Relocalisation Robuste de Caméra en Temps Réel pour la Réalité Augmentée par une Approche Hybride combinant Réseaux de Neurones et Méthodes Géométriques

Nam-Duong Duong, Amine Kacete, Catherine Soladie, Pierre-Yves Richard, Jérôme Royan

Relocalisation Robuste de Caméra en Temps Réel pour la Réalité Augmentée par une Approche Hybride combinant Réseaux de Neurones et Méthodes Géométriques

Nam-Duong Duong[1] Amine Kacete[1] Catherine Sodalie[1] Pierre-Yves Richard[1] Jérôme Royan[1]

[1] IRT b<>com

nam-duong.duong@b-com.com

Résumé

La relocalisation des caméras se signale comme une problématique centrale dans le domaine émergent de la réalité augmentée. Les approches les plus courantes pour la traiter, regroupées sous l'appellation générique de méthodes géométriques, ont pour noms SLAM (Simultaneous Localization And Mapping) et SfM (Structure from Motion). Les rapides progrès de l'apprentissage automatique, en particulier ceux de l'apprentissage en profondeur, ont également offert de nouvelles perspectives prometteuses à cette problématique. De premières tentatives ont récemment été faites pour combiner les deux types d'approches. Cependant, la lourdeur des algorithmes utilisés rend difficile leur exploitation dans le contexte temps réel sous-jacent à la réalité augmentée. De plus, les prédictions concernant la pose d'une caméra restent incertaines, n'étant encore assorties d'aucun score de confiance. Dans cet article, nous proposons une méthode hybride mélangeant à la fois les approches de l'apprentissage en profondeur et les approches géométriques pour estimer la pose d'une caméra indépendamment image par image. Nous présentons un réseau de neurones convolutif (CNN) léger, appelé xyzNet pour calculer en temps réel et robustement par régression les coordonnées dans le repère du monde des points réels associés aux pixels d'une image. Ensuite, l'information géométrique sur les correspondances 2D-3D permet l'élimination des prédictions ambiguës et le calcul d'une pose de caméra plus précise. De plus, nous montrons des résultats favorables quant à l'exactitude et la performance de notre méthode sur des ensembles de données différents ainsi que sa capacité à relever les défis concernant la scène dynamique.

Mots Clef

Relocalisation de caméra en temps réel, régression de l'apprentissage en profondeur.

Abstract

Camera relocalization is a central issue in augmented reality. The most common approaches for camera relocalization known as the geometric-based methods are Simultaneous Localization And Mapping (SLAM) and Structure from Motion (SfM). Also, camera relocalization has recently obtained many promising results thanks to progress in machine learning, especially in deep learning. First attempts to combine both kinds of approaches have recently been published. However, the latter are not suitable for a real time use, because of time consuming algorithms. Besides, prediction about camera pose keeps uncertain with no confidence score provided. In this paper, we propose a hybrid method merging both deep learning and geometric approaches to estimate camera pose in real time. We present a light Convolutional Neural Network (CNN) called xyzNet to efficiently and robustly regress world coordinates of pixels in an image. Then, the geometric information about 2D-3D correspondences allows the removal of ambiguous predictions and the calculation of more accurate camera pose. Moreover, we show favorable results about the accuracy and the performance of our method on different datasets as well as the capacity to address challenges concerning dynamic scene.

Keywords

Real time camera relocalisation, deep learning regression.

1 Introduction

Les deux solutions les plus courantes d'estimation de pose de caméra pour les systèmes commerciaux sont *Structure from Motion* (SfM) [16, 5] et *Simultaneously Localization And Mapping* (SLAM) [10, 3, 17]. Les méthodes SfM traitent hors ligne un ensemble d'images non ordonnées sans contrainte temporelle pour estimer la pose de la caméra en utilisant des caractéristiques correspondantes parmi des paires d'images. Grâce à ces appariements, ils peuvent à la fois reconstruire un modèle de scène 3D et estimer la pose d'une caméra. Inversement, les méthodes SLAM peuvent fonctionner en temps réel sur une séquence ordonnée d'images acquises à partir d'une caméra, potentiellement associées à une unité de mesure inertielle. La majorité des systèmes SLAM sont basés sur le suivi de mouvement de la caméra, en utilisant des trames consécutives pour calculer robustement la pose de la caméra. Mal-

heureusement, dans le cas d'un mouvement de caméra rapide ou d'un changement de point de vue soudain, l'échec du suivi interrompt l'estimation de la pose de la caméra. Et les méthodes SLAM doivent stocker un grand nombre de points-clés ou de images-clés pour réaliser la localisation et la cartographie. Par conséquent, l'utilisation de la mémoire ainsi que le temps de traitement augmentent linéairement par rapport à la taille des modèles. De plus, le déplacement de caméras le long de trajectoires non refermées sur elles-mêmes pour l'observation de grandes scènes génère un cumul d'erreur dans le suivi trame-à-trame qui provoque des dérives dans ces méthodes.

