Estimation de la pose d'une caméra à partir d'un flux vidéo en s'approchant du temps réel

El Rhabi Youssef[2] Simon Loic[1] Brun Luc[1]

[1] Laboratoire du GREYC
[2] 44screens société d'application mobile et de réalité augmentée 3D

6 Boulevard du Maréchal Juin CS 45053 14050 CAEN cedex 4
{loic.simon, luc.brun}@ensicaen.fr, yer@44screens.com

Résumé

Trouver une méthode permettant de calculer la pose d'une image avec robustesse et rapidité est essentiel en réalité augmentée. Nous allons présenter ici l'approche que nous avons suivie pour nous rapprocher du temps réel en utilisant les points caractéristiques SIFT [4]. Nous proposons de filtrer à la fois les points SIFT mais aussi les images à utiliser, afin de concentrer nos calculs sur les données pertinentes.

Mots Clef

Réalité augmentée,SfM,SLAM, calcul de pose temps réel, recalage 2D/3D

Abstract

Finding a way to estimate quickly and robustly the pose of an image is essential in augmented reality. Here we will discuss the approach we chose in order to get closer to real time by using SIFT points [4]. We propose a method based on filtering both SIFT points and images on which to focus on. Hence we will focus on relevant data.

Keywords

Augmented Reality, SFM,SLAM, real time pose computation, 2D/3D registration

1 Introduction

Le principe de la réalité augmentée (RA) est de fusionner des éléments virtuels avec le monde réel, afin de mieux guider l'utilisateur ou de rendre la scène plus attractive. L'émergence des nouveaux smartphones ou tablettes a considérablement accru l'engouement pour ce type de technologies. En effet, ces terminaux permettent de regrouper facilement les composantes nécessaires à une expérience en RA :

- Une caméra pour filmer la scène
- Un processeur permettant le calcul de la pose
- Un écran pour afficher la scène augmentée

D'un point de vue algorithmique, la problématique principale se situe au niveau du calcul de la pose de la caméra. Cette tâche est rendue plus ardue de par les contraintes temps-réel inhérentes à la réalité augmentée. L'état de l'art dans l'industrie repose sur l'utilisation de marqueurs artificiels afin de déterminer la pose d'une caméra [1]. Cependant, cette technique peut conduire à occulter des parties essentielles de la scène ou encore à dégrader un site culturel ou historique. Le développement d'une technologie de RA sans marqueur s'avère donc indispensable. C'est une chose à laquelle le secteur industriel commence à s'intéresser.

Dans cette optique, deux courants majeurs sont actuellement privilégiés. Une première approche consiste à prendre un modèle 3D identique à l'objet autour duquel nous voulons créer notre scène en RA. Il s'agit ensuite d'utiliser une mesure de similarité entre le modèle 3D et l'objet réel sur l'image afin de déterminer la pose de la caméra [2,3,13]. Pour ce faire, les contours de l'objet sont utilisés comme support principal. L'idée est de projeter l'objet 3D dans l'image et trouver des correspondances entre les contours détectés dans l'image et ceux projetés. Ensuite, en maximisant la similarité entre les contours, on en déduit une approximation de la pose réelle. Cette méthode requiert tout de même de disposer d'un modèle 3D assez précis de l'objet en question, ce qui n'est pas toujours le cas. De plus pour des objets non manufacturés ces contours sont difficiles à extraire.

Pour s'affranchir de ce type de limitations il est possible de prendre en compte la texture des objets dans l'image. Ces textures sont usuellement exploitées afin d'extraire des points d'intérêts. Dans le contexte de la robotique [8], une approche de localisation basée sur les points SIFT [4] est présentée. L'idée est créer une carte 3D hors ligne à partir d'une base d'images en se basant sur la géométrie épipolaire [5]. L'implémentation d'une telle méthode est expliquée dans [7]. Cette construction

de la carte fournit des liens 2D/3D entre les images de la base et la carte 3D. Ensuite en appariant les points 2D de chaque image entrante à ceux de la base, un lien 2D/3D est créé par transitivité. Les liens 2D/3D donnent un système résolu par un algorithme de minimisation tel que Levenberg-Marquardt [14], afin de calculer la pose de l'appareil ayant pris l'image entrante. Il faut toutefois savoir que la contrainte temps réel est plus compliquée en RA, ne serait ce que de par les supports limités en mémoire et en puissance de calcul. Metaio [12] proposent des points caractéristiques nécessitant moins de ressource de calculs que les points SIFT[4]. Leur but est d'obtenir des performances compatibles avec le temps réel. Le manque de robustesse des points caractéristiques utilisés ainsi que celle du tracking expliquent toutefois, en partie, les phénomènes de jittering (vibration du modèle 3D) dont est victime cette méthode dans le cadre de scènes en extérieur.

