Une étude comparative de sept solutions du problème P-n-P

A comparative study of seven solutions to the P-n-P problem

A. Pagani D. Stricker

Fraunhofer Institut
für Graphische Datenverarbeitung

Fraunhoferstraße 5
D-64283 Darmstadt
Allemagne
stricker@igd.fhg.de

Résumé

Dans le cadre de l'augmentation d'images avec des objets virtuels, nous étudions le problème classique de calcul de pose à partir de n correspondences de points 2D/3D aussi connu sous le nom "problème P-n-P". En particulier, nous présentons sept solutions à ce problème. Ces solutions sont évaluées au moyen de simulations afin de tester leur robustesse. Nous analysons les résultats obtenus de manière à faciliter le choix d'une solution particulière.

Mots Clés

Calcul du point de vue, Réalité Augmentée, Problème P-n-P

Abstract

We study the classical problem of viewpoint computation from n 2D/3D point correspondences (P-n-P problem) for image augmentation with virtual objects. In particular, we present seven solutions to this problem. We evaluate these solutions with simulations in order to test their robustness. We analyze the obtained results to make the choice of a particular solution easier.

Keywords

Viewpoint computation, Augmented Reality, P-n-P problem

1 Introduction

La Réalité Augmentée est une technique permettant d'ajouter des éléments informatifs non disponibles *a priori* dans le champ perceptuel d'un utilisateur. Les applications recouvrent la médecine, l'architecture, le jeu, les effets spéciaux. Dans le cas de la réalité augmentée appliquée à la vision, des objets virtuels sont incrustés dans des images réelles de manière à correspondre aux contraintes imposées par la camera et la scène réelle, à savoir la distance focale, la position et l'orientation dans la scène, mais aussi la luminosité et la réflexion de la lumière entre les objets. Les objets virtuels doivent en particulier être générés à partir d'une caméra virtuelle aux paramètres identiques à ceux de la caméra réelle lors de la prise de vue. Si l'on considère le modèle de caméra en trou d'épingle, les paramètres à retrouver sont les paramètres internes, *ie.* la distance focale et le point principal, et les paramètres externes *ie.* la position et l'orientation de la caméra. Dans cet article, nous nous intéressons au problème de détermination des paramètres externes de la caméra dans le cas de prises de vues uniques, c'est-à-dire lorsqu'une image seulement est disponible. Nous supposons dans le restant de l'article que la caméra est calibrée, c'est-à-dire que ses paramètres internes sont connus.

1.1 Calcul de pose dans des vues uniques

L'estimation des paramètres externes de la caméra est également connue sous le nom de calcul de pose. Celui-ci revient à déterminer la matrice de rotation $\boldsymbol{R}$ et le vecteur de translation $\boldsymbol{t}$ décrivant le passage d'un repère lié à la scène au repère de la caméra.
Différentes méthodes ont été proposées pour le calcul de pose dans le cas des points de vues uniques. [2] utilise trois points de fuite correspondant à des directions orthogonales. Ces points de fuite peuvent être détectés à l'aide des images de lignes parallèles. Cette méthode permet de déterminer complètement $\boldsymbol{R}$ et le vecteur $\boldsymbol{t}$ est retrouvé à un facteur près. Une autre méthode, décrite par [12], fait usage des homographies entre un plan réel et son image. Une extension de cette technique, proposée par [11], permet de retrouver la pose très facilement en

détectant un rectangle sur un plan de référence. Si ces méthodes fournissent des résultats rapides et sont efficaces lorsque l'image contient les éléments nécessaires (lignes parallèles, plans), elles ne sont pas utilisables lorsque ces informations sont manquantes.
Un cas de figure plus fréquent est celui où l'information est donnée sous forme d'un jeu de correspondances entre des points, chacune étant composée des coordonnées 3D d'un point réel exprimées dans le repère de la scène et des coordonnés 2D de son image. Fishler et Bolles ont été les premiers à mentionner dans [6] le problème de *Perspective en n Points* (problème P-n-P) comme le problème consistant à trouver la pose d'une caméra à l'aide de n correspondances. Des solutions au problème P-3-P sont données par [7] et [3]. Cependant, pour 3 points, les algorithmes doivent résoudre des équations de degré 4. Ils sont donc généralement lents et ne fournissent pas de solution unique. A partir de 4 points, la solution est unique, et il existe des solutions linéaires [10].
Dans cet article, nous étudions sept solutions proposées au problème P-n-P, pour $n \geq 4$. Les méthodes de résolutions sont diverses et si elles fournissent toutes des solutions exactes en absence de bruit, leurs robustesses sont inégales, ainsi que leurs complexités.

1.2 Organisation de l'article

Nous rappelons en section 2 les équations fondamentales mises en jeu, ainsi que le problème d'orientation absolue. La section 3 présente les différentes solutions étudiées. La méthode utilisée pour les tests comparatifs est décrite dans la section 4, et les résultats expérimentaux sont présentés dans la section 5.