Par ailleurs, ces dernières années, les approches d'apprentissage automatique se sont répandues dans le domaine de la vision par ordinateur et fournissent d'excellents résultats pour des applications nombreuses et variées, parmi lesquelles la relocalisation de caméra. Cette problématique y est traitée comme un problème de régression résolu par un apprentissage supervisé basé sur les informations connues à l'avance sur chaque scène. Bien que les méthodes basées sur l'apprentissage automatique ne puissent fonctionner que sur des scènes connues, elles ne nécessitent aucune pose d'estimation initiale et peuvent gérer des trames individuelles à la différence des méthodes SLAM. En particulier, de telles méthodes [9, 7, 19, 2] permettent d'estimer rapidement la pose d'une caméra à partir de chaque image entière. Cependant, les limitations de ces méthodes résident dans leur précision et l'absence de score de confiance pour chaque estimation de pose. D'autres auteurs, [15, 4, 18] ont également proposé des méthodes utilisant les forêts aléatoires (random forest) pour la relocalisation de caméra avec une grande précision. Toutefois la nécessaire disponibilité d'une image de profondeur pour l'estimation restreint leur champ d'application.

Dans cet article, pour aider à surmonter les limitations précédentes, nous proposons une méthode hybride, basée à la fois sur l'approche de l'apprentissage automatique supervisé et sur l'approche géométrique et permettant de cumuler leurs avantages respectifs, dans le but d'estimer la pose de la caméra pour chaque image indépendamment des autres. Cela nous conduit à définir un pipeline compact pour la relocalisation d'une caméra.

2 Travaux connexes

Méthodes basées sur les caractéristiques pour la relocalisation de la caméra

Les approches basées sur les caractéristiques clairsemés (sparse features) utilisent des caractéristiques invariantes au facteur d'échelle et à la rotation telles que SIFT, SURF, ORB, etc. Dans de telles approches, la pose d'une caméra est estimée selon un pipeline commun qui comprend trois étapes : extraction de caractéristiques ; appariement de caractéristiques et calcul de pose. La pose finale de la caméra est déterminée en minimisant l'erreur de re-projection. Pour résoudre le problème de perte de suivi, les approches trame-à-modèle construisent d'abord un modèle de scène comprenant un ensemble de points-clés en appliquant SfM [14] ou *Bags of Words* (BoW) [10] une fois sur un ensemble d'images de la scène. Ensuite, la relocalisation de la caméra peut être effectuée à partir de chaque image indépendamment pour améliorer la relocalisation dans les méthodes SLAM. Même ainsi, le temps de appariement dépend de la taille du modèle, ce qui réduit considérablement l'évolutivité.

Le SLAM semi-dense ou dense est connu comme une nouvelle approche des méthodes SLAM proposée ces dernières années pour reconstruire une carte plus dense. Les approches denses utilisent directement l'intensité des pixels de l'image RVB ou de l'image de profondeur sans extraction de caractéristiques. En utilisant un capteur de profondeur, un nuage de points dense est créé à partir des cartes de profondeur. La pose de la caméra est estimée par le suivi de nuage de points en utilisant l'algorithme ICP [11] et l'algorithme de Lucas-Kanade [13]. Cependant, les caméras de profondeur ne sont pas aussi omniprésentes que les caméras couleur. Par conséquent, une grande partie de l'intérêt de la recherche a porté sur les méthodes SLAM denses [12] et semi-denses [3] à partir d'une seule caméra RVB.

Apprentissage en profondeur pour la relocalisation de la caméra

[9, 7] ont été les premiers à proposer l'utilisation de l'apprentissage en profondeur comme approche d'estimation de pose de caméra de bout en bout. Dans leurs méthodes, un modèle de réseau de neurones convolutif (CNN) est appris à partir d'images entières labellisées à l'aide des poses de la caméra. Ensuite, le modèle entraîné prédit directement la pose de la caméra à partir de chaque image RVB. [9] présente la façon d'adapter le modèle de GoogleNet de la classification à la régression en modifiant ses couches finales afin de régresser la pose de la caméra. La phase d'entraînement est réalisée avec une fonction de perte qui est la somme de l'erreur de localisation et de l'erreur d'orientation. Un facteur d'échelle est utilisé pour maintenir les deux valeurs d'erreur à peu près égales. Cependant, la valeur du facteur d'échelle dépend de chaque scène, ce qui rend les paramètres d'une nouvelle scène difficiles à configurer. [7] donne un moyen de générer une estimation de pose probabiliste en utilisant le dropout après chaque couche de convolution comme moyen d'échantillonnage des poids du modèle. [8] résout une ambiguïté du facteur d'échelle entre erreur de localisation et erreur d'orientation dans la fonction de perte de [9] par une nouvelle fonction de perte basée sur une erreur de re-projection. [2] exploite l'information temporelle en utilisant plusieurs images pour la prédiction de pose. Un LSTM (Long Short Term Memory) qui est l'extension du réseau de neurones récurrent (RNN) est utilisé pour prédire la pose de caméra pour chaque image de la séquence à partir du vecteur de caractéristiques du CNN.

Fusion de méthodes basées sur les caractéristiques et de méthodes basées sur l'apprentissage automatique pour la relocalisation de caméra