L'enjeu est donc de trouver une méthode permettant de calculer rapidement la pose des caméras. En plus de la rapidité il faut assurer la robustesse du calcul de la pose en environnement extérieur. Pour ce faire, nous avons utilisé la méthode SIFT[4]. Elle permet d'apparier des points dans différentes images d'une même scène. SIFT est connu pour sa robustesse, mais pas pour sa rapidité. Afin de nous rapprocher autant que possible du temps réel nous avons travaillé sur une approche permettant de réduire le nombre de points à utiliser dans chaque image, mais également de réduire le champ des images que nous utilisons dans la base de donnée construite hors ligne. En effet pour une frame donnée toutes les images de la base ne seront pas utiles. Les utiliser est donc une perte de temps.

La suite de cet article détaille ces différents points et est structurée comme suit. Les détails concernant la formulation du problème de calcul de pose seront exprimés dans 2. Ensuite nous aborderons la mise en place d'optimisation permettant de réduire le temps de calcul des poses dans la section 3. Une fois toutes ces étapes présentées nous montrerons en quoi nos optimisations ont été efficaces. Ce sera l'objet de la section 4.

2 Positionnement du problème

Dans la RA le but est de pouvoir insérer un objet virtuel dans un emplacement donné d'une scène réelle. Nous pouvons résumer ce problème au fait de projeter cet objet dans l'image. Afin de réaliser cela, la connaissance de la pose de la prise de vue par rapport à la scène est nécessaire. Dans la mesure où le calcul de la pose revient à calculer des correspondances 2D/3D, le problème est mal posé. Afin de simplifier ce problème nous construisons en pré-traitement une liste de correspondances 2D/3D en utilisant la structure from motion (SFM). Par la suite, nous pourrons nous contenter de lier des points 2D, d'une image entrante, à ceux du pré-traitement pour retrouver les liens 2D/3D permettant de calculer la pose de cette image.

La méthodologie suivie pour la mise en place du pré-traitement est vue dans la section 2.1. Nous discutons de la démarche suivie pour calculer la pose d'une frame d'une vidéo dans la section 2.2.

2.1 Pré-traitement

À un point 2D de l'image peut correspondre chacun des points singuliers présents sur la droite passant par ce point et le centre optique de la caméra (voir droite d figure 2). Cela en fait un problème mal posé. Pour palier à ce problème, l'idée que nous avons exploitée est de mettre en place un lien entre des points des images 2D et la 3D.

Nous procédons comme suit (les étapes sont illustrées dans la figure 1) :

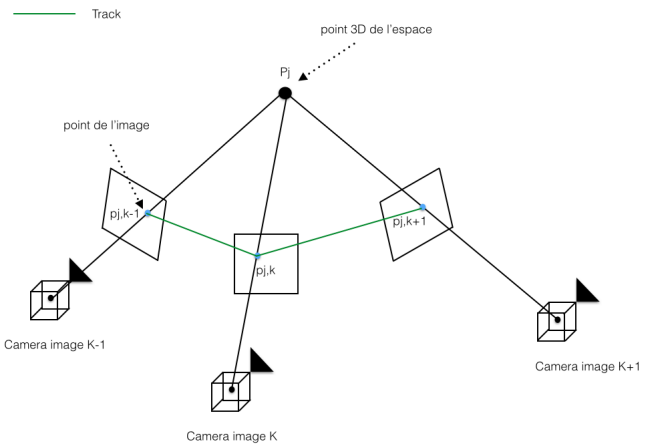

FIGURE 1 – Triangulation de la scène

Nous prenons des photos de l'objet sur lequel nous voulons superposer un élément virtuel. Puis sur ces images nous extrayons des points d'intérêts stables aux changements de pose (points de l'image pj). Nous mettons ces points d'intérêts en correspondance afin de définir des listes de points 2D dans les images correspondant à un même point 3D (tracks en vert). C'est une étape cruciale que nous détaillons à la fin de cette sous section.

Appariement des points 2D des images.

Puisque le processus de prise de photo peut être assimilé

à une projection d'une scène 3D dans un plan 2D, on peut le modéliser selon le modèle sténopé (figure 2). Les smartphones du marché configurés de façon usuelle respectent suffisamment ce modèle.