2 Equations de base

Dans la suit de cet article, nous utilisons le modèle de caméra en trou d'épingle, qui associe à un point $\boldsymbol{X}_i$ de la scène le point image $\boldsymbol{x}_i$ tel que

$$\boldsymbol{x}_i \sim \boldsymbol{P}\boldsymbol{X}_i \sim \boldsymbol{K}\left[\begin{array}{cc}\boldsymbol{R} & \boldsymbol{t}\end{array}\right]\boldsymbol{X}_i. \qquad (1)$$

$\boldsymbol{X}$ est un vecteur inhomogène de dimension 4 et $\boldsymbol{x}$ est un vecteur homogène de dimension 3. $\boldsymbol{P}$ est la matrice de projection perspective dans l'image, $\boldsymbol{K}$ la matrice des paramètres internes de la caméra (supposée connue), et $\left[\begin{array}{cc}\boldsymbol{R} & \boldsymbol{t}\end{array}\right]$ la matrice du point de vue que nous recherchons.
L'équation homogène (1) est l'équation de base du problème P-n-P, car elle fait intervenir directement les données du problème ($\boldsymbol{x}_i$ et $\boldsymbol{X}_i$) et les inconnues ($\boldsymbol{R}$ et $\boldsymbol{t}$). En notant $\tilde{\boldsymbol{x}_i} = \boldsymbol{K^{-1}}\boldsymbol{x}_i$, on obtient l'équation homogène :

$$l_i\tilde{\boldsymbol{x}_i} = \boldsymbol{R}\boldsymbol{X}_i + \boldsymbol{t} \qquad (2)$$

Si de plus $\tilde{\boldsymbol{x}_i}$ est normalisé de façon à avoir $\|\tilde{\boldsymbol{x}_i}\| = 1$, l_i représente la longueur du centre de la caméra au point $\boldsymbol{X}_i$.
En supposant que la longueur l_i est connue pour chaque jeu de points $(\boldsymbol{x}_i, \boldsymbol{X}_i)$, le vecteur $l_i\boldsymbol{K^{-1}}\boldsymbol{x}_i$ représente les coordonnées du point $\boldsymbol{X}_i$ dans le repère lié à la caméra. Le calcul de pose se réduit alors au problème d'*orientation absolue*, consistant à retrouver la transformation entre deux repères à partir de mesures des coordonnées de points dans chacun des repères. Ce problème possède une solution à partir de 3 points, dont la démonstration dépasse le cadre de cet article et peut être trouvée dans [8]. Ainsi certaines des solutions au problème P-n-P s'attachent à déterminer en premier lieu les longueurs l_i.
Nous supposons maintenant connues les coordonnées de n jeux de points $(\boldsymbol{x}_i, \boldsymbol{X}_i)$. Dans la sections suivante, nous présentons les solutions retenues au problème P-n-P.

3 Solutions proposées

Nous décrivons maintenant sept solutions au problème P-n-P. Quatre d'entre elles sont linéaires, trois sont des solutions itératives consistant à minimiser une erreur.

3.1 Transformation linéaire directe

L'algorithme de transformation linéaire directe (DLT) utilise l'équation (1). L'égalité homogène dans cette équation implique la nullité du produit vectoriel des deux composantes de l'équation. On obtient ainsi pour chaque jeu de points trois équations linéaires dans les éléments de $\boldsymbol{R}$ et $\boldsymbol{t}$, dont deux sont linéarement indépendantes. Six points fournissent un système de 12 équations pour 12 inconnues et permettent de retrouver entièrement $\boldsymbol{R}$ et $\boldsymbol{t}$.

3.2 Matrice de pondération

Fiore utilise dans [5] les équations de type (2) et une matrice de pondération. Nous définissons la

matrice de données

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{X}_1 & \dots & \boldsymbol{X}_n \\ 1 & \dots & 1 \end{bmatrix}$$

. $\boldsymbol{D}$ a un rang de 4 si $n \geq 4$ et si les points ne sont pas coplanaires. La matrice de pondération $\boldsymbol{W}$ est la matrice de dimensions $n \times (n-4)$ dont les vecteurs colonnes forment une base orthonormale du noyau de $\boldsymbol{D}$. On observe alors les égalités suivantes :

$$\sum_{i=1}^{n} w_{ij} \boldsymbol{X}_i = \boldsymbol{0}$$

$$\sum_{i=1}^{n} w_{ij} = 0$$

$$\sum_{i=1}^{n} w_{ij}^2 = 1$$

On peut alors former $(n-4)$ combinaisons linéaires des équations de type (2), en utilisant la $j$ème colonne de $\boldsymbol{W}$ pour la $j$ème combinaison :

$$\begin{array}{rcl} \sum_{i=1}^{n} w_{ij} l_i \tilde{\boldsymbol{x}}_i & = & \sum_{i=1}^{n} w_{ij} (\boldsymbol{R}\boldsymbol{X}_i + \boldsymbol{t}) \\ & = & \boldsymbol{R} \sum_{i=1}^{n} w_{ij} \boldsymbol{X}_i + \boldsymbol{t} \sum_{i=1}^{n} w_{ij} \\ & = & \boldsymbol{R}\boldsymbol{0} + \boldsymbol{t}.0 \\ & = & \boldsymbol{0} \end{array}$$

On obtient ainsi $3(n-4)$ équations linéaires pour les n longueurs l_i. Le système est résolvable à partir de 6 points. Une fois les longueurs l_i déterminées, on trouve $\boldsymbol{R}$ et $\boldsymbol{t}$ en résolvant le problème d'orientation absolue.