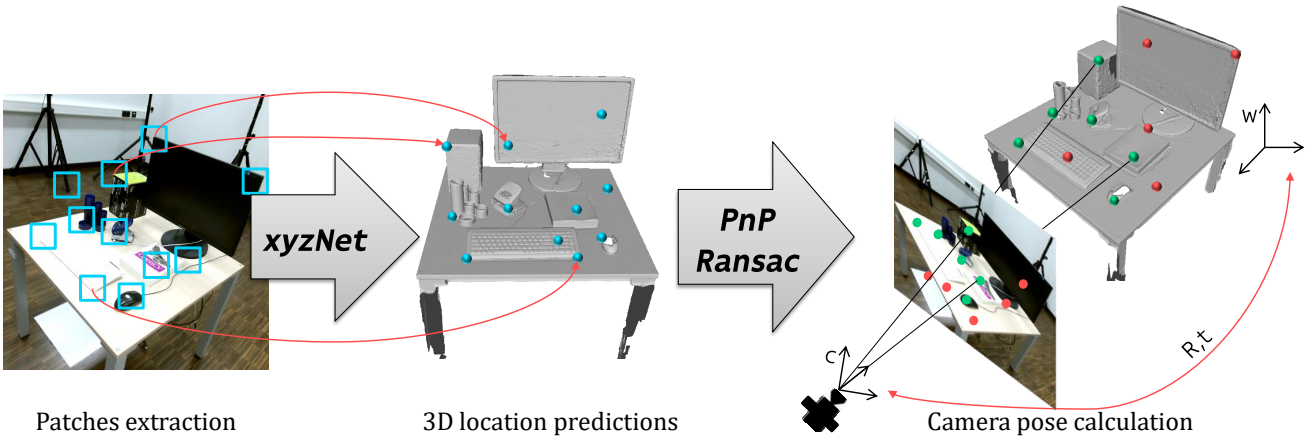


FIGURE 1 – ***xyzNet* Pipeline de la relocalisation de la caméra** : A partir d'un ensemble de patchs (carrés bleus) extraits sur chaque image RVB, nous les passons à travers *xyzNet* pour prédire un ensemble de positions 3D (points bleus) dans le système de coordonnées du monde. Ensuite, les algorithmes PnP et Ransac sont utilisés pour filtrer les inliers (points verts) et éliminer les outliers (points rouges). Enfin, la pose de la caméra est calculée en relançant PnP une fois sur tous les inliers.

Les méthodes présentées dans ce paragraphe apprennent et prédisent la position 3D de chaque pixel d'une image. Dans [15, 4, 18], chaque pixel effectue une prédiction continue de sa propre position 3D dans le système de coordonnées du monde. Ensuite, la pose de la caméra est calculée en utilisant l'algorithme de Kabsch sur les correspondances de points 3D-3D, à savoir la prédiction 3D en coordonnées du monde avec le point 3D correspondant dans les coordonnées de la caméra déduites de l'image de profondeur. [17] propose une méthode CNN-SLAM, où un CNN est intégré pour prédire une carte de profondeur à partir d'une seule image RVB. Cependant, la prédiction par CNN de la carte de profondeur est très consommatrice de temps. Ce travail est donc effectué uniquement sur les images clés et les poses de caméra pour les images intermédiaires sont estimées en fonction de l'image-clé la plus proche.

3 Méthode proposée

3.1 Aperçu

Dans cette section, nous présentons notre propre méthode hybride combinant une approche d'apprentissage en profondeur et une approche géométrique pour la relocalisation de caméra. La figure 1 illustre le pipeline correspondant, qui peut se résumer en deux étapes principales : localisation des points 3D associés aux pixels centraux de patchs locaux dans le système de coordonnées du monde pour définir les correspondances de points 2D-3D ; calcul géométrique de la pose de la caméra à partir de ces correspondances.

3.2 Apprentissage en profondeur pour la localisation de points 3D basée sur des patchs locaux

Nous inspirant du succès de l'apprentissage en profondeur basé sur des patchs locaux pour la détection et la segmentation d'objets [6], nous proposons un réseau de neurones convolutifs léger pour inférer des positions 3D de pixels spécifiques dans le système de coordonnées du monde basé sur l'apparence des patchs RVB qui les entourent. Nous extrayons d'images RVB des patchs locaux porteurs d'informations significatives. Les patchs homogènes tels que des portions de ciel ou de route ne fournissent pas d'informations distinctives sur la pose de la caméra dans la scène. L'utilisation de ces patchs n'est pas recommandée car elle produit du bruit. Un problème posé par la régression de l'apprentissage en profondeur pour la relocalisation de caméra réside dans le manque de score de confiance, qui rend les prédictions incertaines. Pour résoudre ce problème, [7, 2] crée un modèle probabiliste de résultats en utilisant une couche de dropout après chaque couche de convolution comme moyen d'échantillonnage des poids du modèle. Dans notre méthode, nous préférons utiliser un ensemble de patchs pour générer un ensemble de résultats probabilistes à partir des données.

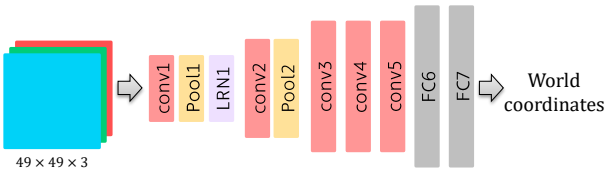


FIGURE 2 – ***xyzNet*** : Un CNN de régression pour prédire les coordonnées du monde de patchs RVB.

Extraction et labellisation des patchs

À partir de chaque image RVB, nous extrayons un ensemble de patchs pour la relocalisation de la caméra. Au lieu de les choisir au hasard, nous sélectionnons délibérément des patchs autour de points-clés pour cibler uni-

quement les régions géométriquement pertinentes. Ainsi, chaque patch retenu est une portion d'image de taille fixe autour d'un point-clé . Nous utilisons le détecteur SURF (Speed Up Robust Features) [1] pour détecter automatiquement des points clairsemés invariants à l'échelle et la rotation, en tant que représentants répétitifs dans une scène. Cela améliore la capacité à localiser les positions 3D à partir des patchs. À partir de chaque image, nous obtenons ainsi un ensemble de patchs $\mathcal{P} = \{\mathcal{P}_i\}$ centrés sur les points-clés de SURF $p = \{p_i = (u_i, v_i)\}$, où p_i définit les coordonnées d'image.

