Contributions à l'intégration vision robotique : calibrage, localisation et asservissement

Fadi Dornaika

THESE

présentée par

Fadi DORNAIKA

pour obtenir le grade de **DOCTEUR**

de l'**INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE**

(Arrêté ministériel du 30 Mars 1992)

Spécialité : **Signal Image Parole**

CONTRIBUTIONS A L'INTEGRATION VISION / ROBOTIQUE : CALIBRAGE, LOCALISATION ET ASSERVISSEMENT

Date de soutenance : 25 Septembre 1995

Composition du jury :

Président :	**Bernard ESPIAU**
Rapporteurs :	**Rachid ALAMI**
	Jean-Thierry LAPRESTE
Examinateurs :	**Stéphane LAVALLEE**
	Radu HORAUD

Thèse préparée au sein du laboratoire LIFIA - IMAG - INRIA Rhône - Alpes

sous la direction de **Radu HORAUD**

Remerciements

Je tiens à remercier Messieurs Bernard Espiau, Stéphane Lavallée, Rachid Alami et Jean-Thierry Lapresté qui m'ont fait l'honneur de participer au jury de ma thèse, et plus particulièrement ces deux derniers pour avoir accepté d'être les rapporteurs de mon manuscrit.

Je tiens ensuite à remercier Radu Horaud, qui a encadré et soutenu mon travail de thèse, pour ses conseils et ses encouragements qu'il m'a prodigués pendant ces trois années. Je remercie également Roger Mohr pour m'avoir accueilli dans son équipe et pour sa gentillesse. Je remercie tous les membres de l'équipe MOVI pour leur amitié et leur accueil.

Table des matières

Introduction

La vision est un de nos sens les plus puissants. Elle nous fournit une quantité importante d'informations qui nous permettent d'interagir intelligemment avec notre environnement. Grâce à elle, nous sommes capables en un instant d'identifier la plupart des objets situés dans notre champ visuel, de repérer leur position et enfin d'ébaucher une réflexion sur les tâches que nous devons effectuer. Mais c'est aussi un de nos sens les plus compliqués. Son mécanisme reste mal connu malgré les progrès réalisés par la psychologie cognitive. Ainsi, notre système de perception visuelle règle en permanence et sans même que nous nous en apercevions des problèmes aussi essentiels à notre survie que de savoir où nous sommes et comment nous bougeons dans l'espace, de voir en trois dimensions ... C'est ainsi encore que les bébés commencent, dès l'âge de 5 mois, à contrôler visuellement les mouvements de leurs mains dans le but d'accéder à un objet placé à côté d'eux.

D'une manière similaire, la vision par ordinateur vise à donner aux robots la capacité de percevoir l'espace qui les entoure. Cependant, malgré les efforts des chercheurs aucune machine ne parvient à rivaliser - et de loin - avec la vision humaine. La vision par ordinateur ne vise pas la reproduction sur machine de la vision humaine. Elle vise plutôt à arriver par des moyens informatiques à des résultats similaires.

Depuis quelques années, les développements technologiques des capteurs ont permis leur implantation directement sur l'effecteur des robots mobiles ou manipulateurs. Les tâches, telles que positionnement et suivi d'objets en utilisant la vision paraissent maintenant tout à fait envisageables. Le capteur de vision fournit une perception de l'environnement alors que le robot fournit l'action réflexe.

L'utilisation du capteur de vision, à laquelle nous nous sommes attachés, est particulièrement intéressante en raison de la grande richesse des informations qu'une caméra peut fournir et en raison de la grande variété des tâches qu'elle permet de réaliser (l'identification, l'inspection, la localisation). Vu les progrès technologiques, les applications utilisant le guidage visuel des robots deviennent de plus en plus nombreuses. Les travaux précurseurs ont porté sur des applications industrielles simples. Actuellement, le capteur de vision fait partie de la boucle de commande. Ainsi, la vision contribue-t-elle au développement de la télérobotique qui vise à remplacer les hommes par des robots dans des sites hostiles à la présence humaine [FDD94], comme l'espace et les sites nucléaires. L'utilisation de la vision en robotique va, non seulement augmenter l'autonomie et l'efficacité des robots, mais également les doter des capacités de reconnaissance et de raisonnement basées sur les informations visuelles. Certes, ces objectifs ont posé de nouveaux problèmes (traitement d'image, reconnaissance d'objets, ...) qui ont déjà fait, et font encore, l'objet de nombreux travaux.

La vision et la robotique peuvent coopérer dans les procédures de calibrage en ce sens

qu'une caméra peut être un excellent moyen de calibrage d'un robot et, inversement, un robot calibré peut être utilisé pour identifier les paramètres de la caméra.

Un capteur de vision fournit les données sous forme d'images. Les traitements effectués sur ces images sont généralement classés en trois niveaux : le premier appelé, "vision bas-niveau", est la segmentation ou l'extraction des indices images ; le deuxième niveau concerne les domaines de la reconstruction, de la localisation et de l'analyse du mouvement ; le troisième concerne le contenu sémantique de la scène observée. Bien entendu, plus les informations sont de haut niveau, plus les algorithmes de traitement d'image nécessaires à leur extraction sont complexes.

Le travail décrit dans cette thèse concerne l'intégration de fonctionnalités de perception et d'action au sein d'un système robotique. Deux sujets liés à cette intégration ont été traités : le calibrage capteur/robot et la localisation d'objets à partir d'images.

Le calibrage capteur/robot s'avère intéressant dans beaucoup d'applications où le mouvement du robot est guidé par ce capteur. De plus, ce calibrage permet la conversion des mesures visuelles dans l'espace du robot.

La localisation d'objets permet de calculer la relation géométrique entre une scène 3D et une caméra. Une information visuelle ne permet à elle seule de remonter à l'information tridimensionnelle de l'environnement : sur un pixel de l'image peuvent se projeter une infinité de points de la scène. Par contre, connaissant les dimensions de l'objet, il est possible de retrouver la localisation de cet objet dans l'espace tridimensionnel. Par conséquent, cette localisation est importante dans la mesure où elle fournit la relation géométrique entre une scène 3D et la caméra considérée.

Généralement, l'utilisation de la vision en robotique était la suivante : traitement des données visuelles dans un repère lié au capteur, conversion des données dans un repère lié à la scène et calcul du vecteur de contrôle du robot dans un repère de la scène. Cependant, ce schéma fonctionne en boucle ouverte vis à vis des mesures visuelles et ne permet pas de tenir compte des erreurs de mesures ou d'estimation de la position du robot. Bien que les données visuelles soient utilisées, le robot reste aveugle lorsque son modèle est affecté d'erreurs.

Pour illustrer le problème de la coordination entre une caméra et un robot, nous considérons la tâche suivante : soit un sujet qui doit soulever un verre d'eau placé sur une table devant lui. Face à ce problème deux approches sont envisageables :

1. Après un regard plus ou moins long, le sujet communique à sa mémoire la valeur de l'angle que doit prendre chaque articulation de son bras. Ensuite, il ferme ses yeux et commence à exécuter les mouvements articulaires tels qu'ils sont estimés visuellement.

2. Le sujet commence à déplacer son bras vers le verre en le regardant tout au long de sa trajectoire et en faisant les corrections nécessaires afin que sa main aboutisse au verre d'une manière satisfaisante.

Il est évident que la première approche se heurte, le plus souvent, à un échec. En effet, même si les angles sont correctement estimés, les déplacements articulaires du bras ne sont pas nécessairement égaux aux valeurs estimées. Par contre, la deuxième approche fournit un retour réel sur l'état du bras par rapport à sa position finale.

Le travail décrit dans cette thèse s'inscrit dans le cadre de la deuxième approche.

Contributions majeures

Les principales contributions de cette thèse sont :

1. mise en œuvre d'une nouvelle méthode pour calculer la transformation caméra-pince.
2. mise en œuvre d'une nouvelle méthode pour calculer les transformations géométriques liant la base et l'effecteur d'un robot à une scène 3D.
3. étude et réalisation d'un algorithme permettant l'autocalibrage d'un système caméra-pince.
4. étude et comparaison de trois méthodes de localisation caméra/objet.
5. étude et réalisation d'une méthode pour le contrôle visuel de robots en boucle fermée.

Plan de la thèse

Le premier chapitre est consacré aux différents modes de description mathématiques d'un déplacement rigide : représentation matricielle, représentation géométrique et représentation quaternion. Il s'agit des notations pratiques pour exprimer les transformations de repère. Ces représentations sont à la base de toute modélisation des éléments géométriques constituant le robot et son environnement. Nous décrivons aussi les conversions entre ces différentes représentations.

Le deuxième chapitre comporte deux parties. La première partie présente une nouvelle formulation pour le calcul de la relation géométrique caméra-pince. Cette relation est un concept clé dans le cas des robots munis d'une ou plusieurs caméras. Cette nouvelle formulation permet de s'affranchir de la connaissance explicite des paramètres intrinsèques et extrinsèques de la caméra. Cette formulation possède la même forme d'équations associées à la formulation classique ($AX = XB$). Dans cette partie, nous proposons une nouvelle méthode de résolution qui estime simultanément les rotation et translation inconnues. Les expériences réalisées avec des données réelles et simulées montrent que, par rapport aux méthodes algébriques, cette méthode permet une meilleure stabilité de la solution.

La deuxième partie est consacrée à une généralisation du calibrage capteur-robot. Outre la relation géométrique caméra-pince, cette généralisation est très utile pour déterminer la relation géométrique entre la base du robot et une scène 3D. Nous y proposons une nouvelle méthode de résolution qui est plus robuste que la méthode linéaire existante.

Le troisième chapitre présente une approche originale permettant l'autocalibrage d'un système caméra-pince (les paramètres de la caméra et la relation caméra-pince). La méthode résout aussi le problème du passage d'une reconstruction projective à une reconstruction euclidienne.

Le quatrième chapitre présente le problème de localisation à partir d'images. Nous traitons le cas des points et des droites. Nous rappelons succinctement la méthode linéaire itérative proposée récemment par Dementhon. La convergence de cette dernière méthode

dépend de la position latérale de l'objet par rapport à l'axe optique de la caméra. Nous proposons une nouvelle méthode basée sur des approximations successives par une projection para-perspective. Cette nouvelle méthode permet le calcul de la pose associée au modèle perspectif de la caméra. La convergence de cette méthode ne dépend pas de la translation latérale. Nous présentons également une méthode numérique pour résoudre ce problème. Nous montrons que la méthode linéaire proposée a une robustesse comparable à celle des méthodes numériques. De plus, cette méthode est tout à fait adaptée aux applications temps réel.

Le dernier chapitre présente une nouvelle méthode de contrôle visuel de robots. Cette méthode constitue une extension de l'approche *commande référencée capteur* où l'on contrôle les mouvements 3D de la caméra. Nous montrons qu'il est possible de positionner précisément l'effecteur d'un robot en dépit des erreurs affectant son modèle et en dépit de l'utilisation d'une caméra non ou mal calibrée. Ainsi, pour chaque tâche, on peut construire un motif, constitué d'un ensemble de signaux élémentaires et correspondant à une bonne réalisation de la tâche. En comparant le motif à atteindre et l'image perçue par la caméra, le problème du contrôle peut alors se traduire sous la forme d'une régulation dans l'image. La précision du positionnement dépend de la précision avec laquelle les primitives 2D sont extraites. C'est un problème qui se situe au bas niveau. Nous montrons également que cette coordination entre la caméra et le robot met en jeu les deux aspects de calibrage présentés dans les deuxième et quatrième chapitres. Le premier aspect sert à modéliser la relation géométrique entre le repère de la pince et les primitives 3D (dans le cas où ces primitives ne sont pas données par le constructeur du robot). Cet aspect appartient à une étape hors ligne. Le deuxième aspect concerne la localisation de la pince du robot par rapport à la caméra. Cet aspect appartient à l'étape en ligne (inséré dans la boucle de contrôle) et est nécessaire pour l'estimation en temps réel du Jacobien d'image qui est à la base de contrôle. Il est à noter que cette localisation n'influe pas sur la précision du positionnement final puisque la boucle de contrôle est pilotée par des informations 2D. Toutefois, cette localisation s'avère intéressante pour que le comportement dynamique de l'asservissement soit optimal et robuste.

Enfin, la conclusion présente un résumé du travail accompli et des résultats obtenus ainsi qu'une vue prospective.

Chapitre 1

Paramétrisation des rotations et des déplacements

Chaque fois qu'on a affaire à déterminer la position relative des objets 3D, on doit utiliser une représentation qui donne l'orientation et la position d'un objet par rapport à un autre. Cette position relative se traduit par une transformation géométrique entre deux repères, chacun de ces deux repères étant lié à un objet. Plusieurs types de paramétrisation peuvent être utilisés pour décrire cette transformation. La plus classique et la plus familière est la représentation matricielle: l'orientation relative est donnée par une matrice orthogonale, de dimension 3×3, appelée matrice de rotation ; la position relative est donnée par un vecteur, de dimension 3, appelé vecteur de translation. L'ensemble des rotations constitue un groupe, il en est de même pour l'ensemble des déplacements (rotation et translation). Dans ce chapitre nous nous intéressons à la paramétrisation de ces deux groupes. Nous décrivons la paramétrisation (vectorielle et quaternion) des rotations ainsi que des déplacements. Nous présentons également leur équivalence avec la représentation matricielle. La représentation vectorielle utilise des paramètres impliqués directement dans le déplacement, autrement dit, ces paramètres constituent les éléments géométriques du déplacement concerné. La représentation quaternion utilise des objets mathématiques, les quaternions et les quaternions duaux, qui sont les plus adéquats aux problèmes de la localisation 3D. Grâce à eux, les solutions du problème de la localisation (analytiques ou numériques) deviennent plus élégantes.

1.1 Introduction

En robotique et en vision par ordinateur on associe à tout élément du poste de travail un ou plusieurs repères. Ces repères sont généralement définis de telle sorte que leurs axes et leurs origines correspondent respectivement à des directions et à des points privilégiés ayant un rôle fonctionnel lors de l'exécution de la tâche: direction d'insertion ou centre de gravité d'une pièce par exemple. Ils permettent de situer dans l'espace les objets fixes de l'environnement ainsi que les corps mobiles comme les éléments constitutifs d'un robot ou transportés par lui par exemple.

Soit une transformation quelconque qui amène le repère R_i sur le repère R_j (voir

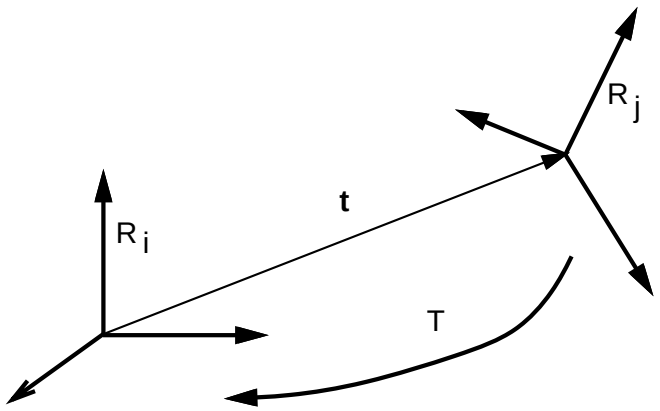


FIG. 1.1 - *Une transformation T permettant l'expression des coordonnées du repère R_j dans le repère R_i ; géométriquement, cette transformation amène le repére R_i sur le repère R_j.*

Figure 1.1). La représentation la plus familière de cette transformation est la représentation matricielle. Elle est définie par une matrice de dimension 4×4 appelée matrice de transformation homogène, telle que :

$$T = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{1.1}$$

- R représente la matrice de rotation, elle est de dimension 3×3 et possède les propriétés suivantes :
  1. Son déterminant est égal à 1.
  2. Son inverse est égal à sa transposée.
  3. Ses valeurs propres sont : 1, $e^{i\theta}$, $e^{-i\theta}$ (complexes conjugués).
- $\mathbf{t}$ représente le vecteur de translation.

Les 9 éléments de la matrice R vérifient 6 contraintes d'orthonormalité qui expriment que les vecteurs ligne (ou colonne) sont de norme égale à 1 et orthogonaux entre eux. On a ainsi 3 éléments, parmi les 9, qui sont indépendants.

Si un point P est exprimé dans le repère R_i par le vecteur $\mathbf{p}_i$ et dans le repère R_j par le vecteur $\mathbf{p}_j$, on aura la relation suivante :

$$\mathbf{p}_i = R\,\mathbf{p}_j + \mathbf{t}$$

Si nous utilisons les coordonnées homogènes, la dernière équation devient:

$$\begin{bmatrix} \mathbf{p}_i \\ 1 \end{bmatrix} = T \begin{bmatrix} \mathbf{p}_j \\ 1 \end{bmatrix}$$

Cette représentation nous permet :

- De représenter dans des différents repères des caractéristiques telles que les points, les vecteurs, les droites et les plans.

– De calculer la transformée de ces caractéristiques par une certaine transformation donnée.

Cependant, cette représentation matricielle n'est pas la plus adéquate pour résoudre les problèmes de la localisation 3D qui consiste à estimer la rotation et la translation entre deux repères différents.

Dans la suite, nous allons décrire deux représentations de la rotation : la représentation vectorielle et la représentation quaternion. Ensuite, nous allons décrire deux représentations du déplacement (rotation et translation) : le vissage et le quaternion dual.

1.2 Rotation et représentation vectorielle

Chaque rotation dont la matrice est donnée par R peut être effectuée par une rotation d'un angle θ autour d'un axe passant par l'origine. Soit $\mathbf{n}$ le vecteur directeur de cet axe.

L'axe de rotation est tel qu'il n'est pas modifié par la rotation. C'est donc la direction propre associée à la valeur propre unitaire de la matrice R :

$$R\,\mathbf{n} = \mathbf{n}$$

L'angle de rotation θ est fourni par les valeurs propres complexes conjuguées.

En notation matricielle, un vecteur $\mathbf{v}$ se transforme en un vecteur $\mathbf{v}'$ grâce à la formule suivante :

$$\mathbf{v}' = R\,\mathbf{v}$$

En notation vectorielle, un vecteur subissant une rotation d'axe $\mathbf{n}$ ($\mathbf{n}\cdot\mathbf{n} = 1$) et d'angle θ devient un vecteur $\mathbf{v}'$ qui s'obtient par la formule de *Rodrigues* :

$$\mathbf{v}' = \mathbf{v} + \sin\theta\ \mathbf{n} \times \mathbf{v} + (1 - \cos\theta)\ \mathbf{n} \times (\mathbf{n} \times \mathbf{v}) \tag{1.2}$$

où $\times$ dénote le produit vectoriel et $\cdot$ le produit scalaire.

Pour démontrer cette formule nous allons décomposer le vecteur $\mathbf{v}$ en une composante parallèle à $\mathbf{n}$ et une composante orthogonale à $\mathbf{n}$ (Figure 1.2) [Aya89] :

$$\begin{aligned} \mathbf{v} &= \mathbf{v}_{\|} + \mathbf{v}_{\perp} \\ &= (\mathbf{v}\cdot\mathbf{n})\,\mathbf{n} + (\mathbf{v} - (\mathbf{v}\cdot\mathbf{n})\,\mathbf{n}) \end{aligned}$$

Le vecteur $\mathbf{v}'$ sera donc obtenu en additionnant $\mathbf{v}'_{\|}$ et $\mathbf{v}'_{\perp}$. En remarquant que $\mathbf{v}'_{\|} = \mathbf{v}_{\|}$ (puisque $\mathbf{v}_{\|}$ a la même direction que $\mathbf{n}$), on aura :

$$\mathbf{v}' = \mathbf{v}_{\|} + \mathbf{v}'_{\perp}$$

$\mathbf{v}'_{\perp}$ est obtenu en faisant tourner $\mathbf{v}_{\perp}$ d'un angle θ dans un plan orthogonal à $\mathbf{n}$. Dans la base $\mathbf{n}$, $\mathbf{v}_{\perp}$ et $\mathbf{n} \times \mathbf{v}_{\perp}$, on a :

$$\mathbf{v}'_{\perp} = \cos\theta\ \mathbf{v}_{\perp} + \sin\theta\ \mathbf{n} \times \mathbf{v}_{\perp}$$

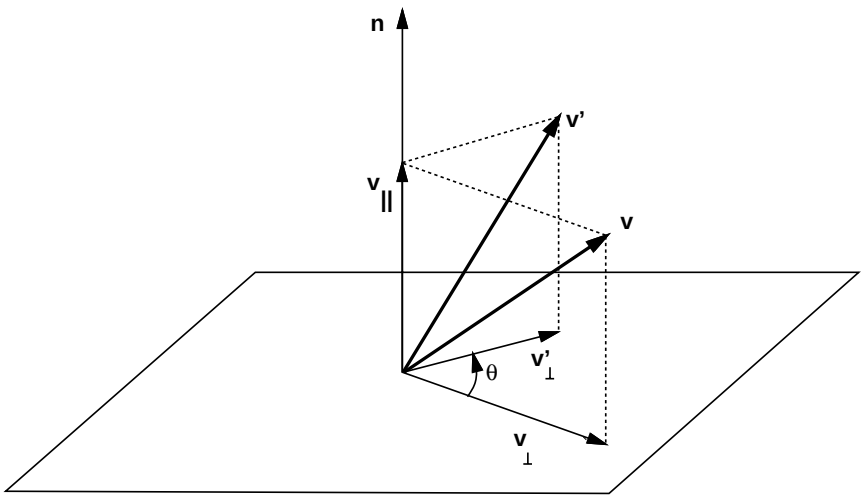


FIG. 1.2 - *Décomposition vectorielle permettant de démontrer facilement la formule de Rodrigues.*

Finalement, on obtient pour $\mathbf{v}'$:

$$\begin{aligned}
\mathbf{v}' &= \mathbf{v}_{\parallel} + \mathbf{v}'_{\perp} \\
&= \mathbf{v}_{\parallel} + \cos\theta\ \mathbf{v}_{\perp} + \sin\theta\ \mathbf{n} \times \mathbf{v}_{\perp} \\
&= (\mathbf{v}\cdot\mathbf{n})\,\mathbf{n} + \cos\theta\,(\mathbf{v} - (\mathbf{v}\cdot\mathbf{n})\,\mathbf{n}) + \sin\theta\,(\mathbf{n} \times (\mathbf{v} - (\mathbf{v}\cdot\mathbf{n})\,\mathbf{n})) \\
&= \cos\theta\ \mathbf{v} + \sin\theta\ \mathbf{n} \times \mathbf{v} + (1 - \cos\theta)(\mathbf{v}\cdot\mathbf{n})\,\mathbf{n}
\end{aligned}$$

Par ailleurs, on a la relation vectorielle suivante :

$$\mathbf{n} \times (\mathbf{n} \times \mathbf{v}) = (\mathbf{v}\cdot\mathbf{n})\,\mathbf{n} - (\mathbf{n}\cdot\mathbf{n})\,\mathbf{v}$$

Mais comme $\mathbf{n}$ est unitaire, on obtient:

$$(\mathbf{v}\cdot\mathbf{n})\,\mathbf{n} = \mathbf{v} + \mathbf{n} \times (\mathbf{n} \times \mathbf{v})$$

et en substituant dans l'équation précédente on obtient bien la formule de *Rodrigues*, soit l'équation (1.2).

La représentation d'une rotation par un axe et un angle est très pratique pour résoudre le problème de la recherche d'une transformation rigide optimale ainsi que pour représenter le déplacement (rotation et translation) avec un vissage et un quaternion dual comme nous allons le voir dans les paragraphes suivants.

Nous présentons ici le passage entre les deux représentations : vectorielle et matricielle.

1.2.1 De $(\mathbf{n}, \theta)$ à R

Remarquons tout d'abord qu'un produit vectoriel peut se mettre sous forme matricielle :

$$\mathbf{n} \times \mathbf{v} = \begin{bmatrix} -n_z v_y + n_y v_z \\ n_z v_x - n_x v_z \\ -n_y v_x + n_x v_y \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

$K(\mathbf{n})$ est la matrice antisymétrique associée au vecteur $\mathbf{n}$:

$$K(\mathbf{n}) = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}$$

Le produit vectoriel devient alors :

$$\mathbf{n} \times \mathbf{v} = K(\mathbf{n})\,\mathbf{v}$$

En utilisant cette égalité nous obtenons la version matricielle de la formule de *Rodrigues* :

$$\mathbf{v}' = \left(I + \sin\theta\, K(\mathbf{n}) + (1 - \cos\theta)\, K^2(\mathbf{n})\right) \mathbf{v}$$

Soit pour la matrice de rotation :

$$R = I + \sin\theta\, K(\mathbf{n}) + (1 - \cos\theta)\, K^2(\mathbf{n})$$

En explicitant les termes et avec les notations $s = \sin\theta$ et $c = \cos\theta$ on obtient pour R :

$$R = \begin{bmatrix} c + (1-c)n_x^2 & -n_z s + (1-c)n_x n_y & n_y s + (1-c)n_x n_z \\ n_z s + (1-c)n_x n_y & c + (1-c)n_y^2 & -n_x s + (1-c)n_z n_y \\ -n_y s + (1-c)n_x n_z & n_x s + (1-c)n_z n_y & c + (1-c)n_z^2 \end{bmatrix} \tag{1.3}$$

Il est intéressant de remarquer que $(\mathbf{n}, \theta)$ représente la même rotation que $(-\mathbf{n}, -\theta)$ ou que $(-\mathbf{n}, 2\pi - \theta)$. Il suffit de substituer $\mathbf{n}$ par $-\mathbf{n}$ et θ par $-\theta$ ou par $2\pi - \theta$ dans l'équation précédente.

1.2.2 De R à $(\mathbf{n}, \theta)$

Il sera également utile de pouvoir extraire l'axe et l'angle de rotation à partir de la matrice de rotation. Pour cela il y a deux possibilités :

1. Extraire les valeurs propres de cette matrice. Calculer le vecteur associé à la valeur propre 1 et normaliser ce vecteur. Ce qui donne la direction de l'axe de rotation. L'angle de rotation se calcule facilement à partir des valeurs propres complexes conjuguées.

2. Identifier les éléments R_{ij} de la matrice de rotation avec l'expression de cette matrice telle qu'elle est fournie par la formule de *Rodrigues*, équation (1.3).

En additionnant les termes diagonaux dans l'équation (1.3) on obtient :

$$R_{11} + R_{22} + R_{33} = 1 + 2\cos\theta$$

Par ailleurs, on a les relations suivantes :

$$\begin{aligned} R_{21} - R_{12} &= 2\, n_z \sin\theta \\ R_{13} - R_{31} &= 2\, n_y \sin\theta \\ R_{32} - R_{23} &= 2\, n_x \sin\theta \end{aligned}$$

En tenant compte du fait que le vecteur $\mathbf{n}$ doit être unitaire, on obtient :

$$\begin{aligned} \cos\theta &= \frac{1}{2}\left(R_{11} + R_{22} + R_{33} - 1\right) \\ \sin^2\theta &= \frac{1}{4}\left((R_{21} - R_{12})^2 + (R_{13} - R_{31})^2 + (R_{32} - R_{23})^2\right) \end{aligned}$$

Nous avons ainsi la valeur du cosinus et la valeur absolue du sinus, ce qui implique qu'il y a deux angles appartenant à l'intervalle $[0, 2\pi]$ qui vérifient les deux équations précédentes. La somme de ces deux angles est égale à 2π. Ce qui ne pose aucun problème puisque $(\mathbf{n},\ \theta)$ et $(-\mathbf{n},\ 2\pi - \theta)$ représente la même rotation. On peut ainsi imposer à l'angle θ d'être compris entre 0 et π (l'axe de rotation sera déterminé d'une façon unique si $\sin\theta \neq 0$). Nous pouvons donc choisir la valeur positive du sinus et nous avons finalement :

$$\theta = \arctan\left(\frac{\sin\theta}{\cos\theta}\right) \quad \text{avec} \quad 0 \leq \theta \leq \pi$$

La direction $\mathbf{n}$, correspondant à l'angle θ ainsi calculée, est alors donnée par (si $\sin\theta \neq 0$) :

$$\begin{aligned} n_x &= (R_{32} - R_{23})/2\sin\theta \\ n_y &= (R_{13} - R_{31})/2\sin\theta \\ n_z &= (R_{21} - R_{12})/2\sin\theta \end{aligned}$$

Deuxième méthode

Lorsque $\sin\theta$ est petit, les paramètres n_x, n_y et n_z ne sont pas déterminés avec précision par les équations citées ci-dessus. Une autre méthode plus exacte peut être utilisée et la solution est donnée par :
Si R_{11} est la plus positive des éléments diagonaux de la matrice R, alors :

$$\begin{aligned} n_x &= sgn(R_{32} - R_{23})\sqrt{\frac{R_{11} - \cos\theta}{1 - \cos\theta}} \\ n_y &= \frac{R_{12} + R_{21}}{2n_x(1 - \cos\theta)} \\ n_z &= \frac{R_{13} + R_{31}}{2n_x(1 - \cos\theta)} \end{aligned}$$

Si R_{22} est la plus positive des éléments diagonaux de la matrice R, alors :

$$n_y = sgn(R_{13} - R_{31})\sqrt{\frac{R_{22} - \cos\theta}{1 - \cos\theta}}$$

$$
\begin{aligned}
n_x &= \frac{R_{21} + R_{12}}{2n_y(1 - \cos\theta)} \\
n_z &= \frac{R_{32} + R_{23}}{2n_y(1 - \cos\theta)}
\end{aligned}
$$

Si R_{33} est la plus positive des éléments diagonaux de la matrice R, alors :

$$
\begin{aligned}
n_z &= sgn(R_{21} - R_{12}) \sqrt{\frac{R_{33} - \cos\theta}{1 - \cos\theta}} \\
n_x &= \frac{R_{13} + R_{31}}{2n_z(1 - \cos\theta)} \\
n_y &= \frac{R_{32} + R_{23}}{2n_z(1 - \cos\theta)}
\end{aligned}
$$

où le symbole sgn() désigne la fonction signe.

En résumé, la représentation ($\mathbf{n}$, θ) comporte 4 paramètres. Si, de plus, on choisit le module de $\mathbf{n}$ tel qu'il soit égal à une fonction de l'angle de rotation:

$$\|\mathbf{n}\| = f(\theta)$$

on n'a alors plus que trois paramètres pour décrire une rotation et cette représentation est minimale. Ainsi, dans la représentation d'une rotation avec l'exponentielle d'une matrice antisymétrique, le module du vecteur associé est égal à θ [PYM89] ; dans le cas où l'on cherche à estimer la rotation optimale, ce module est égal à $\arctan\frac{\theta}{2}$ [TL89].

1.2.3 Angles de Roulis - Tangage - Lacet

Une autre représentation des rotations est celle utilisant une décomposition de la matrice de rotation en trois matrices de rotation autour de chacun des trois axes. Ces trois matrices sont :

$$
R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
R_y = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}
$$

$$
R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix}
$$

Les trois angles θ, ϕ et ψ désignent respectivement le Roulis, le Tangage et le Lacet. La matrice R est donnée par ($C = \cos$, $S = \sin$) :

$$
\begin{aligned}
R &= R_z\, R_y\, R_x \\
&= \begin{bmatrix} C\theta\, C\phi & C\theta\, S\phi\, S\psi - S\theta\, C\psi & C\theta\, S\phi\, C\psi + S\theta\, S\psi \\ S\theta\, C\phi & S\theta\, S\phi\, S\psi + C\theta\, C\psi & S\theta\, S\phi\, C\psi - C\theta\, S\psi \\ -S\phi & C\phi\, S\psi & C\phi\, C\psi \end{bmatrix}
\end{aligned}
$$

1.3 Rotation et quaternion unitaire

Une autre représentation possible des rotations est celle utilisant les quaternions unitaires. L'utilisation des quaternions se justifiera par une formulation très élégante du problème d'optimisation associé à la localisation 3D. Dans un premier temps, nous allons étudier quelques propriétés des quaternions, ensuite nous les utiliserons pour décrire une rotation et nous établirons également l'équivalence avec les autres représentations déjà utilisées [Cas87] [Hor87] [SBE91] [HM93] [Cho92] [DK88].

1.3.1 Les quaternions

les quaternions peuvent être considérés comme des couples $(q_0, \vec{\mathbf{q}})$ où q_0 est un élément de $\mathbb{R}$ et $\vec{\mathbf{q}}$ est un vecteur de dimension 3. L'ensemble de quaternions est alors l'ensemble $\mathbb{R}^4 = \mathbb{R} \times \mathbb{R}^3$.

Les quaternions peuvent être vus également comme des nombres complexes à trois parties imaginaires :

$$\mathbf{q} = q_0 + iq_x + jq_y + kq_z$$

avec :

$$i^2 = j^2 = k^2 = ijk = -1$$

On peut remarquer qu'on obtient également :

$$\begin{aligned} ij &= -ji = k \\ jk &= -kj = i \\ ki &= -ik = j \end{aligned}$$

Grâce à ces formules, on peut facilement calculer le produit de deux quaternions, noté "$*$" :

$$\mathbf{r} * \mathbf{q} = (r_0 + ir_x + jr_y + kr_z)(q_0 + iq_x + jq_y + kq_z)$$

En général, le produit de deux quaternions n'est pas commutatif. Comme le produit vectoriel, le produit de deux quaternions peut s'écrire sous forme matricielle :

$$\mathbf{r} * \mathbf{q} = Q(\mathbf{r})\mathbf{q}$$

avec $Q(\mathbf{r})$ donnée par :

$$Q(\mathbf{r}) = \begin{bmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & -r_z & r_y \\ r_y & r_z & r_0 & -r_x \\ r_z & -r_y & r_x & r_0 \end{bmatrix} \tag{1.4}$$

On peut également écrire le produit sous la forme suivante :

$$\mathbf{q} * \mathbf{r} = W(\mathbf{r})\mathbf{q}$$

avec $W(\mathbf{r})$ définie par :

$$W(\mathbf{r}) = \begin{bmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & r_z & -r_y \\ r_y & -r_z & r_0 & r_x \\ r_z & r_y & -r_x & r_0 \end{bmatrix} \tag{1.5}$$

La matrice $W(\mathbf{r})$ est obtenue à partir de la matrice $Q(\mathbf{r})$ en transposant sa sous-matrice :

$$\begin{bmatrix} . & . & . & . \\ . & r_0 & -r_z & r_y \\ . & r_z & r_0 & -r_x \\ . & -r_y & r_x & r_0 \end{bmatrix}$$

Si le quaternion $\mathbf{r}$ est écrit comme une concaténation d'un réel et d'un vecteur de dimension 3, $\mathbf{r} = (r_0, \vec{\mathbf{r}})$, on peut alors exprimer les matrices $Q(\mathbf{r})$ et $W(\mathbf{r})$ par :

$$Q(\mathbf{r}) = \begin{bmatrix} r_0 & -\vec{\mathbf{r}}^{\,T} \\ \vec{\mathbf{r}} & r_0\, I + K(\vec{\mathbf{r}}) \end{bmatrix}$$

$$W(\mathbf{r}) = \begin{bmatrix} r_0 & -\vec{\mathbf{r}}^{\,T} \\ \vec{\mathbf{r}} & r_0\, I - K(\vec{\mathbf{r}}) \end{bmatrix}$$

$K(\vec{\mathbf{r}})$ est la matrice antisymétrique associée au vecteur $\vec{\mathbf{r}}$:

$$K(\vec{\mathbf{r}}) = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

On peut facilement vérifier que les matrices $Q(\mathbf{r})$ et $W(\mathbf{r})$ possèdent les propriétés suivantes :

$$\begin{aligned} Q(\mathbf{r})^T Q(\mathbf{r}) = Q(\mathbf{r}) Q(\mathbf{r})^T &= \mathbf{r}^T \mathbf{r} I \\ W(\mathbf{r})^T W(\mathbf{r}) = W(\mathbf{r}) W(\mathbf{r})^T &= \mathbf{r}^T \mathbf{r} I \\ Q(\mathbf{r})\mathbf{q} &= W(\mathbf{q})\mathbf{r} \\ Q(\mathbf{r})^T \mathbf{r} = W(\mathbf{r})^T \mathbf{r} &= \mathbf{r}^T \mathbf{r}\, \mathbf{e} \\ Q(\mathbf{r}) Q(\mathbf{q}) &= Q(Q(\mathbf{r})\mathbf{q}) \\ W(\mathbf{r}) W(\mathbf{q}) &= W(W(\mathbf{r})\mathbf{q}) \\ Q(\mathbf{r}) W(\mathbf{q})^T &= W(\mathbf{q})^T Q(\mathbf{r}) \end{aligned}$$

$\mathbf{r}$ et $\mathbf{q}$ étant deux quaternions quelconques, $\mathbf{e}$ étant le quaternion unité : $\mathbf{e} = (1\ 0\ 0\ 0)^T$.