3.3 Equation de rigidité

La rigidité de la scène peut être exprimée en termes d'invariance de la distance entre les points. En effet, la distance connue $d_{ij} = \|\boldsymbol{X}_i - \boldsymbol{X}_j\|$ est la même dans les deux répères (scène et caméra) et induit une contrainte sur les distances l_i et l_j :

$$d_{ij}^2 = l_i^2 + l_j^2 - 2 l_i l_j \cos\theta_{ij} \qquad (3)$$

où θ_{ij} est l'angle entre les deux rayons (voir figure 1).
Le cosinus de l'angle est obtenu directement à partir des points $\boldsymbol{x}_i$ et $\boldsymbol{x}_j$ et de la matrice $\omega = \boldsymbol{K}^{-\top}\boldsymbol{K}^{-1}$:

$$\cos\theta_{ij} = \frac{\boldsymbol{x}_i^\top \omega \boldsymbol{x}_j}{\sqrt{\boldsymbol{x}_i^\top \omega \boldsymbol{x}_i}\sqrt{\boldsymbol{x}_j^\top \omega \boldsymbol{x}_j}}$$

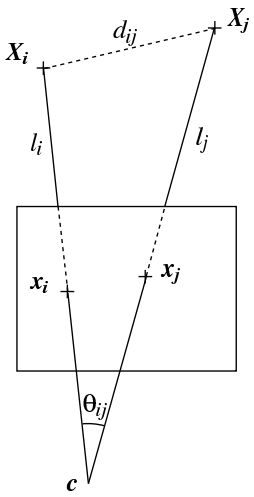


FIG. 1 – Projection de deux points

Pour n points, $\frac{n(n-1)}{2}$ équations du second degré en l_i de type (3) peuvent être déduites. Les deux méthodes suivantes résolvent ces équations de manière linéaire afin de se ramener au problème d'orientation absolue.

Résultants. Quan et Lan montrent dans [10] comment utiliser le résultat classique des résultants de Sylvester pour obtenir une équation de degré 4 en l_1^2 à partir de trois équations de type (3). On arrive ainsi à un système de $\frac{(n-1)(n-2)}{2}$ équations de degré 4 en l_1^2 que l'on peut résoudre linéairement pour les variables indépendantes $l_1^{2j}, j = 1...4$ à partir de $n = 5$ points. Le cas de $n = 4$ est également solvable en réimposant des égalités quadriques entre les variables que l'on supposait indépendantes. Ayant trouvé l_1, on répète le processus n fois pour trouver successivement les autres longueurs.

Contraintes quadratiques. Dans [1], Ansar et Daniilidis résolvent le système des équations de type (3) linéairement pour les $\frac{n(n+1)}{2}$ variables $l_{ij} = l_i l_j$, supposées indépendantes. Le nombre d'équations étant inférieur à celui des variables, on obtient un espace de solutions à $n+1$ dimensions. La solution réelle est retrouvée dans cet espace en réimposant les contraintes quadratiques de type $l_{ab} l_{cd} = l_{a'b'} l_{c'd'}$ pour la permutation d'entiers $(a,b,c,d) \mapsto (a',b',c',d')$. Ceci permet de trouver les longueurs l_i, au moyen d'un résolution d'un nouveau système de

$(\frac{n^2(n-1)}{2})$ équations linéaires de $(\frac{(n+1)(n+2)}{2})$ variables. Cette méthode permet de trouver la pose en 4 points.

3.4 Algorithmes itératifs

Les algorithmes itératifs définissent en général des fonctions d'erreur réflectant la distance à la solution correcte, et cherchent à minimiser cette erreur par un procédé itératif. Nous décrivons ces algorithmes au moyen de leurs fonctions d'erreur.

Erreur dans l'image. La première méthode consiste à minimiser l'erreur de reprojection dans l'image. A partir de valeurs initiales de $\boldsymbol{R}$ et $\boldsymbol{t}$, un point image est calculé et on mesure la distance de ce point au point réel de l'image (voir figure 2). L'erreur est donc définie par

$$e(\boldsymbol{R},\boldsymbol{t}) = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{x}'_i\|^2$$

Les paramètres de la pose sont ensuite optimisés par un procédé itératif comme Levenberg-Macquart.

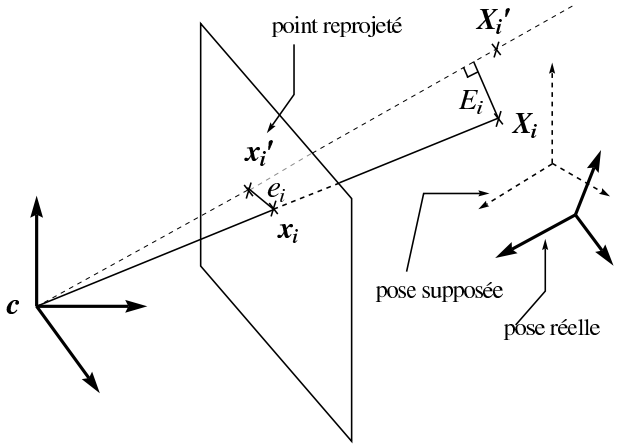


FIG. 2 – Erreur dans l'image et erreur 3D

Erreur 3D. Lu, Hager and Mjolsness ont présenté un algorithme itératif qui minimise l'erreur 3D (voir figure 2). Cette erreur est définie par

$$E(\boldsymbol{R},\boldsymbol{t}) = \sum_{i=1}^{n} \|(\boldsymbol{I} - \boldsymbol{V}_i)(\boldsymbol{R}\boldsymbol{M}_i + \boldsymbol{t})\|^2$$

où $\boldsymbol{V}_i$ est la matrice de projection sur la ligne de vue, définie par

$$\boldsymbol{V}_i = \frac{\tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^{\top}}{\tilde{\boldsymbol{x}}_i^{\top} \tilde{\boldsymbol{x}}_i}$$

Le problème est alors analytiquement similaire à un problème d'orientation absolue où les points image dépendraient de $\boldsymbol{R}$. On résout donc itérativement : à l'étape k, $\boldsymbol{R}^{(k)}$ permet de calculer les points image fictifs, et l'algorithme d'orientation absolue est utilisé pour calculer $\boldsymbol{R}^{(k+1)}$. Plus de précisions peuvent être trouvées dans [9].
Ce procédé est globalement convergent, mais il nécessite des bonnes valeurs initiales pour ne pas converger vers un point fixe autre que la solution.

