objets de manière très robuste dans les images, de pratiquement n'importe quel point de vue. Nous avons adopté une modélisation légère des objets sous la forme d'ellipsoïdes et nous voulons en tirer profit pour améliorer la robustesse de la localisation visuelle.

Dans un premier temps, nous avons cherché à améliorer la détection des objets par des ellipses, qui constituait l'une des principales sources d'imprécision du calcul de pose. Ainsi, nous avons remplacé les ellipses inscrites dans les boîtes de détection alignées avec les axes de l'image par des ellipses orientées cohérentes avec la projection des modèles ellipsoïdaux des objets. Nos expériences ont montré que notre approche améliore nettement la précision des méthodes existantes basées sur les objets et surpasse la robustesse des méthodes par points. Dans un second temps, nous avons proposé une étape de raffinement de la pose de la caméra par la minimisation d'une erreur de reprojection des objets, qui permet de prendre en considération tous les objets détectés dans l'image. Contrairement à une distance entre des points, établir un coût entre des ellipses n'est pas trivial. Nous avons analysé différentes métriques et nous avons proposé une nouvelle formulation basée sur des ensembles de niveaux. Nos expériences ont mis en avant ses bonnes propriétés de convergence et de gestion des objets partiellement visibles dans l'image. Nous avons également montré que cette étape de raffinement permet d'améliorer considérablement la solution analytique du calcul de pose basé sur les objets. Enfin, nous avons intégré ce concept d'objet dans un SLAM et développé un système capable de cartographier les objets à la volée. L'intérêt est double, avec la possibilité de les utiliser comme balises de relocalisation et avec l'ajout d'une information sémantique à la carte offrant une meilleure compréhension de la scène. Notre système fait collaborer les objets et les points et bénéficie de leurs avantages respectifs, la robustesse et la précision. Nous avons montré, dans nos expériences, que cela permet d'étendre considérablement la capacité de relocalisation de notre système.

Mots-clés: Vision par ordinateur, Réalité augmentée, Localisation visuelle, Apprentissage profond, SLAM.

Abstract

Visual localization is a well-known problem in computer vision, which has many applications, for example, in robotics for autonomous navigation (robots, drones, vehicles) or in augmented reality. It consists in estimating the position and orientation of the camera in a scene. Classical approaches are generally based on the geometric structure of the scene and seek to match 2D keypoints, detected in the images, with 3D points of the scene. This matching is however a complex problem in practice, especially because it relies on local information, extracted in the neighborhood of the keypoints. Depending on the size of the scene, these methods can be very computationally expensive. They are also sensitive to large changes in viewpoints, to degraded visual conditions and fail in weakly textured areas.

In this thesis, we are interested in using objects as semantic landmarks for visual positioning. Thanks to recent advances, especially with deep learning, it is possible to detect objects very robustly in images, from almost any viewpoint. We have adopted a lightweight modeling of objects as ellipsoids and want to leverage this to improve the robustness of visual localization.

As a first step, we sought to improve the detection of objects by ellipsoids, which was one of the main sources of inaccuracy in the pose computation. Thus, we replaced the ellipses inscribed in the detection boxes aligned with the image axes by oriented ellipses consistent with the projection of the ellipsoidal models of the objects. Our experiments have shown that our approach significantly improves the accuracy of existing object-based methods and outperforms the robustness of point-based methods. Secondly, we proposed a refinement step of the camera pose by minimizing a reprojection error, which allows us to take into account all the detected objects in the image. Unlike a distance between points, establishing a cost between ellipses is not trivial. We analyzed different metrics and proposed a new formulation based on level sets. Our experiments highlighted its good convergence properties and a better handling of partially visible objects in the image. We have also shown that this refinement step allows us to considerably improve the analytical solution of the object-based pose calculation. Finally, we integrated this concept of object into SLAM and developed a system capable of mapping objects on the fly. The interest is twofold, with the possibility to use them as relocalization landmarks and with the introduction of semantic information in the map, offering a better scene understanding. Our system makes objects and points collaborate and benefits from their respective advantages, robustness and accuracy. We have shown, in our experiments, that this allows us to significantly extend the relocalization capability of our system.

Keywords: Computer Vision, Augmented Reality, Visual Localization, Deep Learning, SLAM.