Pour la phase d'entraînement de *xyzNet*, nous devons labelliser les données d'entraînement avec les coordonnées du monde 3D correspondantes, $P_i^w = (X_i^w, Y_i^w, Z_i^w)$. Une solution possible pour labelliser serait l'exécution de SfM une fois sur tous les ensembles de données d'apprentissage pour effectuer une cartographie entre les points-clés et le nuage de points. Cependant, nos expériences dans la section 4 sont effectuées sur des ensembles de données RVB-D. Donc, nous pouvons utiliser les images RVB-D de la caméra calibrée avec leurs poses de caméra correspondantes pour définir les labels pour la phase d'entraînement. Il convient de souligner que dans la phase de test, nous utilisons uniquement des images RVB et nous n'avons pas besoin d'informations de profondeur. Dans la phase d'entraînement, à partir de la position $p_i = (u_i, v_i)$ de chaque point-clé détecté dans l'image RVB et de la valeur de profondeur correspondante D_i dans l'image de profondeur, une position 3D $P_i^c = (X_i^c, Y_i^c, Z_i^c)$ du système de coordonnées de la caméra est calculée en utilisant le modèle de caméra sténopé standard :

$$P_i^c = D_i K^{-1} \begin{bmatrix} p_i \\ 1 \end{bmatrix}$$

Où K est une matrice de paramètres intrinsèques de la caméra. La pose $T = [R|t]$ de la caméra incluant la matrice de rotation R et le vecteur de translation t pour chaque trame est supposée être connue à l'avance. Les coordonnées du monde $P_i^w = (X_i^w, Y_i^w, Z_i^w)$ correspondant au pixel p_i sont définies en fonction de l'équation de transformation dans le système de coordonnées homogènes :

$$\begin{bmatrix} P_i^w \\ 1 \end{bmatrix} = \left[\begin{array}{c|c} R & t \\ 0 & 1 \end{array}\right] \begin{bmatrix} P_i^c \\ 1 \end{bmatrix}$$

xyzNet

Nous avons conçu un nouveau réseau de régression pour prédire directement à partir de pixels 2D les correspondances 3D dans le système de coordonnées du monde. Il prend en entrée des patchs d'image RVB avec une taille fixe de 49×49 pixels. Les réseaux les plus courants utilisés dans l'apprentissage en profondeur ne sont pas adaptés au traitement de plusieurs patchs en termes de temps de calcul. Nous réalisons un ConvNet léger dédié à la relocalisation de caméra, à la fois efficace, robuste et compatible avec le temps réel. xyzNet consiste en cinq couches qui effectuent la convolution de l'entrée avec un ensemble de filtres de noyaux 3×3, des couches de sous-échantillonnage (max pooling) sur une zone de 3×3 et des couches de fonction d'activation (ReLU). Dans la première étape, une normalisation de réponse locale (LRN) est utilisée pour normaliser les patchs avec des conditions de lumière différentes. Ces cinq couches sont suivies par deux couches entièrement connectées pour régresser les coordonnées du monde. Une couche de dropout est ajoutée après chaque couche entièrement connectée pour gérer l'*over-fitting*. xyzNet est illustré sur la figure 2.

Dans la phase d'apprentissage, les poids de xyzNet sont appris en minimisant une fonction de perte euclidienne avec un algorithme d'optimisation qui est une descente de gradient stochastique. La fonction de perte est définie comme suit :

$$l(p) = \sum_{p_i \in p} \|P_i^w - \hat{P_i^w}\|_2^2$$

Où P_i^w et $\hat{P_i^w}$ sont respectivement la vérité et la prédiction des coordonnées 3D du pixel p_i dans le système de coordonnées du monde. Chaque patch founit une prédiction continue de sa propre position 3D dans les coordonnées du monde plutôt que la pose de la caméra (6-DOF) comme dans [9]. Ceci réduit considérablement la complexité de la fonction de perte, en produisant une optimisation plus efficace.

3.3 Calcul de la pose de la caméra

Dans cette section, nous allons montrer comment estimer la pose de la caméra à partir des prédictions de *xyzNet*. Quand tous les patchs sont passés par *xyzNet*, nous obtenons un ensemble de correspondances 2D-3D. Un triplet de prédictions exactes est théoriquement suffisant pour déduire la pose de la caméra. En pratique, l'algorithme Perspective-n-Points (PnP), méthode bien connue de vision par ordinateur, permet de résoudre ce problème. Toutefois, comme les prédictions produites par xyzNet peuvent être bruitées, nous ne considérons pas directement toutes les correspondances de points 2D-3D pour calculer la pose de la caméra. Au lieu de cela, nous utilisons d'abord PnP et Ransac (Random sample consensus) pour supprimer les bruit (outliers) et conserver les prédictions exactes (inliers). Plus prÃl'cisÃl'ment, Ransac génère un ensemble de poses hypothétiques $\mathcal{T} = \{T_i\}$ en effectuant PnP sur des sous-ensembles aléatoires de correspondances de points 2D-3D. Pour chacune des poses ainsi obtenues, une erreur de re-projection est calculée, permettant de séparer les inliers des outliers à l'aide d'un seuil. On retient alors les inliers associés à l'hypothèse qui en fournit le plus grand nombre, à savoir :

$$\max_{\forall T_i \in \mathcal{T}} \sum_{p_j \in p} \rho(\alpha_{ij})$$

$$\rho(\alpha_{ij}) = \begin{cases} 1, & \text{if } \alpha_{ij} < \tau \\ 0, & \text{otherwise} \end{cases}$$