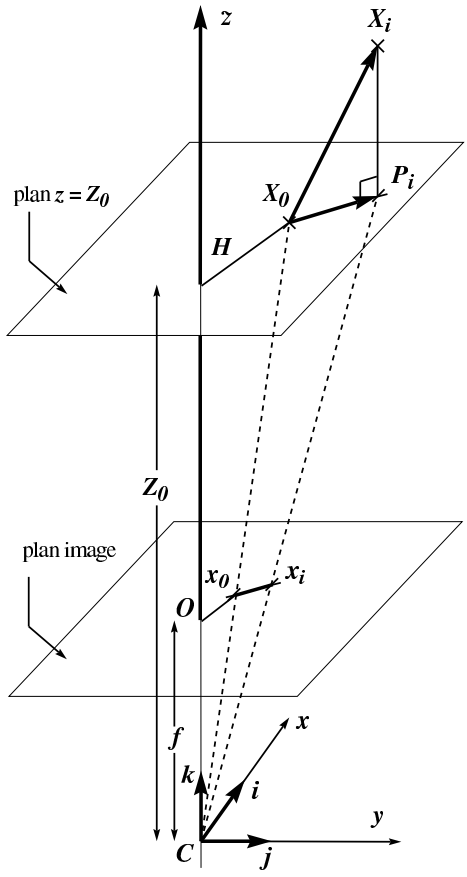


FIG. 3 – Le modèle de projection orthogonale mis à l'échelle

POSIT. L'algorithme POSIT, developpé par DeMenthon et Davis, utilise le modèle de projection orthogonale mise à l'échelle [4]. Dans ce modèle, les points de la scène sont soumis d'abord à une projection orthogonale parallèle à l'axe de la caméra sur un plan d'équation $z = s$, puis à une projection perspective sur le plan image. Dans la figure 3, s est la profondeur Z_0 du premier point $\boldsymbol{X}_0$.

Avec les notations de la figure, on peut montrer géométriquement les égalités suivantes :

$$\boldsymbol{M_0M_i}.\boldsymbol{I} = x_i(1+\epsilon_i) - x_0 \quad (4)$$

$$\boldsymbol{M_0M_i}.\boldsymbol{J} = y_i(1+\epsilon_i) - y_0 \quad (5)$$

où (x_i, y_i, f) sont les coordonnées de $\boldsymbol{x}_i$ par rapport au centre de la caméra, et :

$$\boldsymbol{I} = \frac{f}{Z_0}\boldsymbol{i}, \boldsymbol{J} = \frac{f}{Z_0}\boldsymbol{j} \quad (6)$$

$$\epsilon_i = \frac{1}{Z_0}\boldsymbol{M_0M_i}.\boldsymbol{k} \quad (7)$$

Dans un premier algorithme (POS), des valeurs sont données aux ϵ_i dans les équations (4) et (5), induisant des équations linéaires en les éléments de $\boldsymbol{I}$ et $\boldsymbol{J}$. En résolvant ces équations et en normalisant $\boldsymbol{I}$ et $\boldsymbol{J}$, on trouve $\boldsymbol{i}$ et $\boldsymbol{j}$, qui sont les deux premiers vecteurs lignes de la matrice $\boldsymbol{R}$, et Z_0, qui conduit à $\boldsymbol{t}$. La troisième ligne de $\boldsymbol{R}$ est $\boldsymbol{k} = \boldsymbol{i} \times \boldsymbol{j}$. Lorsque tous les ϵ_i ont pour valeur 0, on est dans le cas du modèle classique de *perspective faible*.
L'algorithme itératif POSIT prend $\epsilon_i = 0$ comme valeurs initiales et POS est appliqué. A chaque étape, une pose est trouvée, ce qui permet de calculer de nouvelles valeurs des ϵ_i avec l'équation (7). On arrête les itérations lorsque les ϵ_i restent inchangés.

4 Description des tests comparatifs

Les sept algorithmes présentés dans la section 3 ont été comparés aux moyens de tests. Nous donnons dans cette section la description du protocole expérimental.

4.1 Jeu de données

Nous utilisons les matrices de calibration et de pose d'une image réelle. Les valeurs utilisées sont les suivantes :

$$\boldsymbol{R} = \begin{bmatrix} -0.5931 & 0.8049 & -0.0167 \\ 0.2190 & 0.1413 & -0.9654 \\ -0.7747 & -0.5763 & -0.2601 \end{bmatrix}$$

$$\boldsymbol{t} = \begin{pmatrix} -14.1343 & 10.1104 & 114.8236 \end{pmatrix}^\top$$

et :

$$\boldsymbol{K} = \begin{bmatrix} 796.099 & 0 & 421.584 \\ 0 & 796.099 & 318.655 \\ 0 & 0 & 1 \end{bmatrix}$$

Les points considérés sont situés sur une mire de calibration dans l'image réelle. Pour les différents tests, nous sélectionnons un jeu de 4, 5 ou 6 points réels de la scène. A partir de ces positions, les points images exacts sont calculés avec l'équation (1) et les matrices $\boldsymbol{R}$ et $\boldsymbol{t}$ réels. Nous obtenons donc un jeu de correspondances exactes $(\boldsymbol{x}_i, \boldsymbol{X}_i)$.