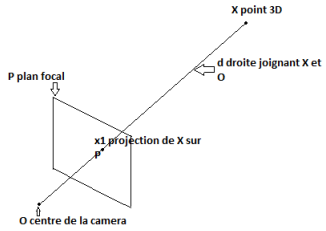

FIGURE 2 – modèle caméra sténopé

Plusieurs stratégies peuvent être utilisées pour apparier efficacement des points 2D dans plusieurs images. Dans [19] l'approche est d'utiliser des méthodes d'apprentissages afin de mettre les points en correspondance. Ils effectuent une sélection d'un certain nombre de points d'intérêt qui sont souvent vus dans des images prises au préalable. Sur cette base d'images nous avons l'information sur les points qui sont en correspondance. Dans une phase d'apprentissage, un classifieur est créé afin de reconnaître une classe donnée. Le classifieur est basé sur un arbre binaire encodant la distribution a posteriori. Chaque classe représente dans les faits une famille de points 2D représentant le même point de l'espace. L'apprentissage se fait sur une base d'image connue. Ensuite pour la phase de mise en correspondance l'idée est de se servir du classifieur afin de savoir si le point testé fait partie d'une classe ou non, et si oui laquelle. Dans [20] la même approche est abordée mais cette fois-ci en s'affranchissant de la structure d'arbre. L'idée est de trouver des sous groupes de classes indépendantes au sens des probabilités. Ils appellent ces sous groupes FERN. Cette approche leur permet de rendre leur algorithme plus rapide encore.

Dans notre approche nous avons besoin d'extraire des points d'intérêt dans des images, puis d'être en mesure d'apparier ces points. Les points SIFT sont connus pour être les plus robustes [18]. Nous comptons vérifier leur compatibilité avec le temps réel en nous basant sur FLANN [21, 22, 23], afin d'estimer le plus proche voisin d'un descripteur.

Nuage de point 3D et liaison avec points 2D.

Le principe consiste à se servir d'une base d'images de la scène prise dans une étape de pré-traitement hors ligne. Dans ces images nous extrayons des points SIFT [4] afin d'y trouver des points caractéristiques. Une fois ces points obtenus, il nous est possible de les mettre en correspondance (section 2.1), nous tombons ainsi dans une géométrie particulière : la géométrie épipolaire.

La géométrie épipolaire permet, par triangulation des points des images appariés, de retrouver des informations sur la configuration 3D de la scène, et de créer un nuage de point 3D. Pour mener à bien cette étape nous nous sommes appuyés sur l'implémentation traitée dans [7], et dont la théorie est amplement expliquée dans [5]. Elle consiste à utiliser SIFT [4] et le procédé de Lowe (section 2.1) pour établir une correspondance entre les points d'intérêt de deux images. On en déduit la matrice fondamentale F qui lie les deux images, en passant par la méthode ACRANSAC [26]. L'utilité de ACRANSAC est de rejeter les points faussement mis en correspondance, tout en fournissant un modèle F. Pour ce faire, il tire aléatoire des échantillons parmi la base de données, en tire un modèle F puis retient le modèle ayant le meilleur consensus. Les méthodes à contrario permettent à partir d'un modèle d'en juger la pertinence, en se basant sur l'idée de discriminer les modèles improbables. L'apport de la méthodologie à contrario à RANSAC [6] est de se concentrer sur les données afin de s'affranchir du paramètrage de RANSAC. A partir de la matrice fondamentale il est possible de déterminer les matrices de projection relatives entre ces deux images. En disposant des matrices de projection nous pouvons, par triangulation, retrouver la position du point 3D correspondant à deux points 2D des images liées. L'ajustement de faisceaux [10] est ensuite utilisé afin de réduire les erreurs de re-projection sur les différentes images. Cela permet d'obtenir la configuration optimale du nuage de points et des caméras. Par conséquent nous avons la position des caméras de la base et du nuage de points 3D (figure 1), ainsi que la calibration à priori de la caméra ayant capturé la scène. Nous nommons track l'ensemble constitué par un point 3D et des points 2D des images de la base qui ont servi à le calculer.