Où $\alpha_{ij} = \|p_j - K T_i^{-1} \hat{P_j^w}\|^2$ et τ est le seuil maximal d'erreur de re-projection qui définit les inliers. Ainsi le

pixel j est-il considéré comme un inlier de l'hypothèse T_i si $\rho(\alpha_{ij}) = 1$. Soit $\mathcal{I}$ l'ensemble des indices des inliers associés à la meilleure hypothèse (celle qui maximise le nombre de ces inliers). La pose de caméra finale est calculée en exécutant PnP une fois sur tous ces inliers, afin de minimiser la fonction d'erreur de re-projection :

$$E(T) = \sum_{i \in \mathcal{I}} \|p_i - KT^{-1}\hat{P}_i^w\|^2$$

Inférer la pose de la caméra à partir de plusieurs patchs avec une méthode de filtrage basée sur les algorithmes PnP et Ransac permet d'éliminer les outliers sur les objets en mouvement dans la scène, afin de relever le défi des scènes avec occlusion partielle. De plus, le nombre d'inliers peut être utilisé comme un score de confiance de l'estimation finale qui n'est pas fourni par les méthodes basées sur l'apprentissage en profondeur précédentes. Nos résultats seront présentés dans la section suivante.

4 Expérimentations

Nous évaluons notre méthode sur un ensemble de données de 7 scènes [15] et un ensemble de données CoRBS [20]. Les résultats obtenus permettent d'illustrer sa précision ainsi que ses performances, en comparaison avec les méthodes de l'état de l'art : SCoRe Forest [15], PoseNet 1 [9], PoseNet 2 [8], LSTM-Pose [19], VidLoc [2].

4.1 Ensembles de données

Les deux ensembles de données que nous utilisons dans nos expériences représentent des scènes d'intérieur. Chacun fournit des images RVB-D (plus de 2000 images), une matrice intrinsèque de caméra et des annotations (pose de caméra pour chaque image).

7 scènes est un ensemble de données présenté par [15] contient sept scènes, dont chacune comprend des séquences capturées autour d'une seule pièce et annotées à l'aide de Kinect Fusion [11]. Ces données sont extrêmement difficiles à traiter en raison de la rotation pure ou du mouvement rapide de la caméra qui fournit de nombreuses images sans texture.

CoRBS [20] est plus précis grâce à l'utilisation de plusieurs capteurs. Les données visuelles sont archivées en utilisant un Kinect v2. La vérité de terrain est obtenue par un système de capture de mouvement externe. Chaque scène contient un modèle de scène 3D dense créé via un scanner 3D externe.

4.2 Bases de Référence

Nous évaluons notre méthode sur les ensembles de données ci-dessus pour la comparer aux méthodes de l'état de l'art.

PoseNet 1 [9] et **PoseNet 2** [8] sont des méthodes qui prennent en entrée une image RVB entière pour prédire la pose de la caméra. Alors que PoseNet 1 utilise une fonction de perte combinant rotation et position par un facteur d'échelle qui dépend de chaque scène, PoseNet 2 introduit une nouvelle fonction de perte utilisant l'erreur de re-projection pour améliorer ses performances.

Scene	Chess	Fire	Heads	Office	Pumpkin	RedKitchen	Stairs
Err_P	0.25m	0.19m	0.14m	0.65m	0.27m	0.44m	0.34m
Err_I	0.13m	0.11m	0.06m	0.26m	0.11m	0.14m	0.13m

TABLE 1 – **Erreur de *xyzNet*** : Erreur de distance moyenne entre les prédictions et les vérités de terrain sur l'ensemble de tous les prédictions (Err_P) et l'ensemble des inliers (Err_I).

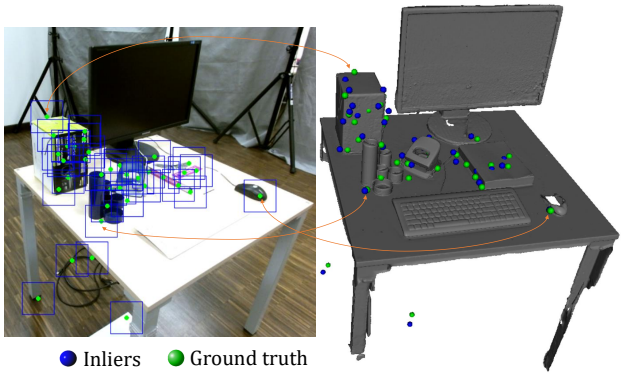


FIGURE 3 – **Exactitude du *xyzNet*** : À partir d'un ensemble de patchs (carrés bleus) extraits autour de points clés (points 2D verts) correspondant à la vérité terrain (points 3D verts) dans le système de coordonnées du monde, xyzNet prédit un ensemble de coordonnées du monde (points 3D bleus).

LSTM-Pose [19] est une amélioration de l'architecture de PoseNet avec des LSTM spatiaux ajoutés après les couches CNN, qui vise à réduire la dimension du vecteur de caractéristiques, fournissant ainsi des améliorations importantes dans les performances de localisation.

VidLoc [2] est similaire à LSTM-Pose. Cependant, cette méthode nécessite une séquence d'images en entrée pour augmenter les informations temporelles.