4.2 Bruit gaussien

Afin de tester la robustesse des algorithmes, nous devons ajouter un bruit aux points images exacts. Ce bruit est un bruit gaussien que nous obtenons de la manière suivante :
Pour une valeur de l'écart-type σ donnée, on construit la fonction de répartition de la loi gaussienne de moyenne nulle et d'écart-type σ. Cette construction est faite par une intégration discrète par palliers représentant 1/100ème de pixel. La fonction de répartition étant strictement croissante, elle représente une bijection de l'ensemble des réels vers l'intervalle $]0, 1[$. Pour obtenir une valeur aléatoire de répartition gaussienne, on choisit un réel aléatoire de répartition uniforme entre 0 et 1, et on utilise la fonction de répartition gaussienne pour retrouver par dichotomie le réel correspondant. On obtient donc une répartition gaussienne d'écart-type voulu à partir d'une répartition uniforme.

4.3 Mesure d'erreurs

A partir d'un jeu de correspondances exactes $(\boldsymbol{x}_i, \boldsymbol{X}_i)$ et d'un écart-type σ donné, nous ajoutons à chacun des points $\boldsymbol{x}_i$ un bruit gaussien dans les directions x et y pour obtenir des points bruités $\boldsymbol{x}'_i$. On applique alors chacun des algorithmes qui fournit à chaque fois de nouvelles valeurs $\boldsymbol{R}'$ et $\boldsymbol{t}'$. Il est alors possible de mesurer les erreurs dans la translation, la rotation et de reprojection. L'erreur de translation est $E_t = \|\boldsymbol{t} - \boldsymbol{t}'\|$, l'erreur de rotation est $E_R = Fnorm(\boldsymbol{R}^\top\boldsymbol{R}' - \boldsymbol{I})$, où $Fnorm$ représente la norme de Frobenius. L'erreur de reprojection est la distance en pixels entre $\boldsymbol{x}'_i$ et $\boldsymbol{R}'\boldsymbol{X}_i + \boldsymbol{t}'$. L'experience est renouvelée pour 100 valeurs de bruits aléatoires et on établit la moyenne des erreurs sur 100 tentatives.
Tout ce processus est répété pour des valeurs croissantes d'écart-type, allant de 0 à 10 pixels par pas de 0.5 pixels. On peut alors tracer les courbes d'erreurs moyennes en fonction de

l'écart-type pour chaque jeu de correspondances selectionné.
Les expériences ont été menées en particulier pour des jeux de 4, 5 et 6 points. Les résultats obtenus sont présentés dans la section suivante.

5 Résultats expérimentaux

Les figures 4 à 7 montrent les erreurs de calcul de pose en fonction du niveau de bruit pour des jeux de 4, 5 et 6 points. Dans cette section, nous analysons les résultats.
La figure 4 présente les erreurs pour un premier jeu de 4 points non coplanaires. La première remarque est l'infériorité de l'algorithme de Quan et Lan (**Quan-Lan**) par rapport aux autres. En fait, cet algorithme présente des problèmes numériques importants, dus à la résolutions d'équations linéaires mettant en jeu des variables aux ordres de grandeurs très différents. Même après une homogénéisation des variables, l'instabilité numérique reste importante.
Deux algorithmes itératifs, celui de Lu, Hager et Mjolsness (**Lu-Hager**) et l'optimisation de l'erreur de reprojection (**NLLS**), fournissent exactement les mêmes résultats, supposés être la meilleure solution atteignable. Ils sont en effet initialisés avec les résultats d'algorithmes linéaires et donnés à titre de référence pour les algorithmes linéaires. Il est intéressant de noter que l'algorithme de DeMenthon et Davis (**POSIT**) et celui de Ansar et Daniilidis (**Daniilidis**) ont des résultats similaires, et sont très proches de la meilleure solution itérative. Il faut remarquer que même si POSIT est aussi un algorithme itératif, sa convergence est assez rapide pour qu'il soit comparé aux algorithmes linéaires pour la question du temps de calcul. En termes d'erreurs de reprojection, Daniilidis obtient de meilleurs résultats.
Dans la figure 5, un autre jeu de 4 points est considéré. Tandis que Quan-Lan reste instable, POSIT donne toujours de bon résultats. Daniilidis diverge pourtant très rapidement. Cet exemple montre la dépendance à la configuration de points de Daniilidis, en particulier dans le cas minimal de 4 points.
Le cas de 5 points est étudié dans la figure 6. Encore une fois, Daniilidis et POSIT ont des résultats similaires pour les erreurs de translation et de rotation. En ce qui concerne l'erreur de repro-