2.2 Calcul de la pose d'une nouvelle image

Notre but est de calculer la matrice de pose P_{frame}=[R | t] de la caméra au moment de la capture de la frame (où R représente la rotation de celle-ci et t sa position). On s'appuie pour cela sur le modèle sténopé (figure 2), et des liens 2D/3D fournis par la section 2.1. L'approche est la suivante : utiliser SIFT pour détecter des points caractéristiques $\{x_1,x_2,...,x_n\}$ dans la frame exprimée en coordonnées rétinales, puis les mettre en correspondance avec l'ensemble des points SIFT des images de la base. Dans notre méthode nous avons évalué la matrice de calibration K lors du prétraitement. Nous supposons pour la suite que nous avons utilisé la même caméra et que la calibration est donc connue. Ensuite nous filtrons les fausses correspon-

dances par le biais de ACRANSAC. Les points des images du pré-traitement étant liés aux points 3D par transitivité on obtient un lien entre un point 2D x_i de l'image entrante (en coordonnées rétinales) et un point 3D (figure 3). La relation liant un point X_i et un point x_i de la frame est pour tout i :

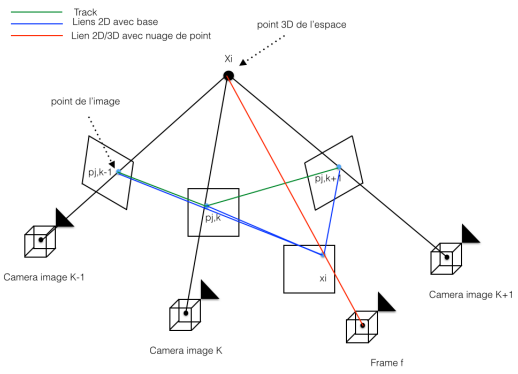

FIGURE 3 – Schéma expliquant la liaison 2D/3D

$$P_{frame}X_i = x_i \tag{1}$$

La matrice P_{frame} est de taille 3 × 4, elle possède donc 12 éléments. Or nous avons ici un système qui possède autant d'équations que de correspondances SIFT calculées (sachant l'équation 1). Notre problème est donc sur contraint. Nous avons donc recours à une méthode de moindres carrés permettant de trouver la matrice P_{frame} répondant au mieux à l'ensemble des contraintes. Nous chercherons donc à minimiser la fonction de coût :

$$costFun(P_{frame}) = \sum_i ||PX_i - x_i||^2 \tag{2}$$

Le problème à résoudre étant convexe, nous le résolvons donc par une méthode de type Levenberg-Marquardt [14].

3 Optimisation des calculs

En utilisant le mode opératoire vu dans 2 nous pouvons prédire avec robustesse la position d'une frame. Par contre ce mode opératoire est très lent. Le temps de calcul de la position d'une frame est de 8s sur un PC dernière génération. Il est donc nécessaire de trouver des moyens d'optimiser les calculs. Pour cela nous avons proposé deux méthodes d'optimisations que nous verrons dans 3.1 et 3.2.

3.1 Filtrage des points SIFT

Dans notre base de donnée, nous avons un nombre de descripteurs conséquent. Réduire le nombre de descripteurs à prendre en compte devient alors intéressant. Il faut tout de même choisir avec soin ceux que l'on décide de garder et ceux que l'on décide de retirer. Il est en effet souligné dans [15] que pour l'appariement de points, les points d'intérêt de plus grande échelle sont de meilleurs candidats pour les raisons suivantes :

- Les points d'intérêt de grande échelle couvrent une plus grande zone de l'image. Ainsi, même en petit nombre, ces points de grande échelle représenteront une partie conséquente de l'image.
- Dans le processus d'appariement de SIFT la structure est telle que les grandes échelles seront appariées entre elles. Cela est dû au fait que la variance des échelles des points caractéristiques appariés est en général basse.

Nous avons donc décidé de filtrer la base d'images prise hors ligne. Nous avons classé, pour chaque image de la base, les points d'intérêt par ordre d'échelle décroissante. Enfin nous en avons gardé un pourcentage fixé parmi les plus grandes échelles.

Partant toujours de cette idée, pour gagner du temps lors du calcul des points SIFT et descripteurs pour l'image entrante, nous décidons de ne calculer que ceux de grandes échelles en ne les cherchant que dans les octaves élevées. Dans notre cas le calcul se fait sur 3 octaves, nous ne chercherons les points d'intérêts que dans les octaves 2 et 3.

3.2 Voisinage d'une image de la base

Afin de pouvoir continuer dans nos optimisations, nous avons besoin de créer une structure permettant de lier les images de la base les unes aux autres. Pour cela, nous nous sommes penchés sur la théorie des graphes qui nous donne une base idéale.