SCoRe Forest [15] utilise une forêt aléatoire plutôt qu'un CNN afin de régresser les coordonnées de la scène de n'importe quel pixel de l'image. La pose d'une caméra est estimée en optimisant une fonction d'objectif qui utilise l'image de profondeur comme entrée. Par conséquent, cette méthode, conceptuellement la plus proche de la nôtre, est contrainte par l'utilisation d'un capteur RVB-D pendant la phase de test.

4.3 Exactitude de *xyzNet*

xyzNet est le noyau de notre pipeline et a une influence puissante sur la précision de la pose de la caméra. Nous considérons l'exactitude de *xyzNet* à travers l'erreur de distance entre les prédictions et la vérité de terrain. Nous donnons quelques résultats pour évaluer l'efficacité et la robustesse de *xyzNet*.

Dans le tableau 1, nous affichons une erreur de localisation moyenne sur les données de test de *7 scènes*. Nous calcu-

Name of scene	RVB-D		RVB				
	SCoRe Forest [15]	Ours with refinement	VidLoc [2] (Temporal)	PoseNet [9]	PoseNet 2 [8]	LSTM-Pose [19]	Ours
Chess	0.03m, 0.66°	0.09m, 3.21°	0.16m	0.32m, 8.12°	**0.13m, 4.48°**	0.24m, 5.77°	0.18m, 4.80°
Fire	0.05m, 1.50°	0.11m, 4.26°	0.24m	0.47m, 14.4°	0.27m, 11.3°	0.34m, 11.9°	**0.21m, 6.72°**
Heads	0.06m, 5.50°	0.16m, 6.84°	0.19m	0.29m, 12.0°	0.17m, 13.0°	0.21m, 13.7°	**0.15m, 8.08°**
Office	0.04m, 0.78°	0.40m, 8.62°	0.33m	0.48m, 7.68°	**0.19m, 5.55°**	0.30m, 8.08°	0.42m, 9.59°
Pumpkin	0.04m, 0.68°	0.09m, 2.26°	0.28m	0.47m, 8.42°	0.26m, 4.75°	0.33m, 7.00°	**0.17m, 4.18°**
Red Kitchen	0.04m, 0.76°	0.15m, 2.89°	0.24m	0.59m, 8.84°	0.23m, 5.35°	0.37m, 8.83°	**0.20m, 4.65°**
Stairs	0.32m, 1.32°	0.16m, 2.69°	**0.13m**	0.47m, 13.8°	0.35m, 12.4°	0.40m, 13.7°	0.24m, **4.63°**
Average	0.08m, 1.62°	0.17m, 4.40°	**0.22m**	0.44m, 10.4°	0.23m, 8.12°	0.31m, 9.85°	**0.22m, 6.09°**

TABLE 2 – **Erreurs médiane de pose** : Comparaison des résultats de notre méthode avec les méthodes de l'état de l'art

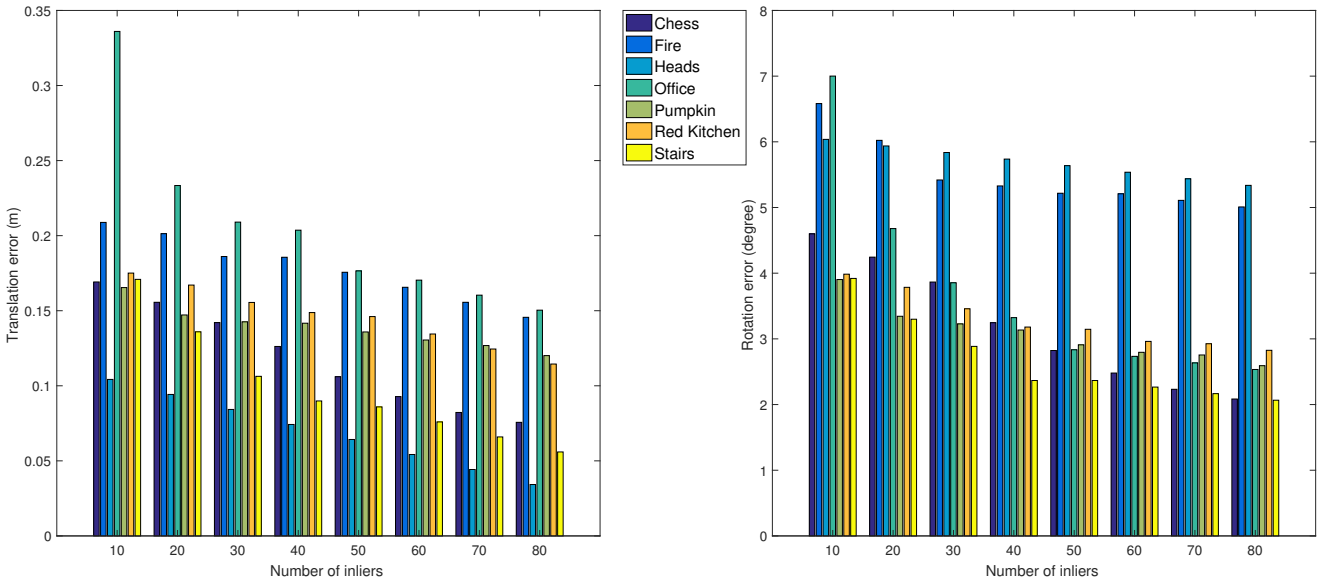


FIGURE 4 – **Erreur de pose de la caméra en fonction du nombre d'inliers** : Pour chaque scène, nous calculons l'erreur médiane de translation (à gauche) et de rotation (à droite) sur les trames qui ont au moins x inliers.

lons deux erreurs de distance sur toutes les prédictions et tous les inliers qui sont $0.33m$ et $0.13m$ respectivement. Sur la figure 3, nous visualisons un exemple sur la scène de *desk* de *CoRBS* concernant les prédictions d'un ensemble d'inliers. Les résultats indiquent que xyzNet fournit une précision plus élevée s'il est efficacement filtré avec Ransac et PnP. Pour les scènes de *office*, *red kitchen*, *stairs*, nous n'avons pas obtenu de bons résultats. Cela provient probablement de la répétitivité des scènes, qui rend ambigus les patchs extraits sur des objets similaires. Cependant, le filtrage avec les algorithmes Ransac et PnP améliore grandement la précision de l'estimation en diminuant les prédictions ambiguës.