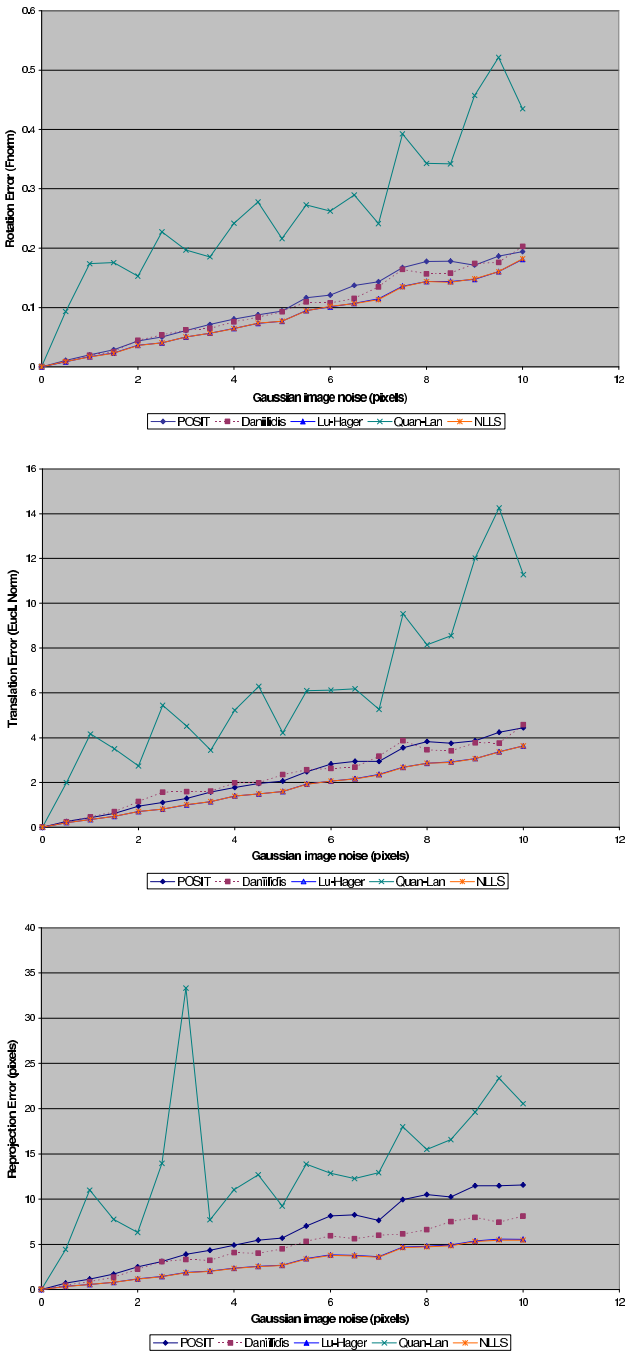


FIG. 4 – Erreurs de calcul de pose pour 4 points (a) *Erreur de rotation (en haut), erreur de translation (au milieu) et erreur de reprojection (en bas)*

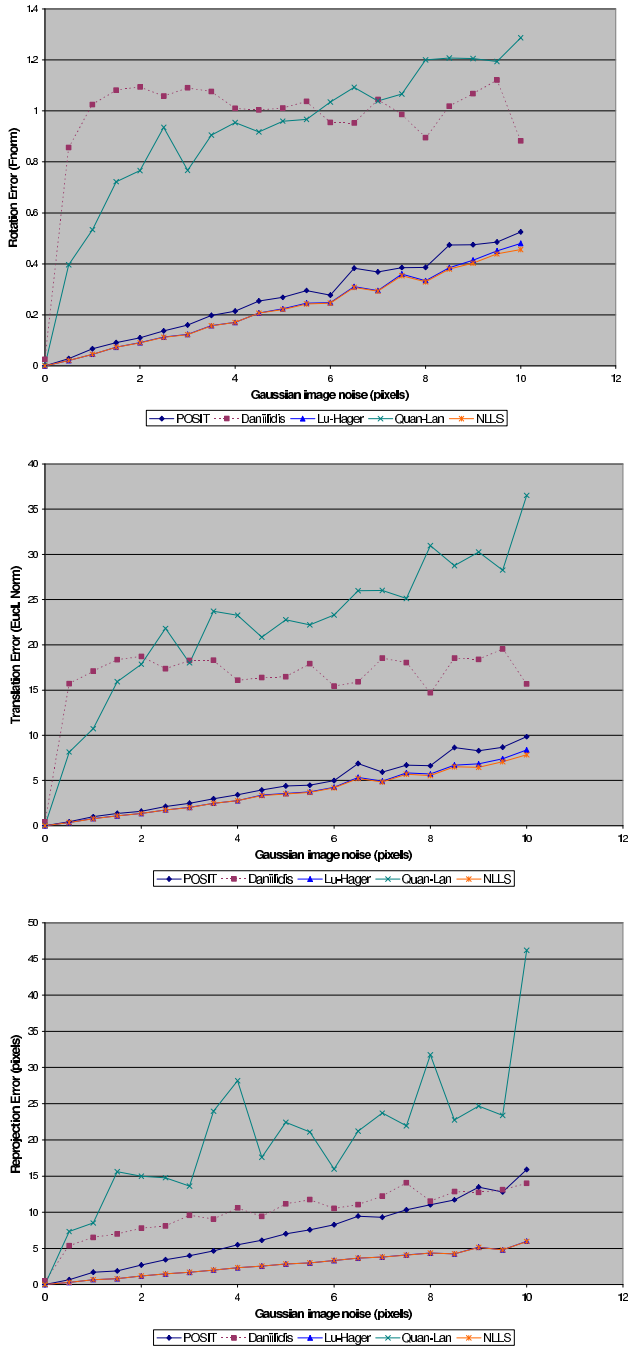


FIG. 5 – Erreurs de calcul de pose pour 4 points (b) *Erreur de rotation (en haut), erreur de translation (au milieu) et erreur de reprojection (en bas)*