Soit G=(V,E,p) un graphe orienté pondéré composé d'un ensemble V=$\{i_1, i_2, ..., i_n\}$ de n noeuds et E $\subset$ V $\times$ V l'ensemble des arêtes pondérées. Une arête (i,j) $\in$ E code la similarité entre deux noeuds voisins et est pondérée par la fonction de poids p : E $\rightarrow \mathbb{R}^+$.

Dans notre cas, chaque noeud représente une image et chaque arête relie deux images voisines. Pour notre notion de poids nous avons choisi de calculer le nombre de tracks (voir section 2.1) que les images partagent. Dans notre représentation, nous poserons T = $\{t_1,t_2,...,t_m\}$ l'ensemble des m tracks créées. Considérons f : E $\times$ T $\rightarrow$ $\mathbb{R}^+$ telle que :

$$\begin{aligned} &\text{soit k} \in \{1,...,\text{m}\} \\ &\forall\,(i,j), tk \in \text{E} \times \text{T} \\ &f(i,j,t_k) = \begin{cases} 1 \text{ si } i \text{ et } j \text{ sont dans } t_k \\ 0 \text{ sinon} \end{cases} \end{aligned} \tag{3}$$

Nous posons notre fonction de poids ainsi :

$$p(i_1, i_2) = \sum_{k=1}^{m} f(i_1, i_2, t_k) \quad (4)$$

Nous choisissons comme voisins de notre noeud i_r les noeuds i_{l1} et i_{l2}, qui sont les deux noeuds qui ont le plus de tracks communes avec i_r. Cela nous donne une notion de voisinage basée sur la mise en correspondance de points SIFT. Si une frame prise en ligne est fortement liée à i_r, il y a de fortes chances qu'elle le soit aussi à ses voisins.

3.3 Filtrage des images de la base

Lors d'une acquisition vidéo, pour une image f_k suite au filtrage de ACRANSAC, seul un nombre limité d'images $SI_k \subset$ V sont utiles au calcul de sa pose. Dans cet esprit, nous avons décidé de limiter la recherche aux images qui semblent les plus pertinentes.

Pour une frame f_k, k$\in$ [1,p], nous voulons avoir la liste SI_k des images qui seront susceptibles de fournir le plus de bonnes correspondances. Pour sélectionner ces images, nous partons de l'idée que d'une frame à l'autre les points de vue sont proches. Posons E_{k-1} l'ensemble des images de la base de donnée fournissant de bonnes correspondances avec f_{k-1}. Posons aussi N(E_{k-1}) l'ensemble des voisins de chaque image présente dans E_{k-1}. Alors pour f_k, nous chercherons les correspondances des points d'intérêts avec les éléments de E_{k-1} ainsi que N(E_{k-1}). Ce qui revient à dire que SI_k=$\{E_{k-1} \cup$ N(E_{k-1})$\}$ (voir figure 4).

Le gain de temps de cette approche est multiple. D'une part cela réduit le temps de mise en correspondance puisque nous avons éliminé les candidats qui n'offrent pas de bonnes correspondances. Nous gagnons aussi du temps au niveau du filtrage géométrique lié à ACRANSAC voir section 2.2. Dans les cas où l'image avec laquelle f_k est comparée n'est pas pertinente ACRANSAC ne trouve pas de solution. Alors il tire plus d'échantillons aléatoirement parmi la base. Le temps mis à filtrer les correspondances pour une image non pertinente est donc considérablement plus long.

4 Résultats

Nous avons éprouvé la robustesse de notre approximation en utilisant la base de données de Strecha [17]. Nous avons utilisé la fontaine, constituée de 11 images. Pour chaque image, nous avons la vérité terrain, c'est à dire la matrice de projection réelle de l'appareil l'ayant capturée. Nous avons également créé notre propre jeu de données constitué de 41 images. Sur cette dernière base, nous pouvons tester nos optimisations et prendre des séquences vidéos à notre guise. Le nuage de point est composé d'environ 6000 à 12000 tracks dans notre cas. Pour une image entrante, nous avons de 1000 à 5000 points mis en correspondance.

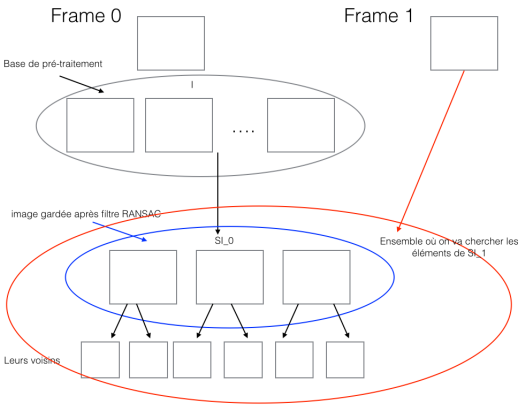

FIGURE 4 – schéma représentant le filtrage des images de la base.