4.4 Résultats sur *7 scènes*

Exactitude

Dans le tableau 2, nous avons comparé nos résultats sur un ensemble de données de *7 scènes* à des méthodes de l'état de l'art. Nous séparons les comparaisons en deux groupes : les méthodes basées sur RVB et les méthodes basées sur RVB-D. Pour le groupe de méthodes utilisant uniquement des images RVB qui sont aussi des méthodes d'apprentissage en profondeur, notre méthode surpasse les estimations de position et d'orientation pour les scènes de *fire*, *heads*, *pumpkin* et *red kitchen*. Nous obtenons un résultat plus mauvais sur la scène de *office*. Comme dans l'évaluation de l'exactitude de *xyzNet*, Ransac peut éliminer les ambiguïtés sur les scènes répétitives telles que les scènes de *office*, *red kitchen* et *stairs*. Malheureusement, trop de prédictions sont considérées comme des outliers sur la scène de *office* et sont supprimées pour le PnP final.

Dans l'autre expérience, nous utilisons également des images de profondeur et un suivi d'image à image pour améliorer nos résultats. Premièrement, au lieu de prendre

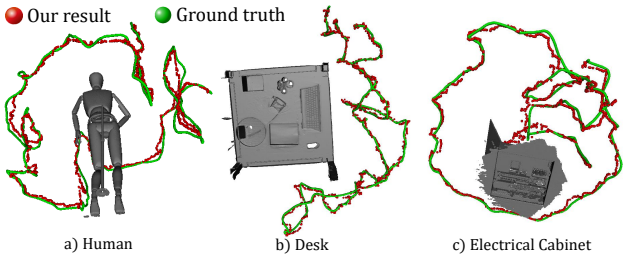


FIGURE 5 – **Notre résultat sur les scènes de CoRBS** : nos résultats par les trajectoires rouges et la vérité terrain en vert.

directement le résultat de PnP, nous appliquons un filtre de Kalman sur la translation et la rotation de la caméra pour chaque séquence. Le filtre de Kalman rend la trajectoire de la caméra plus lisse et plus stable sur les images où nous obtenons peu d'inliers. Ensuite, nous effectuons quelques itérations d'ICP pour affiner la pose de la caméra en appariant le nuage de points de chaque image avec le nuage de points de référence.

Confiance

Le score de confiance de la pose d'une caméra est un problème important dans la relocalisation de la caméra ainsi que dans la régression de l'apprentissage en profondeur et qui n'est pas fournie dans les méthodes de l'état de l'art. Dans notre solution, aucun score de confiance n'est donné par *xyzNet*. Cependant, nous utilisons le nombre d'inliers pour quantifier la précision de notre méthode. Le nombre d'inliers n'est pas un score de confiance absolu, mais la précision y est corrélée. La figure 4 montre la précision croissante de notre méthode en fonction du nombre d'inliers.

Temps de calcul

Nous mesurons le temps de traitement de notre expérimentation. Cela prend environ $60ms$ pour chaque trame avec $10ms$ pour la détection des points-clés SURF, $25ms$ pour le temps de prédiction de 500 patchs sur GPU et $25ms$ pour 500 itérations de Ransac et PnP. Pour notre raffinement, 15 itérations d'ICP prennent environ $25ms$ sur GPU. En général, notre méthode peut ainsi être effectuée en temps réel.

4.5 Résultats sur *CoRBS*

Dans cette section, nous évaluons notre méthode sur l'ensemble de données *CoRBS*. En ce qui concerne l'échelle, les scènes de ces données sont plus simples que celles de l'ensemble de données de *7 scènes*. Chacune scène se concentre sur un petit environnement, par exemple le voisinage d'un bureau. Cependant, cet ensemble de données se révèle difficile à traiter en raison de la présence de nombreuses surfaces planes. Nous choisissons trois séquences (chaque scène contient plus de 2000 images) correspondant à trois scènes de *humain*, *desk*, *eletrical cabinet* pour notre expérimentation. Pour chaque scène, nous prenons une moitié des images pour l'entraînement.