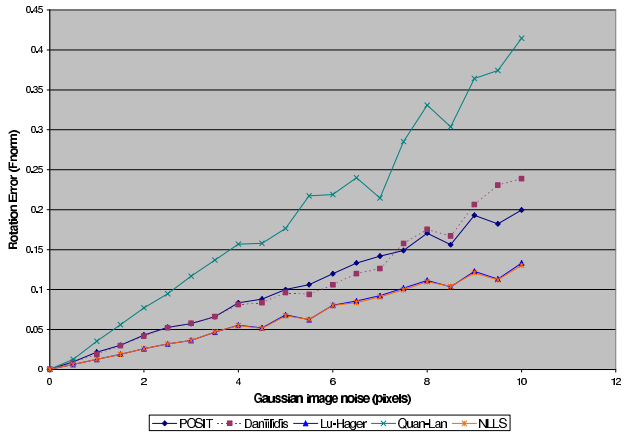
jection, Daniilidis obtient de meilleurs résultats, comparables à ceux des algorithmes itératifs.

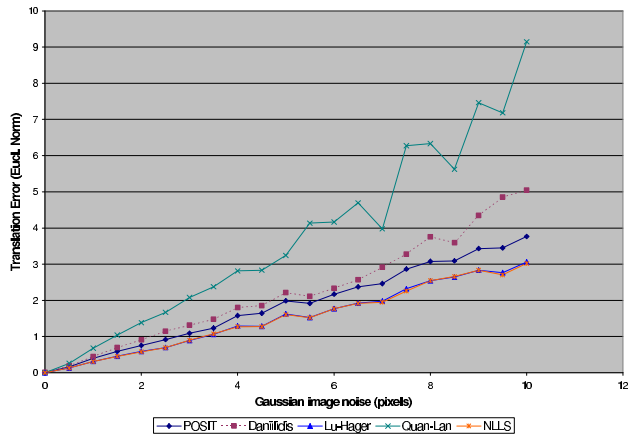
A partir de 6 points, plus d'algorithmes peuvent être appliqués. En plus des précédents, les algorithmes de Fiore (**Fiore**) et la transformation linéaire directe (**DLT**) sont représentés. Nous remarquons d'abord que DLT est particulièrement peu robuste. Dans cet algorithme, l'orthogonalité de la matrice de rotation n'est pas conservée. Lors des tests, nous avons ajouté une étape supplémentaire pour remplacer $\boldsymbol{R}$ par la matrice de rotation la plus proche au sens de la norme de Frobenius, ce qui explique une erreur de reprojection très importante. Force est de constater également que Quan-Lan obtient une meilleur stabilité, avec des résultats comparables à ceux de Fiore. Là encore, Daniilidis montre de très bons résultats, en particulier pour l'erreur de reprojection.

Tous les algorithmes testés n'ont evidemment pas le même temps de calcul. Bien que Daniilidis est dit linéaire, il a le temps de calcul le plus long, car il suppose le calcul de matrices de dimensions de l'ordre de $n^3 \times n^2$. Ainsi, il ne peut pas être utilisé pour des applications en temps réel. Quan-Lan, Fiore et POSIT sont au contraire assez rapide pour le temps réel. Dans une application de réalité augmentée, l'erreur de reprojection a le plus d'importance, car des objets virtuels sont reprojetés dans la scène. Ces trois algorithmes ont des résultats comparables en terme de reprojection. Pour conclure, POSIT est recommandé pour sa rapidité et son exactitude lorsque le temps est crucial. Lorsque le temps a moins d'importance, Daniilidis est indiqué car il donne les meilleurs résultats, sauf dans le cas de 4 points, où POSIT semble plus stable. Il faut toutefois noter que POSIT n'est pas globallement convergent. Lorsque les points sont placés à la périphérie de l'image, POSIT peut échouer, et Fiore ou Quan-Lan doivent être utilisés.

6 Conclusion

Nous avons étudié dans cet article le problème P-n-P, consistant à calculer la pose de la caméra à partir de n correspondances de points 2D/3D. Sept solutions à ce problème ont été envisagées : quatre algorithmes linéaires et trois itératives. L'implémentation de ces méthodes a permis de