4.1 Test de la robustesse de nos estimations sans optimisation

La base de données de Strecha [17] nous fournit une vérité terrain avec une série d'images et leurs poses. Nous nous servons de cette base pour tester la robustesse de nos estimations.

Nous avons mis une image de coté I_i, et nous avons utilisé les 10 autres comme de la base de pré-taitement. Sur cette image I_i nous nous sommes servi de notre implémtentation pour calculer P_{est}, la matrice de projection estimée. La vérité terrain nous donne la vraie matrice de projection P_{VT}. Pour vérifier la précision de nos calculs, nous avons projeté l'ensemble des points $\{X_1,X_2,...,X_n\}$ du nuage sur le plan focal de l'image. Cela se représente comme suit :

$$\begin{aligned} &\forall i \in \{1, ..., n\} \\ &x_{VT_i} = P_{VT} X_i \\ &x_{est_i} = P_{est} X_i \end{aligned} \quad (5)$$

Pour tout i appartenant à $\{1,...,n\}$ nous avons calculé l'erreur de reprojection e_i de la façon suivante :

$$e_i = ||x_{VT_i} - x_{est_i}||^2 \quad (6)$$

Nous avons ainsi une mesure de la précision de nos calculs. L'erreur de reprojection moyenne est de 0.6252 en pixel. Nous avons dressé un histogramme des erreurs de reprojection (figure 5). Nous constatons que, pour l'ensemble des points, elle est inférieure à l'ordre du pixel.

4.2 Test de la robustesse de nos estimation avec optimisations

Nous allons dans un premier temps décrire la méthodologie de test suivie. Ici nous avons pris un ensemble de 10 frames consécutives (chacune de résolution 5312x2988) et avons calculé la pose de chaque frame sous plusieurs conditions :

- Nous avons estimé la pose de la frame f_k en conservant 100%, 80%, 40%, 20%, 10% des points SIFT comme vu dans la section 3.1

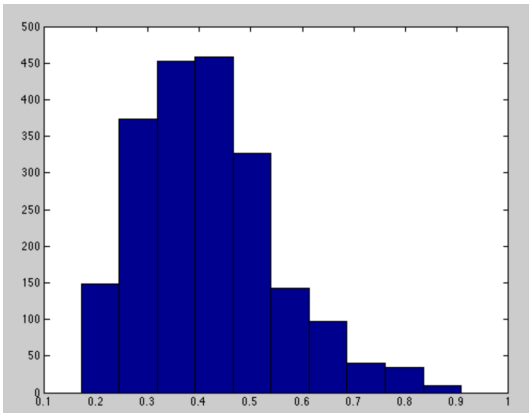

FIGURE 5 – histogramme de la répartition des erreurs avec 100% des points SIFT et se servant de toute la base de donnée. En abscisse : l'erreur de reprojection. En ordonnée : nombre d'éléments dans cette classe

- Pour chacun des seuils, nous avons calculé la pose en prenant respectivement toutes les images de la base ou bien en ne prenant que les images jugées pertinentes par notre méthode comme vu dans 3.3

Une fois tous les calculs effectués, nous comparons les résultats trouvés dans le cas où 100% des points SIFT sont conservés et en utilisant toute la base de données. C'est l'approche que nous avons validée dans la section 4.1 avec la base d'images de Strecha. Un tableau comparatif de ces différents tests est visible table 1. Nos calculs sont effectués sur macbook pro processeur 2.2GHz intel core i7 16 GB de ram. En terme de vitesse de calcul, le détail des temps figure dans la table 2. La table 2 référence les temps obtenus en modifiant le nombre de points SIFT, respectivement en utilisant toute la base ou bien simplement les images pertinentes.

L'analyse des tableaux 1 et 2 nous indique qu'utiliser 20% des points SIFT, en combinant avec un choix cohérent des images que nous sélectionnons, nous offre les meilleurs résultats. Nous atteignons un seuil lorsqu'il n'y a plus assez de points. En effet, le filtrage d'ACRANSAC trouve moins facilement un consensus parmi les données.

Dans la figure 6 nous pouvons observer la reprojection des points 3D sur les images et le recalage effectué. Pour chaque image, nous avons utilisé la matrice de pose estimée afin de reprojeter le nuage de points dessus.