Notons que si nous utilisons la représentation vectorielle d'un quaternion alors le produit de deux quaternions $\mathbf{q} = (q_0, \vec{\mathbf{q}})$ et $\mathbf{r} = (r_0, \vec{\mathbf{r}})$ sera :

$$\mathbf{q} * \mathbf{r} = [q_0 r_0 - \vec{\mathbf{q}} \cdot \vec{\mathbf{r}},\ q_0\, \vec{\mathbf{r}} + r_0\, \vec{\mathbf{q}} + \vec{\mathbf{q}} \times \vec{\mathbf{r}}] \tag{1.6}$$

Si $\mathbf{q}$ et $\mathbf{r}$ sont deux quaternions purement imaginaires (vecteurs de dimension 3) nous aurons les deux égalités suivantes :

$$\vec{\mathbf{q}} \cdot \vec{\mathbf{r}} = -\frac{1}{2}(\mathbf{q} * \mathbf{r} + \mathbf{r} * \mathbf{q})$$

$$\vec{\mathbf{q}} \times \vec{\mathbf{r}} = \frac{1}{2}(\mathbf{q} * \mathbf{r} - \mathbf{r} * \mathbf{q})$$

Le produit scalaire de deux quaternions est donné par :

$$\mathbf{r} \cdot \mathbf{q} = r_0 q_0 + r_x q_x + r_y q_y + r_z q_z$$

On a bien évidemment :

$$\mathbf{q} \cdot \mathbf{q} = \|\mathbf{q}\|^2$$

Le quaternion conjugué de $\mathbf{q}$ est $\overline{\mathbf{q}}$ défini par :

$$\overline{\mathbf{q}} = q_0 - iq_x - jq_y - kq_z$$

Notons que si $Q(\mathbf{q})$ et $W(\mathbf{q})$ sont les matrices associées à $\mathbf{q}$, alors $Q(\mathbf{q})^T$ et $W(\mathbf{q})^T$ seront les matrices associées au quaternion conjugué $\overline{\mathbf{q}}$. On a :

$$\mathbf{q} * \overline{\mathbf{q}} = \mathbf{q} \cdot \mathbf{q} = \|\mathbf{q}\|^2$$

et on obtient ainsi le quaternion inverse de $\mathbf{q}$:

$$\mathbf{q}^{-1} = \frac{1}{\|\mathbf{q}\|^2}\,\overline{\mathbf{q}}$$

On en déduit que l'inverse d'un quaternion unitaire (de norme égale à 1) est égal à son conjugué. Nous allons maintenant établir quelques propriétés du produit qui seront facilement vérifiables si l'on utilise la notation matricielle. On a :

$$(\mathbf{q} * \mathbf{p}) \cdot (\mathbf{q} * \mathbf{r}) = (\mathbf{q} \cdot \mathbf{q})(\mathbf{p} \cdot \mathbf{r}) \tag{1.7}$$

On en déduit facilement les propriétés suivantes :

$$\begin{aligned} (\mathbf{q} * \mathbf{p}) \cdot (\mathbf{q} * \mathbf{p}) &= (\mathbf{q} \cdot \mathbf{q})(\mathbf{p} \cdot \mathbf{p}) \\ \|\mathbf{p} * \mathbf{q}\|^2 &= \|\mathbf{p}\|^2 \|\mathbf{q}\|^2 \\ (\mathbf{p} * \mathbf{q}) \cdot \mathbf{r} &= \mathbf{p} \cdot (\mathbf{r} * \overline{\mathbf{q}}) \end{aligned}$$

Un vecteur de dimension 3 peut s'écrire comme un quaternion purement imaginaire :

$$\mathbf{q} = 0 + iq_x + jq_y + kq_z$$

et les matrices associées à un quaternion purement imaginaire sont des matrices antisymétriques. Par exemple on a :

$$Q(\mathbf{q}) = \begin{bmatrix} 0 & -q_x & -q_y & -q_z \\ q_x & 0 & -q_z & q_y \\ q_y & q_z & 0 & -q_x \\ q_z & -q_y & q_x & 0 \end{bmatrix}$$

avec les propriétés évidentes :

$$\begin{aligned} Q(\mathbf{q})^T &= -Q(\mathbf{q}) \\ W(\mathbf{q})^T &= -W(\mathbf{q}) \end{aligned}$$

1.3.2 Quaternion et rotation

On sait que les transformations géométriques planes de base peuvent être représentées par des opérations agissant sur des nombres complexes. Comme l'ensemble des quaternions contient l'espace $\mathbb{R}^3$, on espère trouver une certaine analogie concernant les transformations géométriques dans $\mathbb{R}^3$. Nous nous intéressons à la rotation.

On peut représenter une rotation avec un quaternion unitaire à condition de pouvoir trouver une transformation qui change un vecteur (un quaternion purement imaginaire) en un vecteur de façon que la transformation préserve la longueur du vecteur transformé ainsi que le produit scalaire et le signe du produit vectoriel.

Soit $\mathbf{r}$ un quaternion purement imaginaire et $\mathbf{q}$ un quaternion unitaire. On peut alors remarquer que le quaternion $\mathbf{r}'$ tel que :

$$\mathbf{r}' = \mathbf{q} * \mathbf{r} * \overline{\mathbf{q}} \tag{1.8}$$

est un quaternion purement imaginaire et de plus on a :

$$\mathbf{q} * \mathbf{r} * \overline{\mathbf{q}} = (Q(\mathbf{q})\mathbf{r}) * \overline{\mathbf{q}} = (W(\mathbf{q})^T Q(\mathbf{q}))\mathbf{r}$$

Par ailleurs notons que :

$$(-\mathbf{q}) * \mathbf{r} * (-\overline{\mathbf{q}}) = \mathbf{q} * \mathbf{r} * \overline{\mathbf{q}}$$

ce qui implique que les quaternions $\mathbf{q}$ et $-\mathbf{q}$ représentent la même rotation. La matrice $W(\mathbf{q})^T Q(\mathbf{q})$ est une matrice orthogonale puisque les deux matrices $W(\mathbf{q})$ et $Q(\mathbf{q})$ sont orthogonales lorsque $\mathbf{q}$ est unitaire :

$$\begin{aligned} [W(\mathbf{q})^T\ Q(\mathbf{q})]^{-1} &= Q(\mathbf{q})^{-1}\ (W(\mathbf{q})^T)^{-1} \\ &= Q(\mathbf{q})^T\ W(\mathbf{q}) \\ &= [W(\mathbf{q})^T\ Q(\mathbf{q})]^T \end{aligned}$$

Elle est donnée par :

$$W(\mathbf{q})^T Q(\mathbf{q}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 0 & 2(q_x q_y + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 0 & 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}$$

On peut maintenant établir l'expression d'une matrice de rotation R en fonction d'un quaternion unitaire $\mathbf{q}$:

$$R = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 2(q_x q_y + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{1.9}$$

et on peut également écrire :

$$W(\mathbf{q})^T Q(\mathbf{q}) = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & R \end{bmatrix}$$

Nous pouvons également extraire un quaternion unitaire à partir d'une matrice de rotation. Soit R_{ij} un élément de la matrice de rotation. En considérant les termes diagonaux de la matrice précédente on obtient les combinaisons suivantes :

$$\begin{aligned} 1 + R_{11} + R_{22} + R_{33} &= 4\,q_0^2 \\ 1 + R_{11} - R_{22} - R_{33} &= 4\,q_x^2 \\ 1 - R_{11} + R_{22} - R_{33} &= 4\,q_y^2 \\ 1 - R_{11} - R_{22} + R_{33} &= 4\,q_z^2 \end{aligned}$$

Ces équations nous donnent les carrés des composantes du quaternion correspondant. On choisit la composante du quaternion ayant la valeur absolue la plus élevée. Ensuite, on injecte la composante choisie dans trois parmi les six expressions suivantes :

$$\begin{aligned} R_{32} - R_{23} &= 4\,q_0 q_x \\ R_{13} - R_{31} &= 4\,q_0 q_y \\ R_{21} - R_{12} &= 4\,q_0 q_z \\ R_{21} + R_{12} &= 4\,q_x q_y \\ R_{32} + R_{23} &= 4\,q_y q_z \\ R_{13} + R_{31} &= 4\,q_z q_x \end{aligned}$$

Cette démarche évite le calcul imprécis des composantes du quaternion puisque les trois autres composantes ne sont plus obtenues par une division de deux nombres dont la valeur absolue est très petite. Notons que le signe de la composante choisie (ayant la valeur absolue la plus élevée) est arbitraire puisque $\mathbf{q}$ et $-\mathbf{q}$ représentent la même rotation.

Remarque

En utilisant les propriétés des matrices Q et W, on peut démontrer la propriété suivante :
Le quaternion unitaire associé au produit de deux rotations est obtenu en multipliant les deux quaternions associés à ces deux matrices de rotation.

démonstration

Soient R_1 et R_2 deux matrices de rotation. Nous désignons par $\mathbf{q}_1$ et $\mathbf{q}_2$ les quaternions unitaires associés à ces deux matrices. La composition de ces deux rotations est donnée par le produit de ces deux matrices. Soit R ce produit, nous avons :

$$\begin{aligned} R &= R_1\,R_2 \\ &= W(\mathbf{q}_1)^T Q(\mathbf{q}_1)\, W(\mathbf{q}_2)^T Q(\mathbf{q}_2) \\ &= W(\mathbf{q}_1)^T W(\mathbf{q}_2)^T\, Q(\mathbf{q}_1) Q(\mathbf{q}_2) \\ &= [W(\mathbf{q}_2) W(\mathbf{q}_1)]^T\, Q(Q(\mathbf{q}_1)\mathbf{q}_2) \\ &= W(W(\mathbf{q}_2)\mathbf{q}_1)^T\, Q(\mathbf{q}_1 * \mathbf{q}_2) \\ &= W(\mathbf{q}_1 * \mathbf{q}_2)^T\, Q(\mathbf{q}_1 * \mathbf{q}_2) \end{aligned}$$

1.3.3 Equivalence avec la formule de *Rodrigues*

Nous venons d'établir l'équivalence entre la représentation matricielle et la représentation quaternion d'une rotation. Il y a une relation très simple entre un quaternion unitaire et l'axe et l'angle d'une rotation. Comme nous venons de l'établir (équation (1.6)) le produit de deux quaternions $\mathbf{p} = (p_0, \vec{\mathbf{p}})$ et $\mathbf{q} = (q_0, \vec{\mathbf{q}})$ peut être exprimé par :

$$\mathbf{p} * \mathbf{q} = [p_0 q_0 - \vec{\mathbf{p}} \cdot \vec{\mathbf{q}},\ p_0 \vec{\mathbf{q}} + q_0 \vec{\mathbf{p}} + \vec{\mathbf{p}} \times \vec{\mathbf{q}}]$$

L'équation (1.8) devient alors :

$$\mathbf{r}' = (0, (q_0^2 - \vec{\mathbf{q}} \cdot \vec{\mathbf{q}})\,\vec{\mathbf{r}} + 2q_0\,\vec{\mathbf{q}} \times \vec{\mathbf{r}} + 2(\vec{\mathbf{q}} \cdot \vec{\mathbf{r}})\,\vec{\mathbf{q}})$$

$\mathbf{r}'$ est donc un quaternion purement imaginaire et sa partie purement imaginaire est :

$$\vec{\mathbf{r}'} = (q_0^2 - \vec{\mathbf{q}} \cdot \vec{\mathbf{q}})\,\vec{\mathbf{r}} + 2q_0\,\vec{\mathbf{q}} \times \vec{\mathbf{r}} + 2(\vec{\mathbf{q}} \cdot \vec{\mathbf{r}})\,\vec{\mathbf{q}}$$

et on peut donc confondre un quaternion purement imaginaire avec un vecteur de dimension 3 :

$$\vec{\mathbf{r}'} = \mathbf{r}'$$

Notons que la formule de *Rodrigues* (équation (1.2)) peut se mettre sous la forme suivante :

$$\vec{\mathbf{r}'} = \cos\theta\ \vec{\mathbf{r}} + \sin\theta\ \vec{\mathbf{n}} \times \vec{\mathbf{r}} + (1 - \cos\theta)\,(\vec{\mathbf{n}} \cdot \vec{\mathbf{r}})\,\vec{\mathbf{n}}$$

En identifiant les deux expressions précédentes de $\vec{\mathbf{r}'}$ et en utilisant les deux égalités suivantes :

$$\begin{aligned} \cos\theta &= (\cos\frac{\theta}{2})^2 - (\sin\frac{\theta}{2})^2 \\ \sin\theta &= 2\ \cos\frac{\theta}{2}\ \sin\frac{\theta}{2} \end{aligned}$$

on obtient :

$$\begin{aligned} q_0 &= \cos\frac{\theta}{2} \\ \vec{\mathbf{q}} &= \sin\frac{\theta}{2}\,\vec{\mathbf{n}} \end{aligned}$$

Le quaternion représentant une rotation d'axe $\mathbf{n}$ et d'angle θ s'écrit donc :

$$\mathbf{q} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\ (i\,n_x + j\,n_y + k\,n_z) \tag{1.10}$$

Enfin, nous pouvons noter qu'à chaque quaternion unitaire correspond une rotation unique.

1.4 Déplacement et vissage

1.4.1 Définition et formulation

Soit une transformation (rotation et/ou translation) dont la représentation matricielle est donnée par la matrice T telle que :

$$T = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Cette transformation amène un repére R_0 sur un repére R_1. Le déplacement correspondant peut être effectué de la façon classique suivante.

Tout d'abord R_0 subit une translation dont le vecteur est égal à $\mathbf{t}$. Ensuite le repère obtenu subit une rotation d'un angle θ autour d'un axe $\mathbf{n}$ passant par l'origine ($\mathbf{n}$ et θ étant la représentation vectorielle de la rotation). L'ordre de la rotation et de la translation peut être inversé.

Ce même déplacement (théorème de Chasles) [Che91] [WSV91] peut être effectué par une translation du repère R_0 d'une distance d le long de l'axe de rotation suivie d'une rotation d'un angle ϕ autour d'une ligne unique ayant $\mathbf{r}$ comme vecteur directeur et passant par le point $\mathbf{p}$ (voir Figure 1.3).

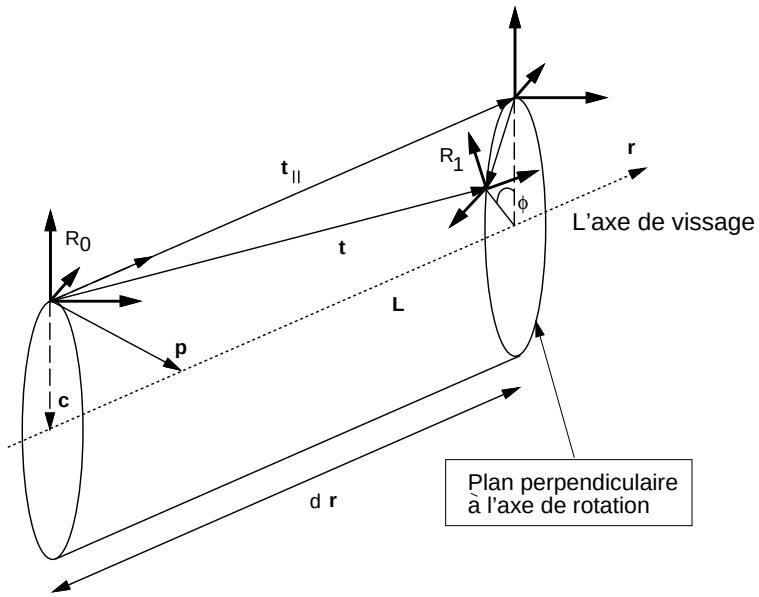


FIG. 1.3 - *Un déplacement donné par le vissage (d, ϕ, $\mathcal{L}$)*

Une telle description est appelée "vissage" et l'axe unique est appelé "axe de vissage", cet axe ne passe pas par l'origine de R_0. Le vissage a été découvert au début du $18^{ème}$ siècle. Depuis il avait constitué un outil adéquat pour la description des mouvements 3D.

Deux quantités sont associées à l'axe de vissage : la distance de translation et l'angle de rotation. Comme l'axe de vissage est une ligne dans l'espace, sa localisation exige quatre paramètres indépendants. On a ainsi six paramètres indépendants définissant complètement le vissage, nombre qui est égal au nombre de degrés de liberté nécessaire pour localiser un corps solide.

Un vissage est donc défini par $(d,\ \phi,\ \mathcal{L})$ où :

1. d est la distance de translation le long de l'axe de vissage.
2. ϕ est l'angle de rotation.
3. $\mathcal{L}$ est l'axe de vissage dont l'équation en représentation paramétrique est donnée par :

$$\mathcal{L}: \quad \mathbf{p} = \mathbf{c} + k\,\mathbf{r} \quad (k \in I\!R)$$

avec :

- $\mathbf{r}$ est le vecteur directeur (unitaire) de l'axe de vissage $\mathcal{L}$.
- $\mathbf{c}$ est la position de cet axe ($\mathbf{c} \cdot \mathbf{r} = 0$).

Il y a une ambiguïté concernant le vecteur directeur de l'axe de vissage. Pour lever cette ambiguïté nous adoptons la convention suivante : le vecteur directeur est choisi de façon qu'il pointe dans le même sens que la translation du vissage d (voir Figure 1.4).

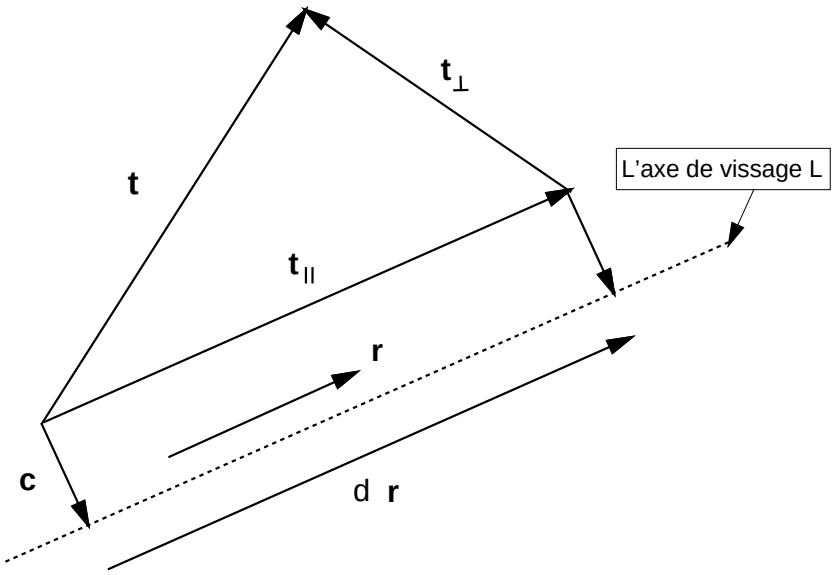


FIG. 1.4 - *Le vecteur directeur* $\mathbf{r}$ *de l'axe de vissage est choisie de telle sorte qu'il ait le même sens que celui de la composante* $\mathbf{t}_{||}$.

1.4.2 Conversion entre $(R,\ \mathbf{t}\ )$ et $(d,\ \phi,\ \mathcal{L})$

Nous décrivons ici la conversion entre la représentation classique donnée par la matrice de transformation homogène $(R, \mathbf{t}\ )$ et la représentation vissage donnée par les paramètres $(d,\ \phi,\ \mathcal{L})$.

Comme nous venons de l'établir dans le paragraphe 1.2.2, on peut extraire l'axe $\mathbf{n}$ et l'angle θ de rotation à partir de la matrice R.

Si nous désignons par $\mathbf{t}_{\|}$ et $\mathbf{t}_{\perp}$ respectivement les composantes parallèle et perpendiculaire à $\mathbf{n}$ du vecteur $\mathbf{t}$, nous aurons :

$$\mathbf{t}_{\|} = (\mathbf{t} \cdot \mathbf{n})\,\mathbf{n} \tag{1.11}$$

$$\mathbf{t}_{\perp} = \mathbf{t} - \mathbf{t}_{\|}$$

Un vecteur de position $\mathbf{p}$ subissant la transformation $(R, \mathbf{t})$ devient un vecteur $\mathbf{p}'$ tel que :

$$\mathbf{p}' = R\,\mathbf{p} + \mathbf{t}_{\|} + \mathbf{t}_{\perp} \tag{1.12}$$

Comme l'axe de vissage ne passe pas par l'origine, nous pouvons faire la translation parallèle à l'axe de rotation, ceci doit être effectué en déplaçant l'axe de rotation de l'origine de telle sorte que la composante perpendiculaire $\mathbf{t}_{\perp}$ soit absorbée par la rotation. Pour effectuer cette absorption, le vecteur de déplacement désigné par $\mathbf{c}$ doit être orthogonal à l'axe de rotation $(\mathbf{c} \cdot \mathbf{n} = 0)$.

Le déplacement de l'axe de rotation peut être vu comme une translation de vecteur $\mathbf{c}$, ce qui donne :

$$\mathbf{p}' - \mathbf{c} = R\,(\mathbf{p} - \mathbf{c}) + a\,\mathbf{n} \quad \text{où } a \text{ est un nombre réel.}$$

Le terme $a\,\mathbf{n}$ représente une translation le long de l'axe de rotation. En réécrivant la dernière équation comme :

$$\mathbf{p}' = R\,\mathbf{p} + (I - R)\,\mathbf{c} + a\,\mathbf{n}$$

et par identification avec l'équation (1.12), nous aurons :

$$\mathbf{t}_{\|} = a\,\mathbf{n} \tag{1.13}$$

$$\mathbf{t}_{\perp} = (I - R)\,\mathbf{c} \tag{1.14}$$

L'équation (1.13) implique que la translation d'un vissage est égale à la composante parallèle du $\mathbf{t}$. L'importance de l'équation (1.14) est illustrée par la Figure 1.5.

Avec les notations suivantes :

$$\begin{aligned} \overrightarrow{OD} &= \mathbf{t}_{\perp} \\ \overrightarrow{DC} &= R\,\mathbf{c} \\ \overrightarrow{OC} &= \mathbf{c} \end{aligned}$$

Nous aurons :

$$\overrightarrow{DC} = R\,\overrightarrow{OC} \Longrightarrow \|\overrightarrow{DC}\| = \|R\,\overrightarrow{OC}\|$$

Le triangle OCD est isocèle et $\widehat{OCD} = \theta$. Le point O peut être transformé en D par la rotation R si l'axe de rotation passe par le point C d'où l'absorption de la composante perpendiculaire $\mathbf{t}_{\perp}$.

Les propriétés citées ci-dessus conduisent à une méthode simple pour le calcul du vecteur de déplacement $\mathbf{c}$: dans le plan contenant $\vec{OD}$ et perpendiculaire à $\mathbf{n}$, on cherche

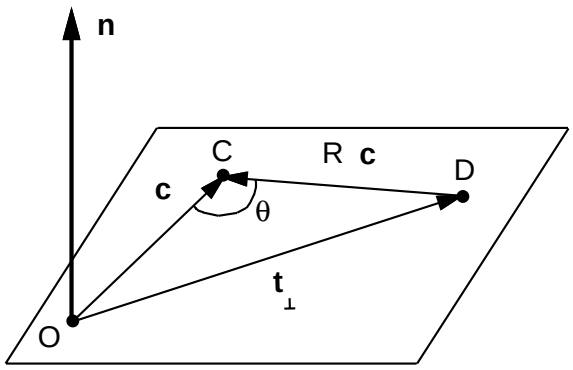


FIG. 1.5 - *Dans un plan perpendiculaire à l'axe de rotation* $\mathbf{n}$ *les trois vecteurs* $\mathbf{t}_\perp$, $\mathbf{c}$ *et* $R\,\mathbf{c}$ *forment un triangle isocèle avec* $\widehat{OCD} = \theta$.

un point C appartenant à la médiatrice de $\vec{OD}$ tel que $\widehat{OCD} = \theta$. On a ainsi deux points, de part et d'autre de $\vec{OD}$, vérifiant cette propriété mais seulement un point a pour sens celui de la rotation R.

D'après la Figure 1.5, nous pouvons calculer ce vecteur qui est alors donné par:

$$\mathbf{c} = [\mathbf{t}_\perp + (\mathbf{n} \times \mathbf{t}_\perp)\cot(\theta/2)]/2 \tag{1.15}$$

En résumé, le calcul de $(d,\, \phi,\, \mathcal{L})$ à partir de $(R,\, \mathbf{t})$ consiste tout d'abord à déterminer i) $\mathbf{t}_\parallel$ à partir de l'équation (1.11) et ii) $\mathbf{c}$ à partir de l'équation (1.15). Ensuite la translation du vissage, l'angle de rotation et le vecteur directeur de l'axe de vissage seront donnés par :

$$\begin{cases} d &= \mathbf{t}\cdot\mathbf{n} \\ \phi &= \theta \qquad\qquad \text{si} \quad \mathbf{t}\cdot\mathbf{n} \geq 0 \\ \mathbf{r} &= \mathbf{n} \end{cases}$$

$$\begin{cases} d &= -(\mathbf{t}\cdot\mathbf{n}) \\ \phi &= 2\pi - \theta \qquad \text{si} \quad \mathbf{t}\cdot\mathbf{n} < 0 \\ \mathbf{r} &= -\mathbf{n} \end{cases}$$

Le calcul de $(R,\, \mathbf{t})$ à partir de $(d,\, \phi,\, \mathcal{L})$ est très simple. Nous avons l'axe et l'angle de rotation, ce qui donne la matrice de rotation R en utilisant l'équation (1.3), le vecteur de translation $\mathbf{t}$ sera donné par :

$$\mathbf{t} = \mathbf{t}_\parallel + \mathbf{t}_\perp = d\,\mathbf{r} + (I - R)\mathbf{c}$$

1.5 Déplacement et quaternion dual

Une autre représentation possible des déplacements est celle utilisant les quaternions duaux. Dans un premier temps nous allons étudier quelques propriétés des nombres duaux, ensuite nous utiliserons les quaternions duaux pour décrire un déplacement [WSV91] [SMH91].

1.5.1 Les nombres duaux

Un nombre dual $\hat{a}$ peut être défini comme un couple de deux nombres réels a et b tel que :

$$\hat{a} = a + \epsilon\, b$$

où ϵ est l'unité duale ($\epsilon \neq 0$) et possède les propriétés de multiplication suivantes :

$$\begin{aligned} 0 \cdot \epsilon &= \epsilon \cdot 0 = 0 \\ 1 \cdot \epsilon &= \epsilon \cdot 1 = \epsilon \\ \epsilon^{\,2} &= 0 \end{aligned}$$

Les règles d'addition et de multiplication de deux nombres duaux sont les suivantes :

$$\begin{aligned} (a + \epsilon\, b) + (c + \epsilon\, d) &= (a + c) + \epsilon\,(b + d) \\ (a + \epsilon\, b) - (c + \epsilon\, d) &= (a - c) + \epsilon\,(b - d) \\ (a + \epsilon\, b) \cdot (c + \epsilon\, d) &= (ac) + \epsilon\,(ad + bc) \\ (a + \epsilon\, b)/(c + \epsilon\, d) &= (a/c) + \epsilon\,(b/c - ad/c^2) \quad (\text{pour } c \neq 0) \end{aligned}$$

Un nombre dual peut spécifier complètement la position relative de deux lignes dans l'espace. Dans ce cas, il est appelé angle dual et il est défini par :

$$\hat{\theta} = \theta + \epsilon\, d \tag{1.16}$$

Avec :

- θ est l'angle entre les directions de deux lignes.
- d est la distance entre ces deux lignes.

Quelques proriétés des nombres duaux

1. Le produit d'un nombre dual $\hat{a}$ avec son conjugué $\overline{\hat{a}} = a - \epsilon\, b$ est égal à a^2. En effet, on a :

$$\begin{aligned} \hat{a} \cdot \overline{\hat{a}} &= (a + \epsilon\, b)(a - \epsilon\, b) \\ &= a^2 + \epsilon\,(ab - ba) \\ &= a^2 \end{aligned}$$

2. Le module d'un nombre dual est donné par :

$$|\hat{a}| = a$$

Ce module peut être négatif.

3. Si $f(\hat{a})$ est une fonction différentiable alors le développement de cette fonction en série de Taylor au voisinage de a sera :

$$f(\hat{a}) = f(a) + \epsilon\, b f'(a) + \frac{\epsilon^2 b^2}{2} f''(a) + \cdots$$

Mais comme $\epsilon^2 = \epsilon^3 = \cdots = 0$, l'expression de $f(\hat{a})$ devient :

$$f(\hat{a}) = f(a) + \epsilon\, b f'(a)$$

4. Pour l'angle dual que nous venons de définir par l'équation (1.16) nous avons :

$$\sin\hat{\theta} = \sin(\theta + \epsilon\, d) = \sin\theta + \epsilon\, d\cos\theta \tag{1.17}$$

$$\cos\hat{\theta} = \cos(\theta + \epsilon\, d) = \cos\theta - \epsilon\, d\sin\theta \tag{1.18}$$

Cette propriété (les deux équations précédentes) est déduite à partir de la précédente.

L'idée des quantités duales peut être étendue pour définir ainsi des vecteurs duaux, des quaternions duaux et des matrices duales. Ces quantités duales combinent deux différentes quantités en une seule. Un vecteur dual, par exemple, peut représenter une droite dans l'espace. La direction et la position d'une droite sont données par le vecteur dual $\hat{\mathbf{n}}$ tel que :

$$\hat{\mathbf{n}} = \mathbf{n} + \epsilon\, \mathbf{p} \times \mathbf{n}$$

- $\mathbf{n}$ est le vecteur directeur de la droite.
- $\mathbf{p}$ est un point appartenant à cette droite.

1.5.2 Déplacement et quaternion dual

Un quaternion dual est composé de deux quaternions ordinaires $\mathbf{q}$ et $\mathbf{s}$:

$$\hat{\mathbf{q}} = \mathbf{q} + \epsilon\, \mathbf{s} \tag{1.19}$$

$\mathbf{q}$ et $\mathbf{s}$ sont appelés respectivement partie réelle et partie duale. L'interprétation géométrique d'un quaternion dual est exactement la même que celle d'un vissage. Un quaternion dual a la même forme qu'un quaternion ordinaire :

$$\hat{\mathbf{q}} = \begin{bmatrix} \cos\frac{\hat{\phi}}{2} \\ \sin\frac{\hat{\phi}}{2}\,\hat{\mathbf{r}} \end{bmatrix} \tag{1.20}$$

Avec :

- $\hat{\mathbf{r}}$ représente la ligne du vissage.
- $\hat{\phi}$ représente l'angle dual de ce vissage.

Les expressions de $\hat{\mathbf{r}}$ et de $\hat{\phi}$ en fonction des paramètres du vissage sont données par :

$$\begin{aligned} \hat{\mathbf{r}} &= \mathbf{r} + \epsilon\, \mathbf{p} \times \mathbf{r} \\ \hat{\phi} &= \phi + \epsilon\, d \end{aligned}$$

- $\mathbf{r}$ est le vecteur directeur de l'axe de vissage.
- $\mathbf{p}$ est un point appartenant à l'axe de vissage.
- ϕ est l'angle de rotation.
- d est la distance de translation le long de la direction $\mathbf{r}$.

Nous pouvons montrer qu'à chaque ensemble ($\mathbf{r}$, $\mathbf{p}$, d, ϕ) nous pouvons trouver une transformation unique ($\mathbf{t}$, $\mathbf{r}$, ϕ). D'autre part, à chaque transformation ($\mathbf{t}$, $\mathbf{r}$, ϕ) nous pouvons trouver un ensemble de solutions ($\mathbf{r}$, $\mathbf{p}$, d, ϕ) puisque chaque point appartenant à la ligne de vissage constitue une solution pour le vecteur $\mathbf{p}$.

En remplaçant $\hat{\mathbf{r}}$ et $\hat{\phi}$ par leur valeur dans l'expression de $\hat{\mathbf{q}}$ donnée par l'équation (1.20) et en utilisant les deux équations (1.17) et (1.18), nous aurons après identification avec l'équation (1.19) :

$$\mathbf{q} = \begin{bmatrix} \cos\frac{\phi}{2} \\ \sin\frac{\phi}{2}\,\mathbf{r} \end{bmatrix} \quad \text{et} \quad \mathbf{s} = \begin{bmatrix} -d\sin\frac{\phi}{2} \\ (d/2)\cos\frac{\phi}{2}\,\mathbf{r} + \sin\frac{\phi}{2}\,\mathbf{p} \times \mathbf{r} \end{bmatrix} \tag{1.21}$$

Un quaternion dual possède 8 éléments mais comme le nombre minimum de variables nécessaire à la représentation d'une situation 3D est égal à 6, cela implique qu'il y a deux éléments, parmi les 8, qui ne sont pas indépendants. En effet, nous pouvons montrer que les composantes d'un quaternion dual vérifient les deux contraintes suivantes :

$$\begin{aligned} \mathbf{q} \cdot \mathbf{q} &= 1 \\ \mathbf{q} \cdot \mathbf{s} &= 0 \end{aligned}$$

De même, on peut trouver une définition de la multiplication de deux quaternions duaux dans [HN92] :

$$\begin{aligned} \hat{\mathbf{q}}_1\, \hat{\mathbf{q}}_2 &= (\mathbf{q}_1 + \epsilon\, \mathbf{s}_1)\,(\mathbf{q}_2 + \epsilon\, \mathbf{s}_2) \\ &= \mathbf{q}_1 * \mathbf{q}_2 + \epsilon\,(\mathbf{q}_1 * \mathbf{s}_2 + \mathbf{s}_1 * \mathbf{q}_2) \end{aligned}$$

1.5.3 Equivalence avec la représentation matricielle

Il s'agit de calculer la matrice de transformation homogène (R, $\mathbf{t}$) en fonction du quaternion dual correspondant $\hat{\mathbf{q}}$.

L'équation (1.21) montre que la partie réelle $\mathbf{q}$ du quaternion dual n'est autre chose que le quaternion décrivant la rotation ($\mathbf{r}$, ϕ). Nous pouvons ainsi calculer la matrice de rotation R par la relation suivante :

$$W(\mathbf{q})^T Q(\mathbf{q}) = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & R \end{bmatrix}$$

Le vecteur de translation est lié aux composantes du quaternion dual par la relation suivante :

$$\mathbf{t} = 2\, W(\mathbf{q})^T \mathbf{s} \tag{1.22}$$

Pour démontrer cette relation nous adoptons la représentation vectorielle pour les quaternions $\mathbf{q}$ et $\mathbf{s}$:

$$\begin{aligned} \mathbf{q} &= (q_0, \vec{\mathbf{q}}) \\ \mathbf{s} &= (s_0, \vec{\mathbf{s}}) \end{aligned}$$

Soit un point $\mathbf{p}$ appartenant à l'axe de vissage, $\mathbf{p}'$ sa transformée par le vissage (d, ϕ, $\mathcal{L}$) correspondant au quaternion dual $\hat{\mathbf{q}}$. Le point $\mathbf{p}'$ est obtenu par une translation de $\mathbf{p}$ d'une distance d le long de l'axe de vissage $\mathcal{L}$ suivie d'une rotation d'un angle ϕ autour de cet axe.

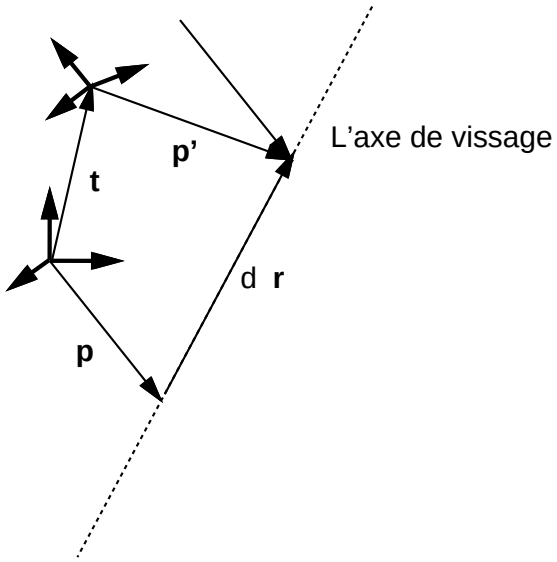


FIG. 1.6 - *Le vecteur* $\boldsymbol{p}$*, ayant son extrémité sur l'axe de vissage et subissant un vissage* (d, ϕ, $\mathcal{L}$) *devient un vecteur* $\boldsymbol{p}'$ *tel que :* $\boldsymbol{p}' = \boldsymbol{p} + d\,\boldsymbol{r} - \boldsymbol{t}$

A partir de la Figure 1.6 on peut écrire :

$$\mathbf{t} = \mathbf{p} + d\,\mathbf{r} - \mathbf{p}'$$

Mais comme $\mathbf{p}' = R\,\mathbf{p}$, on obtient :

$$\mathbf{t} = (I - R)\,\mathbf{p} + d\,\mathbf{r} \tag{1.23}$$

La formule de *Rodrigues* donne l'expression de R en fonction de $\mathbf{r}$ et ϕ :

$$\begin{aligned} R &= I + (1 - \cos\phi)\, K^2(\mathbf{r}) + \sin\phi\, K(\mathbf{r}) \\ &= I + 2\sin^2\frac{\phi}{2}\, K^2(\mathbf{r}) + \sin\phi\, K(\mathbf{r}) \end{aligned}$$

En remplaçant cette dernière expression dans l'équation (1.23), on obtient :

$$\mathbf{t} = 2\sin^2\frac{\phi}{2}\,\mathbf{r} \times (\mathbf{p} \times \mathbf{r}) + \sin\phi\,(\mathbf{p} \times \mathbf{r}) + d\,\mathbf{r} \tag{1.24}$$

Le développement de $W(\mathbf{q})^T\mathbf{s}$ donne :

$$W(\mathbf{q})^T\mathbf{s} = \begin{bmatrix} q_0 & \vec{\mathbf{q}}^{\,T} \\ -\vec{\mathbf{q}} & q_0 I + K(\vec{\mathbf{q}}) \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ \vec{\mathbf{s}} \end{bmatrix} = \begin{bmatrix} q_0 s_0 + \vec{\mathbf{q}}^{\,T}\vec{\mathbf{s}} \\ -s_0\vec{\mathbf{q}} + q_0\vec{\mathbf{s}} + \vec{\mathbf{q}} \times \vec{\mathbf{s}} \end{bmatrix}$$

Mais comme :

$$q_0 s_0 + \vec{\mathbf{q}}^{\,T}\vec{\mathbf{s}} = \mathbf{q} \cdot \mathbf{s} = 0$$

on obtient :

$$W(\mathbf{q})^T\mathbf{s} = q_0\vec{\mathbf{s}} - s_0\vec{\mathbf{q}} + \vec{\mathbf{q}} \times \vec{\mathbf{s}} \tag{1.25}$$

D'autre part, l'équation (1.21) donne :

$$\begin{aligned} q_0\vec{\mathbf{s}} - s_0\vec{\mathbf{q}} &= \cos\frac{\phi}{2}\,[\frac{d}{2}\cos\frac{\phi}{2}\,\mathbf{r} + \sin\frac{\phi}{2}\,\mathbf{p} \times \mathbf{r}] + \frac{d}{2}\sin\frac{\phi}{2}\,[\sin\frac{\phi}{2}\,\mathbf{r}] \\ &= \frac{d}{2}\,\mathbf{r} + \frac{1}{2}\sin\phi\,\mathbf{p} \times \mathbf{r} \end{aligned} \tag{1.26}$$

et

$$\begin{aligned} \vec{\mathbf{q}} \times \vec{\mathbf{s}} &= [\sin\frac{\phi}{2}\,\mathbf{r}] \times [\frac{d}{2}\cos\frac{\phi}{2}\,\mathbf{r} + \sin\frac{\phi}{2}\,\mathbf{p} \times \mathbf{r}] \\ &= \sin^2\frac{\phi}{2}\,\mathbf{r} \times (\mathbf{p} \times \mathbf{r}) \end{aligned} \tag{1.27}$$

En utilisant les deux égalités (1.26) et (1.27) dans l'équation (1.24) on obtient :

$$\mathbf{t} = 2\,(q_0\vec{\mathbf{s}} - s_0\vec{\mathbf{q}}) + 2\,(\vec{\mathbf{q}} \times \vec{\mathbf{s}})$$

et par identification avec l'équation (1.25) on obtient bien l'équation (1.22).