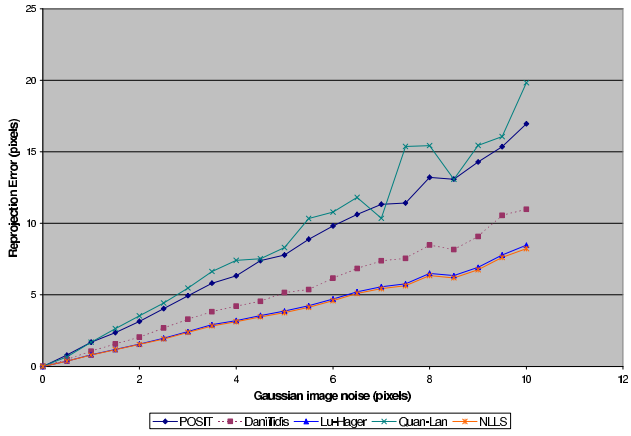
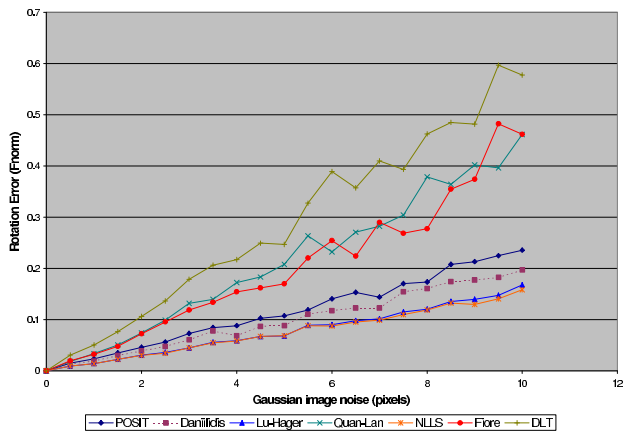
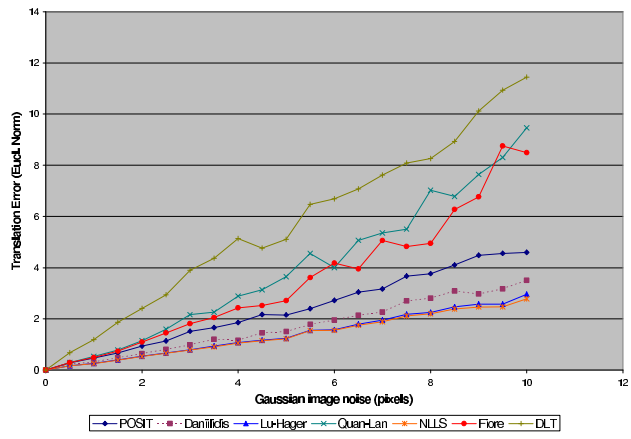


FIG. 6 – Erreurs de calcul de pose pour 5 points
Erreur de rotation (en haut), erreur de translation (au milieu) et erreur de reprojection (en bas)





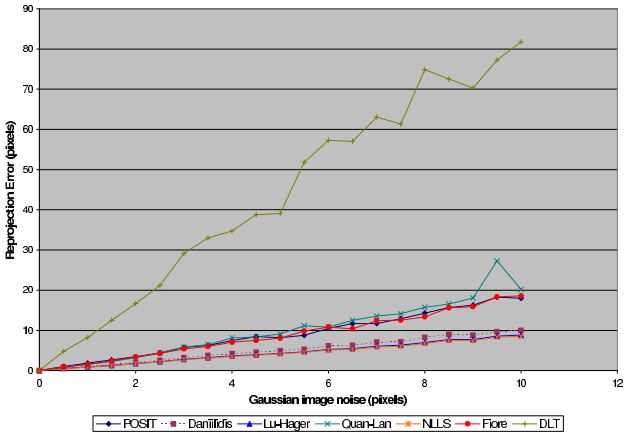


FIG. 7 – Erreurs de calcul de pose pour 6 points
Erreur de rotation (en haut), erreur de translation (au milieu) et erreur de reprojection (en bas)

comparer leur résistance au bruit au moyen de simulations.
Il reste important de bien connaître ces algorithmes afin de choisir la méthode adaptée à un cas de figure. En particulier, certaines méthodes itératives (POSIT) est souvent plus rapide que la méthode linéaire de Daniilidis. La limite de cette étude est donc le choix nécéssaire d'algorithme pour une configuration donnée.
Il serait intéressant d'automatiser ce choix de façon à faciliter l'utilisation de ces algorithmes. Une première idée pourrait consister en une compétition parallèle des algorithmes avec récupération du meilleur résultat. Cette solution directe devrait toutefois être optimisée de façon à se conformer aux contraintes de temps souvent cruciales en réalité augmentée.

Remerciements

Le travail presenté dans cet article a été réalisé dans le cadre du project européen ARIS (Augmented Reality and Image Synthese, IST-2000-28707), http ://www.igd.fhg.de/igd-a4/projects/aris/index.html

Références

[1] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. In *Proc. ECCV*, volume 4, pages 282–296, 2002.

[2] R. Cipolla, T. Drummond, and D. Robertson. Camera calibration from vanishing points in images of architectural scenes. In *Proc. British Machine Vision Conference*, 1999.

[3] D. F. DeMenthon and L. S. Davis. Exact and approximate solutions of the perspective-three-point problem. *IEEE Trans. PAMI*, 14(11) :1100–1105, 1992.

[4] D. F. Dementhon and L. S. Davis. Model-based objects pose in 25 lines of code. *Int. Journal of Computer Vision*, 15(1) :123–141, 1995.

[5] P. D. Fiore. Efficient linear solution of exterior orientation. *IEEE Trans. PAMI*, 23(2) :140–148, February 2001.

[6] M. A. Fishler and R. C. Bolles. Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Assoc. Comp. Mach.*, 24(6) :381–395, 1981.

[7] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions of the three point perspective pose estimation problem. In *Proc. Conf. on CVPR*, pages 592–598, 1991.

[8] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society*, 5(7) :1127–1135, July 1988.

[9] C. P. Lu, G. D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Trans. PAMI*, 22(6) :610–622, 2000.

[10] L. Quan and Z. Lan. Linear n-point camera pose determination. *IEEE Trans. PAMI*, 21 :774–780, 1999.

[11] G. Simon, A. W. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *Proc. ISAR*, 2000.

[12] P. Sturm and S. Maybank. On plane-based camera calibration : A general algorithm, singularities, applications. In *Proc. Conf. CVPR*, pages 432–437, 1999.