5 Conclusion

Dans cet article, nous nous sommes intéressés au calcul de la pose d'une prise de vue dans le contexte de la réalité augmentée. Dans ce cadre, la rapidité des traitements est primordiale, mais elle ne doit pas mettre en péril la robustesse des résultats obtenus. Pour répondre à cette problématique, nous avons d'abord formulé le problème en déportant la tâche de plus grande complexité, à savoir les

Pourcentage de points SIFT		100%	80%	40%	20%	10%
Temps de convergence	Base de donnée complète	9.77 s	7.29 s	3.81 s	1.93 s	1.01 s
	images pertinentes	1.69 s	1.36 s	0.78 s	0.46 s	0.31 s
Précision	Base de donnée complète	0 px	2.5 px	3.6 px	4.1 px	3.9 px
	images pertinentes	2.65 px	3.2 px	3.8 px	3.0 px	4.9 px

TABLE 1 – table mettant en avant les performances constatées suite aux optimisations. La première ligne indique le nombre de points SIFT conservés. La seconde le temps mis pour opérer le calcul de la pose, la troisième l'erreur de reprojection moyenne calculée vis à vis de la pose calculée dans le cas où 100% des points SIFT sont conservés et toutes les images de la base de donnée sont utilisées. Le temps et l'erreur de reprojection sont considérés respectivement en utilisant toute la base ou bien simplement les images pertinentes.

SIFT conservés		mise en pair	filtre ACRANSAC	calcul de la pose
100%	Base de donnée complète	8.43s	1.42s	0.027s
	images pertinentes	1.52s	0.13s	0.027s
80%	Base de donnée complète	6.63s	0.63s	0.023s
	images pertinentes	1.20s	0.14s	0.028s
40%	Base de donnée complète	3.37s	0.41s	0.023s
	images pertinentes	0.62s	0.13s	0.024s
20%	Base de donnée complète	1.68s	0.24s	0.022s
	images pertinentes	0.31s	0.13s	0.023s
10%	Base de donnée complète	0.84s	0.47s	0.016s
	images pertinentes	0.15s	0.13s	0.022s

TABLE 2 – table mettant en avant les performances constatées suite aux optimisations. La première colonne indique le nombre de points SIFT conservés. La seconde le temps mis pour calculer les correspondances, la troisième le temps nécessaire pour opérer le filtre ACRANSAC, la quatrième celui pour calculer la pose.

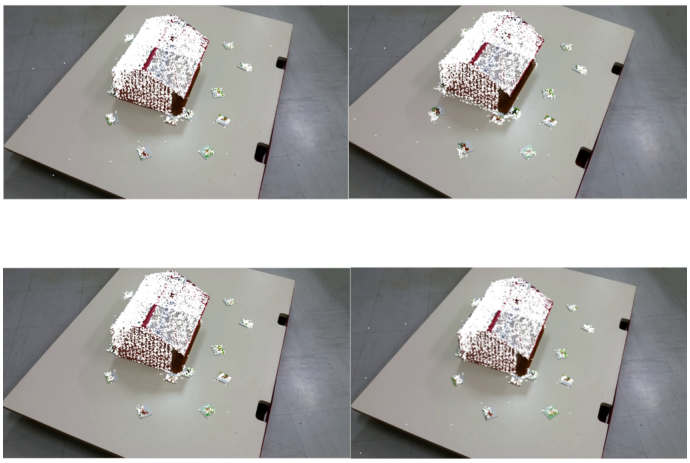

FIGURE 6 – en haut à gauche : reprojection en gardant 100% des points avec toute la base. En haut à droite : reprojection en gardant 10% des points en ne gardant que les images pertinentes. En bas à gauche : reprojection en gardant 20% des points avec toute la base. En bas à droite : reprojection en gardant 20% des points en ne gardant que les images pertinentes.

correspondances 2d-3d, dans une étape de pré-traitement. Nous avons de plus mis en place des structures permettant de réduire le calcul de la pose d'une image, tout en conservant une approche basée sur des points SIFT. Nous sommes ainsi passés de 9.77 s à 0.46 s pour la totalité des étapes du calcul de la pose d'une image. La perte en précision occasionnée par cette optimisation n'est que de 3 pixels sur une image de résolution 5312x2988. Néanmoins, plusieurs approches restent à explorer pour améliorer encore les résultats. On pourra envisager entre autre le tracking via un filtrage de Kalman. Par ailleurs, la réduction du temps d'initialisation étant également essentielle, nous envisageons par la suite d'utiliser notre structure de graphe en passant par des structures hiérarchiques [16].