Le calcul du quaternion dual $\hat{\mathbf{q}}$ en fonction de $(R,\, \mathbf{t})$ s'effectue de la façon suivante : on détermine le quaternion réel $\mathbf{q}$ à partir de la matrice de rotation R (voir section 1.3.2). Une fois qu'on a déterminé ce quaternion, la partie duale $\mathbf{s}$ est donnée par :

$$\mathbf{s} = \frac{1}{2}\,W(\mathbf{q})\,\mathbf{t}$$

1.6 Conclusion

Dans ce chapitre, nous avons présenté quelques paramétrisations des déplacements rigides qui sont utiles pour les prochains chapitres.

De ce chapitre, nous retenons en particulier l'importance des quaternions et des quaternions duaux dans les problèmes de calibrage et de localisation. Ces paramètres constituent une paramétrisation minimale sans singularité, c'est-à-dire, il n'y a pas de dégénération dans la correspondance quaternion/déplacement.

A. Tableau récapitulatif

représentation	rotation	déplacement
matricielle	R	R, $\mathbf{t}$
vectorielle	$\mathbf{n}$, θ	$\mathbf{r}$, $\mathbf{c}$, ϕ, d
quaternion	$\mathbf{q}$	$\mathbf{q}$, $\mathbf{s}$
mixte	-	$\mathbf{n}$, θ, $\mathbf{t}$ / $\mathbf{q}$, $\mathbf{t}$

B. Formules de passage

De ($\mathbf{n}$, θ) à $\boldsymbol{R}$

$$R = \begin{bmatrix} c+(1-c)n_x^2 & -n_z s+(1-c)n_x n_y & n_y s+(1-c)n_x n_z \\ n_z s+(1-c)n_x n_y & c+(1-c)n_y^2 & -n_x s+(1-c)n_z n_y \\ -n_y s+(1-c)n_x n_z & n_x s+(1-c)n_z n_y & c+(1-c)n_z^2 \end{bmatrix}$$

De $\boldsymbol{R}$ à ($\mathbf{n}$, θ)

$$\cos\theta = \frac{1}{2}\left(R_{11}+R_{22}+R_{33}-1\right) \quad \text{avec} \quad 0 \le \theta \le \pi$$

$$n_x = (R_{32}-R_{23})/2\sin\theta$$
$$n_y = (R_{13}-R_{31})/2\sin\theta$$
$$n_z = (R_{21}-R_{12})/2\sin\theta$$

De ($\mathbf{n}$, θ, $\mathbf{t}$) à ($\mathbf{r}$, $\mathbf{c}$, ϕ, d)

$$\mathbf{c} = [\mathbf{t}_\perp + (\mathbf{n}\times\mathbf{t}_\perp)\cot(\theta/2)]/2$$

$$\begin{cases} d &= \mathbf{t}\cdot\mathbf{n} \\ \phi &= \theta \qquad\qquad \text{si} \quad \mathbf{t}\cdot\mathbf{n} \ge 0 \\ \mathbf{r} &= \mathbf{n} \end{cases}$$

$$\begin{cases} d &= -(\mathbf{t}\cdot\mathbf{n}) \\ \phi &= 2\pi-\theta \qquad \text{si} \quad \mathbf{t}\cdot\mathbf{n} < 0 \\ \mathbf{r} &= -\mathbf{n} \end{cases}$$

De q à *R*

$$R = W(\mathbf{q})^T Q(\mathbf{q})$$

De *R* à q

$$1 + R_{11} + R_{22} + R_{33} = 4\,q_0^2$$

$$1 + R_{11} - R_{22} - R_{33} = 4\,q_x^2$$
$$1 - R_{11} + R_{22} - R_{33} = 4\,q_y^2$$
$$1 - R_{11} - R_{22} + R_{33} = 4\,q_z^2$$

$$R_{32} - R_{23} = 4\,q_0 q_x$$
$$R_{13} - R_{31} = 4\,q_0 q_y$$
$$R_{21} - R_{12} = 4\,q_0 q_z$$
$$R_{21} + R_{12} = 4\,q_x q_y$$
$$R_{32} + R_{23} = 4\,q_y q_z$$
$$R_{13} + R_{31} = 4\,q_z q_x$$

De (n, θ) à q

$$\mathbf{q} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\ (i\,n_x + j\,n_y + k\,n_z)$$

De $\hat{\mathbf{q}} = (\mathbf{q}, \mathbf{s})$ à t

$$\mathbf{t} = 2\,W(\mathbf{q})^T\,\mathbf{s}$$

De t à s

$$\mathbf{s} = \frac{1}{2}\,W(\mathbf{q})\,\mathbf{t}$$

Chapitre 2

Calibrage Capteur/Robot

2.1 Introduction

L'enjeu majeur des systèmes de vision par ordinateur est l'analyse et la compréhension de l'environnement 3D qui nous entoure en vue d'applications comme le contrôle industriel ou la navigation autonome de robots. La vision active, qui est une branche de la vision par ordinateur, est caractérisée par sa capacité à contrôler ses paramètres pour extraire au mieux les informations visuelles de la scène. Parmi ces paramètres, on peut citer la position du capteur dans son environnement. La caractéristique principale de la vision active est le mouvement du capteur. Dans la plupart des cas, ce mouvement est réalisé avec l'aide d'un système robotique. Le capteur embarqué possède ainsi le même nombre de degrés de liberté que l'effecteur du robot considéré. On accède ainsi à des informations sensorielles directes sur l'interaction qui relie le robot à son environnement local.

Dans la suite, nous nous plaçons dans le cas où le capteur est formé d'une ou plusieurs caméras. Il est donc très important de connaître la relation géométrique entre la caméra et l'effecteur du robot. Lorsque la vision est utilisée pour connaître la relation géométrique entre les objets 3D et la base du robot, les capteurs visuels fournissent ces mesures dans des repères attachés à ces capteurs. La connaissance de la relation géométrique entre la caméra et l'effecteur va nous permettre d'exprimer ces mesures dans le repère de l'effecteur. Enfin, ces mesures pourraient être exprimées par rapport à la base du robot si l'on connaît le modèle géométrique direct du robot.

De plus, la reconstruction d'une scène 3D à partir d'une caméra mobile [BMV93] va être considérablement simplifiée si l'on connaît la transformation caméra-effecteur. En effet, si cette transformation est connue on peut déduire les déplacements de la caméra à partir des déplacements de l'effecteur.

Un autre domaine intéressant où ce calibrage s'avère très important est celui de l'asservissement visuel [ECR92] [Cor93b] [SG95] [ZFL93] [CMCK93] [KD94]. On cherche à commander le mouvement 3D de la caméra à partir des informations visuelles. La connaissance de la relation géométrique caméra-effecteur va nous permettre d'obtenir précisément le mouvement correspondant du robot.

Dans ce chapitre, nous appelons "calibrage caméra-pince" le processus avec lequel on détermine la relation géométrique entre une caméra embarquée et l'effecteur du robot. Notons que le calibrage "caméra-pince" s'applique à un large éventail de problèmes dans

lesquels l'inconnue est la transformation rigide entre deux solides qui sont rigidement liés (voir Figures 2.1 et 2.2). Par exemple, le calibrage d'une tête stéréo peut être réalisé avec l'aide du calibrage caméra-pince. Les deux caméras de la tête constituent les deux solides liés.

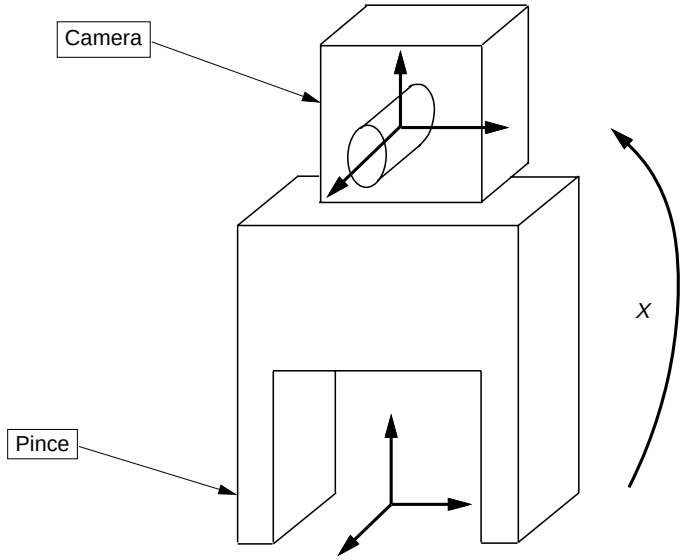


FIG. 2.1 - *Le système caméra-pince est formé par la caméra qui est, soit saisie par la pince, soit montée sur cette pince. Le calibrage caméra-pince consiste à estimer la transformation rigide (rotation et translation) entre un repère attaché à la caméra et un repère attaché à la pince.*

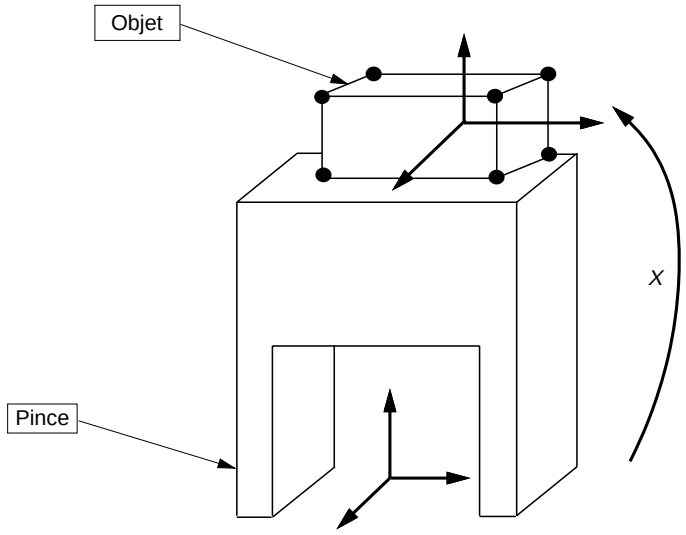


FIG. 2.2 - *Un exemple où le calibrage caméra-pince peut être appliqué pour déterminer la transformation rigide entre une pince et un objet.*

2.2 Formulation

Le problème du calibrage caméra-pince consiste à déterminer la transformation rigide (rotation et translation) entre une caméra montée sur la pince d'un robot et la pince elle même. En d'autres termes, on cherche à déterminer la transformation rigide entre le repère de la caméra et le repère de la pince.

2.2.1 Formulation classique

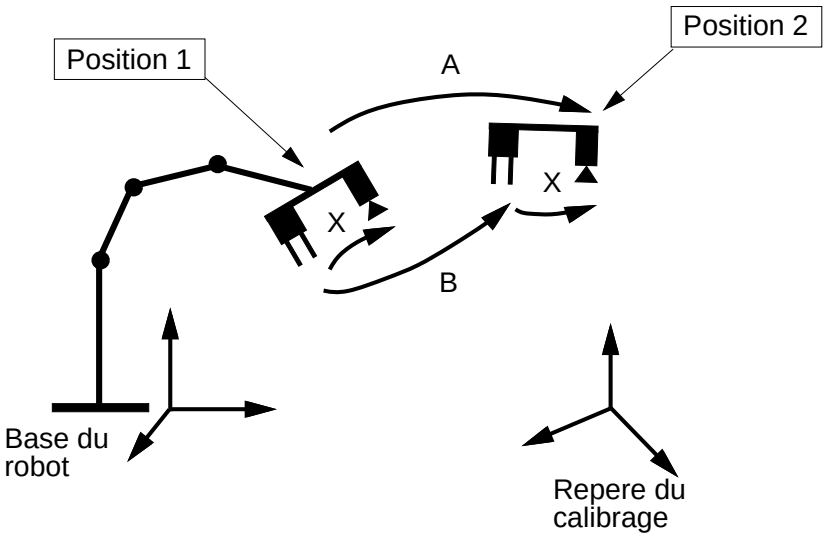


FIG. 2.3 - *La relation géométrique, notée X, entre le repère de la caméra et le repère de la pince est déterminée par la résolution d'un système d'équations homogène de la forme $AX = XB$, où B représente le déplacement de la pince du robot, A représente le déplacement résultant de la caméra.*

Considérons deux positions différentes de la pince du robot (la caméra est rigidement attachée à la pince). Ces deux positions sont dénotées position 1 et position 2 (voir Figure 2.3). Soit A (respectivement B) le déplacement rigide du repère caméra (du repère pince) entre ces deux positions. Soit X la transformation repère pince-repère caméra. A, B et X vérifient l'équation suivante :

$$AX = XB \tag{2.1}$$

Ces matrices, de dimension 4×4, ont la forme suivante :

$$A = \begin{bmatrix} R_A & \mathbf{t}_A \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Dans cette expression, R_A est une matrice orthogonale, de dimension 3×3, et représente une rotation, $\mathbf{t}_A$ est un vecteur colonne décrivant une translation.

Dans ce chapitre, nous adoptons la convention suivante : la matrice T (A, B, X, Y, ...) est la transformation homogène *du* repère b *vers* le repère a :

$$\begin{bmatrix} x_a \\ y_a \\ z_a \\ 1 \end{bmatrix} = T \begin{bmatrix} x_b \\ y_b \\ z_b \\ 1 \end{bmatrix}$$

où l'indice a indique que les coordonnées 3D sont exprimées dans le repère a. La matrice A est obtenue en effectuant deux fois le calibrage de la caméra. Ce calibrage se fait par le biais de l'introduction d'un repère supplémentaire fixe : le repère objet dans lequel la scène vue par la caméra est référencée. Ce repère est également appelé "repère du calibrage" ou "repère mire". Soient A_1 et A_2 les transformations repère calibrage-repère caméra associés aux deux positions du système caméra-pince. Nous avons :

$$A = A_2 A_1^{-1} \tag{2.2}$$

La matrice B est obtenue en faisant déplacer la pince du robot de la position 1 vers la position 2. Soient B_1 et B_2 les transformations pince-base du robot associées à ces deux positions. Nous avons :

$$B = B_2^{-1} B_1 \tag{2.3}$$

Les matrices B_i sont obtenues à partir du modèle géométrique du robot, c'est-à-dire la relation entre les coordonnées articulaires et la position cartésienne de la pince.

2.2.2 Etat de l'art

La formulation du problème du calibrage est très claire, mais l'aspect mathématique de ce problème n'est pas facile à cause des paramètres non linéaires des matrices de rotation. De nombreux travaux ont déjà été effectués dans ce domaine. Ils ont essayé de résoudre l'équation $(AX = XB)$ en la décomposant en deux équations : une équation matricielle qui dépend des rotations et une équation vectorielle qui dépend des rotations et des translations :

$$R_A R_X = R_X R_B \tag{2.4}$$

et:

$$(R_A - I)\,\mathbf{t}_X = R_X\,\mathbf{t}_B - \mathbf{t}_A \tag{2.5}$$

Dans cette équation I est la matrice identité de dimension 3×3.

Une fois la rotation estimée, l'estimation de la translation devient un problème trivial : il suffit de résoudre au sens des moindres carrés le système linéaire donné par l'équation (2.5).

La résolution de l'équation (2.4) est loin d'être triviale mais les propriétés algébriques et géométriques des matrices de rotation permettent d'aboutir. En effet, cette équation peut être écrite de la façon suivante :

$$R_A = R_X R_B R_X^T$$

qui représente une transformation similaire puisque la matrice R_X est orthogonale. Ainsi, les matrices R_A et R_B ont les mêmes valeurs propres puisqu'elles sont deux matrices semblables. On sait qu'une matrice de rotation a l'une de ses valeurs propres qui est égale à 1. Soit $\mathbf{n}_B$ le vecteur propre de R_B associé à la valeur propre 1. En multipliant à droite les deux termes de l'équation (2.4) par $\mathbf{n}_B$, on obtient :

$$\begin{aligned} R_A R_X \mathbf{n}_B &= R_X R_B \mathbf{n}_B \\ &= R_X \mathbf{n}_B \end{aligned}$$

On en déduit que le vecteur $R_X \mathbf{n}_B$ est le vecteur propre de la matrice de rotation R_A associé à la valeur propre 1.

$$\mathbf{n}_A = R_X \mathbf{n}_B \tag{2.6}$$

Dans [SA89], la solution pour la rotation est donnée en résolvant un système linéaire de neuf équations en 4 inconnues. Ils considèrent deux déplacements du système caméra-pince. Si l'on considère un autre déplacement, alors le nombre d'inconnues va augmenter de deux puisque les auteurs considèrent que le cosinus et le sinus comme indépendants.

Dans [PM94], les auteurs utilisent la théorie des algèbres de Lie. Ils proposent une solution analytique qui fournit directement la matrice orthogonale représentant la rotation. Dans [CK91], les auteurs représentent la rotation avec un quaternion unitaire. Ils utilisent la décomposition en valeurs singulières pour résoudre les systèmes linéaires obtenus. Tsai & Lenz [TL89] proposent de représenter la rotation par son axe et son angle de rotation. Ils utilisent les propriétés algébriques et géométriques des matrices de rotation pour obtenir un système linéaire en trois inconnues. Cette méthode sera exposée en détails dans le paragraphe 2.6.1. Wang [Wan92] a proposé trois méthodes analytiques qui ressemblent à la méthode de Tsai.

Tous ces travaux et les travaux présentés dans [Che91] [CLA93] [NTG92] [ZR91] ont étudié les conditions dans lesquelles le système homogène ($AX = XB$) possède une solution unique : *le robot doit exécuter, au moins, deux déplacements dont les axes de rotation sont différents.*

Toutes ces méthodes ont les deux points communs suivants :

- l'estimation de la rotation est séparée de l'estimation de la translation ;
- l'utilisation de solutions analytiques dans l'estimation de la rotation et de la translation.

Remarque

Il existe une méthode simple pour calculer la rotation R_X entre la pince et la caméra : supposons que le déplacement de la pince est une translation pure, on a alors $R_B = R_A = I$ et l'équation (2.5) se réécrit :

$$R_X \mathbf{t}_B = \mathbf{t}_A$$

Ainsi deux translations perpendiculaires de la pince permettent-elles de déterminer la matrice R_X : par exemple, la première colonne est donnée par $\mathbf{t}_A$ correspondant à

$\mathbf{t}_B = (1\ 0\ 0)^T$, la seconde colonne par $\mathbf{t}_A$ correspondant à $\mathbf{t}_B = (0\ 1\ 0)^T$ et la troisième par le produit vectoriel des deux premières. En pratique, cette méthode s'avère instable: elle ne permet aucune correction des erreurs de mesure, d'autant plus que les déplacements du robot sont moins précis en translation qu'en rotation.

2.2.3 Nouvelle formulation

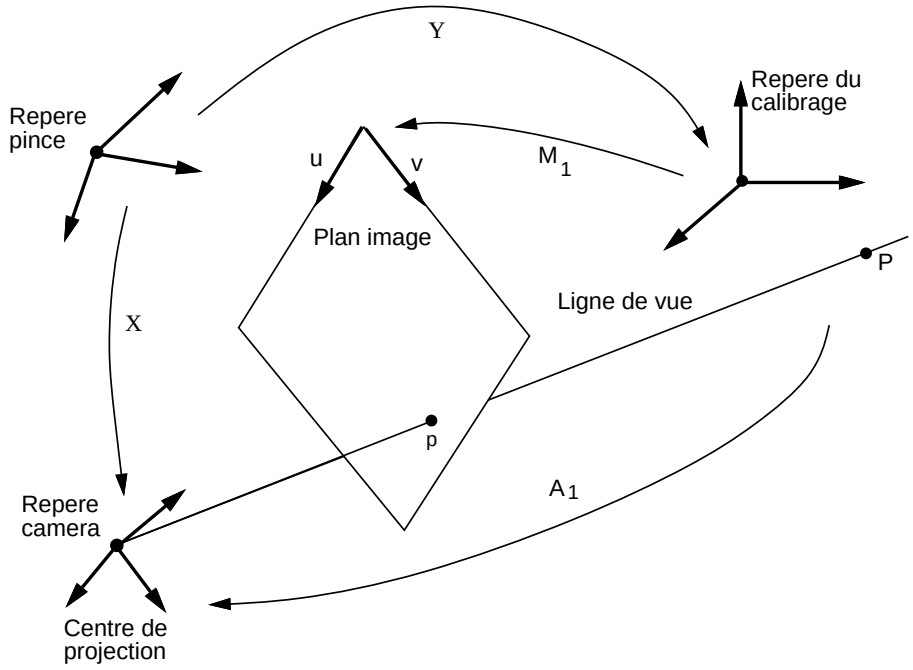


FIG. 2.4 - *La ligne de vue passant par le centre de projection et le point image p sera exprimé dans le repère du calibrage en utilisant les coefficients de la matrice de projection perspective M_1.*

L'intérêt de la méthode que nous proposons ici est de s'affranchir de la connaissance explicite des paramètres intrinsèques et de remplacer l'estimation de la transformation repère caméra-repère pince par l'estimation directe de la transformation entre le repère du calibrage et le repère de la pince (Figure 2.4).

La formulation classique ($AX = XB$) implique que la caméra doit être calibrée dans chaque position i du système caméra-pince. Une fois ce calibrage effectué, les paramètres extrinsèques de la caméra (transformation repère du calibrage-repère caméra) sont obtenus en décomposant la matrice de projection M_i de la manière suivante [FT86] [Tsa87a] [HM93] :

$$\begin{aligned} M_i &= C\,A_i \\ &= \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_A^i & \mathbf{t}_A^i \\ \mathbf{0}^T & 1 \end{bmatrix} \end{aligned} \tag{2.7}$$

Les paramètres α_u, α_v, u_0 et v_0 décrivent la transformation affine entre le repère de la caméra et le repère de l'image. Cette décomposition suppose que la caméra est décrite par le modèle sténopé et que l'axe optique associé à ce modèle est perpendiculaire au plan image.

Soit Y la transformation rigide entre le repère pince et le repère du calibrage. Nous avons (voir Figure 2.4) :

$$X = A_1 Y \tag{2.8}$$

Ainsi, la transformation Y est l'équivalent de X à une transformation rigide près. En écrivant le système d'équations matricielles associé à la formulation classique et en considérant les déplacements par rapport à la position initiale (position 1), nous pouvons écrire :

$$\begin{cases} A_{12} X &= X B_{12} \\ & \vdots \\ A_{1i} X &= X B_{1i} \\ & \vdots \\ A_{1n} X &= X B_{1n} \end{cases} \tag{2.9}$$

En injectant l'équation (2.8) dans ce système, on obtient ($A_{1i} A_1 = A_i$) :

$$\begin{cases} A_2 Y &= A_1 Y B_{12} \\ & \vdots \\ A_i Y &= A_1 Y B_{1i} \\ & \vdots \\ A_n Y &= A_1 Y B_{1n} \end{cases} \tag{2.10}$$

En multipliant à gauche les termes de ce système par la matrice C et en utilisant l'équation (2.7) pour $i = 1 \ldots n$, nous obtenons :

$$\begin{cases} M_2 Y &= M_1 Y B_{12} \\ & \vdots \\ M_i Y &= M_1 Y B_{1i} \\ & \vdots \\ M_n Y &= M_1 Y B_{1n} \end{cases} \tag{2.11}$$

Dans ce système, l'inconnue est la matrice Y qui représente la transformation rigide entre le repère du calibrage et le repère pince. De point de vue mathématique, le remplacement du repère de la caméra par le repère du calibrage est équivalent à l'emploi de la matrice de projection la plus générale M_1 au lieu de la matrice C. L'avantage de ce remplacement est que l'on peut manipuler des situations où le modèle sténopé n'est pas valide, de plus, on évite la décomposition de la matrice de projection, cette opération étant très instable en présence de bruit.

Etant donné P un point de la scène et p son image par une projection (Figure 2.4), les coordonnées (u, v) dans le repère image s'expriment en fonction des coordonnées (x,

y, z) de P et de la matrice de projection M_1 par la relation :

$$\begin{bmatrix} s\,u \\ s\,v \\ s \end{bmatrix} = M_1 \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

ou :

$$u = \frac{m_{11}\,x + m_{12}\,y + m_{13}\,z + m_{14}}{m_{31}\,x + m_{32}\,y + m_{33}\,z + m_{34}} \tag{2.12}$$

$$v = \frac{m_{21}\,x + m_{22}\,y + m_{23}\,z + m_{24}}{m_{31}\,x + m_{32}\,y + m_{33}\,z + m_{34}} \tag{2.13}$$

où x, y et z sont les coordonnées du point P dans le repère du calibrage, s est un scalaire non nul et les m_{ij} sont les coefficients de la matrice M_1. Ces deux équations peuvent être écrites sous la forme suivante :

$$(m_{11} - u\,m_{31})\,x + (m_{12} - u\,m_{32})\,y + (m_{13} - u\,m_{33})\,z = u\,m_{34} - m_{14} \tag{2.14}$$

$$(m_{21} - v\,m_{31})\,x + (m_{22} - v\,m_{32})\,y + (m_{23} - v\,m_{33})\,z = v\,m_{34} - m_{24} \tag{2.15}$$

Ces deux équations peuvent être interprétées de la façon suivante : étant donné une matrice de projection M_1 et un point image p, les deux équations (2.14) et (2.15) décrivent la ligne de vue passant par le centre de projection et le point p. Cette ligne de vue est exprimée dans le repère du calibrage.

Ainsi, la connaissance de la géométrie caméra/pince (la matrice X dans la formulation classique ou la matrice Y dans notre nouvelle formulation) permet d'exprimer la ligne de vue associée à chaque point image dans le repère de la pince et donc dans la base du robot.

2.3 Equivalence avec le problème de localisation 3D

Nous allons établir l'équivalence entre le problème du calibrage caméra-pince et le problème de localisation 3D. Nous montrons que cette formulation est équivalente au problème de localisation 3D à partir des correspondances droites/droites.

Comme le déplacement de la caméra et celui de la pince représentent le même mouvement mais exprimé dans deux repères différents, il existe une relation géométrique plus explicite que la relation ($AX = XB$). En effet, cette relation peut être établie si l'on représente les déplacements de la caméra et de la pince par des vissages [Che91]. On représente le déplacement A par son vissage correspondant (d_A, ϕ_A, $\mathcal{L}_A$) et B par son vissage (d_B, ϕ_B, $\mathcal{L}_B$). Les deux scalaires d_A, ϕ_A et le vecteur directeur de l'axe $\mathcal{L}_A$ sont donnés par (voir chapitre 1) :

$$\begin{cases} d_A &= \mathbf{t}_A \cdot \mathbf{n}_A \\ \phi_A &= \theta_A \qquad\qquad \text{si} \quad \mathbf{t}_A \cdot \mathbf{n}_A \geq 0 \\ \mathbf{r}_A &= \mathbf{n}_A \end{cases}$$

$$\left\{ \begin{array}{lcll} d_A & = & -(\mathbf{t}_A \cdot \mathbf{n}_A) & \\ \phi_A & = & 2\pi - \theta_A & \text{si} \quad \mathbf{t}_A \cdot \mathbf{n}_A < 0 \\ \mathbf{r}_A & = & -\mathbf{n}_A & \end{array} \right.$$

où $\mathbf{n}_A$ et θ_A sont l'axe et l'angle de rotation associés à la matrice R_A.

Les positions de ces deux axes sont données par les deux vecteurs $\mathbf{c}_A$ et $\mathbf{c}_B$:

$$\begin{array}{rcl} \mathbf{c}_A & = & [\mathbf{t}_{A\perp} + \mathbf{n}_A \times \mathbf{t}_{A\perp} \ \cot(\theta_A/2)]/2 \\ \mathbf{c}_B & = & [\mathbf{t}_{B\perp} + \mathbf{n}_B \times \mathbf{t}_{B\perp} \ \cot(\theta_B/2)]/2 \end{array}$$

avec :

$$\begin{array}{rcl} \mathbf{c}_A \cdot \mathbf{r}_A & = & 0 \\ \mathbf{c}_B \cdot \mathbf{r}_B & = & 0 \end{array}$$

L'équation $R_A = R_X R_B R_X^T$ montre que les deux matrices de rotation, R_A et R_B, ont les mêmes valeurs propres et que le vecteur $R_X \, \mathbf{r}_B$ est le vecteur propre de la matrice R_A associé à sa valeur propre 1. R_A possède les valeurs propres suivantes : 1, $e^{i\theta_A}$ et $e^{-i\theta_A}$. En tenant compte des signes des axes de rotation, on peut écrire :

$$\left\{ \begin{array}{rcl} \theta_A & = & \theta_B \\ \mathbf{r}_A & = & R_X \, \mathbf{r}_B \end{array} \right. \tag{2.16}$$

ou:

$$\left\{ \begin{array}{rcl} \theta_A & = & 2\pi - \theta_B \\ \mathbf{r}_A & = & -R_X \, \mathbf{r}_B \end{array} \right. \tag{2.17}$$

Nous allons montrer que seule l'équation (2.16) est acceptable. En réécrivant l'équation (2.5), on obtient :

$$R_A \, \mathbf{t}_X + \mathbf{t}_{A\perp} + \mathbf{t}_{A\parallel} = R_X \, (\mathbf{t}_{B\perp} + \mathbf{t}_{B\parallel}) + \mathbf{t}_X \tag{2.18}$$

où $\mathbf{t}_{A\parallel}$ est la composante de $\mathbf{t}_A$ sur l'axe de rotation et $\mathbf{t}_{A\perp}$ est la composante normale à cet axe. Nous avons :

$$\begin{array}{rcl} \mathbf{t}_{A\perp} & = & (I - R_A) \, \mathbf{c}_A \\ \mathbf{t}_{B\perp} & = & (I - R_B) \, \mathbf{c}_B \\ \mathbf{t}_{A\parallel} & = & d_A \, \mathbf{r}_A \\ \mathbf{t}_{B\parallel} & = & d_B \, \mathbf{r}_B \end{array}$$

En injectant ces équations dans l'équation (2.18) et en utilisant les deux égalités $R_A R_X = R_X R_B$ et $\mathbf{r}_B = R_X^T \, \mathbf{r}_A$, on obtient :

$$(I - R_A) \, (R_X \, \mathbf{c}_B - \mathbf{c}_A + \mathbf{t}_X) = (d_A - d_B) \, \mathbf{r}_A \tag{2.19}$$

En posant :

$$\mathbf{w} = R_X \, \mathbf{c}_B - \mathbf{c}_A + \mathbf{t}_X$$

et en multipliant à gauche l'équation (2.19) par $\mathbf{r}_A^T$, on aura :

$$\mathbf{r}_A^T \, \mathbf{w} - \mathbf{r}_A^T R_A \, \mathbf{w} = d_A - d_B$$

mais :

$$\begin{aligned}\mathbf{r}_A^T\,\mathbf{w} - \mathbf{r}_A^T R_A\,\mathbf{w} &= \mathbf{r}_A^T\,\mathbf{w} - \mathbf{w}^T R_A^T\,\mathbf{r}_A \\ &= \mathbf{r}_A^T\,\mathbf{w} - \mathbf{w}^T\mathbf{r}_A \\ &= 0\end{aligned}$$

Ce qui donne :

$$d_A - d_B = 0$$

Notons que si nous utilisons l'égalité $\mathbf{r}_B = -R_X^T\,\mathbf{r}_A$, alors on obtient $d_A + d_B = 0$ ce qui est absurde puisque d_A et d_B sont positifs.

L'équation (2.18) devient donc :

$$R_A\,\mathbf{w} = \mathbf{w}$$

On en déduit que le vecteur $\mathbf{w}$ est proportionnel au vecteur propre de R_A associé à sa valeur propre 1 :

$$R_X\,\mathbf{c}_B - \mathbf{c}_A + \mathbf{t}_X = \nu\,\mathbf{r}_A \tag{2.20}$$

où ν est un scalaire réel.

En multipliant à gauche cette équation par $\mathbf{r}_A^T$, on obtient :

$$\nu = \mathbf{r}_A^T R_X\,\mathbf{c}_B - \mathbf{r}_A^T\mathbf{c}_A + \mathbf{r}_A^T\mathbf{t}_X$$

Compte tenu des égalités suivantes :

$$\begin{aligned}\mathbf{r}_A^T\,\mathbf{c}_A &= 0 \\ \mathbf{r}_A^T R_X\,\mathbf{c}_B &= \mathbf{r}_B^T R_X^T R_X\,\mathbf{c}_B \\ &= \mathbf{r}_B^T\,\mathbf{c}_B \\ &= 0\end{aligned}$$

on a :

$$\nu = \mathbf{r}_A^T\,\mathbf{t}_X$$

En injectant cette égalité dans l'équation (2.20), on aura :

$$\mathbf{c}_A = R_X\,\mathbf{c}_B + \mathbf{t}_X - (\mathbf{r}_A^T\,\mathbf{t}_X)\,\mathbf{r}_A \tag{2.21}$$

En résumé, les paramètres des deux vissages vérifient les propriétés suivantes :

$$d_A = d_B \tag{2.22}$$

$$\phi_A = \phi_B \tag{2.23}$$

$$\mathbf{r}_A = R_X\,\mathbf{r}_B \tag{2.24}$$

$$\mathbf{c}_A = R_X\,\mathbf{c}_B + \mathbf{t}_X - (\mathbf{r}_A^T\,\mathbf{t}_X)\,\mathbf{r}_A \tag{2.25}$$

Les deux équations (2.22) et (2.23) montrent que les déplacements angulaire et translationnel des deux déplacements, A et B, restent invariants. Les deux équations (2.24) et (2.25) montrent que les deux axes de vissage sont transformés l'un de l'autre et la

transformation qui fait passer de l'un à l'autre n'est autre que la transformation rigide caméra-pince. Ainsi, chaque déplacement du système caméra-pince fournit une correspondance droite/droite. Les deux équations (2.22) et (2.23) sont très utiles puisqu'elles permettent de quantifier les incertitudes qui affectent les mesures fournies par les déplacements de la caméra et de la pince. De plus, elles pourraient être utilisées comme un moyen puissant de mise en correspondance entre les déplacements de la caméra et ceux de la pince.

2.4 Décomposition de la nouvelle formulation

Nous montrons à présent que la nouvelle formulation introduite dans le paragraphe 2.2.3 a la même structure mathématique que celle de la formulation classique. Ce qui va nous permettre de formuler une approche unifiée applicable aux deux formulations. Pour ce faire, nous écrivons la matrice de projection perspective M en fonction des paramètres intrinsèques et extrinsèques :

$$M = \begin{bmatrix} \alpha_u r_{11} + u_0 r_{31} & \alpha_u r_{12} + u_0 r_{32} & \alpha_u r_{13} + u_0 r_{33} & \alpha_u t_x + u_0 t_z \\ \alpha_v r_{21} + v_0 r_{31} & \alpha_v r_{22} + v_0 r_{32} & \alpha_v r_{23} + v_0 r_{33} & \alpha_v t_y + v_0 t_z \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

Notons qu'une matrice M_i de cette forme pourrait être écrite de la manière suivante :

$$M_i = \begin{bmatrix} N_i & \mathbf{n}_i \end{bmatrix}$$

où N_i est une matrice de dimension 3×3 et $\mathbf{n}_i$ est un vecteur de dimension 3. Dans le cas général, la matrice N_i possède une inverse puisque les vecteurs $(r_{11}\ r_{12}\ r_{13})^T$, $(r_{21}\ r_{22}\ r_{23})^T$ et $(r_{31}\ r_{32}\ r_{33})^T$ sont mutuellement orthogonaux et $\alpha_u \neq 0$, $\alpha_v \neq 0$. Avec ces notations la première équation du système (2.11) peut être décomposée en une équation matricielle :

$$N_2 R_Y = N_1 R_Y R_B \tag{2.26}$$

et une équation vectorielle :

$$N_2\, \mathbf{t}_Y + \mathbf{n}_2 = N_1 R_Y\, \mathbf{t}_B + N_1\, \mathbf{t}_Y + \mathbf{n}_1 \tag{2.27}$$

En introduisant la notation suivante :

$$N = N_1^{-1} N_2$$

l'équation (2.26) devient :

$$N R_Y = R_Y R_B \tag{2.28}$$

d'où l'on déduit :

$$N = R_Y R_B R_Y^T$$

La matrice N possède deux propriétés :

1. N est le produit de trois matrices de rotation, il s'ensuit que cette matrice est elle-même une matrice de rotation :

$$N^{-1} = N^T$$

2. Comme R_Y est orthogonale, les matrices N et R_B ont les mêmes valeurs propres. Soit $\mathbf{n}_N$ le vecteur propre de N associé à sa valeur propre 1. A partir de l'équation (2.28), on peut écrire :

$$\begin{aligned} N R_Y \mathbf{n}_B &= R_Y R_B \mathbf{n}_B \\ &= R_Y \mathbf{n}_B \end{aligned}$$

On en déduit que :

$$\mathbf{n}_N = R_Y \mathbf{n}_B \tag{2.29}$$

Cette équation est équivalente à l'équation (2.6) de la formulation classique.