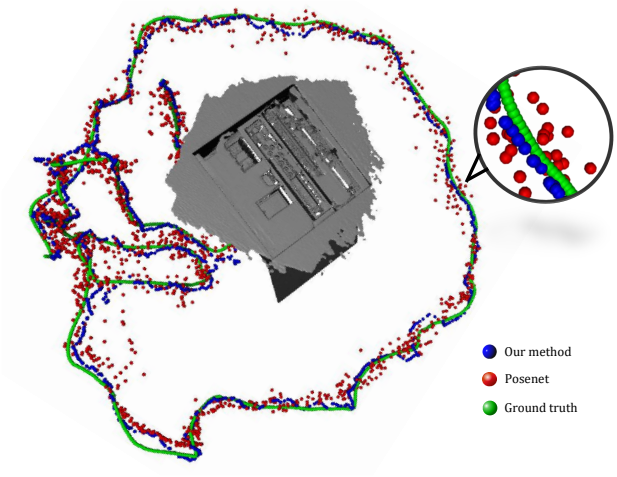


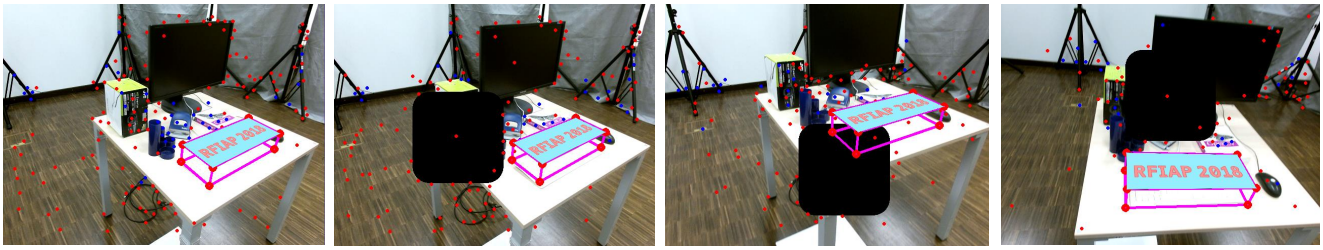
FIGURE 6 – Résultat comparatif entre notre méthode (bleu) et PoseNet (rouge) sur la précision en translation.

La figure 5 présente trois trajectoires résultantes pour l'estimation de la translation. Nous obtenons respectivement les valeurs moyennes $0.04m$ et $1.1°$ pour l'erreur de translation et de rotation. Alors que [9] obtient $0.12m$ et $4.7°$. De plus, sur la figure 6, nous comparons les résultats de notre méthode avec ceux de [9] sur la scène de *eletrical cabinet*. Nos résultats s'avèrent plus stables.

Finalement, nous soulignons un autre avantage intéressant de notre méthode lié à l'utilisation de patchs au lieu d'une image entière. Pour la relocalisation de caméra, la manipulation de scènes non rigides, bien que fréquente, se révèle plus difficile. En effet, lorsque les objets se déplacent à travers une scène, cela provoque des occlusions partielles de la scène. Dans la méthode, nous extrayons un ensemble de patchs pour générer des prédictions sur leur location 3D, sans se préoccuper de savoir s'ils correspondent ou non à des objets en mouvement. Par exemple sur la figure 7, nous extrayons des patchs sur le masque noir qui se déplace dans la scène. C'est en effet l'utilisation de Ransac et PnP qui permet d'éliminer comme outliers les correspondances sur les objets en mouvement tout en conservant les correspondances sur les objets rigides. La pose de la caméra est donc toujours estimée de manière stable, ce qui permet d'appliquer la méthode dans le contexte de la réalité augmentée, comme l'illustre la figure 7.

5 Conclusion

Dans cet article, nous avons proposé une nouvelle méthode hybride combinant l'approche d'apprentissage en profondeur et l'approche géométrique pour la relocalisation de caméra. Nous avons présenté un réseau de neurones convolutifs léger pour définir de manière efficace et robuste les correspondances de pixels 2D dans le système de coordonnées du monde. Un ensemble de résultats probabilistes est généré pour traiter l'incertitude de la régression de l'apprentissage en profondeur dans la relocalisation de caméra.



a) Frame #001 b) Frame #005 c) Frame #050 d) Frame #110

FIGURE 7 – Capacité de traitement d'occlusion partielle et d'application dans la réalité augmentée de notre méthode.

Simultanément, nous exploitons des informations géométriques sur les correspondances 2D-3D pour résoudre le problème de l'occlusion partielle et calculer la pose de la caméra en utilisant les algorithmes Ransac et PnP. Nous considérons également le nombre d'inliers comme score de confiance pour chaque image.

Références

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3) :346–359, 2008.

[2] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen. Vidloc : A deep spatio-temporal model for 6-dof video-clip relocalization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[3] J. Engel, T. Schöps, and D. Cremers. Lsd-slam : Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[4] A. Guzman-Rivera, P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi. Multi-output learning for camera relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1114–1121, 2014.

[5] N. Jiang, Z. Cui, and P. Tan. A global linear method for camera pose registration. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 481–488, 2013.

[6] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *European Conference on Computer Vision*, pages 205–220. Springer, 2016.

[7] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 4762–4769. IEEE, 2016.

[8] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[9] A. Kendall, M. Grimes, and R. Cipolla. Posenet : A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2938–2946, 2015.

[10] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam : a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5) :1147–1163, 2015.

[11] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion : Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[12] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam : Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[13] B. Peasley and S. Birchfield. Rgbd point cloud alignment using lucas–kanade data association and automatic error metric selection. *IEEE Transactions on Robotics*, 31(6) :1548–1554, 2015.

[14] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *2011 International Conference on Computer Vision*, pages 667–674. IEEE, 2011.

[15] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013.

[16] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2) :189–210, 2008.

[17] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam : Real-time dense monocular slam with learned depth prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[18] J. Valentin, M. Nießner, J. Shotton, A. Fitzgibbon, S. Izadi, and P. H. Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4400–4408, 2015.

[19] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[20] O. Wasenmüller, M. Meyer, and D. Stricker. Corbs : Comprehensive rgb-d benchmark for slam using kinect v2. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–7. IEEE, 2016.