Références

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

[2] C. Wiedemann, M. Ulrich, and C. Steger, *Recognition and tracking of 3D objects, Lect. Notes Comput. Sci.,* vol. 5096, pp. 132-141, 2008.

[3] Lima, J. P., Simões, F., Figueiredo, L., Kelner, J. (2010). *Model based markerless 3D tracking applied to augmented reality. Journal on 3D Interactive Systems, 1.*

[4] Lowe, D. G. (1999). Object recognition from local scale-invariant features. *In Computer vision, 1999. The proceedings of the seventh IEEE international conference on (Vol. 2, pp. 1150-1157).*

[5] Hartley, R., Zisserman, A. (2000). Multiple view geometry in computer vision (Vol. 2). Cambridge.

[6] Fischler, M. A., Bolles, R. C. (1981). Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. . *Communications of the ACM, 24(6), 381-395.*

[7] Moulon Pierre, Monasse Pascal and Marlet Renaud. ACCV 2012. Adaptive Structure from Motion with a contrario model estimation *In Computer Vision ?ACCV 2012 (pp. 257-270). Springer Berlin Heidelberg.*

[8] Se, S., Lowe, D. G., Little, J. J. (2005). Vision-based global localization and mapping for mobile robots. *Robotics, IEEE Transactions on, 21(3), 364-375*

[9] Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., Schmalstieg, D. (2008, September). Pose tracking from natural features on mobile phones *In Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality (pp. 125-134). IEEE Computer Society.*

[10] Lourakis, M. I., Argyros, A. A. (2009). SBA : A software package for generic sparse bundle adjustment *ACM Transactions on Mathematical Software (TOMS), 36(1), 2.*

[11] Ceres-solver http ://ceres-solver.org/license.html

[12] Kurz, D., Meier, P. G., Plopski, A., Klinker, G. (2014). Absolute spatial context-aware visual feature descriptors for outdoor handheld camera localization. *In International conference on computer vision theory and applications (pp. 36-42).*

[13] M. Armstrong and A. Zisserman. Robust object tracking. . *In Proc. Asian Conference on Computer Vision, volume I, pages 58,61, 1995.*

[14] Moré, J. J. (1978). The Levenberg-Marquardt algorithm : implementation and theory. . *In Numerical analysis (pp. 105-116). Springer Berlin Heidelberg.*

[15] Wu, C. (2013, June). Towards linear-time incremental structure from motion. *In 3DTV-Conference, 2013 International Conference on (pp. 127-134).*

[16] BRUN, L., KROPATSCH, W., Hierarchical Graph Encodings. *In Image processing and analysis with graphs : theory and practice, 2012 CRC Press.*

[17] Strecha, C., von Hansen, W., Van Gool, L., Fua, P., Thoennessen, U. (2008, June). On benchmarking camera calibration and multi-view stereo for high resolution imagery. *In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on (pp. 1-8).*

[18] Mikolajczyk, K., Schmid, C. (2005). A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 27(10), 1615-1630.*

[19] Lepetit, V., Lagger, P., Fua, P. (2005, June). Randomized trees for real-time keypoint recognition. *In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Vol. 2, pp. 775-781).*

[20] Ozuysal, M., Calonder, M., Lepetit, V., Fua, P. (2010). Fast keypoint recognition using random ferns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 32(3), 448-461.*

[21] Marius Muja and David G. Lowe : "Scalable Nearest Neighbor Algorithms for High Dimensional Data". *Pattern Analysis and Machine Intelligence (PAMI), Vol. 36, 2014.*

[22] Marius Muja and David G. Lowe : "Fast Matching of Binary Features". *Conference on Computer and Robot Vision (CRV) 2012.*

[23] Marius Muja and David G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration" *in International Conference on Computer Vision Theory and Applications (VISAPP'09), 2009*

[24] Moisan, L., Stival, B. : A probabilistic criterion to detect rigid point matches between two images and estimate the fundamental matrix. *Int. J. of Computer Vision (IJCV) 57 (2004) 201-218*

[25] Julien Rabin, Julie Delon, Yann Gousseau, Lionel Moisan, et al. Mac-ransac : a robust algorithm for the recognition of multiple objects. *Proceedings of 3DPTV, 2010.*

[26] Moisan, Lionel, Pierre Moulon, and Pascal Monasse, Automatic homographic registration of a pair of images, with a contrario elimination of outliers. *Image Processing On Line 10 (2012)*