En multipliant à gauche l'équation (2.27) par N_1^{-1}, nous obtenons :

$$(N - I)\,\mathbf{t}_Y = R_Y \mathbf{t}_B - \mathbf{t}_N \tag{2.30}$$

avec:

$$\mathbf{t}_N = N_1^{-1}(\mathbf{n}_2 - \mathbf{n}_1)$$

Cette équation est identique à l'équation (2.5) de la formulation classique.

2.5 Solution optimale commune

Dans le paragraphe précédent nous avons démontré que les deux formulations sont équivalentes de point de vue mathématique. En effet, la formulation classique, $AX = XB$, se décompose en (2.4) et (2.5) :

$$\begin{aligned} \mathbf{n}_A &= R_X \mathbf{n}_B \\ (R_A - I)\,\mathbf{t}_X &= R_X \mathbf{t}_B - \mathbf{t}_A \end{aligned}$$

et la nouvelle formulation, $MY = M'YB$, se décompose en (2.29) et (2.30) :

$$\begin{aligned} \mathbf{n}_N &= R_Y \mathbf{n}_B \\ (N - I)\,\mathbf{t}_Y &= R_Y \mathbf{t}_B - \mathbf{t}_N \end{aligned}$$

Ces deux ensembles d'équations sont de la forme :

$$\mathbf{v}' = R\,\mathbf{v} \tag{2.31}$$

$$(K - I)\,\mathbf{t} = R\,\mathbf{p} - \mathbf{p}' \tag{2.32}$$

où R et $\mathbf{t}$ sont les paramètres à estimer (rotation et translation), $\mathbf{v}'$, $\mathbf{v}$, $\mathbf{p}'$, $\mathbf{p}$ sont des vecteurs de dimension 3, K est une matrice orthogonale de dimension 3×3 et I est la matrice identité.

Les équations (2.31) et (2.32) sont associées à un seul déplacement du système caméra-pince. Nous avons vu qu'un seul déplacement de la pince est insuffisant pour résoudre ces équations. Un deuxième déplacement de la pince, au moins, est donc nécessaire pour pouvoir estimer R et $\mathbf{t}$. Dans le cas général où l'on a n déplacements de la pince le

problème de résolution des $2n$ équations devient un problème de minimisation de deux fonctions d'erreur positives :

$$f_1(R) = \sum_{i=1}^{n} \|\mathbf{v}'_i - R\,\mathbf{v}_i\|^2 \tag{2.33}$$

et

$$f_2(R, \mathbf{t}) = \sum_{i=1}^{n} \|R\,\mathbf{p}_i - (K_i - I)\,\mathbf{t} - \mathbf{p}'_i\|^2 \tag{2.34}$$

Ainsi, deux approches sont possibles :

1. *R puis $\mathbf{t}$.* La rotation est estimée en minimisant f_1. Ce problème de minimisation a une solution analytique simple que nous allons détailler dans le paragraphe suivant. Une fois la rotation estimée, la minimisation de la fonction f_2, qui permet d'estimer la translation $\mathbf{t}$, devient un problème linéaire que l'on peut résoudre au sens des moindres carrés.

2. *R et $\mathbf{t}$.* La rotation et la translation sont simultanément estimées en minimisant $f_1 + f_2$. Cette minimisation est non linéaire mais elle permet d'obtenir la solution la plus stable.

La Figure 2.5 représente le principe du calibrage pour les deux formulations (classique et nouvelle).

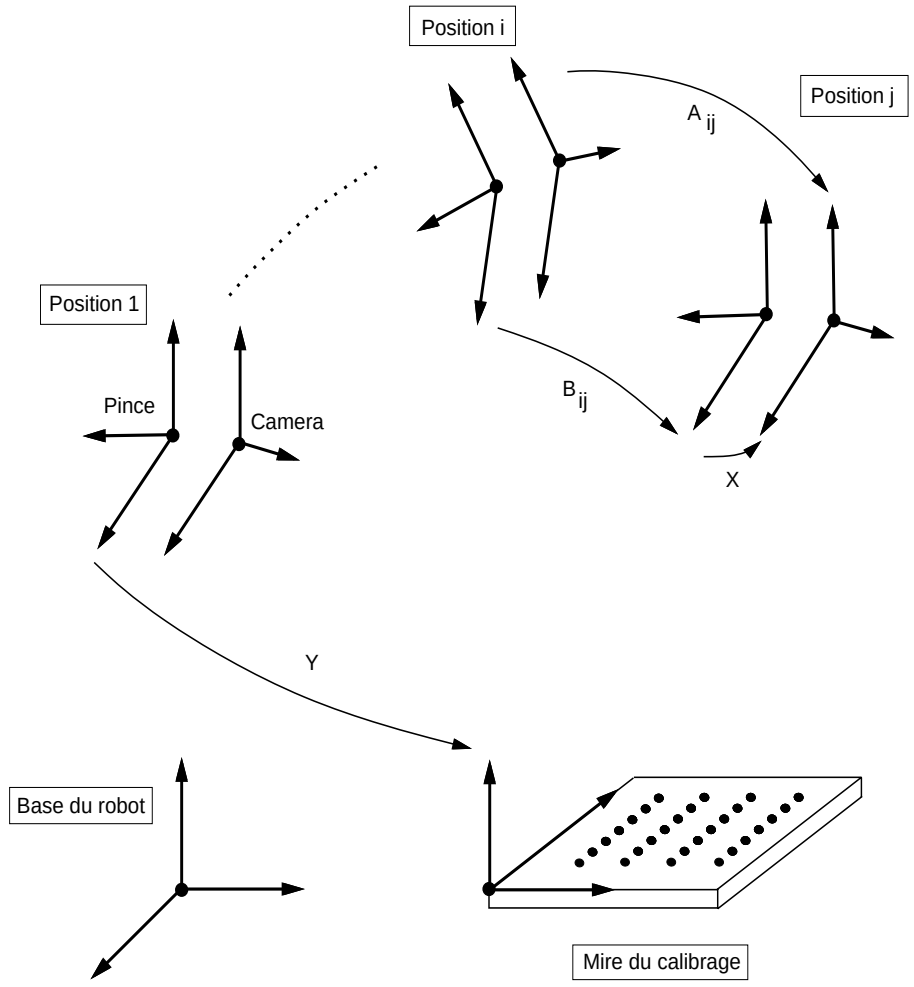


FIG. 2.5 - *Calibrage caméra-pince : on doit connaître les déplacements de la pince et le modèle de la scène 3D observée par la caméra. La formulation classique consiste à déterminer la transformation X grâce au système d'équations $(AX = XB)$, la nouvelle formulation consiste à déterminer la transformation Y grâce au système d'équations $(MY = M'YB)$.*

2.6 Solutions analytiques

2.6.1 La rotation

La rotation optimale est obtenue par minimisation de la fonction f_1 :

$$\min_R \left(\sum_{i=1}^{n} \|\mathbf{v}'_i - R\,\mathbf{v}_i\|^2 \right) \tag{2.35}$$

La minimisation du critère de l'équation (2.35) dépend du choix d'une représentation pour la rotation. Comme nous connaissons 4 représentations possibles pour la rotation, nous avons donc quatre possibilités :

- *matrice orthogonale.* Dans ce cas une rotation est décrite par 9 paramètres qui sont liés par les 6 contraintes d'orthonormalité. Un certain nombre de solutions a été proposé parmi lesquelles on peut citer [LHB+86] [AHB87] [Ume91].

- *angles d'Euler.* Certains auteurs ont proposé de trouver les angles d'Euler optimaux. Cette formulation conduit à un problème de minimisation non linéaire.

- *axe et angle.* Cette méthode est proposée par Tsai [TL88] [TL89]. Dans ce cas nous aurons à estimer 3 paramètres, soit les composantes du vecteur directeur de l'axe de rotation, la valeur de l'angle étant proportionnelle au module de ce vecteur. Il s'agit d'une représentation intéressante car minimale en ce qui concerne le nombre des paramètres à estimer. Cette méthode ramène le problème à la résolution d'un système linéaire surcontraint.

- *quaternion unitaire.* La représentation par des quaternions unitaires est également intéressante car, d'une part, il n'y a que 4 paramètres à estimer et, d'autre part, le calcul se ramène à la détermination de la plus petite valeur propre d'une matrice symétrique de taille 4 [FH86] [Fau88].

Axe et angle de rotation optimaux

Le critère de l'équation (2.35) signifie que nous cherchons une solution approchée d'un système de n équations où chaque équation du système est du type :

$$\mathbf{v}'_i = R\,\mathbf{v}_i$$

Soit $\mathbf{n}$ le vecteur directeur de l'axe de rotation et utilisons le fait que la rotation conserve le produit scalaire. On obtient :

$$\mathbf{n} \cdot \mathbf{v}_i = (R\,\mathbf{n}) \cdot (R\,\mathbf{v}_i)$$

et si on se souvient que $\mathbf{n}$ est un vecteur propre de R associé à la valeur propre 1, on obtient :

$$\mathbf{n} \cdot \mathbf{v}_i = \mathbf{n} \cdot \mathbf{v}'_i$$

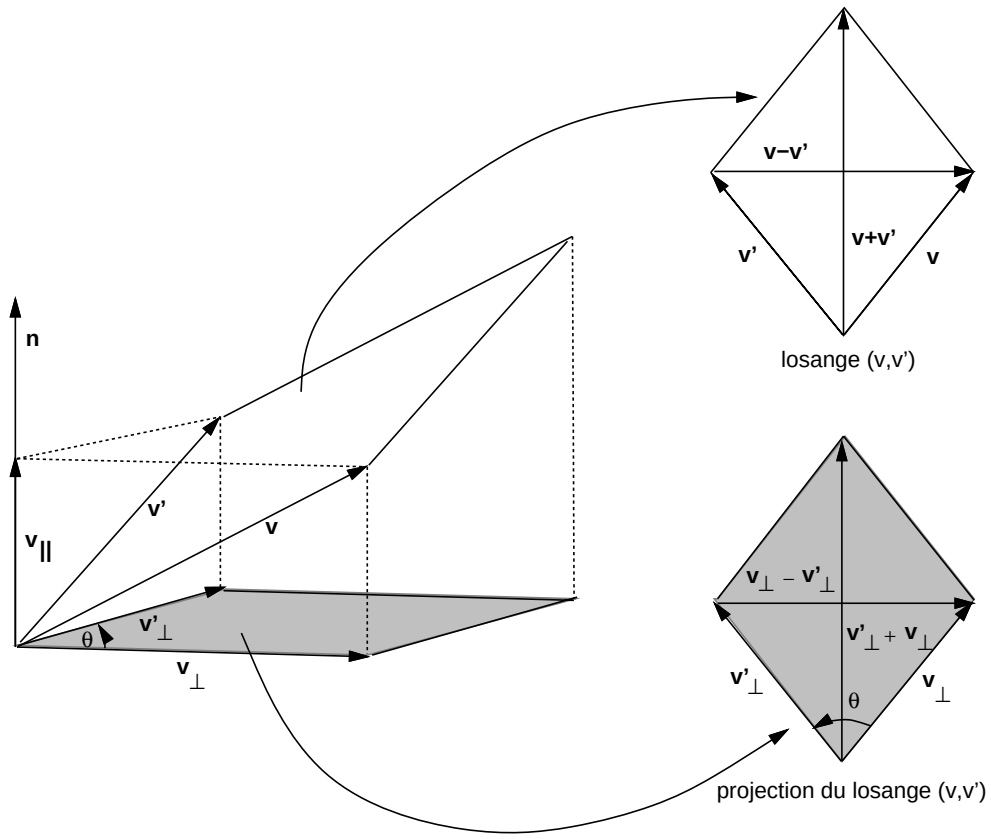


FIG. 2.6 - *Calcul de la rotation optimale utilisant l'axe et l'angle de rotation.*

d'où on déduit la propriété de perpendicularité suivante :

$$\mathbf{n} \cdot (\mathbf{v}_i - \mathbf{v}'_i) = 0 \tag{2.36}$$

Par ailleurs le module d'un vecteur est conservé par rotation. On a donc, pour tout i :

$$\|\mathbf{v}'_i\| = \|\mathbf{v}_i\|$$

On déduit facilement une deuxième propriété de perpendicularité :

$$(\mathbf{v}_i + \mathbf{v}'_i) \cdot (\mathbf{v}_i - \mathbf{v}'_i) = 0 \tag{2.37}$$

Puisque le vecteur $\mathbf{v}_i - \mathbf{v}'_i$ est perpendiculaire aux vecteurs $\mathbf{n}$ et $\mathbf{v}_i + \mathbf{v}'_i$ il est donc proportionnel à leur produit vectoriel :

$$(\mathbf{v}_i + \mathbf{v}'_i) \wedge \mathbf{n} = k\,(\mathbf{v}_i - \mathbf{v}'_i) \tag{2.38}$$

Afin de déterminer la valeur de k nous allons, décomposer les vecteurs $\mathbf{v}_i$ et $\mathbf{v}'_i$ selon deux directions, une direction colinéaire à l'axe de rotation $\mathbf{n}$ et une autre direction orthogonale à cet axe de rotation, (voir Figure 2.6) :

$$\begin{aligned} \mathbf{v}_i &= \mathbf{v}_{i\perp} + \mathbf{v}_{i\|} \\ \mathbf{v}'_i &= \mathbf{v}'_{i\perp} + \mathbf{v}'_{i\|} \end{aligned}$$

En substituant ces formules dans l'équation (2.38), on obtient :

$$(\mathbf{v}'_{i\perp} + \mathbf{v}'_{i\parallel} + \mathbf{v}_{i\perp} + \mathbf{v}_{i\parallel}) \wedge \mathbf{n} = k\,(\mathbf{v}_{i\perp} + \mathbf{v}_{i\parallel} - \mathbf{v}'_{i\perp} - \mathbf{v}'_{i\parallel})$$

En développant et en remarquant que $\mathbf{v}'_{i\parallel} = \mathbf{v}_{i\parallel}$, on obtient :

$$(\mathbf{v}_{i\perp} + \mathbf{v}'_{i\perp}) \wedge \mathbf{n} = k\,(\mathbf{v}_{i\perp} - \mathbf{v}'_{i\perp})$$

On peut remarquer que les vecteurs $\mathbf{v}_{i\perp} + \mathbf{v}'_{i\perp}$ et $\mathbf{v}_{i\perp} - \mathbf{v}'_{i\perp}$ sont orthogonaux et par ailleurs ils sont tous les deux orthogonaux au vecteur unitaire $\mathbf{n}$. On obtient donc k comme le rapport suivant :

$$k = \frac{\|\mathbf{v}_{i\perp} + \mathbf{v}'_{i\perp}\|}{\|\mathbf{v}_{i\perp} - \mathbf{v}'_{i\perp}\|}$$

Or, l'angle entre les vecteurs $\mathbf{v}_{i\perp}$ et $\mathbf{v}'_{i\perp}$ est l'angle de la rotation que l'on cherche, soit θ. D'après la Figure 2.6 on remarque qu'il s'agit d'un des angles d'un losange qui a comme diagonales $\mathbf{v}_{i\perp} + \mathbf{v}'_{i\perp}$ et $\mathbf{v}_{i\perp} - \mathbf{v}'_{i\perp}$. La relation entre cet angle et le rapport des longueurs de ces diagonales est :

$$k = \frac{1}{\tan(\theta/2)} \tag{2.39}$$

En posant :

$$\mathbf{m} = \tan\frac{\theta}{2}\ \mathbf{n}$$

on obtient un système de trois équations et trois inconnues pour chaque i :

$$S(\mathbf{v}_i + \mathbf{v}'_i)\,\mathbf{m} = \mathbf{v}'_i - \mathbf{v}_i \tag{2.40}$$

Le rang de la matrice antisymétrique $S()$ est 2 et chaque mise en correspondance i fournit donc 2 équations et non pas 3 équations. Il faut donc au moins 2 mises en correspondance pour trouver un axe de rotation. En général, on dispose de n mises en correspondance. On obtient alors un système de $2n$ équations à 3 inconnues. Si on considère les deux premières équations du système d'équations (2.40), on obtiendra :

$$\begin{bmatrix} \vdots & \vdots & \vdots \\ 0 & -(\mathbf{v}_{zi} + \mathbf{v}'_{zi}) & \mathbf{v}_{yi} + \mathbf{v}'_{yi} \\ \mathbf{v}_{zi} + \mathbf{v}'_{zi} & 0 & -(\mathbf{v}_{xi} + \mathbf{v}'_{xi}) \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \mathbf{m}_x \\ \mathbf{m}_y \\ \mathbf{m}_z \end{bmatrix} = \begin{bmatrix} \vdots \\ \mathbf{v}'_{xi} - \mathbf{v}_{xi} \\ \mathbf{v}'_{yi} - \mathbf{v}_{yi} \\ \vdots \end{bmatrix}$$

La solution optimale pour $\mathbf{m}$ est obtenue en résolvant ce système d'équations surcontraint. On obtient finalement le vecteur unitaire de l'axe de rotation et l'angle de rotation :

$$\begin{aligned} \mathbf{n} &= \frac{\mathbf{m}}{\|\mathbf{m}\|} \\ \theta &= 2\arctan(\|\mathbf{m}\|) \end{aligned}$$

et grâce à la formule de Rodrigues on obtient facilement la matrice de rotation à partir du vecteur unitaire $\mathbf{n}$ et de l'angle θ.

La solution, proposée ici, est particulièrement simple car la rotation est représentée par un vecteur dont le module est fonction de l'angle de rotation. On a donc une représentation minimale particulièrement intéressante pour la recherche d'une rotation optimale.

Quaternion unitaire optimal

Nous représentons la rotation par un quaternion unitaire. La rotation du vecteur $\mathbf{v}$ s'écrit alors :

$$\begin{aligned} \mathbf{v}' &= R\,\mathbf{v} \\ &= \mathbf{q} * \mathbf{v} * \overline{\mathbf{q}} \end{aligned}$$

où $*$ dénote le produit quaternion.

Le critère de l'équation (2.35) devient :

$$\begin{gathered} \min_{\mathbf{q}} \left(\sum_{i=1}^{n} \|\mathbf{v}'_i - \mathbf{q} * \mathbf{v}_i * \overline{\mathbf{q}}\|^2\right) \\ \text{avec } \|\mathbf{q}\|^2 = 1 \end{gathered}$$

Comme le produit des modules des quaternions est égal au module des produits, on a successivement :

$$\begin{aligned} \|\mathbf{v}'_i - \mathbf{q} * \mathbf{v}_i * \overline{\mathbf{q}}\|^2 &= \|\mathbf{v}'_i - \mathbf{q} * \mathbf{v}_i * \overline{\mathbf{q}}\|^2 \|\mathbf{q}\|^2 \\ &= \|\mathbf{v}'_i * \mathbf{q} - \mathbf{q} * \mathbf{v}_i\|^2 \\ &= (Q(\mathbf{v}'_i)\,\mathbf{q} - W(\mathbf{v}_i)\,\mathbf{q})^T (Q(\mathbf{v}'_i)\,\mathbf{q} - W(\mathbf{v}_i)\,\mathbf{q}) \\ &= \mathbf{q}^T \mathcal{A}_i\,\mathbf{q} \end{aligned}$$

où A_i est la matrice symétrique de dimension 4 :

$$A_i = (Q(\mathbf{v}'_i) - W(\mathbf{v}_i))^t (Q(\mathbf{v}'_i) - W(\mathbf{v}_i))$$

Le critère de l'équation (2.35) devient donc :

$$\begin{aligned} \min_{\mathbf{q}} \left(\sum_{i=1}^{n} \mathbf{q}^t A_i\,\mathbf{q}\right) &= \min_{\mathbf{q}} \left(\mathbf{q}^t (\sum_{i=1}^{n} A_i)\,\mathbf{q}\right) \\ &= \min_{\mathbf{q}} \left(\mathbf{q}^t \mathcal{A}\,\mathbf{q}\right) \end{aligned}$$

avec :

$$\mathcal{A} = \sum_{i=1}^{n} A_i = \sum_{i=1}^{n} (Q(\mathbf{v}'_i) - W(\mathbf{v}_i))^t (Q(\mathbf{v}'_i) - W(\mathbf{v}_i)) \tag{2.41}$$

étant une matrice symétrique positive. Sous la contrainte que le quaternion doit être unitaire, on obtient finalement le critère suivant :

$$\min_{\mathbf{q}} Q = \min_{\mathbf{q}} \left(\mathbf{q}^t \mathcal{A}\,\mathbf{q} + \lambda\,(1 - \mathbf{q}^t \mathbf{q})\right) \tag{2.42}$$

En dérivant Q par rapport à $\mathbf{q}$ on obtient :

$$\mathcal{A}\,\mathbf{q} - \lambda\,\mathbf{q} = \mathbf{0}$$

et en substituant cette solution dans l'équation (2.42), on obtient :

$$Q = \lambda$$

Le quaternion unitaire qui minimise Q est donc le vecteur propre unitaire de la matrice $\mathcal{A}$ associé à la plus petite valeur propre de $\mathcal{A}$, soit λ. Nous rappelons qu'une matrice symétrique a des valeurs propres réelles et une matrice symétrique positive a toutes ses valeurs propres positives. Soient alors les vecteurs propres de $\mathcal{A}$ qui forment une base orthogonale, $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_3$ et $\mathbf{w}_4$. Si ces vecteurs sont unitaires, la base est orthonormée. Le quaternion $\mathbf{q}$ peut s'écrire dans cette base :

$$\mathbf{q} = \mu_1 \mathbf{w}_1 + \mu_2 \mathbf{w}_2 + \mu_3 \mathbf{w}_3 + \mu_4 \mathbf{w}_4$$

Pour tout i on a :

$$\mathcal{A}\,\mathbf{w}_i = \lambda_i \mathbf{w}_i$$

λ_1, λ_2, λ_3 et λ_4 étant les valeurs propres de $\mathcal{A}$ avec :

$$\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$$

En tenant compte du fait que les vecteurs propres forment une base orthonormée, on peut écrire :

$$\mathbf{q}^t \mathcal{A}\,\mathbf{q} = \mu_1^2\,\lambda_1 + \mu_2^2\,\lambda_2 + \mu_3^2\,\lambda_3 + \mu_4^2\,\lambda_4$$

Cette expression est minimale pour $\mu_1 = 1$ et $\mu_2 = \mu_3 = \mu_4 = 0$. On obtient donc :

$$\mathbf{q} = \mathbf{w}_1$$

et

$$\mathbf{q}^t \mathcal{A}\,\mathbf{q} = \lambda_1$$

Le quaternion unitaire optimal est donc le vecteur propre unitaire de $\mathcal{A}$ associé à sa plus petite valeur propre.

2.6.2 La translation

Une fois que nous avons estimé la rotation optimale R^*, le calcul de la translation optimale est possible grâce au critère suivant :

$$\min_{\mathbf{t}} \left(\sum_{i=1}^{n} \| R^*\,\mathbf{p}_i - (K_i - I)\,\mathbf{t} - \mathbf{p}'_i \|^2 \right)$$

C'est un système linéaire surcontraint. En dérivant par rapport aux trois composantes de $\mathbf{t}$ et en annulant ces dérivées on obtient la solution pour le vecteur de translation :

$$\mathbf{t} = G^{\dagger}\,\mathbf{y}$$

où $G^{\dagger}$ est la pseudo-inverse de la matrice G.

La matrice G et le vecteur $\mathbf{y}$ sont donnés par :

$$G = \underbrace{\begin{bmatrix} \vdots \\ (K_i - I) \\ \vdots \end{bmatrix}}_{3n \times 3}, \quad \mathbf{y} = \underbrace{\begin{bmatrix} \vdots \\ R^*\,\mathbf{p}_i - \mathbf{p}'_i \\ \vdots \end{bmatrix}}_{3n \times 1}$$

2.7 Solutions numériques

Nous venons de voir que le problème de détermination de la rotation et de la translation est divisé en deux problèmes. On minimise séparément deux critères : on estime d'abord la rotation optimale et ensuite on utilise cette rotation pour déterminer la translation optimale. Cette décomposition a des avantages et des inconvénients. L'avantage est que chaque critère est plus simple et, comme nous venons de le voir, la solution au problème d'optimisation de la rotation est très élégante et très fiable d'un point de vue numérique. Le critère de translation se ramène à un problème d'optimisation linéaire très classique. L'inconvénient est que les erreurs éventuelles associées au calcul de la rotation vont se répercuter sur le calcul de la translation. Nous proposons une alternative intéressante qui consiste à estimer simultanément et d'une façon optimale les paramètres de rotation et translation.

Les rotation et translation optimales sont données par :

$$\min_{R,\mathbf{t}} (\lambda_1 f_1 + \lambda_2 f_2)$$

Grâce à la représentation de la rotation par un quaternion unitaire, ce critère devient :

$$\min_{\mathbf{q},\mathbf{t}} (f(\mathbf{q},\mathbf{t}) + \lambda (1 - \mathbf{q}^T\mathbf{q})^2) \tag{2.43}$$

avec :

$$\begin{aligned} f(\mathbf{q},\mathbf{t}) &= \lambda_1 f_1(\mathbf{q}) + \lambda_2 f_2(\mathbf{q},\mathbf{t}) \\ &= \lambda_1 \sum_{i=1}^{n} \|\mathbf{v}'_i - \mathbf{q} * \mathbf{v}_i * \overline{\mathbf{q}}\|^2 + \lambda_2 \sum_{i=1}^{n} \|\mathbf{q} * \mathbf{p}_i * \overline{\mathbf{q}} - (K_i - I)\,\mathbf{t} - \mathbf{p}'_i\|^2 \end{aligned}$$

qui est une somme de carrés de fonctions non linéaires. Le dernier terme $\lambda(1 - \mathbf{q}^T\mathbf{q})^2$ est un terme de pénalisation avec $\lambda > 0$ qui garantira que le quaternion sera unitaire. λ_1 et λ_2 sont deux coefficients de pondération. Deux possibilités sont envisageables pour minimiser cette fonction d'erreur donnée par l'équation (2.43). La première possibilité consiste à appliquer les techniques non linéaires classiques puisque la fonction d'erreur est une somme de carrés de fonctions [GMW89] [Fle90]. Les résultats de simulation et les résultats expérimentaux ont été obtenus par la méthode de minimisation non linéaire de Levenberg-Marquardt. Cette méthode est décrite dans [PFTW88]. Il est à noter que pour toutes nos expériences les coefficients de pondération ont été fixés de la façon suivante :

$$\begin{cases} \lambda_1 &= \lambda_2 = 1 \\ \lambda &= 2\,10^6 \end{cases}$$

Simplification de la fonction d'erreur

La deuxième possibilité consiste à simplifier la fonction d'erreur. En effet, en utilisant les propriétés associées aux quaternions nous pouvons simplifier l'expression de cette fonction. Nous avons obtenu une forme analytique simple pour la fonction f_1, (l'équation (2.41)). D'une manière similaire, nous pouvons simplifier la fonction f_2.

La fonction f_2 est la somme de termes ayant la forme suivante :

$$\|\mathbf{q} * \mathbf{p}_i * \overline{\mathbf{q}} - (K_i - I)\,\mathbf{t} - \mathbf{p}'_i\|^2$$

et nous avons:

$$\|\mathbf{q} * \mathbf{p}_i * \overline{\mathbf{q}} - (K_i - I)\,\mathbf{t} - \mathbf{p}'_i\|^2 \|\mathbf{q}\|^2 = \|\mathbf{q} * \mathbf{p}_i - (K_i - I)\,\mathbf{t} * \mathbf{q} - \mathbf{p}'_i * \mathbf{q}\|^2$$

En utilisant les formes matricielles du produit de deux quaternions, nous pouvons écrire :

$$\begin{aligned}
&\|\mathbf{q} * \mathbf{p}_i - (K_i - I)\,\mathbf{t} * \mathbf{q} - \mathbf{p}'_i * \mathbf{q}\|^2 \\
&\quad = (W(\mathbf{p}_i)\,\mathbf{q} - W(\mathbf{q})(K_i - I)\,\mathbf{t} - Q(\mathbf{p}'_i)\,\mathbf{q})^T\,(W(\mathbf{p}_i)\,\mathbf{q} - W(\mathbf{q})(K_i - I)\,\mathbf{t} - Q(\mathbf{p}'_i)\,\mathbf{q}) \\
&\quad = \mathbf{q}^T \mathcal{B}_i\,\mathbf{q} + \mathbf{t}^T \mathcal{C}_i\,\mathbf{t} + \delta_i\,\mathbf{t} - 2\,\mathbf{q}^T W(\mathbf{p}_i)^T W(\mathbf{q})(K_i - I)\,\mathbf{t}
\end{aligned}$$

avec :

$$\begin{aligned}
\mathcal{B}_i &= (\mathbf{p}_i^T \mathbf{p}_i + \mathbf{p}_i'^T \mathbf{p}'_i)\,I - W(\mathbf{p}_i)^T Q(\mathbf{p}'_i) - Q(\mathbf{p}'_i)^T W(\mathbf{p}_i) \\
\mathcal{C}_i &= K_i^T K_i - K_i - K_i^T + I \\
\delta_i &= 2\,\mathbf{p}_i'^T (K_i - I)
\end{aligned}$$

Le développement du dernier terme donne :

$$\begin{aligned}
-2\,\mathbf{q}^T W(\mathbf{p}_i)^T W(\mathbf{q})(K_i - I)\,\mathbf{t} &= -2\,\mathbf{p}_i^T Q(\mathbf{q})^T W(\mathbf{q})(K_i - I)\,\mathbf{t} \\
&= -2\,\mathbf{p}_i^T \left(W(\mathbf{q})^T Q(\mathbf{q})\right)^T (K_i - I)\,\mathbf{t}
\end{aligned}$$

La matrice $W(\mathbf{q})^T Q(\mathbf{q})$ n'est autre que la matrice de rotation inconnue (R_X ou R_Y). La matrice K_i est également une matrice de rotation (R_{A_i} ou N_i). A partir des équations (2.4) et (2.28), on peut écrire :

$$\begin{aligned}
R_X^T R_{A_i} &= R_{B_i} R_X^T \\
R_Y^T N_i &= R_{B_i} R_Y^T
\end{aligned}$$

d'où l'on déduit l'égalité suivante :

$$\left(W(\mathbf{q})^T Q(\mathbf{q})\right)^T (K_i - I) = (R_{B_i} - I)\left(W(\mathbf{q})^T Q(\mathbf{q})\right)^T$$

Finalement le dernier terme sera donné par :

$$\begin{aligned}
-2\,\mathbf{q}^T W(\mathbf{p}_i)^T W(\mathbf{q})(K_i - I)\,\mathbf{t} &= -2\,\mathbf{p}_i^T \left(W(\mathbf{q})^T Q(\mathbf{q})\right)^T (K_i - I)\,\mathbf{t} \\
&= -2\,\mathbf{p}_i^T (R_{B_i} - I)\left(W(\mathbf{q})^T Q(\mathbf{q})\right)^T \mathbf{t} \\
&= -2\,\mathbf{p}_i^T (R_{B_i} - I) Q(\mathbf{q})^T W(\mathbf{q})\,\mathbf{t} \\
&= \varepsilon_i\, Q(\mathbf{q})^T W(\mathbf{q})\,\mathbf{t}
\end{aligned}$$

où :

$$\varepsilon_i = -2\,\mathbf{p}_i^T (R_{B_i} - I)$$

Ce qui implique que chaque terme de la fonction f_2 pourrait être écrit sous la forme suivante :

$$\|\mathbf{q} * \mathbf{p}_i * \overline{\mathbf{q}} - (K_i - I)\,\mathbf{t} - \mathbf{p}'_i\|^2 = \mathbf{q}^T \mathcal{B}_i\,\mathbf{q} + \mathbf{t}^T \mathcal{C}_i\,\mathbf{t} + \delta_i\,\mathbf{t} + \varepsilon_i\,Q(\mathbf{q})^T W(\mathbf{q})\,\mathbf{t}$$

La fonction d'erreur f_2 devient donc :

$$\begin{aligned} f_2 &= \sum_{i=1}^{n} (\|\mathbf{q} * \mathbf{p}_i * \overline{\mathbf{q}} - (K_i - I)\,\mathbf{t} - \mathbf{p}'_i\|^2) \\ &= \mathbf{q}^T (\sum_{i=1}^{n} \mathcal{B}_i)\,\mathbf{q} + \mathbf{t}^T (\sum_{i=1}^{n} \mathcal{C}_i)\,\mathbf{t} + (\sum_{i=1}^{n} \delta_i)\,\mathbf{t} + (\sum_{i=1}^{n} \varepsilon_i)\,Q(\mathbf{q})^T W(\mathbf{q})\,\mathbf{t} \end{aligned}$$

La simplification du critère de l'équation (2.43) donne :

$$\min_{\mathbf{q},\mathbf{t}} (\mathbf{q}^T (\lambda_1\,\mathcal{A} + \lambda_2\,\mathcal{B})\,\mathbf{q} + \lambda_2\,\mathbf{t}^T \mathcal{C}\,\mathbf{t} + \lambda_2\,\delta\,\mathbf{t} + \lambda_2\,\varepsilon\,Q(\mathbf{q})^T W(\mathbf{q})\,\mathbf{t} + \lambda\,(1 - \mathbf{q}^T\mathbf{q})^2) \quad (2.44)$$

avec $\mathcal{B} = \sum_{i=1}^{n} \mathcal{B}_i$, $\mathcal{C} = \sum_{i=1}^{n} \mathcal{C}_i$, $\delta = \sum_{i=1}^{n} \delta_i$, $\varepsilon = \sum_{i=1}^{n} \varepsilon_i$, et $\mathcal{A}$ est donnée par l'équation (2.41).

Notons que cette fonction d'erreur (équation (2.44)) est une somme de 5 termes. Ce nombre est indépendant du nombre des déplacements du système caméra-pince. Afin de minimiser une telle fonction, on peut utiliser les méthodes *constrained step* comme la méthode de la région de confiance [Fle90] [Yas89].

2.8 Etude de la stabilité

Une des propriétés les plus importantes du calibrage caméra-pince est la stabilité en présence d'erreurs de mesures. Comme la transformation caméra-pince est déterminée à partir des déplacements de la caméra et de la pince, deux sources principales d'erreurs peuvent affecter le résultat du calibrage :

- les erreurs associées au calibrage de la caméra qui impliquent que le déplacement réel de la caméra ne correspond pas exactement avec celle qui est mesurée,

- les erreurs de modèle du robot, dûes aux erreurs de construction ou à une imprécision dans la détermination du modèle cinématique direct du robot, qui impliquent que la situation réelle de la pince du robot ne correspond pas exactement avec celle qui est mesurée.

Par conséquent, l'estimation de la transformation caméra-pince est entachée d'erreurs. La quantification de ces erreurs va déterminer la stabilité de la méthode utilisée ainsi que la comparaison avec d'autres méthodes.

Pour ce faire, nous adoptons la démarche suivante :

- On synthétise une transformation caméra-pince X ;

- On synthétise également $n+1$ positions de la caméra $A_1, \ldots, A_{n+1}$. Les n déplacements de la pince correspondant sont calculés par :

$$B_{ii+1} = X^{-1} A_{i+1} A_i^{-1} X$$

- On ajoute un bruit uniforme ou gaussien aux différents déplacements de la caméra et de la pince. La transformation caméra-pince est estimée en présence de ces perturbations en utilisant les trois méthodes présentées dans ce chapitre : la méthode linéaire de Tsai, la méthode analytique et la méthode non linéaire ;

- Pour chaque méthode on évalue l'erreur d'estimation en fonction du bruit et/ou en fonction du nombre de déplacements n.

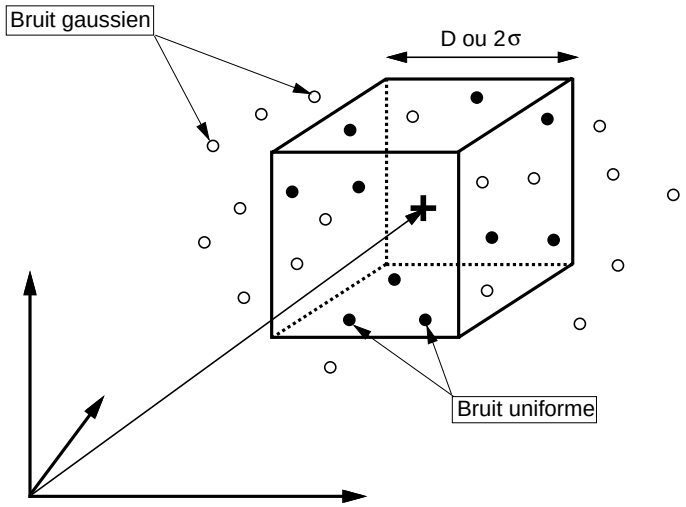


FIG. 2.7 - *Perturbation d'un vecteur 3D qui consiste à remplacer son extrémité réelle (+) par une autre choisie aléatoirement.*

Comme les translations et les rotations peuvent être représentées par des vecteurs de dimension 3, la perturbation d'un déplacement donné (rotation et translation) revient donc à perturber les vecteurs associés. La perturbation d'un vecteur consiste à ajouter à ses trois composantes des nombres aléatoires centrés qui sont choisis par un générateur de nombres aléatoires (voir Figure 2.7). La distribution de ces nombres aléatoires peut être, soit uniforme dans l'intervalle $[-D/2, +D/2]$, soit gaussienne d'écart type σ. Dans ce qui suit, le niveau du bruit est représenté comme un rapport qui est égal à l'amplitude D ou 2σ divisé par l'amplitude nominale du paramètre perturbé. Dans le cas des rotations, les vecteurs associés sont unitaires, il en résulte que le rapport exprimant le niveau du bruit est tout simplement donné par D ou 2σ. Dans le cas des translations, le niveau du bruit est donné par l'amplitude D ou 2σ divisé par une valeur nominale $\|t_{nominal}\|$ donnée par :

$$\|t_{nominal}\| = \frac{\sum_{i=1}^{n} (\|\mathbf{t}_{A_{ii+1}}\| + \|\mathbf{t}_{B_{ii+1}}\|)}{2n}$$

où $\mathbf{t}_{A_{ii+1}}$ est le vecteur de translation associé au déplacement A_{ii+1}. Pour chaque niveau de bruit, nous calculons les erreurs d'estimation pour 1000 configurations aléatoires. Les

diagrammes suivants illustrent la moyenne de ces erreurs. L'erreur de la rotation et de position sont calculées de la manière suivante:

$$e_{rot} = \sqrt{\frac{1}{1000}\sum_{j=1}^{1000} \|\tilde{R}_j - R\|^2}$$

et:

$$e_{tr} = \frac{\sqrt{\frac{1}{1000}\sum_{j=1}^{1000} \|\tilde{\mathbf{t}}_j - \mathbf{t}\|^2}}{\|\mathbf{t}\|}$$

où R et $\mathbf{t}$ sont les valeurs théoriques de la rotation et de la translation associées à la transformation X, $\tilde{R}_j$ et $\tilde{t}_j$ sont les estimations de la rotation et de la translation correspondant à la configuration aléatoire j. Dans toutes les simulations présentées dans ce paragraphe nous avons $\|\mathbf{t}\| = 157$mm.

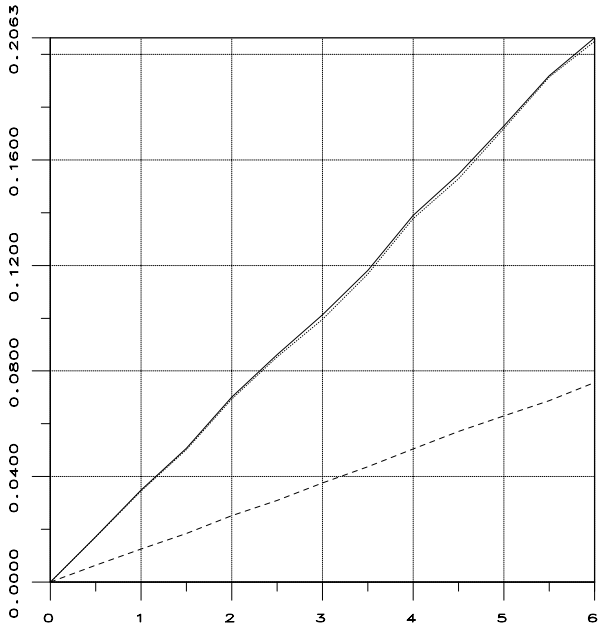
Les figures suivantes illustrent les erreurs d'estimation en fonction du pourcentage du bruit. Le pourcentage du bruit varie entre 0.5% et 6%. Les courbes continues (—) correspondent à la méthode linéaire de Tsai, les courbes pointillées (...) correspondent à la méthode analytique, les courbes interrompues (- - -) à la méthode non linéaire. Les Figures 2.8,..., 2.11 montrent les résultats de simulation associés à deux déplacements du système caméra-pince ($n = 2$) alors que la Figure 2.12 correspond à n variant de 2 à 9.

Les Figures 2.8.a et 2.8.b présentent les erreurs de rotation et de translation en fonction du pourcentage d'un bruit uniforme qui affecte uniquement les parties rotation des déplacements de la caméra et de la pince. Les Figures 2.9.a et 2.9.b présentent les erreurs de rotation et de translation en fonction du pourcentage d'un bruit uniforme qui affecte les parties rotation et translation des déplacements de la caméra et de la pince. L'axe horizontale présente le pourcentage du bruit affectant ces deux parties.

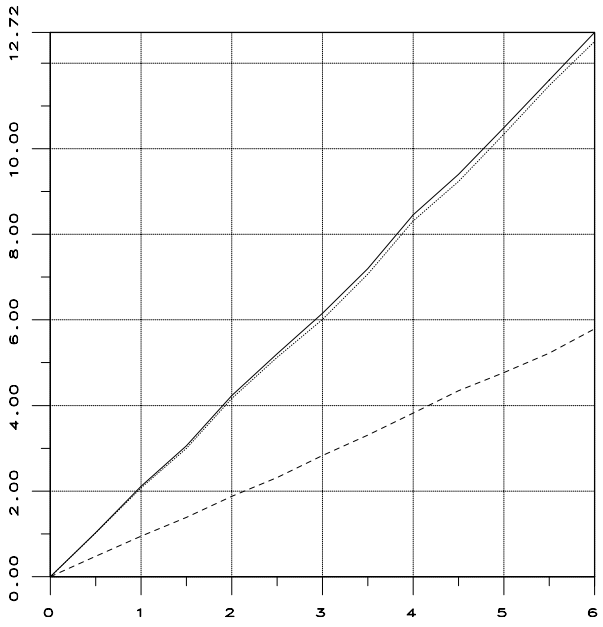
Les Figures 2.10 et 2.11 sont respectivement équivalentes aux Figures 2.8 et 2.9 mais le bruit ajouté est un bruit gaussien. Par exemple, sur la Figure 2.9.b lorsque le pourcentage du bruit uniforme affectant les axes de rotation et les vecteurs de translation est égal à 3% alors l'erreur de translation (moyenne) est égale à (5.8%= 9.1mm) pour la méthode non linéaire et à (7.5% = 11.8mm) pour les deux autres méthodes.

Il est clair que la méthode de Tsai et la méthode analytique ont presque la même stabilité en présence de bruit tandis que la méthode non linéaire fournit la meilleure stabilité. Ceci reste valable pour tous les cas de figure présentés ici. D'autre part, la suprématie de la méthode non linéaire est d'autant plus importante que les mesures des parties translation sont précises. Le fait que la stabilité des deux premières méthodes est la même est expliqué par le fait que ces deux méthodes divisent le problème en deux étapes.

Les Figures 2.12.a et 2.12.b montrent les erreurs de rotation et de translation en fonction de la racine carrée du nombre de déplacements ($\sqrt{n}$ varie de 1.414 à 3). Le bruit gaussien affectant les mesures est de 6% pour les axes de rotation et de 2% pour les vecteurs de translation. Par exemple, pour 4 déplacements l'erreur de translation est égale à 4% pour la méthode non linéaire et à 6.5% pour les deux autres méthodes.

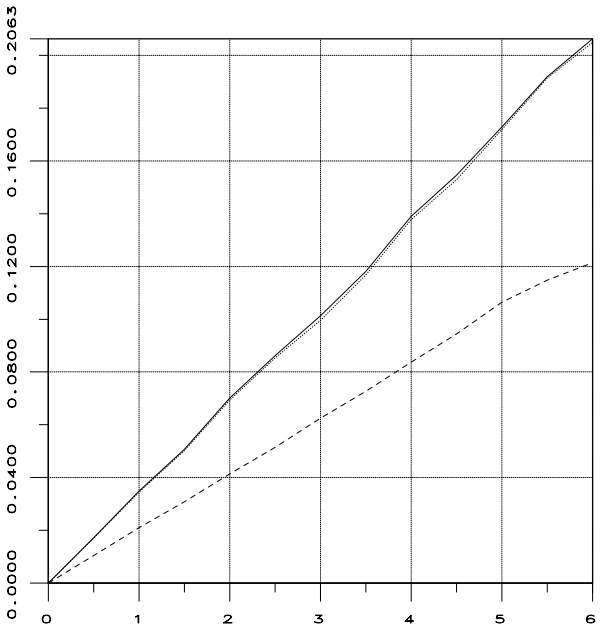


a) Erreur d'orientation.

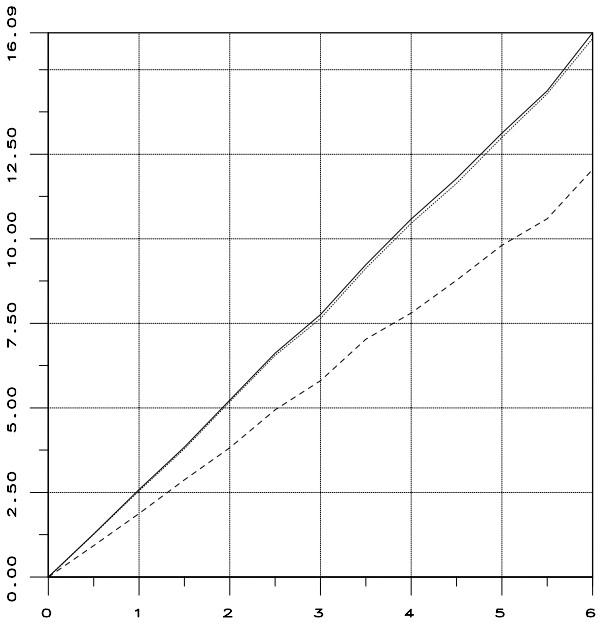


b) Erreur de translation.

FIG. 2.8 - *Erreurs d'orientation et de translation en présence d'un bruit uniforme affectant les axes de rotation. L'axe horizontal présente le pourcentage du bruit.*

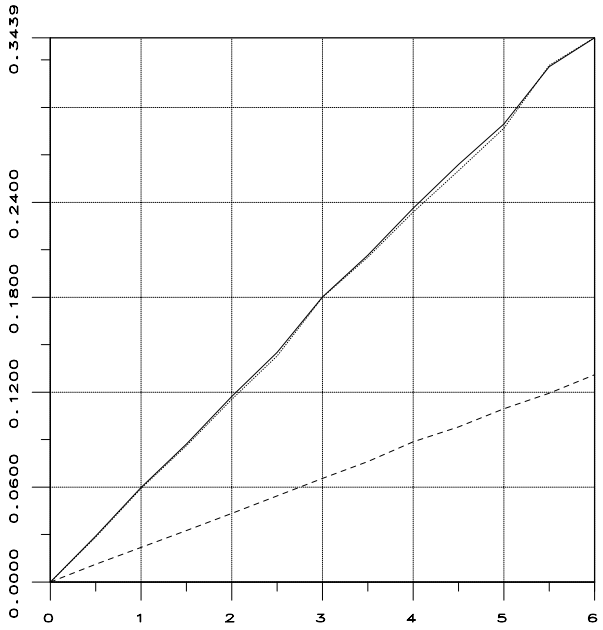


a) Erreur d'orientation.

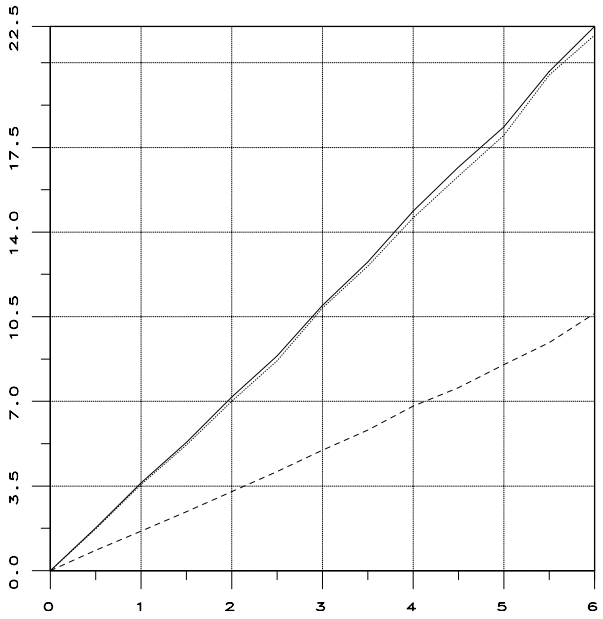


b) Erreur de translation.

FIG. 2.9 - *Erreurs d'orientation et de translation en présence d'un bruit uniforme affectant les axes de rotation et les vecteurs de translation. L'axe horizontal présente le pourcentage du bruit.*

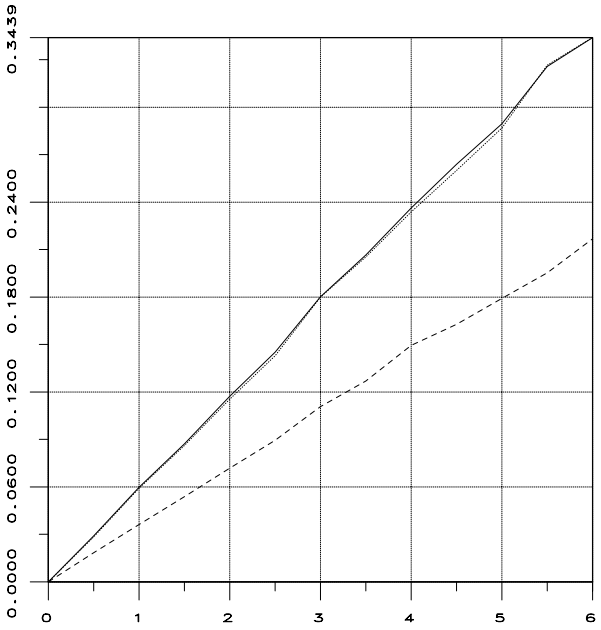


a) Erreur d'orientation.

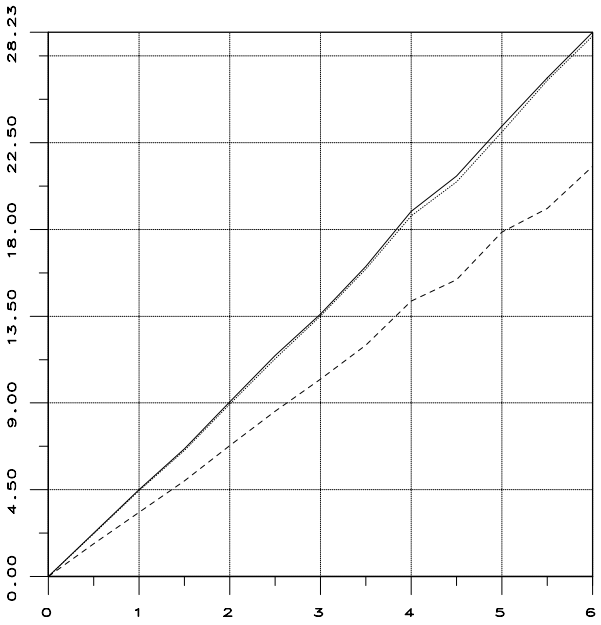


b) Erreur de translation.

FIG. 2.10 - *Erreurs d'orientation et de translation en présence d'un bruit gaussien affectant les axes de rotation. L'axe horizontal présente le pourcentage du bruit.*

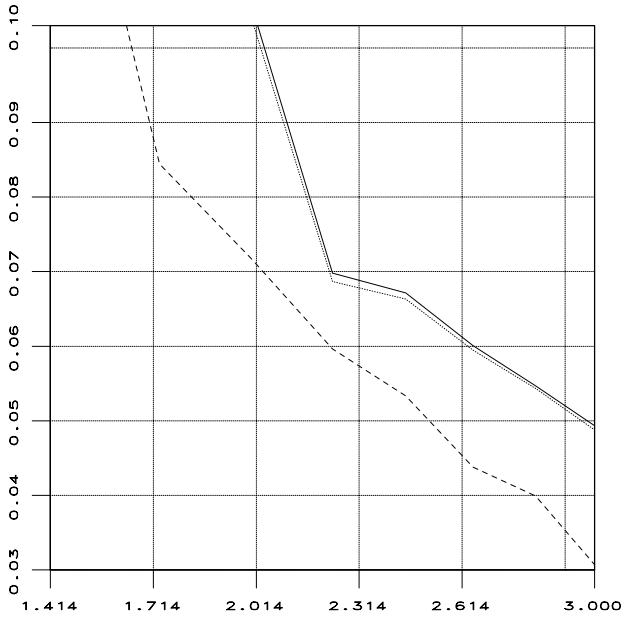


a) Erreur d'orientation.

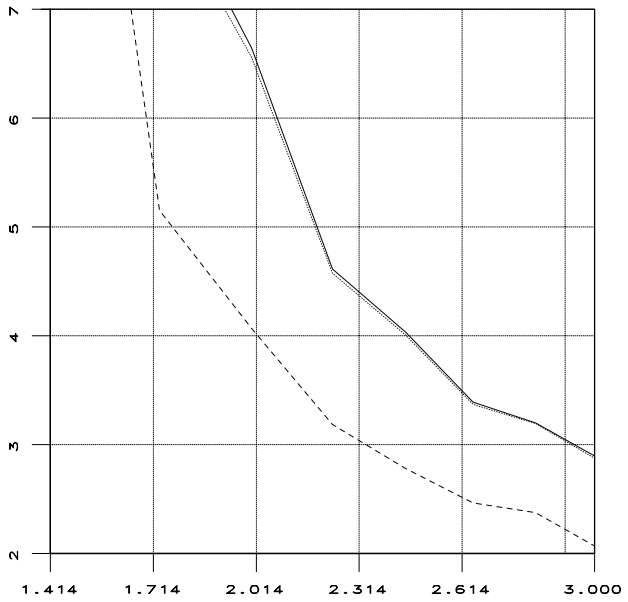


b) Erreur de translation.

FIG. 2.11 - *Erreurs d'orientation et de translation en présence d'un bruit gaussien affectant les axes de rotation et les vecteurs de translation. L'axe horizontal présente le pourcentage du bruit.*



a) Erreur d'orientation.



b) Erreur de translation.

FIG. 2.12 - *Erreurs d'orientation et de translation en fonction du nombre de déplacements. Le bruit gaussien affectant les déplacements est de 6% pour les axes de rotations et de 2% pour les vecteurs de translation.*

2.9 Expérimentations

Pour l'expérimentation en site réel, nous avons testé les méthodes du calibrage sur trois ensembles de données. Le premier a été fourni par François Chaumette de l'IRISA [1], les deux autres ensembles ont été obtenus au LIFIA [2]. Le premier ensemble est formé de 17 positions du système caméra-pince. Le deuxième ensemble est formé de 7 positions. Le troisième ensemble est formé de 6 positions. Pour les deux derniers ensembles nous avons également les matrices de projection. Par conséquent, ces deux derniers ensembles nous permettent d'effectuer une comparaison entre les deux formulations : $(AX = XB)$ et $(MY = M'YB)$. Les positions de la pince sont choisies de telle sorte que la mire du calibrage soit dans le champ de vue de la caméra embarquée. Pour une position quelconque, la situation de la pince par rapport à la base du robot est obtenue en utilisant les mesures fournies par les capteurs articulaires et le modèle géométrique direct du robot.

Les conditions expérimentales associées aux deux ensembles obtenus au LIFIA sont les suivantes. La matrice de projection correspondant à une position donnée de la pince est estimée à partir de l'observation de la projection d'un certain nombre de points 3D connus dans le repère du calibrage "repère mire" (voir Figure 2.14). La mire utilisée est formée d'une grille rectangulaire de 200×300 mm que l'on peut déplacer très précisément par rapport à un axe normal à son plan. La grille est constituée de 23 carrés noirs sur un fond blanc. Les 92 sommets de ces carrés forment les points de mesure utilisés lors du calibrage de la caméra. Dans le plan de la grille, ces points sont mesurés avec une précision de 0.1 mm. Cinq positions équidistantes ont été choisies sur un intervalle de 50 mm, de manière à avoir un ensemble de 460 points de mesure lorsque tous les points sont visibles par la caméra. Lors du calibrage caméra-pince la distance de la mire par rapport à la caméra varie entre 600 mm et 1000 mm. L'estimation des matrices de projection est effectuée grâce à la méthode de Faugeras-Toscani [FT86]. Vu le grand nombre de points de mesure (460 points) et la précision avec laquelle on extrait leur image (entre 0.1 et 0.2 pixel), l'estimation des matrices de projection par la méthode de Faugeras-Toscani est assez précise. Par contre, les mesures des déplacements de la pince du robot ne sont pas assez précises (modèle géométrique du robot est entachée d'erreurs).

Pour chaque ensemble de données, nous avons testé les trois méthodes de résolution. Comme la valeur exacte de la matrice caméra-pince n'est pas connue, nous devons comparer les erreurs résiduelles du système d'équations matricielles. Les tableaux 2.1, 2.2 et 2.3 présentent les résultats obtenus avec les trois ensembles de données. Ainsi, la deuxième colonne de ces trois tableaux présente la somme au carré de l'erreur résiduelle de la partie rotation à savoir $(R_A R_X = R_X R_B)$. La troisième colonne de ces trois tableaux présentent l'erreur relative de la partie translation.

D'après ces résultats expérimentaux, il s'avère que, d'une part, la méthode non linéaire fournit une estimation de la translation qui est la plus précise (parfois, au détriment de la rotation, mais ceci peut être évité par un choix judicieux des coefficients de pondération) ; d'autre part, la nouvelle formulation fournit des résultats plus précis que celle de la formulation classique. La quatrième colonne du tableau 2.3 présente le temps de calcul de chaque algorithme.

1. Institut de Recherche en Informatique et Systèmes Aléatoires.
2. Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle.

FIG. 2.13 - *Système caméra-pince.*

FIG. 2.14 - *Mire utilisée.*

$AX = XB$	$\sum \|R_A R_X - R_X R_B\|^2$	$\frac{\sum \|(R_A - I)t_X - R_X t_B + t_A\|^2}{\sum \|R_X t_B - t_A\|^2}$
Tsai-Lenz	0.0006	0.032
Solution analytique	0.0005	0.029
Optimisation non linéaire	0.0003	0.019

TAB. 2.1 - *La formulation classique utilisée avec le premier ensemble de mesures (17 positions du système caméra-pince).*

$AX = XB$	$\sum \|R_A R_X - R_X R_B\|^2$	$\frac{\sum \|(R_A - I)t_X - R_X t_B + t_A\|^2}{\sum \|R_X t_B - t_A\|^2}$
Tsai-Lenz	0.0014	0.036
Solution analytique	0.0014	0.023
Optimisation non linéaire	0.0017	0.015
$MY = M'YB$	$\sum \|NR_Y - R_Y R_B\|^2$	$\frac{\sum \|(N - I)t_Y - R_Y t_B + t_N\|^2}{\sum \|R_Y t_B - t_N\|^2}$
Tsai-Lenz	0.0031	0.0021
Solution analytique	0.0015	0.001
Optimisation non linéaire	0.0013	0.0006

TAB. 2.2 - *Les deux formulations utilisées avec le deuxième ensemble de mesures (7 positions).*

$AX = XB$	$\sum \|R_A R_X - R_X R_B\|^2$	$\frac{\sum \|(R_A - I)t_X - R_X t_B + t_A\|^2}{\sum \|R_X t_B - t_A\|^2}$	temps CPU [a]
Tsai-Lenz	0.014	0.23	0.08
Solution analytique	0.036	0.223	0.06
Optimisation non linéaire	0.258	0.058	0.21
$MY = M'YB$	$\sum \|NR_Y - R_Y R_B\|^2$	$\frac{\sum \|(N - I)t_Y - R_Y t_B + t_N\|^2}{\sum \|R_Y t_B - t_N\|^2}$	
Tsai-Lenz	0.038	0.039	0.06
Solution analytique	0.035	0.037	0.08
Optimisation non linéaire	0.04	0.034	0.25

TAB. 2.3 - *Les deux formulations utilisées avec le troisième ensemble de mesures (6 positions).*

[a] Sparc-10.

2.10 Calibrage robot/environnement

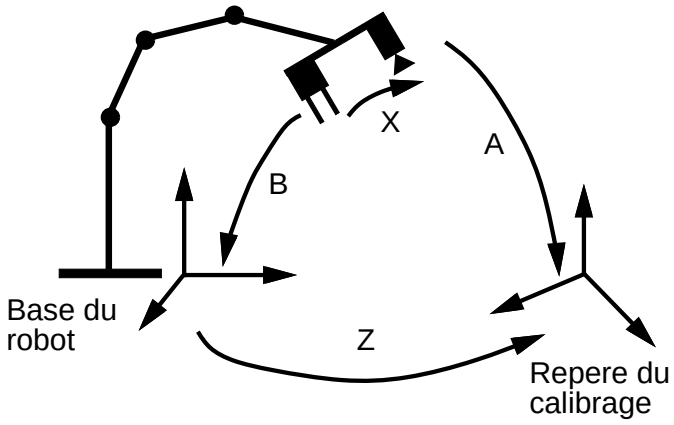


FIG. 2.15 - *Calibrage caméra-pince et robot-environnement.*

Nous allons à présent généraliser le concept du calibrage caméra-pince et proposer de déterminer simultanément deux transformations rigides : la transformation caméra-pince (X) et la transformation entre la base du robot et le repère du calibrage (Z) (voir Figure 2.15). Notons que ce calibrage peut être utilisé pour déterminer la relation entre deux robots ou la relation entre une caméra fixe et un robot. Il est à noter que la transformation Z pourrait être déterminée :

- soit à partir de la transformation caméra-pince ;
- soit par utilisation de la formulation classique ($AZ = ZB$).

Cependant, dans les deux cas, l'estimation des deux transformations va être effectuée en deux étapes : i) une transformation (X ou Z) va être estimée en utilisant la formulation du calibrage caméra-pince, ii) l'estimation de l'autre transformation est déduite à partir de la première. Par conséquent, la transformation qui est estimée dans la deuxième étape va dépendre de la position choisie et va être influencée par les erreurs commises sur l'estimation de l'autre transformation. Dans [HV93], on peut trouver une méthode analytique permettant de déterminer la relation base du robot-repère du calibrage. A partir de la Figure 2.15, on peut écrire :

$$AX = ZB \tag{2.45}$$

où X et Z sont les deux transformations rigides inconnues, A est la transformation caméra-repère du calibrage et B est la transformation pince-base du robot.

2.10.1 Méthode linéaire

Cette méthode a été proposée par Zhuang et al. [ZRS94]. Soient R_A, R_B, R_X et R_Z les matrices de rotation associées aux déplacements A, B, X et Z. Soient $\mathbf{t}_A$, $\mathbf{t}_B$, $\mathbf{t}_X$ et

$\mathbf{t}_Z$ les vecteurs de translation de ces déplacements. La décomposition de l'équation (2.45) donne :

$$\begin{bmatrix} R_A & \mathbf{t}_A \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} R_X & \mathbf{t}_X \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} R_Z & \mathbf{t}_Z \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} R_B & \mathbf{t}_B \\ \mathbf{0}^T & 1 \end{bmatrix}$$

d'où :

$$R_A R_X = R_Z R_B \tag{2.46}$$
$$R_A \mathbf{t}_X + \mathbf{t}_A = R_Z \mathbf{t}_B + \mathbf{t}_Z \tag{2.47}$$

Dans un premier temps, on estime les rotations R_X et R_Z en résolvant un système linéaire surcontraint. Une fois ces rotations calculées, la détermination des vecteurs de translation $\mathbf{t}_X$ et $\mathbf{t}_Z$ devient un problème linéaire classique.

Comme le quaternion associé au produit de deux matrices de rotation est égal au produit des quaternions associés à ces deux matrices, l'équation (2.46) donne :

$$\mathbf{q}_A * \mathbf{q}_X = \mathbf{q}_Z * \mathbf{q}_B \tag{2.48}$$

où les inconnues sont les quaternions $\mathbf{q}_X$ et $\mathbf{q}_Z$.

On sait qu'un quaternion peut s'écrire comme la concaténation d'un réel avec un vecteur de dimension 3. Avec cette notation, les quatre quaternions de l'équation (2.48) peuvent s'écrire sous la forme suivante :

$$\mathbf{q}_A = \begin{bmatrix} a_0 \\ \mathbf{a} \end{bmatrix}, \quad \mathbf{q}_B = \begin{bmatrix} b_0 \\ \mathbf{b} \end{bmatrix}, \quad \mathbf{q}_X = \begin{bmatrix} x_0 \\ \mathbf{x} \end{bmatrix} \text{ et } \quad \mathbf{q}_Z = \begin{bmatrix} z_0 \\ \mathbf{z} \end{bmatrix}$$

En développant l'équation (2.48), on obtient :

$$[a_0 x_0 - \mathbf{a} \cdot \mathbf{x},\ a_0 \mathbf{x} + x_0 \mathbf{a} + \mathbf{a} \times \mathbf{x}] = [z_0 b_0 - \mathbf{b} \cdot \mathbf{z},\ z_0 \mathbf{b} + b_0 \mathbf{z} - \mathbf{b} \times \mathbf{z}]$$

L'identification des deux termes de cette équation fournit :

$$a_0 x_0 - \mathbf{a} \cdot \mathbf{x} = z_0 b_0 - \mathbf{b} \cdot \mathbf{z} \tag{2.49}$$
$$a_0 \mathbf{x} + x_0 \mathbf{a} + \mathbf{a} \times \mathbf{x} = z_0 \mathbf{b} + b_0 \mathbf{z} - \mathbf{b} \times \mathbf{z} \tag{2.50}$$

Si $a_0 \neq 0$, alors on peut calculer x_0 à partir de l'équation (2.49) :

$$x_0 = (\mathbf{a}/a_0) \cdot \mathbf{x} + (b_0/a_0)\, z_0 - (\mathbf{b}/a_0) \cdot \mathbf{z} \tag{2.51}$$

En injectant cette expression dans l'équation (2.50), on obtient :

$$[(b_0/a_0)\, \mathbf{a} - \mathbf{b}]\, z_0 + a_0 \mathbf{x} + \mathbf{a}(\mathbf{a}/a_0) \cdot \mathbf{x} + \mathbf{a} \times \mathbf{x} + \mathbf{b} \times \mathbf{z} - (\mathbf{a}/a_0)\mathbf{b} \cdot \mathbf{z} - b_0 \mathbf{z} = \mathbf{0} \tag{2.52}$$

En utilisant la représentation matricielle pour le produit scalaire et le produit vectoriel et, en divisant par z_0 l'équation (2.52) (on suppose que $z_0 \neq 0$), l'équation (2.52) se réécrit comme suit :

$$(a_0 I + \mathbf{a}\,\mathbf{a}^T/a_0 + S(\mathbf{a}))\, \mathbf{x}/z_0 + (-b_0 I - \mathbf{a}\,\mathbf{b}^T/a_0 + S(\mathbf{b}))\, \mathbf{z}/z_0 = \mathbf{b} - (b_0/a_0)\, \mathbf{a} \tag{2.53}$$

où $S(\mathbf{a})$ est la matrice antisymétrique associée au vecteur $\mathbf{a}$.

En posant :

$$\mathbf{u} = \underbrace{\begin{bmatrix} \mathbf{x}/z_0 \\ \mathbf{z}/z_0 \end{bmatrix}}_{6\times 1} = [\ u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5 \quad u_6\ ]^T$$

et:

$$\underbrace{J_i}_{3\times 6} \equiv (a_0 I + \mathbf{a}\,\mathbf{a}^T/a_0 + S(\mathbf{a}),\ -b_0 I - \mathbf{a}\,\mathbf{b}^T/a_0 + S(\mathbf{b}))$$

l'équation vectorielle (2.53) donne un système linéaire en six inconnues (les composantes de $\mathbf{u}$) :

$$J_i\,\mathbf{u} = \mathbf{b} - (b_0/a_0)\,\mathbf{a}\}_i$$

Pour n positions de la pince, on doit avoir le système linéaire surcontraint suivant :

$$\underbrace{\begin{bmatrix} \vdots \\ J_i \\ \vdots \end{bmatrix}}_{3n\times 6} \mathbf{u} = \begin{bmatrix} \vdots \\ \mathbf{b} - (b_0/a_0)\,\mathbf{a}\}_i \\ \vdots \end{bmatrix}$$

Une fois qu'on a déterminé le vecteur $\mathbf{u}$ au sens des moindres carrés, les composantes des quaternions unitaires $\mathbf{q}_X$ et $\mathbf{q}_Z$ sont données par :

$$\begin{aligned} z_0 &= \pm\{1 + u_4^2 + u_5^2 + u_6^2\}^{-1/2} \\ \mathbf{z} &= z_0[\ u_4 \quad u_5 \quad u_6\ ] \\ \mathbf{x} &= z_0[\ u_1 \quad u_2 \quad u_3\ ] \\ x_0 &= \pm\{1 - x_1^2 - x_2^2 - x_3^2\}^{1/2} \end{aligned}$$

Le signe de x_0 est déterminé à partir de l'équation (2.51).

Notons que les conditions d'unicité de la solution sont les mêmes que celles de la solution caméra-pince ($AX = XB$).

2.10.2 Méthode non linéaire

Dans le paragraphe précédent, nous venons de voir que les vecteurs de translation sont estimés après l'estimation des matrices de rotation. Nous proposons d'estimer simultanément les rotations et les translations associées aux transformations X et Z.

Le produit de deux quaternions peut s'écrire sous forme matricielle. On a :

$$\begin{aligned} \mathbf{q}_A * \mathbf{q}_X &= Q(\mathbf{q}_A)\,\mathbf{q}_X \\ \mathbf{q}_Z * \mathbf{q}_B &= W(\mathbf{q}_B)\,\mathbf{q}_Z \end{aligned}$$

En remplaçant ces deux équations dans l'équation (2.48), on obtient :

$$Q(\mathbf{q}_A)\,\mathbf{q}_X - W(\mathbf{q}_B)\,\mathbf{q}_Z = \mathbf{0}$$

La matrice de rotation R_Z est donnée en fonction du quaternion $\mathbf{q}_Z = (z_0\ z_1\ z_2\ z_3)^T$:

$$R_Z = R(\mathbf{q}_Z) = \begin{bmatrix} z_0^2+z_1^2-z_2^2-z_3^2 & 2(z_1z_2-z_0z_3) & 2(z_1z_3+z_0z_2) \\ 2(z_1z_2+z_0z_3) & z_0^2-z_1^2+z_2^2-z_3^2 & 2(z_2z_3-z_0z_1) \\ 2(z_1z_3-z_0z_2) & 2(z_2z_3+z_0z_1) & z_0^2-z_1^2-z_2^2+z_3^2 \end{bmatrix}$$

Chaque position du système caméra-pince i fournit les deux contraintes suivantes :

$$\begin{aligned} Q(\mathbf{q}_{Ai})\,\mathbf{q}_X - W(\mathbf{q}_{Bi})\,\mathbf{q}_Z &= \mathbf{0} \\ R_{Ai}\,\mathbf{t}_X + \mathbf{t}_{Ai} - R_Z(\mathbf{q}_Z)\,\mathbf{t}_{Bi} - \mathbf{t}_Z &= \mathbf{0} \end{aligned}$$

Pour n positions du système, la fonction d'erreur est :

$$\begin{aligned} & f(\mathbf{q}_X, \mathbf{q}_Z, \mathbf{t}_X, \mathbf{t}_Z) \\ = & \sum_{i=1}^{n} \lambda_1 \left(\|Q(\mathbf{q}_{Ai})\,\mathbf{q}_X - W(\mathbf{q}_{Bi}))\,\mathbf{q}_Z\|^2\right) + \sum_{i=1}^{n} \lambda_2 \left(\|R_{Ai}\,\mathbf{t}_X + \mathbf{t}_{Ai} - R_Z(\mathbf{q}_Z)\,\mathbf{t}_{Bi} - \mathbf{t}_Z\|^2\right) + \\ & \lambda_X \left(1 - \|\mathbf{q}_X\|^2\right)^2 + \lambda_Z \left(1 - \|\mathbf{q}_Z\|^2\right)^2 \end{aligned}$$

où les éléments à estimer sont les éléments des quaternions unitaires $\mathbf{q}_X$ et $\mathbf{q}_Z$ et les éléments des vecteurs de translation $\mathbf{t}_X$ et $\mathbf{t}_Z$. Il est clair que la minimisation de cette fonction d'erreur est un problème d'optimisation non linéaire. Pour toutes nos expériences les coefficients de pondération de la fonction d'erreur ont été fixés de la façon suivante :

$$\begin{cases} \lambda_1 = \lambda_2 = 1 \\ \lambda_X = \lambda_Z = 2\ 10^6 \end{cases}$$

2.10.3 Comparaison des deux méthodes

Ce paragraphe contient deux types de résultats. Le premier, avec des données synthétiques ; le second, avec des données réelles. Ces deux types visent à comparer la précision des deux méthodes présentées ci-dessus.

Données synthétiques

Par une démarche similaire à celle du paragraphe 2.8, nous évaluons les erreurs d'orientation et de position des deux méthodes (linéaire et non linéaire). Ainsi, des positions synthétiques sont créées (les transformations A_i, B_i, X et Z) et les solutions calculées en présence de bruit sont comparées avec les transformations réelles qui ont servi à calculer les positions synthétiques. L'erreur d'orientation est donnée par l'angle de rotation nécessaire pour aligner les repères dans ses positions estimées avec ces mêmes repères dans ses positions réelles. L'erreur de translation est définie comme la norme du vecteur de translation nécessaire pour faire coïncider la translation estimée des repères avec la translation réelle, divisée par la norme de la translation réelle. Nous considérons 500 configurations aléatoires du bruit. Nous représentons la moyenne de ces erreurs.

Les Figures 2.16 et 2.17 montrent les résultats de simulation associés à trois positions du système caméra-pince ($n = 3$) alors que les Figures 2.18 et 2.19 correspondent à n

variant de 3 à 8. Les courbes continues (—) correspondent à la méthode linéaire, les courbes interrompues (- - -) à la méthode non linéaire. Les translations réelles $\|\mathbf{t}_X\|$ et $\|\mathbf{t}_Z\|$ sont respectivement : 229 mm et 768 mm.

La Figure 2.16 présente les erreurs de rotation et de translation (pour les deux transformations X et Z) en fonction du pourcentage d'un bruit uniforme affectant les axes de rotation. La Figure 2.17 est équivalente à la Figure 2.16 mais le bruit ajouté est un bruit gaussien.

Les Figures 2.18 et 2.19 présente les erreurs de rotation et de translation en fonction de la racine carrée du nombre de positions ($\sqrt{n}$ varie de 1.732 à 2.828). Le bruit gaussien affectant les mesures est de 6% pour les axes de rotation et de 2% pour les vecteurs de translation. La Figure 2.18 correspond à une perturbation qui affecte uniquement les positions du robot ; la Figure 2.19 correspond à une perturbation qui affecte les positions du robot et de la caméra.

Données réelles

Ces données sont fournies par les mêmes mesures utilisées dans le calibrage caméra-pince. Ainsi, nous reprenons les deux premiers ensembles de données du paragraphe 2.9. Les tableaux 2.4 et 2.5 présentent les résultats obtenus avec ces deux ensembles de données. Ainsi, la deuxième colonne de ces deux tableaux présente la somme au carré de l'erreur résiduelle de la partie rotation à savoir ($R_A R_X = R_Z R_B$). La troisième colonne de ces trois tableaux présentent l'erreur relative de la partie translation.

	$\sum \|R_A R_X - R_Z R_B\|^2$	$\left(\frac{\sum \|(R_A t_X + t_A - R_Z t_B - t_Z\|^2}{\sum \|R_A t_X + t_A\|^2}\right)^{1/2}$
Méthode linéaire	0.00031	0.00068
Méthode non linéaire	0.00071	0.00021

TAB. 2.4 - *La formulation $AX = ZB$ utilisée avec le premier ensemble de mesures (17 positions du système caméra-pince).*

	$\sum \|R_A R_X - R_Z R_B\|^2$	$\left(\frac{\sum \|(R_A t_X + t_A - R_Z t_B - t_Z\|^2}{\sum \|R_A t_X + t_A\|^2}\right)^{1/2}$
Méthode linéaire	0.12174	0.00738
Méthode non linéaire	0.00109	0.00451

TAB. 2.5 - *La formulation $AX = ZB$ utilisée avec le deuxième ensemble de mesures (7 positions du système caméra-pince).*

Il arrive parfois que la méthode non linéaire fournisse une erreur résiduelle, en la partie rotation, plus grande que celle de la méthode linéaire (voir tableau 2.4). Cela est dû au

fait que la méthode non linéaire minimise la somme des erreurs résiduelles associées aux parties rotation et translation.

Les deux transformations X et Z correspondant au deuxième ensemble sont données par (les translations sont données en millimètres) :

- **Méthode linéaire :**

$$\begin{cases} X = \begin{bmatrix} -0.9799 & -0.1990 & 0.0092 & -12.1 \\ 0.1960 & -0.9714 & -0.1339 & 66.8 \\ 0.0356 & -0.1294 & 0.9909 & 36.1 \end{bmatrix} \\ Z = \begin{bmatrix} -0.0633 & -0.0274 & -0.9976 & 437.8 \\ 0.9959 & 0.0619 & -0.0649 & 218.4 \\ 0.0635 & -0.9977 & 0.0234 & 1419.7 \end{bmatrix} \end{cases}$$

- **Méthode non linéaire :**

$$\begin{cases} X = \begin{bmatrix} -0.9965 & -0.0766 & 0.0340 & -12.3 \\ 0.0703 & -0.9846 & -0.1597 & 64.5 \\ 0.0457 & -0.1567 & 0.9866 & 39.0 \end{bmatrix} \\ Z = \begin{bmatrix} -0.0244 & -0.0320 & -0.9988 & 446.0 \\ 0.9986 & 0.0385 & -0.0256 & 211.9 \\ 0.0393 & -0.9984 & 0.0310 & 1411.4 \end{bmatrix} \end{cases}$$

Pour vérifier les résultats obtenus, nous pouvons effectuer une comparaison avec la solution pour la transformation X obtenue grâce à la formulation $(AX = XB)$; nous rappelons que les données des deux formulations $(AX = XB)$ et $(AX = ZB)$ sont les mêmes. Ainsi, pour le deuxième ensemble de données, la matrice X obtenue grâce à la formulation $(AX = XB)$ est la suivante :

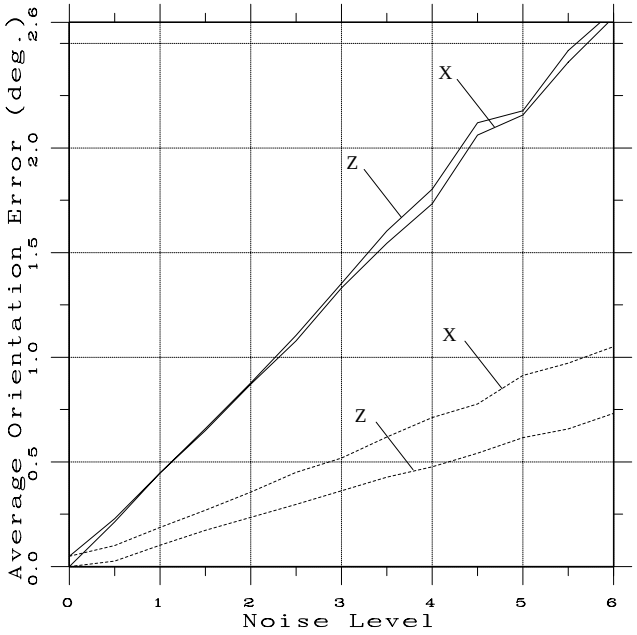
$$X = \begin{bmatrix} -0.9964 & -0.0752 & 0.0398 & -13.2 \\ 0.0681 & -0.9852 & -0.1569 & 64.2 \\ 0.0510 & -0.1536 & 0.9868 & 39.2 \end{bmatrix}$$

Cette transformation est pratiquement la même matrice obtenue avec la méthode non linéaire associée à la formulation $(AX = ZB)$. Ce qui tend à montrer que les résultats de la méthode non linéaire sont plus stables que ceux de la méthode linéaire.

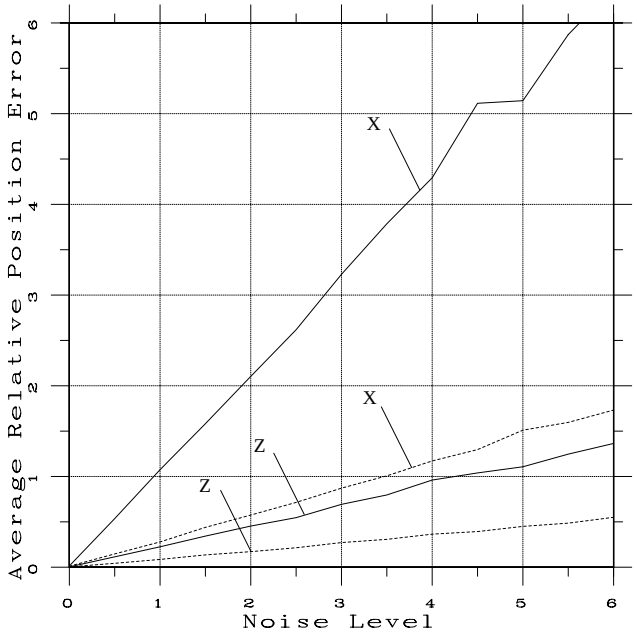
Il est à noter que dans le cas où les rotations R_{Ai} n'ont pas la forme particulière associée à l'angle de rotation π, on peut initialiser la méthode non linéaire avec les solutions données par la méthode linéaire, sinon, on initialise la méthode non linéaire par les solutions associées à la formulation $(AX = XB)$.

En résumé, la méthode non linéaire présente deux avantages par rapport à la méthode linéaire :

- elle est plus stable en présence de bruit ;
- elle peut s'affranchir des contraintes imposées par la méthode linéaire qui exige que les matrices de rotation de chaque position (R_{Ai}) doivent avoir leur angle de rotation différent de π.

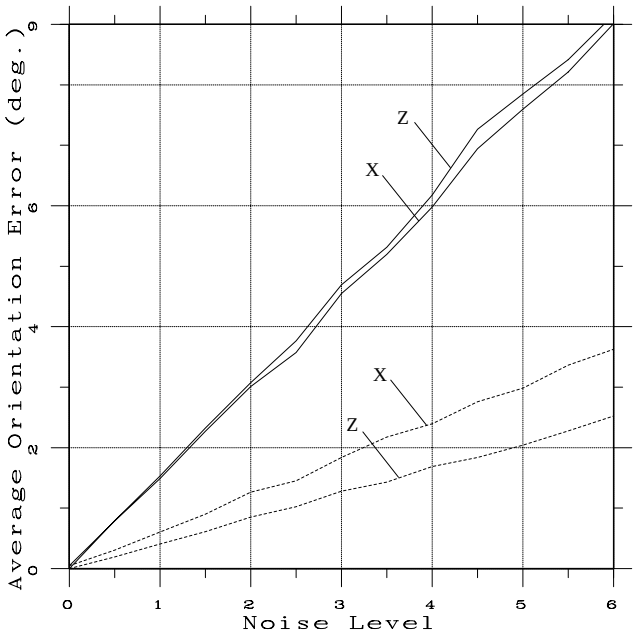


a) Erreur d'orientation.

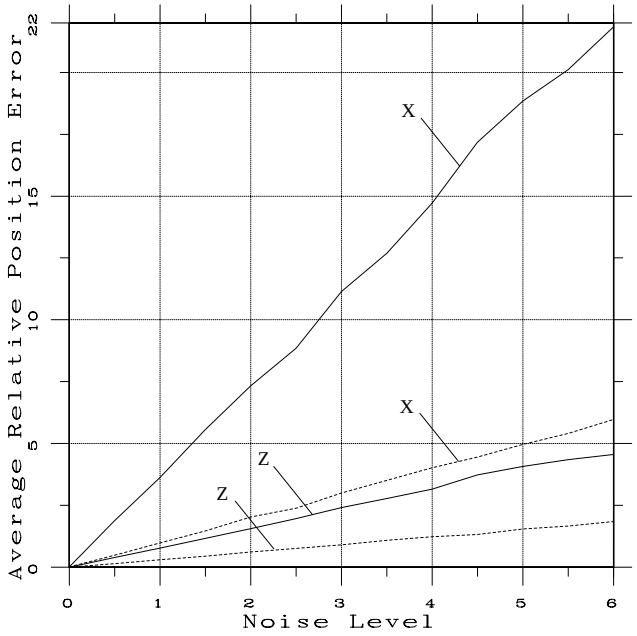


b) Erreur de translation.

FIG. 2.16 - *Erreurs d'orientation et de translation en fonction d'un bruit uniforme affectant les axes de rotation. Les courbes continues (—) correspondent à la méthode linéaire, les courbes interrompues (- - -) à la méthode non linéaire.*

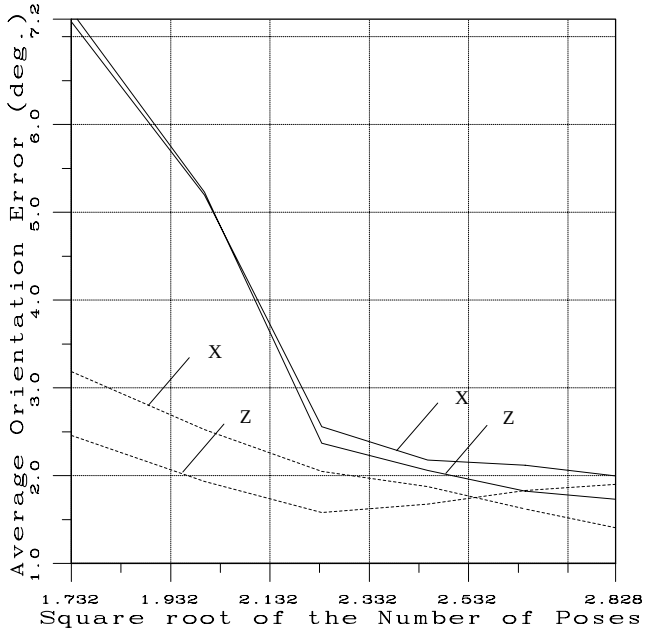


a) Erreur d'orientation.

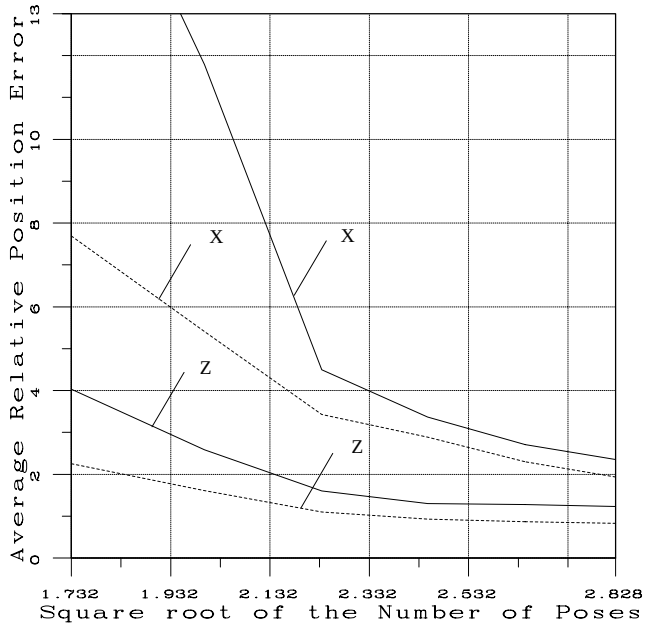


b) Erreur de translation.

FIG. 2.17 - *Erreurs d'orientation et de translation en fonction d'un bruit gaussien affectant les axes de rotation. Les courbes continues (—) correspondent à la méthode linéaire, les courbes interrompues (- - -) à la méthode non linéaire.*

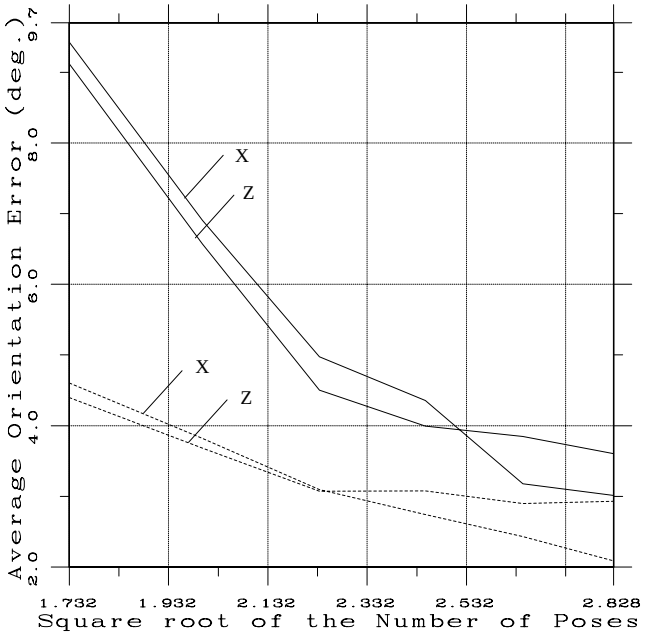


a) Erreur d'orientation.

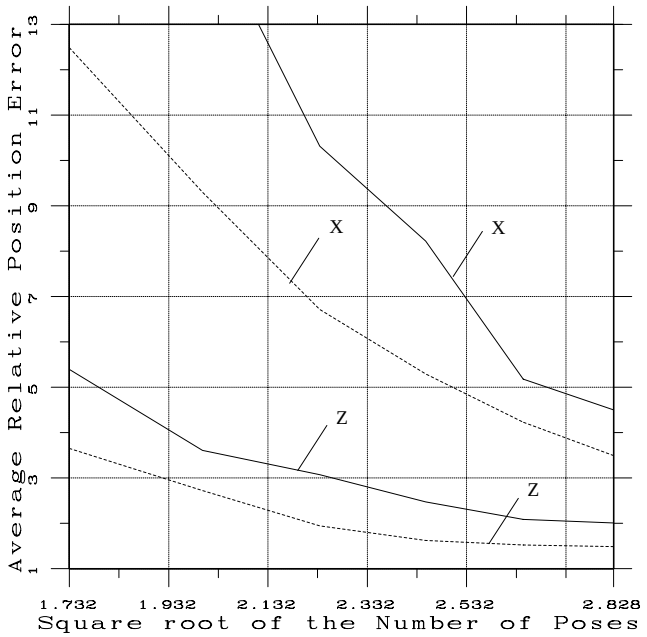


b) Erreur de translation.

FIG. 2.18 - *Erreurs d'orientation et de translation en fonction du nombre de positions. Le bruit gaussien, qui affecte les positions du robot, est de 6% pour les axes de rotations et de 2% pour les vecteurs de translation. Les courbes continues (—) correspondent à la méthode linéaire, les courbes interrompues (- - -) à la méthode non linéaire.*



a) Erreur d'orientation.



b) Erreur de translation.

FIG. 2.19 - *Erreurs d'orientation et de translation en fonction du nombre de positions. Le bruit gaussien, qui affecte les positions de la caméra et du roobot, est de 6% pour les axes de rotations et de 2% pour les vecteurs de translation. Les courbes continues (—) correspondent à la méthode linéaire, les courbes interrompues (- - -) à la méthode non linéaire.*

2.11 Conclusion

Dans ce chapitre nous avons proposé une nouvelle formulation du problème du calibrage caméra-pince. Cette formulation est équivalente à la résolution d'un système d'équations matricielles de la forme $(MY = M'YB)$. La formulation classique de ce problème consistait à résoudre un système d'équations matricielles de la forme $(AX = XB)$. L'avantage de cette nouvelle formulation par rapport à la formulation classique est qu'elle n'a pas besoin d'un calcul explicite des déplacements de la caméra, seules les matrices de projection perspective sont utilisées. Du coup, les erreurs éventuelles associées à la décomposition des matrices de projection en paramètres intrinsèques et extrinsèques seront éliminées. Les deux formulations permettent d'exprimer les lignes de vue dans le repère de la pince.

Nous avons démontré que ces deux formulations ont la même structure mathématique en les inconnues (une rotation et une translation). Nous avons également présenté deux méthodes de résolution algébriques. La première a été proposé par Tsai & Lenz [TL89]. La deuxième a été proposée par Faugeras & Hebert [FH86]. Nous avons proposé une autre méthode numérique pour estimer simultanément la rotation et la translation.

Ces trois méthodes de résolution ont été comparées sur des données synthétiques et sur des données réelles. Nous avons montré que la méthode non linéaire fournit les résultats les plus stables. Il faut noter que les solutions algébriques peuvent être utilisées comme solution initiale du processus non linéaire. Ainsi, on se passe du problème du minimum local que l'on peut rencontrer en initialisant la méthode non linéaire avec une solution aléatoire. De plus, la convergence deviendrait plus rapide. Nous avons également vu que la nouvelle formulation $(MY = M'YB)$ fournit des résultats plus précis que la formulation classique $(AX = XB)$.

La dernière partie de ce chapitre a été consacrée au calibrage robot-environnement $(AX = ZB)$. Ce calibrage est plus général que le calibrage caméra-pince $(AX = XB)$. Nous avons présenté la méthode linéaire proposé par Zhuang [ZRS94]. Cette méthode estime les rotations et les translations en deux étapes successives. Nous avons proposé une alternative qui consiste à estimer simultanément les rotations et les translations. Nous avons également effectué une comparaison entre ces deux méthodes. Nous avons montré que la méthode non linéaire est plus précise que la méthode linéaire.

Chapitre 3

Autocalibrage Capteur/Robot

Nous nous posons, dans ce chapitre, le problème suivant : étant donné une reconstruction projective d'une scène inconnue, obtenue à partir d'une caméra non calibrée et montée sur un robot, comment calculer la transformation qui, appliquée à cette reconstruction, donnerait une reconstruction euclidienne. Plus précisément, nous proposons une méthode qui permet d'obtenir (i) la structure euclidienne d'une scène à partir d'une reconstruction projective de cette scène, (ii) les paramètres intrinsèques de la caméra utilisée et (iii) la relation géométrique caméra-pince. Nous montrerons que ce problème est équivalent à la résolution d'un système d'équations matricielles. Nous présenterons les conditions nécessaires que doivent vérifier les déplacements du robot pour que la solution soit unique. Nous montrerons qu'il s'agit d'une généralisation du problème de calibrage caméra-pince traité dans le deuxième chapitre.

3.1 Introduction

L'une des tâches majeures de la vision par ordinateur est la reconstruction euclidienne de la structure d'une scène à l'aide d'une caméra mobile. Deux approches sont envisageables : la première utilise une caméra calibrée, la seconde utilise une caméra non calibrée pour constituer une reconstruction projective qui est transformée en une structure euclidienne. Avec la première approche, la qualité de le reconstruction dépend de la qualité du calibrage de la caméra et de l'hypothèse de l'invariance des paramètres de calibrage. Cependant, même si l'on dispose des techniques numériques qui permettent d'obtenir un calibrage de bonne qualité, le problème de la dérive temporelle des paramètres subsiste. Avec la seconde approche, la notion d'autocalibrage est introduite : l'idée est d'utiliser les points observés eux-mêmes pour constituer le repère dans lequel on positionne la scène. Le calibrage de la caméra est alors effectué simultanément avec la reconstruction, et on évite ainsi les problèmes liés à la dérive des paramètres. Cependant, si le problème de la reconstruction projective est résolu il n'en est pas de même pour le problème du passage du projectif à l'euclidien. Des implantations numériques peuvent être trouvées à partir des travaux théoriques de Maybank & Faugeras [MF92]. Ces implantations fournissent des résultats qui sont partiellement satisfaisants [FLM92].

Dans la suite, nous supposons que la caméra est rigidement liée à la pince du robot.

Nous ne connaissons ni les paramètres de la caméra ni la relation caméra-pince. Dans le passé, le problème du calibrage caméra-pince a été associé au problème d'estimation des paramètres extrinsèques d'une caméra [TL89] [HD95]. Dans [HDBM94], les auteurs ont proposé une méthode de calibrage caméra-pince en conjonction avec la reconstruction euclidienne. Ici, nous proposons une méthode de calibrage basée sur une reconstruction projective. Cette méthode estime simultanément :

1. Le passage d'une structure projective à une structure euclidienne.

2. Les paramètres intrinsèques de la caméra.

3. La transformation caméra-pince.

3.2 Préliminaires

3.2.1 Calibrage des caméras

Nous considérons le problème du calibrage de la caméra et supposons que la caméra est décrite par le modèle sténopé. Dans le domaine de la vision par ordinateur, la précision de ce modèle est généralement suffisante. De plus, la transformation effectuée est alors une projection perspective pure, qui appartient aux transformations projectives de $I\!P^3$ dans $I\!P^2$ ($I\!P^k$ dénote l'espace projectif de dimension k). Le calibrage d'une telle caméra consiste à calculer la relation qui lie les coordonnées tridimensionnelles d'un point aux coordonnées bidimensionnelles de son image. Cette relation est donnée par une matrice M de dimension 3×4. Nous avons :

$$p = \lambda\, MP \tag{3.1}$$

où $p = (u\ v\ w)^T$ est un point image et $P = (x\ y\ z\ s)^T$ un point 3D dont les coordonnées euclidiennes sont x/s, y/s et z/s. λ est un facteur d'échelle.

La matrice M peut être déterminée par un processus linéaire [FT87] [Fau93] pourvu qu'on ait, au moins, 6 correspondances point 3D/point 2D. Cette matrice a la forme suivante :

$$M = \underbrace{\begin{bmatrix} \alpha_u & -\alpha_u \cot\theta & u_c \\ 0 & \alpha_v \sin\theta & v_c \\ 0 & 0 & 1 \end{bmatrix}}_{C} \begin{bmatrix} \underbrace{R}_{3\times3} & \underbrace{\mathrm{t}}_{3\times1} \end{bmatrix} \tag{3.2}$$

La matrice C regroupe les 5 paramètres intrinsèques de la caméra. La deuxième matrice est la transformation rigide entre le repère de la caméra et le repère de la scène (6 paramètres indépendants). Une fois M estimée, l'extraction des paramètres intrinsèques et extrinsèques sera possible. Notons qu'il existe de nombreuses méthodes qui permettent d'obtenir uniquement les paramètres intrinsèques [CLA93] [BMZ92].

3.2.2 Reconstruction 3D

La reconstruction 3D vise à déterminer la structure 3D de la scène observée. Les étapes suivantes sont nécessaires pour obtenir une telle reconstruction :

1. Une séquence d'images (au moins deux) d'une scène rigide est prise à l'aide d'une caméra.

2. Des points d'intérêt sont extraits et mis en correspondance dans la séquence.

3. Cinq points non 4 à 4 coplanaires sont choisis comme base pour un repère relatif et se voient donc affecter des coordonnées. Ces dernières peuvent être euclidiennes, affines ou projectives.

4. L'ensemble des observations (coordonnées images) est traduit en un système d'équations non linéaires dont la résolution donne directement la structure tridimensionnelle de la scène (euclidienne, affine ou projective) et les matrices de projection.

Considérons une caméra mobile observant une scène rigide comprenant k points. Notons les points P_j, $j \in \{1, \ldots, k\}$. Considérons également n positions de cette caméra. Les images correspondant à ces positions sont indexées par i, $i \in \{1, \ldots, n\}$. Par conséquent, l'équation (3.2) se réécrit :

$$p_{ij} = \lambda_{ij}\ M_i P_j, \quad i = 1, \ldots, n, \ \ j = 1, \ldots, k \tag{3.3}$$

Le système (3.3) tel qu'il est posé, sans contrainte additionnelle, n'admet pas de solution unique. En effet, si M_i et P_j sont solutions du système, il en est de même pour $M_i W$ et $W^{-1} P_j$ où W est une matrice 4×4 inversible quelconque. On a :

$$p_{ij} = \lambda_{ij}\ M_i W W^{-1} P_j$$

Par conséquent, pour assurer l'unicité de la solution, une base quelconque peut être choisie dans l'espace projectif $\mathbb{P}^3$, ce qui revient à choisir 5 points indépendants (non quatre à quatre coplanaires) pour définir une telle base. Il est à noter que le type de reconstruction dépend des coordonnées des 5 points utilisés comme base. Par exemple quand ces coordonnées sont euclidiennes, la reconstruction le sera aussi.

Afin d'éviter toute confusion, S_j dénote un point de l'espace projectif $\mathbb{P}^3$, P_j un point de l'espace euclidien 3D. N_i dénote une matrice qui projette les points de l'espace projectif 3D dans l'espace projectif de l'image, M_i la matrice qui projette les points de l'espace euclidien 3D dans l'espace projectif de l'image. L'équation (3.3) peut être écrite sous une autre forme plus utilisée dans la littérature :

$$\begin{aligned}
\frac{u_{ij}}{w_{ij}} &= \frac{n_{11}^{(i)} x_j + n_{12}^{(i)} y_j + n_{13}^{(i)} z_j + n_{14}^{(i)} s_j}{n_{31}^{(i)} x_j + n_{32}^{(i)} y_j + n_{33}^{(i)} z_j + n_{34}^{(i)} s_j} \\
\frac{v_{ij}}{w_{ij}} &= \frac{n_{21}^{(i)} x_j + n_{22}^{(i)} y_j + n_{23}^{(i)} z_j + n_{24}^{(i)} s_j}{n_{31}^{(i)} x_j + n_{32}^{(i)} y_j + n_{33}^{(i)} z_j + n_{34}^{(i)} s_j}
\end{aligned}$$

Comme nous avons k points et n images, ceci nous donne un total de $2 \times k \times n$ équations. Nous avons $11 \times n$ inconnues pour les matrices de projection et $3 \times k$ pour les points de l'espace. Dès que le nombre d'équations est plus grand ou égal que celui des inconnues, le système (3.3) peut être résolu en théorie.

Le problème décrit par les équations ci-dessus peut être formulé comme un problème d'estimation de paramètres [Bou94]. Nous avons à estimer un ensemble des paramètres, ici les N_i et les S_i, à partir des mesures non exactes, ici les coordonnées images. L'estimation du maximum de vraisemblance des paramètres est obtenue en minimisant la fonction d'erreur suivante :

$$f(N_1, \ldots N_i, \ldots N_n, S_1, \ldots S_j, \ldots S_k) =$$
$$\sum_{ij} \left(\left(\frac{u_{ij}}{w_{ij}} - \frac{n_{11}^{(i)} x_j + n_{12}^{(i)} y_j + n_{13}^{(i)} z_j + n_{14}^{(i)} s_j}{n_{31}^{(i)} x_j + n_{32}^{(i)} y_j + n_{33}^{(i)} z_j + n_{34}^{(i)} s_j} \right)^2 + \left(\frac{v_{ij}}{w_{ij}} - \frac{n_{21}^{(i)} x_j + n_{22}^{(i)} y_j + n_{23}^{(i)} z_j + n_{24}^{(i)} s_j}{n_{31}^{(i)} x_j + n_{32}^{(i)} y_j + n_{33}^{(i)} z_j + n_{34}^{(i)} s_j} \right)^2 \right)$$

De nombreuses méthodes existent pour résoudre ce type de problème qui n'est autre que la résolution d'un système d'équations non linéaires au sens des moindres carrés. L'algorithme de Levenberg-Marquardt pourrait être utilisé pour ce cas [BMV93] [Bou94] [MBB94] [Har94b] [Har94a] [Fau92].

3.3 Formulation du problème

Considérons une caméra montée sur l'organe terminal d'un robot (pince). La pince est dotée d'un repère euclidien dont l'orientation et la position par rapport à la base du robot sont données par le modèle cinématique direct du robot. Supposons que la liaison caméra-pince est une liaison rigide. Soit $X = (R\ \mathbf{t})$ la transformation rigide entre le repère caméra et le repère pince. Les paramètres de X sont les paramètres associés au problème du calibrage caméra-pince. Si nous combinons X avec les paramètres intrinsèques de la caméra données par la matrice C, nous obtenons la matrice projective suivante :

$$M = CX \tag{3.4}$$

Il est à noter que la matrice M regroupe le problème du calibrage de la caméra (paramètres intrinsèques) et le problème du calibrage caméra-pince. Cette matrice ne peut pas être estimée directement. Dans la suite, nous montrerons comment on peut l'estimer d'une manière indirecte.

Supposons maintenant que le système caméra-pince a réalisé une séquence de déplacements rigides et que la matrice C n'a pas varié pendant ces déplacements. Comme la caméra observe une scène 3D, on en obtient une séquence d'images. Nous pouvons y établir la mise en correspondance des indices 2D. Ainsi, nous pouvons effectuer une reconstruction projective de la scène 3D grâce à la méthode présentée dans la section précédente. Nous associons un repère projectif à la scène 3D. Soit S_j un point de cette scène exprimé dans ce repère projectif et N_i la matrice de projection entre ce repère et le repère associé à la $i^{ème}$ image (voir Figure 3.1).

Soit Y_i une matrice 4×4 inversible associée à la transformation entre le repère projectif de la scène et le repère euclidien de la pince (dans son $i^{ème}$ position). Par conséquent, la

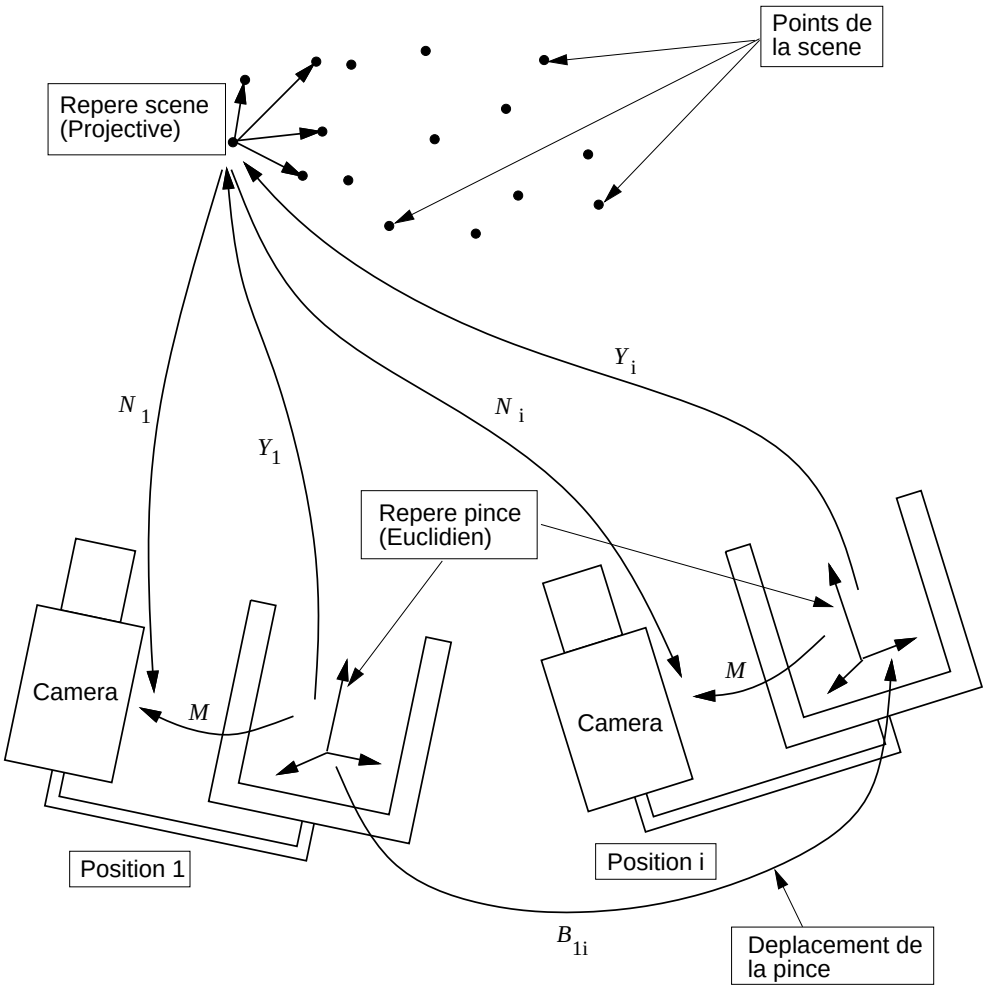


FIG. 3.1 - *Une caméra montée sur la pince d'un robot. Si la pince effectue un déplacement rigide, il en est de même pour la caméra. La scène observée par cette caméra est matérialisée par un ensemble de points. Les seuls paramètres connus sont les coordonnées images et les déplacements de la pince.*

matrice M peut être écrite comme le produit de deux transformations projectives :

$$\begin{array}{rcl} M & \cong & N_1 Y_1 \\ & \cong & \dots \\ & \cong & N_i Y_i \\ & \cong & \dots \\ & \cong & N_n Y_n \end{array} \tag{3.5}$$

où $\cong$ indique que ces équations sont définies à un facteur multiplicatif près.

Soit B_{1i} le déplacement effectué par la pince du robot entre la première position -1- et une autre position quelconque -i. A partir de la Figure 3.1, on peut écrire :

$$Y_i \cong Y_1 B_{1i}^{-1}$$

En remplaçant l'expression de Y_i dans l'équation (3.5), nous obtenons un ensemble de $n-1$ équations matricielles :

$$\left\{ \begin{array}{rcl} N_1 Y_1 B_{12} & = & \lambda_2 N_2 Y_1 \\ & \vdots & \\ N_1 Y_1 B_{1n} & = & \lambda_n N_n Y_1 \end{array} \right. \tag{3.6}$$

Dans ce système, les inconnues sont les scalaires $\lambda_2, \dots, \lambda_n$ et la matrice Y_1 de dimension 4×4 qui représente la relation entre la structure projective de la scène et la structure euclidienne de cette scène exprimée dans le repère de la pince. En d'autres termes, la matrice Y_1 est une transformation projective qui fait passer d'une reconstruction projective à une reconstruction euclidienne. Si nous sommes capables d'estimer la matrice Y_1, alors le problème du passage de la reconstruction projective à la reconstruction euclidienne sera résolu. La matrice Y_1 a la forme suivante :

$$Y_1 = \left[\begin{array}{cccc} y_{11} & y_{12} & y_{13} & y_{14} \\ y_{21} & y_{22} & y_{23} & y_{24} \\ y_{31} & y_{32} & y_{33} & y_{34} \\ y_{41} & y_{42} & y_{43} & y_{44} \end{array} \right]$$

Chaque équation matricielle du système (3.6) fournit 12 contraintes. Les inconnues $\lambda_2, \dots, \lambda_n$ peuvent être éliminées en divisant, par exemple, les 11 premières contraintes par la dernière. En considérant la première équation matricielle et en désignant par V la matrice $N_1 Y_1 B_{12}$ et par T la matrice $N_2 Y_1$, nous obtenons les 12 contraintes suivantes :

$$\left\{ \begin{array}{rcl} v_{11} & = & \lambda_2 \, t_{11} \\ & \vdots & \\ v_{34} & = & \lambda_2 \, t_{34} \end{array} \right.$$

avec :

$$v_{11} = n_{11}^{(1)} \left(b_{11}y_{11} + b_{21}y_{12} + b_{31}y_{13}\right) + n_{12}^{(1)} \left(b_{11}y_{21} + b_{21}y_{22} + b_{31}y_{23}\right) +$$

$$n_{13}^{(1)}\,(b_{11}y_{31}+b_{21}y_{32}+b_{31}y_{33})+n_{14}^{(1)}\,(b_{11}y_{41}+b_{21}y_{42}+b_{31}y_{43})$$

$$\begin{aligned} v_{34} &= n_{31}^{(1)}\,(b_{14}y_{11}+b_{24}y_{12}+b_{34}y_{13}+y_{14})+n_{32}^{(1)}\,(b_{14}y_{21}+b_{24}y_{22}+b_{34}y_{23}+y_{24})+ \\ &\quad n_{33}^{(1)}\,(b_{14}y_{31}+b_{24}y_{32}+b_{34}y_{33}+y_{34})+n_{34}^{(1)}\,(b_{14}y_{41}+b_{24}y_{42}+b_{34}y_{43}+y_{44}) \end{aligned}$$

$$t_{11} = n_{11}^{(2)}y_{11}+n_{12}^{(2)}y_{21}+n_{13}^{(2)}y_{31}+n_{14}^{(2)}y_{41}$$

$$t_{34} = n_{31}^{(2)}y_{14}+n_{32}^{(2)}y_{24}+n_{33}^{(2)}y_{34}+n_{34}^{(2)}y_{44}$$

Après l'élimination de λ_2, la première contrainte devient :

$$v_{11}\,t_{34}-v_{34}\,t_{11}=0$$

Ainsi, le système (3.6) fournit $11\times(n-1)$ contraintes quadratiques en 15 inconnues (la matrice Y_1 est définie à un facteur multiplicatif près). Il en résulte que trois positions du robot, au moins, sont nécessaires pour pouvoir résoudre le système (3.6). Supposons qu'on a trouvé la solution pour Y_1. Nous obtenons immédiatement :

- *La structure euclidienne*, en multipliant les coordonnées projectives par l'inverse de Y_1,
$$\forall j \quad P_j = Y_1^{-1}S_j$$
- *Le calibrage de la caméra*, en estimant la matrice M par l'une des égalités du système (3.5). Une alternative peut être utilisée : la matrice M pourrait être estimée à partir des coordonnées euclidiennes des points de la scène et de leur projection dans la première image (la scène est reconstruite dans le repère de la pince correspondant à la première position).
- *Le calibrage caméra-pince*, en estimant les paramètres extrinsèques (rotation et translation) associés à la matrice M.

Il est intéressant de noter que cette méthode n'est autre que la généralisation de la formulation du calibrage caméra-pince [TL89] [HD95] (voir chapitre 2). En effet, si nous disposons d'une reconstruction euclidienne de la scène, alors la matrice Y_1 décrit un déplacement rigide et les matrices N_i doivent être remplacées par les matrices M_i. Dans ce cas, le système (3.6) se transforme en le système (2.11) donnant la formulation $MY = M'YB$ (voir section 2.2.3).

3.4 Résolution du problème

Comme nous l'avons mentionné dans la section précédente, la solution du problème consiste à résoudre un système d'équations matricielles (le système (3.6)). Une telle équation matricielle s'écrit comme suit :

$$N_1 = \lambda_i N_i Y_1 B_{1i}^{-1} Y_1^{-1}$$

Soit U une matrice 4×4 inversible qui appartient au sous-groupe des matrices du groupe projectif qui commutent avec B_{1i}^{-1} ou B_{1i}. Désignons par $\mathcal{C}(B_{1i})$ ce sous-groupe.

Nous avons :

$$\begin{aligned} N_1 &= \lambda_i N_i Y_1 B_{1i}^{-1} Y_1^{-1} \\ &= \lambda_i N_i Y_1\, U\, U^{-1}\, B_{1i}^{-1} Y_1^{-1} \\ &= \lambda_i N_i\, (Y_1 U)\, B_{1i}^{-1}\, (Y_1 U)^{-1} \end{aligned}$$

Par conséquent, si Y_1 est une solution, la matrice $Y_1\,U$ l'est aussi et la matrice Y_1 n'est pas unique. Afin que la solution soit unique, on doit choisir des déplacements du robot de telle sorte que le seul choix possible pour la matrice U soit la matrice identité :

$$\bigcap_{i>1} \mathcal{C}(B_{1i}) = \{I\}$$

3.4.1 Unicité de la solution

Dans cette section, nous allons établir la condition suffisante permettant d'obtenir le résultat ci-dessus. Nous montrons que cette condition est satisfaite si l'on a, au moins, deux déplacements dont les deux axes de rotation sont différents et qu'un déplacement, au moins, doit avoir une translation non orthogonale à son axe de rotation.

En effet, les deux matrices U et B_{1i} peuvent être écrites sous la forme suivante :

$$U = \begin{bmatrix} W & \mathbf{u} \\ \mathbf{v}^T & w \end{bmatrix} \qquad B_{1i} = \begin{bmatrix} R_i & \mathbf{t}_i \\ 0\,0\,0 & 1 \end{bmatrix}$$

Nous cherchons une matrice U qui vérifie la condition suivante :

$$B_{1i} U \cong U B_{1i}$$

Afin d'éliminer le facteur multiplicatif de cette égalité, nous fixons le déterminant de U de la façon suivante :

$$\det(U) = 1$$

En développant le produit matriciel, nous obtenons :

$$W\,R_i = R_i\,W + \mathbf{t}_i\,\mathbf{v}^T \tag{3.7}$$

$$W\,\mathbf{t}_i + \mathbf{u} = R_i\,\mathbf{u} + w\,\mathbf{t}_i \tag{3.8}$$

$$\mathbf{v}^T R_i = \mathbf{v}^T \tag{3.9}$$

$$\mathbf{v}^T \mathbf{t}_i = 0 \tag{3.10}$$

L'équation (3.9) montre que le vecteur $\mathbf{v}$ doit avoir la même direction que l'axe de rotation associé à la matrice R_i. Ainsi, si la pince du robot exécute deux déplacements dont les deux axes de rotation sont différents, nous devons avoir :

$$\mathbf{v} = 0$$

Les équations (3.7) et (3.8) deviennent donc :

$$W\,R_i = R_i\,W \tag{3.11}$$
$$(W - w\,I)\,\mathbf{t}_i = (R_i - I)\,\mathbf{u} \tag{3.12}$$

Soient $\mathbf{n}_1$, $\mathbf{n}_2$ et $\mathbf{n}_3$ les vecteurs propres de la matrice R_i. Il est bien connu que cette matrice possède les trois valeurs propres suivantes : $\mu_1 = 1$, $\mu_2 = e^{i\theta}$ et $\mu_3 = e^{-i\theta}$. A partir de l'équation (3.11), nous pouvons constater que les deux matrices W et R_i ont les mêmes vecteurs propres mais elles peuvent avoir des valeurs propres différentes. En effet, nous avons :

$$R_i\mathbf{n}_j = \mu_j\mathbf{n}_j$$

et en utilisant l'égalité (3.11), nous obtenons :

$$WR_i\mathbf{n}_j = W\mu_j\mathbf{n}_j = \mu_j W\mathbf{n}_j = R_i W\mathbf{n}_j$$

Par conséquent, $W\mathbf{n}_j = \nu_j\mathbf{n}_j$ avec $j = 1, 2, 3$. Notons que les valeurs propres de la matrice W, ν_1, ν_2 et ν_3 ne peuvent pas être nulles puisque W n'est pas singulière.

Considérons maintenant un autre déplacement de la pince du robot. R_k est la matrice de rotation associée à ce déplacement. Les vecteurs propres de cette matrice sont notés $\mathbf{m}_1$, $\mathbf{m}_2$ et $\mathbf{m}_3$. Soit $\mathbf{m}_1$ le vecteur propre donnant la direction de l'axe de rotation de R_k. Supposons que ce vecteur a une direction différente de celle de $\mathbf{n}_1$ et qu'il n'appartient pas au plan défini par les deux vecteurs $\mathbf{n}_2$ et $\mathbf{n}_3$. Dans ce cas, le vecteur $\mathbf{m}_1$ peut être écrit comme une combinaison linéaire des vecteurs propres de R_i :

$$\mathbf{m}_1 = \sum_{j=1}^{3} \lambda_j\mathbf{n}_j \tag{3.13}$$

Nous obtenons :

$$W\,\mathbf{m}_1 = W(\sum_{j=1}^{3} \lambda_j\mathbf{n}_j) = \sum_{j=1}^{3} \lambda_j\nu_j\mathbf{n}_j \tag{3.14}$$

Mais W et R_k ont les mêmes vecteurs propres, il s'ensuit que :

$$W\,\mathbf{m}_1 = \nu_1\mathbf{m}_1 = \nu_1 \sum_{j=1}^{3} \lambda_j\mathbf{n}_j = \sum_{j=1}^{3} \nu_1\lambda_j\mathbf{n}_j \tag{3.15}$$

En identifiant les deux équations (3.15) et (3.14), nous aurons :

$$\nu_1 = \nu_2 = \nu_3 = \nu$$

Comme la matrice W a trois valeurs propres identiques, elle est nécessairement de la forme :

$$W = \nu I$$

L'équation (3.12) devient donc :

$$(\nu - w)\,\mathbf{t}_i = (R_i - I)\,\mathbf{u}$$

Dans cette expression, $(R_i - I)\,\mathbf{u}$ est un vecteur orthogonal à l'axe de rotation de R_i – le vecteur propre $\mathbf{n}_1$. En supposant que la translation $\mathbf{t}_i$ n'appartient pas au plan perpendiculaire à l'axe de rotation et en éliminant la solution triviale ($R_i = I$ et $\mathbf{t}_i = 0$), nous obtenons :

$$\mathbf{u} = 0$$

$$\nu = w$$

Il en résulte que la matrice U a la forme suivante :

$$U = \nu\, I$$

Comme le déterminant de U est égal à 1, nous obtenons la solution unique pour U :

$$U = I$$

En résumé, la condition suffisante de l'unicité de la solution du système (3.6) est la suivante : *parmi tous les déplacements du robot, (i) au moins deux déplacements doivent avoir des axes de rotation différents et (ii) au moins un déplacement doit avoir son vecteur de translation non orthogonal à son axe de rotation.*

3.4.2 Résolution : aspect pratique

La méthode proposée se résume de la façon suivante :

1. On effectue un certain nombre de déplacements du robot. Ces déplacements doivent satisfaire la condition suffisante établie dans la section précédente. On enregistre ces déplacements qui sont obtenus grâce au modèle cinématique direct du robot.

2. On extrait les points d'intérêt dans la séquence d'images obtenue et on y établit leur mise en correspondance.

3. On effectue une reconstruction projective de la scène observée.

4. On résout le système (3.6) en deux étapes :

   - *Première étape* : On utilise des valeurs approximatives pour les paramètres intrinsèques de la caméra et pour la transformation caméra-pince. Ces valeurs approximatives vont donner une estimation approximative de la matrice M qui est donnée par l'équation (3.4). Ensuite, en remplaçant l'équation (3.5) dans le système (3.6), nous transformons ce dernier en un système linéaire en les coefficients de Y_1.
   - *Deuxième étape* : On résout le système (3.6) en utilisant une technique de minimisation non linéaire. La solution linéaire obtenue à la première étape sera utilisée pour initialiser la technique non linéaire.

5. On effectue le passage de la reconstruction projective de la scène à la reconstruction euclidienne.

6. On détermine la matrice M et on en extrait les paramètres intrinsèques (calibrage caméra) et extrinsèques (calibrage caméra-pince).

3.4.3 Expérimentations

La méthode décrite ci-dessus est testée sur des données réelles. Ainsi, nous reprenons les données associés à la deuxième expérience du chapitre 2. Dans cette expérience, nous avons 7 positions du système caméra-pince. La scène est constituée d'une mire (460 points). Par conséquent, cet ensemble de données va permettre la comparaison de deux méthodes :

1. La méthode hors ligne qui consiste à (i) calibrer la caméra à chaque position (les paramètres intrinsèques et extrinsèques) et à (ii) déterminer la transformation caméra-pince à partir des déplacements mesurés (voir chapitre 2).

2. La méthode d'autocalibrage telle qu'elle est décrite dans ce chapitre.

Les deux méthodes estiment une matrice de projection M (de taille 3×4). A partir de cette matrice on doit extraire les paramètres intrinsèques et extrinsèques. Une telle décomposition pourrait être effectuée par une variante de la décomposition matricielle QR, dénotée RQ [Har94b]. En effet, la matrice M peut être écrite sous la forme suivante :

$$M = \left[\underbrace{CR}_{3\times3} \quad \underbrace{C\,\mathbf{t}}_{3\times1} \right] \tag{3.16}$$

Les matrices C et R peuvent être obtenues en décomposant la sous-matrice 3×3 de M en une matrice triangulaire haute et une matrice orthogonale. Les résultats du calibrage des paramètres intrinsèques, obtenus par les deux méthodes, sont donnés par le tableau 3.1. Pour les deux méthodes nous avons utilisés la décomposition RQ. Les paramètres intrinsèques estimés par autocalibrage ont le même ordre de grandeur que les paramètres estimés par la méthode classique (les 7 premières lignes). Les résultats d'autocalibrage reportés dans le tableau 3.1 peuvent être comparés avec les résultats d'autocalibrage reportés par Faugeras et al. [FLM92] et par Hartley [Har94b]. Tous ces auteurs ont remarqué les grandes variations de la position du centre optique (les coordonnées u_c et v_c) aussi bien que la sensibilité du calibrage au bruit.

Les paramètres extrinsèques associés à la matrice M de la méthode d'autocalibrage fournissent la transformation caméra-pince. La comparaison entre la solution obtenue par la méthode hors ligne ($AX = XB$) et celle de la méthode d'autocalibrage donne :

Calibrage hors ligne ***($AX = XB$)*** **:**

$$\left[R \;\; \mathbf{t}_{(m)}\right] = \begin{bmatrix} -0.9964 & -0.0752 & 0.0398 & -0.0132 \\ 0.0681 & -0.9852 & -0.1569 & 0.0642 \\ 0.0510 & -0.1536 & 0.9868 & 0.0392 \end{bmatrix}$$

Autocalibrage **:**

$$\left[R \;\; \mathbf{t}_{(m)}\right] = \begin{bmatrix} -0.9974 & -0.0574 & 0.0438 & -0.0199 \\ 0.0524 & -0.9927 & -0.1081 & 0.0283 \\ 0.0497 & -0.1055 & 0.9932 & 0.0831 \end{bmatrix}$$

	α_u	α_v	α_u/α_v	$-\alpha_u \cot\theta$	u_c	v_c
Position 1	-2084	1417	-1.4707	5	256	221
Position 2	-2055	1397	-1.4710	-3	257	223
Position 3	-2081	1415	-1.4706	0.6	265	230
Position 4	-2083	1416	-1.4710	0.45	261	213
Position 5	-2094	1422	-1.4726	3.3	270	222
Position 6	-2054	1426	-1.4404	2.1	260	217
Position 7	-2051	1420	-1.4444	2	251	230
Initialisation	-1700	1000	-1.7	0	256	256
Autocalibrage	-1879	1310	-1.4343	46	278	199

TAB. 3.1 - *Ce tableau résume les résultats du calibrage de la caméra. Les résultats obtenus par la méthode hors ligne sont donnés par les 7 premières lignes. Les résultats obtenus par la méthode d'autocalibrage sont donnés par la dernière ligne*

Nous constatons que les deux méthodes fournissent le même résultat pour la matrice de rotation. En revanche, les deux méthodes fournissent deux vecteurs de translation qui sont différents (en particulier, les deux composantes $\mathbf{t}_y$ et $\mathbf{t}_z$). Cette différence est de l'ordre de 4cm. Ce comportement pourrait être expliqué par le fait que les déplacements du robot sont moins précis en translation qu'en rotation.

3.5 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle approche permettant le passage d'une reconstruction projective à une reconstruction euclidienne. Nous avons montré que le calcul de ce passage permet également l'autocalibrage de la caméra et du système caméra-pince. Ce passage est décrit par une matrice inversible de taille 4×4 qui représente une collinéation de l'espace. Nous avons montré comment on peut utiliser les déplacements rigides de la pince du robot pour les traduire en contraintes sur les paramètres de la collinéation cherchée ; le problème devient alors un problème d'optimisation non linéaire résolu au sens des moindres carrés. Nous avons également établi la condition nécessaire pour que la solution soit unique.

Les résultats expérimentaux valident l'approche proposée. La manque de précision sur l'estimation de la translation caméra-pince peut être levée par l'utilisation d'un robot assez précis.

Dans un calibrage classique, les informations euclidiennes sont données non seulement par les déplacements du robot mais également par le modèle 3D d'une mire observée. Dans l'approche d'autocalibrage, ces informations sont uniquement fournies par les déplacements du robot. Ces déplacements sont moins précis que les mesures 3D *a priori* d'une mire de calibrage.

Chapitre 4

Localisation Caméra/Objet

Dans ce chapitre, nous nous focalisons sur le problème de la détermination de la position et l'orientation d'un objet par rapport à une caméra à partir de mesures images. Ce problème est équivalent au problème de la détermination des paramètres extrinsèques d'une caméra lorsque les paramètres intrinsèques sont connus. Nous présentons la méthode linéaire itérative, proposée par DeMenthon [DD95], qui utilise la projection perspective faible. Ensuite, nous allons montrer comment on peut utiliser cette méthode dans le cas des correspondances de droites. Ensuite, nous proposons une nouvelle méthode linéaire itérative basée sur la projection para-perspective. Nous allons montrer que les propriétés de convergence associées à cette méthode sont meilleures que celles de la méthode de DeMenthon. Ensuite, nous proposons une méthode non linéaire permettant le calcul de la pose. Nous proposons également une méthode simple pour tenir compte des contraintes d'orthogonalité de la matrice de rotation lorsque la scène est non coplanaire (pour les deux méthodes linéaires itératives). Enfin, les trois méthodes de localisation sont comparées dans les mêmes conditions : positions de l'objet, bruit d'image et bruit sur les paramètres de la caméra. Enfin, nous présentons quelques résultats expérimentaux qui visent à valider les méthodes de localisation proposées.

4.1 Introduction

En robotique et en vision, la détermination des position et orientation relatives entre une caméra et un objet est un problème central. Elle a suscité de nombreux travaux de recherche, du fait que les applications sont nombreuses et importantes, pour le calibrage, l'asservissement visuel, l'analyse d'images aériennes, la cartographie, la poursuite et la reconnaissance d'objets. Ce problème peut être formulé de la manière suivante : étant donné un ensemble de primitives 3D décrites dans un référentiel objet et leurs projections 2D décrites dans un référentiel caméra et connaissant les paramètres de la caméra, il faut alors déterminer la transformation rigide (rotation et translation) entre le référentiel objet et le référentiel caméra. La connaissance de cette transformation permet de relier les mesures images avec la structure de la scène observée. Notons qu'il existe des méthodes de localisation qui ne nécessitent pas l'extraction explicite des données 2D [Rob94], ni la mise en correspondance [SHK93]. Il est à noter que le problème de calibrage d'une caméra

consiste à estimer deux sortes de paramètres :

1. les paramètres intrinsèques qui dépendent uniquement de la structure physique de la caméra et du convertisseur analogique/numérique,

2. les paramètres extrinsèques qui dépendent uniquement de la position de la caméra dans son environnement.

Ces deux types de paramètres peuvent être déterminés, soit simultanément, soit indépendamment [FT87] [Tsa87b] [LT87] [CLA93] [PS90].

Nous nous intéressons ici à l'estimation des paramètres extrinsèques. Les solutions proposées pour résoudre ce problème peuvent se classer en plusieurs catégories en fonction de :

- la nature des primitives utilisées (points, droites, coniques, ...) ;

- le modèle de la caméra (perspectif/affine, avec ou sans distorsions) ;

- la nature de la solution (analytique ou numérique).

Les solutions analytiques ne sont pas nécessairement robustes. Les méthodes basées sur l'optimisation non linéaire peuvent utiliser un plus grand nombre de primitives que les méthodes analytiques, et tendent à être plus robustes puisque la redondance va atténuer l'effet du bruit de l'image sur le calcul de la pose. Toutefois, ces méthodes numériques souffrent de deux inconvénients : i) la nécessité de connaître une solution initiale proche de la solution cherchée et ii) l'importance du temps de calcul nécessaire à la convergence vers la solution optimale puisqu'il y a inversion d'une matrice à chaque pas d'itération. Ce dernier inconvénient, à lui seul, rend très difficile voire impossible l'utilisation de ces méthodes dans des applications exigeant le calcul de la pose en temps réel (asservissement visuel, poursuite d'objets, traqueurs optiques utilisés dans la réalité virtuelle). Comme les méthodes non linéaires itératives sont mal adaptées aux applications temps réel et que des minima locaux peuvent être trouvés par ces méthodes, il s'avère très intéressant de disposer d'une méthode qui ne nécessite pas d'initialisation et qui est robuste et rapide. Une telle approche a été proposée par DeMenthon et al. [DeM93] [DD92] [DD95]. Cette approche s'appuie sur des techniques de l'algèbre linéaire et est itérative comme les méthodes numériques. Elle ne nécessite pas d'inversions de matrice dans le processus itératif.

4.2 Etat de l'art

Haralick et al. [HJL+89] ont présenté une dissertation qui regroupe 80 solutions différentes proposées dans la photogrammétrie.

On peut classer ces solutions en 2 catégories :

1. *Solutions analytiques.*

   Les chercheurs ont proposé des solutions analytiques lorsque le nombre des primitives est assez réduit (3 à 6) dans des configurations coplanaires ou non coplanaires [FB81] [SK89] [HCLL89] [DRLR89]. [FB81] et [WMSM91] ont montré que pour trois points il peut y avoir jusqu'à 4 solutions. Des solutions existent également pour 4 points coplanaires [HYH85] ou pour 4 points non coplanaires [HCLL89] [HN91].

2. *Solutions numériques.* Lorsque le nombre de points de correspondances est grand, les solutions analytiques ne sont plus efficaces, car on doit résoudre un système d'équations non linéaires où le nombre d'équations est grand par rapport au nombre d'inconnues. Dans ce cas, des solutions numériques sont nécessaires. Nous allons discuter de quelques solutions proposées.

   Yuan [Yua89] a proposé de séparer la rotation de la translation et concentré ses efforts sur la détermination de la rotation . Les six contraintes d'orthonormalité de la matrice de rotation fournissent 6 contraintes quadratiques. La solution commune à ces 6 contraintes est trouvée par la méthode de Newton (descente de gradient). Lowe [Low87] a également utilisé la méthode de Newton. La fonction d'erreur à minimiser est la somme des distances au carré entre chaque projection d'un point (d'une droite) de l'objet et son équivalent dans l'image. Liu et al. [LHF90] ont examiné une méthode itérative, la rotation étant représentée par les angles d'Euler. D'abord, ils constatent que si l'on met en correspondance des droites et non pas des points, la rotation est séparée de la translation et qu'une fois qu'on a déterminé la rotation, la détermination de la translation devient un problème linéaire. Cependant, les auteurs constatent que leur méthode marche bien lorsque les angles de la rotation sont inférieurs à 30^0. Dhome et al. [DYL93] ont utilisé une technique itérative pour déterminer les orientations externes (par rapport à la caméra) et les orientations internes entre les composants rigides constituant un robot articulé à partir des correspondances droites/droites. Les paramètres des translations sont déterminés par un système linéaire. Phong et al. [PHYP95] ont présenté le déplacement caméra/objet par un quaternion dual. Ils ont écrit la fonction d'erreur en fonction de ce quaternion aussi bien pour des correspondances points/points que pour des correspondances droites/droites. La solution optimale pour ce quaternion est obtenue en minimisant la fonction d'erreur par la méthode de région de confiance. Cette dernière technique est la moins sensible aux problèmes dûs à l'initialisation. Zhang et al. [ZBC94] ont proposé une méthode itérative qui détermine la nouvelle position de la caméra correspondant à une nouvelle image. La solution initiale n'est autre que la position de la caméra correspondant à l'image initiale. Ils représentent la rotation par les angles de Roulis-Tangage-Lacet et appliquent leur méthode à des images aériennes.

4.3 Formulation

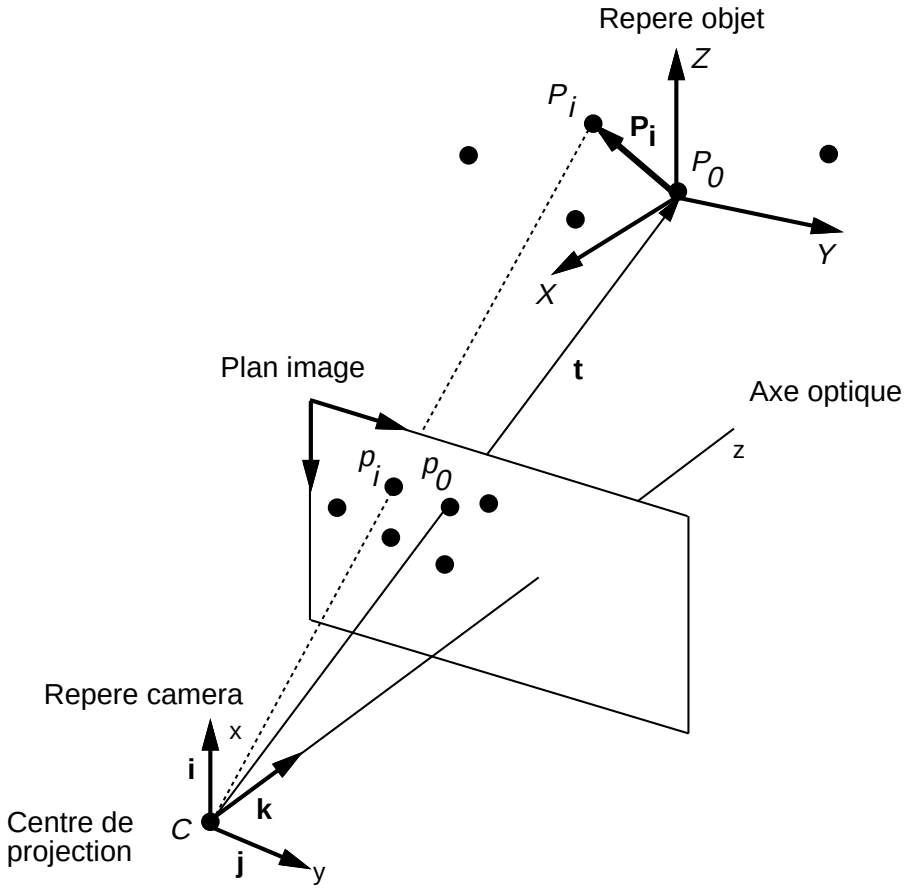


FIG. 4.1 - *Le modèle sténopé pour une caméra.*

Sur la Figure 4.1, nous représentons le modèle classique d'une caméra (sténopé) avec son centre de projection C, son plan d'image à la distance focale de C, ses axes de repère Cx et Cy pointent dans la direction des lignes et colonnes du capteur de la caméra. Nous supposons que le plan d'image est toujours placé entre l'objet et le centre de projection. L'axe optique de la caméra est confondu avec l'axe Cz. Les vecteurs unitaires de ces trois axes sont appelés $\mathbf{i}$, $\mathbf{j}$ et $\mathbf{k}$.

On considère un objet situé dans le champ visuel de la caméra. Cet objet comprend $n+1$ points caractéristiques $(P_0, P_1, \ldots, P_n)$. Soit P_i un point de cet objet dont les coordonnées $(X_i,\ Y_i,\ Z_i)$ sont exprimées dans un repère attaché à cet objet. L'origine de ce repère est désigné par le point P_0. Le point objet P_i se projette sur le plan image en un point p_i ayant les coordonnées caméra (normalisées) x_i et y_i. On a ($\mathbf{P}_i$ représente le vecteur $\vec{P_0P_i}$) :

$$x_i = \frac{\mathbf{i} \cdot \mathbf{P}_i + t_x}{\mathbf{k} \cdot \mathbf{P}_i + t_z} \tag{4.1}$$

$$y_i = \frac{\mathbf{j} \cdot \mathbf{P}_i + t_y}{\mathbf{k} \cdot \mathbf{P}_i + t_z} \tag{4.2}$$

Ces équations décrivent le modèle perspectif de la caméra où la transformation rigide entre le repère de la caméra et celui de l'objet est donnée par :

$$T = \begin{bmatrix} \mathbf{i}^T & t_x \\ \mathbf{j}^T & t_y \\ \mathbf{k}^T & t_z \\ 0\,0\,0 & 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{t} \\ 0\,0\,0 & 1 \end{bmatrix}$$

La relation entre les coordonnées caméra et les coordonnées images peut être obtenue en utilisant les paramètres intrinsèques de la caméra :

$$u_i = \alpha_u x_i + u_c \tag{4.3}$$
$$v_i = \alpha_v y_i + v_c \tag{4.4}$$

Dans ces équations, α_u et α_v représentent la distance focale de la caméra exprimée en pixels verticales et en pixels horizontales respectivement ; u_c et v_c représentent les coordonnées images de l'intersection de l'axe optique avec le plan image.

Nous divisons le numérateur et le dénominateur des équations (4.1) et (4.2) par t_z et nous introduisons les notations suivantes :

- $\mathbf{I} = \mathbf{i}/t_z$ est la première ligne de la matrice de rotation divisée par la composante sur l'axe optique du vecteur de translation ;
- $\mathbf{J} = \mathbf{j}/t_z$ est la deuxième ligne de la matrice de rotation divisée par la composante sur l'axe optique du vecteur de translation ;
- $x_0 = t_x/t_z$ et $y_0 = t_y/t_z$ sont les coordonnées caméra du point p_0 qui est la projection de P_0 – l'origine du repère objet ;
- $\epsilon_i = \mathbf{k} \cdot \mathbf{P}_i/t_z$.

Avec ces notations, Les équations (4.1) et (4.2) deviennent :

$$x_i = \frac{\mathbf{I} \cdot \mathbf{P}_i + x_0}{1 + \epsilon_i} \tag{4.5}$$
$$y_i = \frac{\mathbf{J} \cdot \mathbf{P}_i + y_0}{1 + \epsilon_i} \tag{4.6}$$

4.4 Pose à partir d'une projection perspective faible

4.4.1 Définition et équations fondamentales

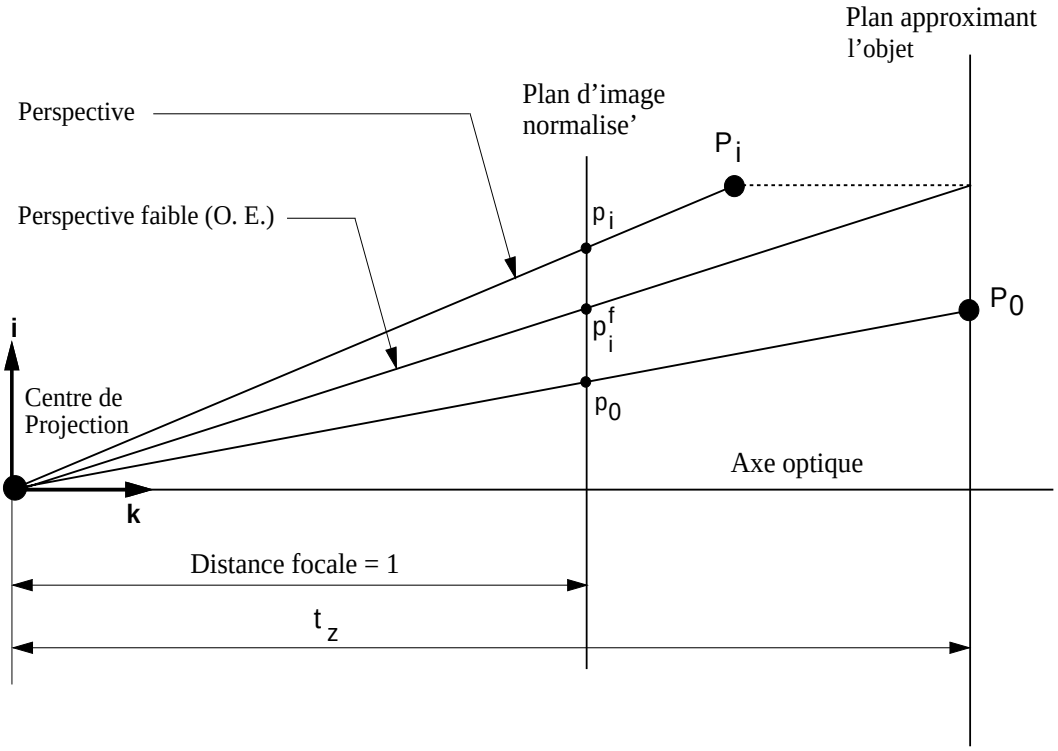


FIG. 4.2 - *Projection perspective faible.*

La projection perspective faible est une approximation de la projection perspective. On l'appelle aussi projection orthographique à l'échelle. Elle permet d'établir un modèle affine pour la caméra. Ce modèle est essentiellement utilisable dans le cas d'objets plans positionnés parallèlement au plan image. Elle suppose que les points de l'objet appartiennent au même plan parallèle au plan de l'image et passant par un point donné appelé point de référence.

Dans ce modèle, les projections de points sont obtenues de la manière suivante (voir Figure 4.2) : on construit un plan parallèle au plan image et passant par le point de référence P_0, les points subissent d'abord une projection orthographique sur ce plan selon la direction de l'axe optique. Ensuite, ces points, devenus coplanaires, subissent une projection perspective sur le plan image. Sur cette figure sont illustrées les projections perspective et perspective faible du point P_i, notées p_i et p_i^f respectivement.

Ainsi, si la taille de la scène est relativement petite par rapport à la distance entre la caméra et la scène la projection perspective faible serait une bonne approximation de la projection perspective. Notons que la profondeur du point de référence P_0 détermine l'échelle de ce type de projection puisque cette échelle est égale à f/t_z (f représente la distance focale de la caméra).

L'approximation d'une projection perspective par une projection perspective faible est

une approximation d'ordre zéro :

$$\frac{1}{1+\epsilon_i} \approx 1 \quad \forall i,\ i \in \{1 \dots n\}$$

En injectant cette égalité dans les équations (4.5) et (4.6), nous obtenons :

$$\begin{aligned} x_i^f &= \mathbf{I} \cdot \mathbf{P}_i + x_0 \\ y_i^f &= \mathbf{J} \cdot \mathbf{P}_i + y_0 \end{aligned}$$

où x_i^f et y_i^f représentent les coordonnées de la projection perspective faible du point P_i. Ainsi, les coordonnées x_i^f et y_i^f ne sont que les numérateurs des fractions données par les équations (4.5) et (4.6). Il en résulte que la relation entre les coordonnées de la projection perspective faible et celles de la projection perspective sera donnée par :

$$x_i^f = x_i\,(1+\epsilon_i) \tag{4.7}$$
$$y_i^f = y_i\,(1+\epsilon_i) \tag{4.8}$$

On peut également retrouver ces deux équations en calculant les coordonnées du point p_i^f à partir de la Figure 4.2. Pour ce faire, on utilise les propriétés géométriques des triangles semblables présents dans cette figure et le fait que $\epsilon_i = \mathbf{k} \cdot \mathbf{P}_i / t_z$.

On pourrait réécrire les équations de la projection perspective (4.5) et (4.6) de la manière suivante :

$$\begin{aligned} \mathbf{P}_i \cdot \mathbf{I} &= x_i\,(1+\epsilon_i) - x_0 \qquad (4.9)\\ &= x_i^f - x_0 \\ \mathbf{P}_i \cdot \mathbf{J} &= y_i\,(1+\epsilon_i) - y_0 \qquad (4.10)\\ &= y_i^f - y_0 \end{aligned}$$

On peut écrire sous forme matricielle les équations (4.9) et (4.10) pour chaque point P_i, avec i allant de 1 à n :

$$\underbrace{P}_{n\times 3}\underbrace{\mathbf{I}}_{3\times 1} = \underbrace{\mathbf{x}}_{n\times 1} \tag{4.11}$$
$$\underbrace{P}_{n\times 3}\underbrace{\mathbf{J}}_{3\times 1} = \underbrace{\mathbf{y}}_{n\times 1} \tag{4.12}$$

avec :

- P est une matrice de dimension $n \times 3$ et formée par les coordonnées 3D des n vecteurs $\mathbf{P}_1 \dots \mathbf{P}_n$. Comme P_0 est l'origine du repère de l'objet, cette matrice peut être écrite sous la forme suivante :

$$P = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n \end{bmatrix}$$

  où X_i, Y_i et Z_i représentent les coordonnées 3D du point P_i.

– $\mathbf{x}$ et $\mathbf{y}$ sont donnés par :

$$\mathbf{x} = \begin{bmatrix} x_1\,(1+\epsilon_1) - x_0 \\ \vdots \\ x_n\,(1+\epsilon_n) - x_0 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1\,(1+\epsilon_1) - y_0 \\ \vdots \\ y_n\,(1+\epsilon_n) - y_0 \end{bmatrix}$$

4.4.2 Pose par approximations successives

L'idée de base proposée par DeMenthon [DD95] est que si des valeurs sont données pour les termes ϵ_i (cela suppose que les projections perspectives faibles des points objets sont connues), les équations (4.11) et (4.12) fournissent alors deux systèmes d'équations linéaires dans lesquels les seules inconnues sont respectivement les coordonnées de $\mathbf{I}$ et $\mathbf{J}$.

Une fois ces deux vecteurs calculés, on peut facilement en déduire la pose. Cependant, la connaissance exacte des ϵ_i suppose que la pose est connue ce qui n'est pas vrai en réalité d'où l'approche heuristique qui consiste à initialiser les ϵ_i par des valeurs quelconques pour déterminer une approximation de la pose. Une fois cette pose déterminée, des valeurs généralement plus exactes peuvent être calculées pour les ϵ_i. Ces nouvelles valeurs des ϵ_i vont mettre à jour les seconds termes des équations (4.11) et (4.12). Si aucune pose approximative n'est disponible, on pourrait poser au départ $\epsilon_i = 0$. Dans ce cas, l'algorithme de pose commence par calculer la pose en supposant que les projections perspectives faibles sont confondues avec les points images. Cette pose est itérativement améliorée de la façon suivante :

1. Pour chaque i, $i \in \{1 \ldots n\}$, $n \geq 3$, $\epsilon_i = 0$.

2. Résoudre les deux systèmes linéaires surcontraints (4.11) et (4.12). La solution pour les deux vecteurs $\mathbf{I}$ et $\mathbf{J}$ est donnée par :

$$\begin{aligned} \mathbf{I} &= (P^T P)^{-1} P^T \mathbf{x} \\ \mathbf{J} &= (P^T P)^{-1} P^T \mathbf{y} \end{aligned}$$

3. Calculer la rotation et la translation entre le repère de l'objet et celui de la caméra :

$$\begin{aligned} t_z &= \frac{1}{2}\left(\frac{1}{\|\mathbf{I}\|} + \frac{1}{\|\mathbf{J}\|}\right) \\ t_x &= x_0\, t_z \\ t_y &= y_0\, t_z \\ \mathbf{i} &= \frac{\mathbf{I}}{\|\mathbf{I}\|} \\ \mathbf{j} &= \frac{\mathbf{J}}{\|\mathbf{J}\|} \\ \mathbf{k} &= \mathbf{i} \times \mathbf{j} \end{aligned}$$

4. Pour chaque i, calculer :

$$\epsilon_i = \frac{\mathbf{k} \cdot \mathbf{P}_i}{t_z}$$

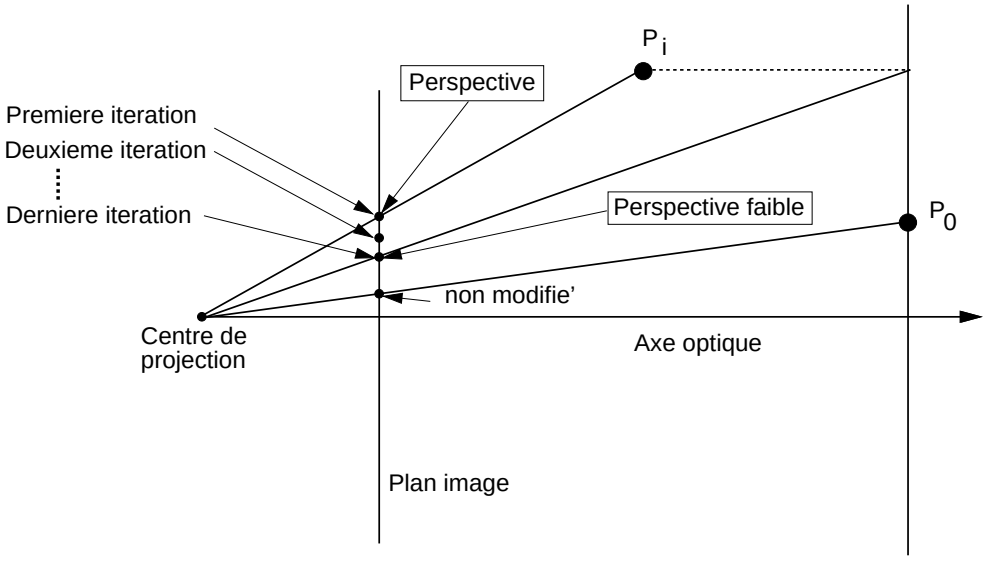


FIG. 4.3 - *Illustration du processus itératif associé à l'algorithme de pose basé sur la projection perspective faible.*

Si les ϵ_i calculés à cette itération sont égaux à ceux calculés à l'itération précédente, alors arrêter la procédure, sinon aller à l'étape 2.

Il est à noter que la matrice pseudo-inverse de P, utilisée dans l'étape 2, est calculée une fois pour toutes. Cet algorithme itératif peut être décrit de manière géométrique (voir Figure 4.3) :

1. Confondre les projections perspectives faibles des points P_i avec les points images p_i. Pour chaque i, $i \in \{1 \ldots n\}$, $p_i^f = p_i$.

2. Calculer une pose approchée de l'objet en supposant que les points P_i se projettent aux points images p_i^f par une projection perspective faible.

3. Déplacer les points de l'objet à partir de leurs positions correspondant à la pose approchée calculée à l'étape 2 pour les placer sur les lignes de vue associées aux images p_i tout en les laissant à la même profondeur (cela correspond à une déformation de l'objet).

   Trouver un autre ensemble de points images p_i^f de ces points déplacés en utilisant une projection perspective faible associée à la pose calculée à l'étape 2.

4. Si ces points p_i^f sont à la même position que les points à l'itération précédente, arrêter la procédure. Sinon aller à l'étape 2.

Notons que dans le repère de la caméra les lignes de vue, associés aux points images p^i sont invariants puisque les paramètres intrinsèques de la caméra et ces points images sont des constantes. La mise à jour des ϵ_i dans les deux systèmes linéaires (4.11) et (4.12) est équivalente à l'étape 3 de la description géométrique. A chaque itération, ce déplacement est nécessaire puisque l'approximation perspective faible déplacent les points objets en

dehors de leur ligne de vue. Il s'effectue perpendiculairement à l'axe optique pour ne pas changer la valeur de ϵ_i correspondant à la pose calculée.

4.4.3 Extension de l'algorithme au cas de droites

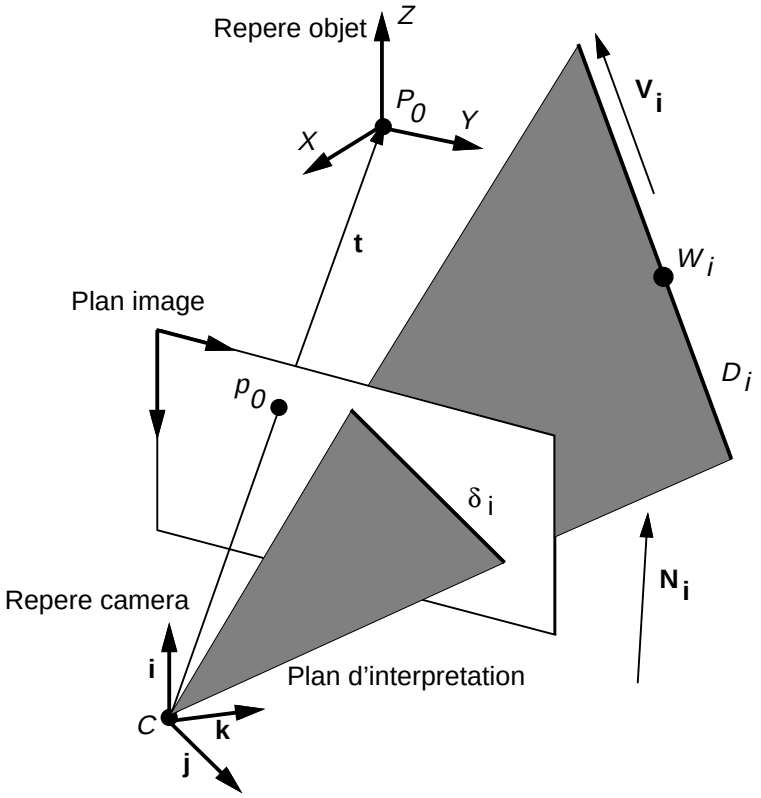


FIG. 4.4 - *Pose à partir des correspondances de droites.*

Nous venons de présenter comment on peut estimer itérativement la pose d'un objet à partir des correspondances points/points. L'algorithme itératif est basé sur l'approximation de la projection perspective par une projection perspective faible. Ici, nous proposons une approche similaire qui, à partir de l'image de droites, va déterminer la pose de l'objet correspondant au modèle perspectif de la caméra. L'avantage majeur de l'utilisation des droites réside dans le fait que l'extraction des position et orientation des segments de droites dans l'image est plus précise que l'extraction de leurs extrémités. Pour ce faire, considérons l'image de n droites de l'objet dont on veut déterminer la pose. On considère également l'image d'un point de l'objet considéré comme l'origine du repère attaché à l'objet (voir Figure 4.4).

La projection perspective, notée δ_i, de la droite objet $\mathcal{D}_i$ est obtenue par l'intersection du plan de l'image avec le plan formé par le centre de projection et la droite objet. Ce dernier plan est le plan d'interprétation associé à la droite objet $\mathcal{D}_i$. L'équation de δ_i est, dans le plan de l'image normalisé, donnée par :

$$a_i\, x + b_i\, y + c_i = 0 \tag{4.13}$$

Considérons maintenant la droite objet $\mathcal{D}_i$. Dans le repère de l'objet cette droite peut être décrite par sa représentation paramétrique. Soient $\mathbf{V}_i$ son vecteur directeur et $\mathbf{W}_i$ son vecteur position. L'équation paramétrique de cette droite est donc donnée par :

$$\mathbf{P}_i = \mathbf{W}_i + \lambda_i \, \mathbf{V}_i \quad (\lambda_i \in I\!R) \tag{4.14}$$

où $\mathbf{P}_i$ est un vecteur joignant l'origine P_0 à un point quelconque de la droite objet $\mathcal{D}_i$.

Tout point appartenant à la droite $\mathcal{D}_i$ se projette sur le plan image en un point dont les coordonnées caméra sont données par :

$$\begin{aligned} x &= \frac{\mathbf{P}_i \cdot \mathbf{I} + x_0}{1 + \epsilon_i} \\ y &= \frac{\mathbf{P}_i \cdot \mathbf{J} + y_0}{1 + \epsilon_i} \end{aligned}$$

où ϵ_i est donné par :

$$\epsilon_i = \mathbf{k} \cdot \mathbf{P}_i / t_z$$

En remplaçant les expressions de x et y dans l'équation (4.13), nous obtenons :

$$a_i \, (\mathbf{P}_i \cdot \mathbf{I} + x_0) + b_i \, (\mathbf{P}_i \cdot \mathbf{J} + y_0) + c_i \, (1 + \mathbf{k} \cdot \mathbf{P}_i / t_z) = 0$$

Si l'on injecte, dans cette équation, l'expression de $\mathbf{P}_i$ donnée par l'équation (4.14), nous obtenons :

$$a_i \, \mathbf{W}_i \cdot \mathbf{I} + b_i \, \mathbf{W}_i \cdot \mathbf{J} + a_i \, x_0 + b_i \, y_0 + c_i \, (1 + \mathbf{k} \cdot \mathbf{W}_i / t_z) + \lambda_i \, (a_i \, \mathbf{V}_i \cdot \mathbf{I} + b_i \, \mathbf{V}_i \cdot \mathbf{J} + c_i \, \mathbf{k} \cdot \mathbf{V}_i / t_z) = 0$$

Comme cette équation est vérifiée pour tout point appartenant à la droite $\mathcal{D}_i$ ($\forall \ \lambda_i$), nous obtenons les deux équations suivantes :

$$a_i \, \mathbf{W}_i \cdot \mathbf{I} + b_i \, \mathbf{W}_i \cdot \mathbf{J} + a_i \, x_0 + b_i \, y_0 + c_i \, (1 + \eta_i) = 0 \tag{4.15}$$

$$a_i \, \mathbf{V}_i \cdot \mathbf{I} + b_i \, \mathbf{V}_i \cdot \mathbf{J} + c_i \, \xi_i = 0 \tag{4.16}$$

où η_i et ξ_i sont donnés par :

$$\begin{aligned} \eta_i &= \mathbf{k} \cdot \mathbf{W}_i / t_z \\ \xi_i &= \mathbf{k} \cdot \mathbf{V}_i / t_z \end{aligned}$$

Comme le vecteur $(a_i \ b_i \ c_i)^T$ donne la direction de la normale N_i au plan d'interprétation, les deux équations (4.15) et (4.16) traduisent le fait que la droite objet $\mathcal{D}_i$, exprimé dans le repère caméra, appartient à ce plan d'interprétation.

Nous avons donc $2n$ équations pour n droites ($n \geq 3$) et nous pouvons écrire ces équations sous forme matricielle :

$$\underbrace{G}_{2n \times 6} \begin{bmatrix} \mathbf{I} \\ \mathbf{J} \end{bmatrix} = \underbrace{\mathbf{z}}_{2n \times 1} \tag{4.17}$$

avec :

- G est une matrice de dimension $2n \times 6$ et est donnée par :

$$G = \begin{bmatrix} a_1 \mathbf{W}_1^T & b_1 \mathbf{W}_1^T \\ \vdots & \vdots \\ a_n \mathbf{W}_n^T & b_n \mathbf{W}_n^T \\ a_1 \mathbf{V}_1^T & b_1 \mathbf{V}_1^T \\ \vdots & \vdots \\ a_n \mathbf{V}_n^T & b_n \mathbf{V}_n^T \end{bmatrix}$$

- $\mathbf{z}$ est un vecteur de dimension $2n$ et est donné par :

$$\mathbf{z} = \begin{bmatrix} -a_1 x_0 - b_1 y_0 - c_1 (1 + \eta_1) \\ \vdots \\ -a_n x_0 - b_n y_0 - c_n (1 + \eta_n) \\ -c_1 \xi_1 \\ \vdots \\ -c_n \xi_n \end{bmatrix}$$

La connaissance des η_i et ξ_i permet de linéariser le système d'équations donné par (4.17). La solution pour $\mathbf{I}$ et $\mathbf{J}$ est donc :

$$\begin{bmatrix} \mathbf{I} \\ \mathbf{J} \end{bmatrix} = G^\dagger \, \mathbf{z}$$

où $G^\dagger$ est la pseudo-inverse de la matrice G. Elle est calculée une fois pour toutes.

La pose de l'objet sera donc obtenue par résolutions successives de cette dernière équation jusqu'à ce que les quantités η_i et ξ_i, qui interviennent dans le calcul du vecteur $\mathbf{z}$, ne varient plus d'une itération à l'autre.

Ces approximations successives peuvent être décrites de manière géométrique :

1. Confondre les projections perspectives faibles des droites objets avec les droites images δ_i. Pour chaque i, $i \in \{1 \ldots n\}$, $\eta_i = \xi_i = 0 \;\Rightarrow\; \delta_i^f = \delta_i$.

2. Calculer une pose approchée de l'objet en supposant que les droites objets $\mathcal{D}_i$ se projettent aux droites images δ_i^f par une projection perspective faible.

3. Déplacer les droites de l'objet à partir de leurs positions correspondant à la pose approchée (calculée à l'étape 2) pour les placer sur les plans d'interprétation associés aux images δ_i tout en les laissant à la même profondeur (cela correspond à une déformation de l'objet).

   Trouver un autre ensemble d'images δ_i^f de ces droites déplacés en utilisant une projection perspective faible associée à la pose de l'étape 2.

4. Si ces droites δ_i^f sont à la même position que les droites à l'itération précédente, arrêter la procédure. Sinon aller à l'étape 2.

4.5 Pose à partir d'une projection para-perspective

4.5.1 Définition et équations fondamentales

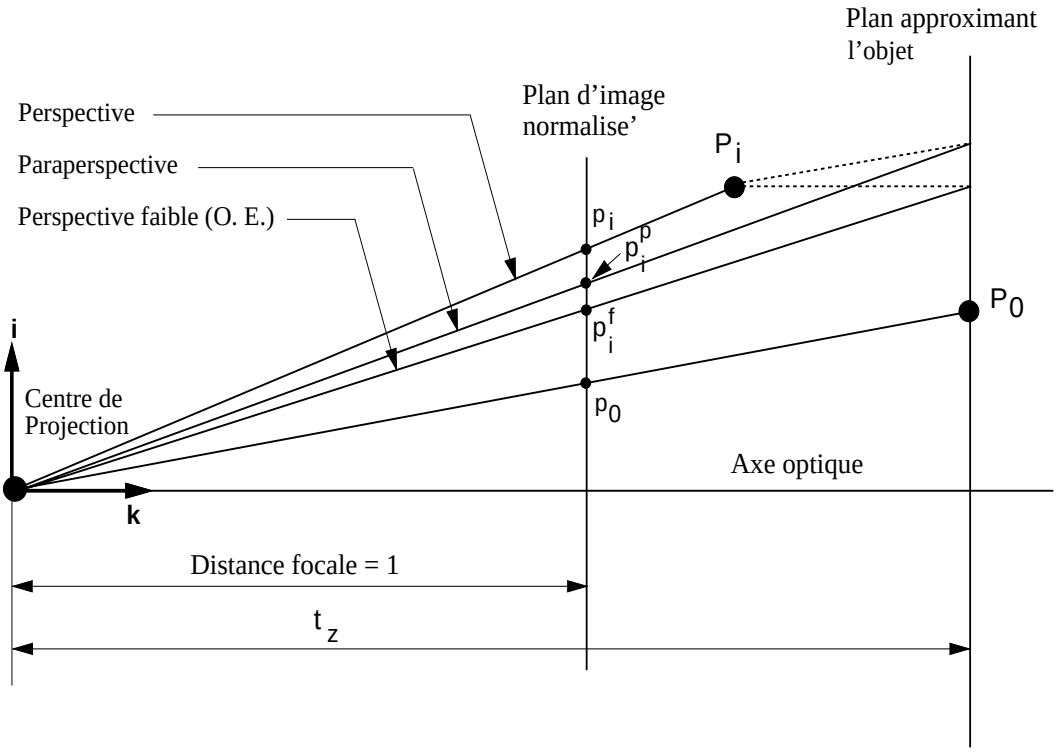


FIG. 4.5 - *Projection perspective faible et projection para-perspective.*

La projection para-perspective est aussi une approximation de la projection perspective. Elle permet d'établir un modèle linéaire pour la caméra tout en tenant compte de la translation latérale de l'objet par rapport à l'axe optique de la caméra [PK94]. En ce sens, pour une même profondeur, la vue para-perspective d'un segment placé très loin de l'axe optique va être différente de celle de ce même segment lorsqu'il est proche de cet axe. La projection para-perspective d'un point P_i est obtenue de la façon suivante (voir Figure 4.5) : on construit un plan parallèle au plan image et passant par le point de référence P_0, le point P_i est projeté sur ce plan suivant la direction définie par le centre de la caméra et ce point de référence. Le point obtenu est ensuite projeté sur le plan image par une projection perspective ce qui donne la projection para-perspective p_i^p.

L'approximation d'une projection perspective par une projection para-perspective est une approximation du premier ordre [Alo90] :

$$\frac{1}{1+\epsilon_i} \approx 1-\epsilon_i \;\; \forall i,\; i \in \{1 \ldots n\}$$

En utilisant cette égalité dans les équations (4.5) et (4.6), nous obtenons les coordonnées de la projection para-perspective du P_i :

$$x_i^p \;\; = \;\; (\mathbf{I} \cdot \mathbf{P}_i + x_0)(1-\epsilon_i)$$

$$\begin{aligned} &\approx \mathbf{I} \cdot \mathbf{P}_i + x_0 - x_0\epsilon_i \\ &= \frac{\mathbf{i} \cdot \mathbf{P}_i}{t_z} + x_0 - x_0\frac{\mathbf{k} \cdot \mathbf{P}_i}{t_z} \end{aligned}$$

où le terme en $1/t_z^2$ est négligé. Il existe une expression similaire pour y_i^p.

Les expressions de ces coordonnées en fonction des paramètres de la pose sont :

$$x_i^p - x_0 = \frac{\mathbf{i} - x_0\,\mathbf{k}}{t_z} \cdot \mathbf{P}_i \tag{4.18}$$

$$y_i^p - y_0 = \frac{\mathbf{j} - y_0\,\mathbf{k}}{t_z} \cdot \mathbf{P}_i \tag{4.19}$$

Afin de trouver la relation entre les coordonnées de la projection para-perspective et celles de la projection perspective, nous réécrivons les équations des coordonnées para-perspectives de la manière suivante :

$$\begin{aligned} x_i^p &= \mathbf{I} \cdot \mathbf{P}_i + x_0 - x_0\epsilon_i \\ y_i^p &= \mathbf{J} \cdot \mathbf{P}_i + y_0 - y_0\epsilon_i \end{aligned}$$

Par identification avec les équations (4.5) et (4.6), nous aurons la relation entre les coordonnées de la projection para-perspective et celles de la projection perspective du point P_i :

$$x_i^p = x_i\,(1 + \epsilon_i) - x_0\epsilon_i \tag{4.20}$$

$$y_i^p = y_i\,(1 + \epsilon_i) - y_0\epsilon_i \tag{4.21}$$

Ces coordonnées peuvent être obtenues d'une manière géométrique à partir de la Figure 4.5 où x_i^p et y_i^p représentent les coordonnées du point p_i^p.

En remplaçant les équations (4.20) et (4.21) dans les équations (4.18) et (4.19), nous aurons :

$$\mathbf{P}_i \cdot \mathbf{I}_p = (x_i - x_0)(1 + \epsilon_i) \tag{4.22}$$

$$\mathbf{P}_i \cdot \mathbf{J}_p = (y_i - y_0)(1 + \epsilon_i) \tag{4.23}$$

avec :

$$\mathbf{I}_p = \frac{\mathbf{i} - x_0\,\mathbf{k}}{t_z} \tag{4.24}$$

$$\mathbf{J}_p = \frac{\mathbf{j} - y_0\,\mathbf{k}}{t_z} \tag{4.25}$$

Si les ϵ_i sont connus alors les équations (4.22) et (4.23) fournissent deux systèmes linéaires qui peuvent être écrits sous la forme matricielle suivante :

$$\underbrace{P}_{n\times 3}\underbrace{\mathbf{I}_p}_{3\times 1} = \underbrace{\mathbf{x}_p}_{n\times 1} \quad (4.26)$$

$$\underbrace{P}_{n\times 3}\underbrace{\mathbf{J}_p}_{3\times 1} = \underbrace{\mathbf{y}_p}_{n\times 1} \quad (4.27)$$

Les deux vecteurs $\mathbf{x}_p$ et $\mathbf{y}_p$ sont donnés par :

$$\mathbf{x}_p = \begin{bmatrix} (x_1 - x_0)\,(1+\epsilon_1) \\ \vdots \\ (x_n - x_0)\,(1+\epsilon_n) \end{bmatrix}, \quad \mathbf{y}_p = \begin{bmatrix} (y_1 - y_0)\,(1+\epsilon_1) \\ \vdots \\ (y_n - y_0)\,(1+\epsilon_n) \end{bmatrix}$$

4.5.2 Pose par approximations successives

Dans le paragraphe précédent, nous venons de démontrer qu'on peut calculer, par un processus linéaire, les deux vecteurs $\mathbf{I}_p$ et $\mathbf{J}_p$ à partir de la projection para-perspective de points. Dans ce qui suit, nous allons montrer comment on peut déterminer les paramètres de la pose à partir de ces deux vecteurs. Ainsi, nous pouvons calculer la pose d'un objet en faisant des approximations successives par une projection para-perspective.

Paramètres de la pose

Premièrement, on a :

$$\begin{aligned} \|\mathbf{I}_p\|^2 &= \frac{(\mathbf{i} - x_0\,\mathbf{k})\cdot(\mathbf{i} - x_0\,\mathbf{k})}{t_z^2} \\ &= \frac{1 + x_0^2}{t_z^2} \end{aligned}$$

et:

$$\|\mathbf{J}_p\|^2 = \frac{1 + y_0^2}{t_z^2}$$

Ainsi, nous obtenons :

$$t_z = \frac{1}{2}\left(\frac{\sqrt{1+x_0^2}}{\|\mathbf{I}_p\|} + \frac{\sqrt{1+y_0^2}}{\|\mathbf{J}_p\|}\right) \quad (4.28)$$

$$t_x = x_0\,t_z \quad (4.29)$$

$$t_y = y_0\,t_z \quad (4.30)$$

Deuxièment, nous calculons les trois vecteurs orthogonaux et unitaires $\mathbf{i}$, $\mathbf{j}$ et $\mathbf{k}$. Les équations (4.24) et (4.25) donnent :

$$\mathbf{i} = t_z\,\mathbf{I}_p + x_0\,\mathbf{k} \quad (4.31)$$

$$\mathbf{j} = t_z\,\mathbf{J}_p + y_0\,\mathbf{k} \quad (4.32)$$

Le vecteur $\mathbf{k}$ est le produit vectoriel de ces deux vecteurs :

$$\begin{aligned}\mathbf{k} &= \mathbf{i} \times \mathbf{j} \\ &= t_z^2\, \mathbf{I}_p \times \mathbf{J}_p + t_z y_0\, \mathbf{I}_p \times \mathbf{k} - t_z x_0\, \mathbf{J}_p \times \mathbf{k}\end{aligned}$$

Soient $\Omega(\mathbf{a})$ la matrice antisymétrique associée au vecteur $\mathbf{a}$ et $I_{3\times 3}$ la matrice identité. L'expression précédente peut être écrite sous la forme matricielle suivante :

$$(I_{3\times 3} - t_z y_0\, \Omega(\mathbf{I}_p) + t_z x_0\, \Omega(\mathbf{J}_p))\ \mathbf{k} = t_z^2\, \mathbf{I}_p \times \mathbf{J}_p \tag{4.33}$$

Cette équation permet de calculer le vecteur $\mathbf{k}$ pourvu que le système linéaire ait un rang plein. En effet, la matrice A :

$$A = I_{3\times 3} - t_z y_0\, \Omega(\mathbf{I}_p) + t_z x_0\, \Omega(\mathbf{J}_p)$$

elle est de la forme :

$$A = \begin{bmatrix} 1 & \gamma & -\beta \\ -\gamma & 1 & \alpha \\ \beta & -\alpha & 1 \end{bmatrix}$$

Son déterminant est toujours strictement positif :

$$det(A) = 1 + \alpha^2 + \beta^2 + \gamma^2$$

Une fois le vecteur $\mathbf{k}$ calculé, les deux vecteurs $\mathbf{i}$ et $\mathbf{j}$ sont donnés par les équations (4.31) et (4.32).

Approximations successives

L'algorithme de pose est le suivant :

1. Pour chaque i, $i \in \{1 \ldots n\}$, $n \geq 3$, $\epsilon_i = 0$.

2. Résoudre les deux systèmes linéaires surcontraints (4.26) et (4.27) qui donnent une estimation des vecteurs $\mathbf{I}_p$ et $\mathbf{J}_p$:

$$\begin{aligned}\mathbf{I}_p &= (P^T P)^{-1} P^T\, \mathbf{x}_p \\ \mathbf{J}_p &= (P^T P)^{-1} P^T\, \mathbf{y}_p\end{aligned}$$

3. Calculer la translation (t_x, t_y et t_z) et la rotation ($\mathbf{i}$, $\mathbf{j}$ et $\mathbf{k}$) de l'objet par rapport au repère de la caméra en utilisant les équations (4.28), (4.29), (4.30), (4.33),(4.31) et (4.32) ;

4. Pour chaque i, calculer :

$$\epsilon_i = \frac{\mathbf{k} \cdot \mathbf{P}_i}{t_z}$$

   Si les ϵ_i calculés à cette itération sont égaux à ceux calculés à l'itération précédente, alors arrêter la procédure, sinon aller à l'étape 2.

4.5.3 Répartition coplanaire de points

Dans le paragraphe précédent, la solution pour les vecteurs $\mathbf{I}_p$ et $\mathbf{J}_p$ est valable dans le cas où les points de l'objet sont non coplanaires. Dans le cas où ces points sont coplanaires, le rang de la matrice P devient égal à 2. Par conséquent, les deux systèmes linéaires (4.26) et (4.27) ne fournissent plus une solution unique. Dans le repère de l'objet, on considère le plan formé par ces points et on décompose les vecteurs $\mathbf{I}_p$ et $\mathbf{J}_p$ en deux composantes : la première composante est contenue dans le plan de l'objet, la deuxième composante est perpendiculaire à ce plan (voir Figure 4.6) :

Il s'ensuit que :

$$\mathbf{I}_p = \mathbf{I}_0 + \lambda\,\mathbf{u} \tag{4.34}$$

$$\mathbf{J}_p = \mathbf{J}_0 + \mu\,\mathbf{u} \tag{4.35}$$

où $\mathbf{u}$ est un vecteur unitaire et normal au plan de l'objet.

Puisque les points sont coplanaires, on a $\vec{P_0P_i} \cdot \mathbf{u} = 0$. Le vecteur $\mathbf{u}$ est donc le vecteur unitaire de l'espace nul de la matrice P. Il peut être obtenu comme le vecteur propre correspondant à la plus petite valeur singulière dans la deuxième matrice orthogonale fournie par la décomposition en valeurs singulières de la matrice P. Ce calcul semble utile dans le cas où les points ne sont pas exactement coplanaires.

En remplaçant les deux équations (4.34) et (4.35) dans (4.26) et (4.27), nous obtenons :

$$P\,\mathbf{I}_0 = \mathbf{x}_p$$

$$P\,\mathbf{J}_0 = \mathbf{y}_p$$

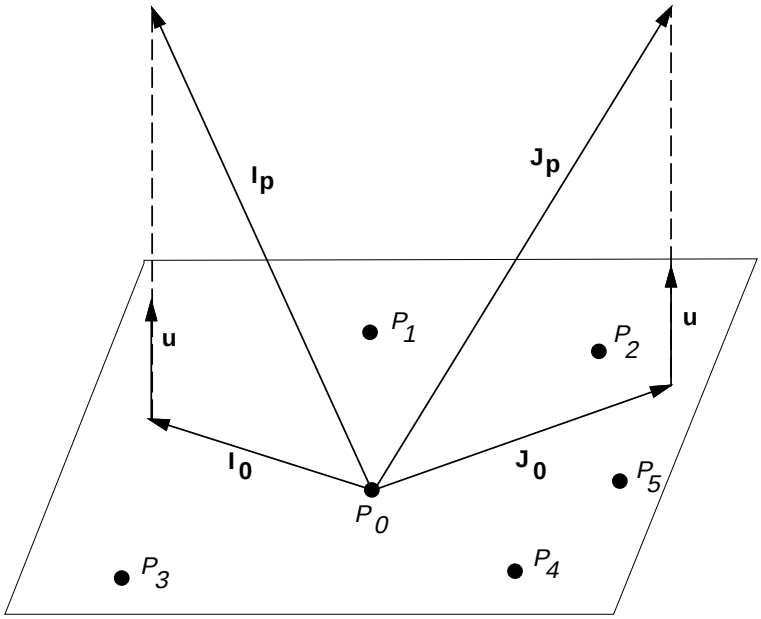


FIG. 4.6 - *Une configuration coplanaire de points.*

Nous constatons que chaque vecteur ayant son origine confondue avec celui du repère de l'objet et ayant son extrémité sur la droite perpendiculaire au plan de l'objet et passant

par l'extrémité du vecteur $\mathbf{I}_0$ est une solution du système linéaire (4.26). Les vecteurs $\mathbf{I}_0$ et $\mathbf{J}_0$, qui représentent la projection sur le plan de l'objet des vecteurs $\mathbf{I}_p$ et $\mathbf{J}_p$, sont calculés à partir des systèmes (4.26) et (4.27) auxquels on a ajouté les deux contraintes supplémentaires suivantes :

$$\mathbf{u} \cdot \mathbf{I}_0 = 0$$

$$\mathbf{u} \cdot \mathbf{J}_0 = 0$$

Les deux vecteurs $\mathbf{I}_0$ et $\mathbf{J}_0$ sont donnés par :

$$\begin{aligned} \mathbf{I}_0 &= (P'^T P')^{-1} P'^T \begin{bmatrix} \mathbf{x}_p \\ 0 \end{bmatrix} \\ \mathbf{J}_0 &= (P'^T P')^{-1} P'^T \begin{bmatrix} \mathbf{y}_p \\ 0 \end{bmatrix} \end{aligned}$$

La matrice P' est définie par :

$$P' = \begin{bmatrix} P \\ \mathbf{u}^T \end{bmatrix}$$

Son rang est égal à 3.

Afin d'estimer $\mathbf{I}_p$ et $\mathbf{J}_p$, nous devons calculer les deux scalaires λ et μ. Dans [ODD93] (projection perspective faible) ces deux scalaires sont déterminés en utilisant les deux contraintes :

$$\begin{aligned} \|\mathbf{I}\| &= \|\mathbf{J}\| \\ \mathbf{I} \cdot \mathbf{J} &= 0 \end{aligned}$$

Dans notre cas, ces deux scalaires peuvent être déterminés en utilisant les contraintes que vérifient les deux vecteurs $\mathbf{I}_p$ et $\mathbf{J}_p$ (déduites des équations (4.24) et (4.25)) :

$$\begin{aligned} \|\mathbf{I}_p\|^2 &= \frac{1 + x_0^2}{t_z^2} \\ \|\mathbf{J}_p\|^2 &= \frac{1 + y_0^2}{t_z^2} \\ \mathbf{I}_p \cdot \mathbf{J}_p &= \frac{x_0 y_0}{t_z^2} \end{aligned}$$

En éliminant t_z de ces trois égalités, nous obtenons les deux contraintes suivantes :

$$\begin{aligned} \mathbf{I}_p \cdot \mathbf{J}_p &= \frac{x_0 y_0}{1 + x_0^2} \|\mathbf{I}_p\|^2 \\ \mathbf{I}_p \cdot \mathbf{J}_p &= \frac{x_0 y_0}{1 + y_0^2} \|\mathbf{J}_p\|^2 \end{aligned}$$

En remplaçant les équations (4.34) et (4.35) dans ces expressions, nous pouvons écrire :

$$\begin{aligned}\mathbf{I}_0 \cdot \mathbf{J}_0 + \lambda\,\mu &= \frac{x_0 y_0}{1+x_0^2}\,(\|\mathbf{I}_0\|^2 + \lambda^2)\\ \mathbf{I}_0 \cdot \mathbf{J}_0 + \lambda\,\mu &= \frac{x_0 y_0}{1+y_0^2}\,(\|\mathbf{J}_0\|^2 + \mu^2)\end{aligned}$$

L'élimination de μ de ces deux équations donne l'équation biquadratique suivante :

$$\mathcal{A}\,\lambda^4 + \mathcal{B}\,\lambda^2 + \mathcal{C} = 0 \tag{4.36}$$

Avec :

$$\begin{aligned}\mathcal{A} &= a^2 - g\\ \mathcal{B} &= 2\,a^2 d - g\,d + e - 2\,a\,c\\ \mathcal{C} &= a^2 d^2 + c^2 - 2\,a\,c\,d\\ a &= \frac{x_0 y_0}{1+x_0^2}\\ b &= \frac{x_0 y_0}{1+y_0^2}\\ c &= \mathbf{I}_0 \cdot \mathbf{J}_0\\ d &= \|\mathbf{I}_0\|^2\\ e &= \|\mathbf{J}_0\|^2\\ g &= \frac{1+y_0^2}{1+x_0^2}\end{aligned}$$

Dans le but d'étudier le nombre de racines réelles de l'équation (4.36), nous remplaçons λ^2 par t :

$$\mathcal{A}\,t^2 + \mathcal{B}\,t + \mathcal{C} = 0$$

Examinons le signe des racines de cette équation. Pour ce faire, nous examinons le signe de leur produit qui est donné par :

$$\begin{aligned}\frac{\mathcal{C}}{\mathcal{A}} &= f(x_0, y_0, c, d)\\ &= \frac{-x_0^2 y_0^2\,d^2 - (1+x_0^2)^2\,c^2 + 2x_0 y_0 (1+x_0^2)\,c\,d}{1+x_0^2+y_0^2}\\ &= -\frac{[x_0 y_0\,d - (1+x_0^2)\,c]^2}{1+x_0^2+y_0^2}\end{aligned}$$

Nous pouvons déduire que la valeur de $\mathcal{C}/\mathcal{A}$ est toujours plus petite ou égale à zéro. Ainsi, l'équation du second degré en t possède une racine positive (ou nulle) et une racine négative (ou nulle). Il en résulte que l'équation biquadratique en λ possède deux racines réelles opposées et deux racines imaginaires pures conjuguées.

Les deux racines réelles pour λ fournissent deux solutions pour μ. Par conséquent, il existe deux solutions pour les vecteurs $\mathbf{I}_p$ et $\mathbf{J}_p$. Le module de ces deux vecteurs reste le même pour ces deux solutions.

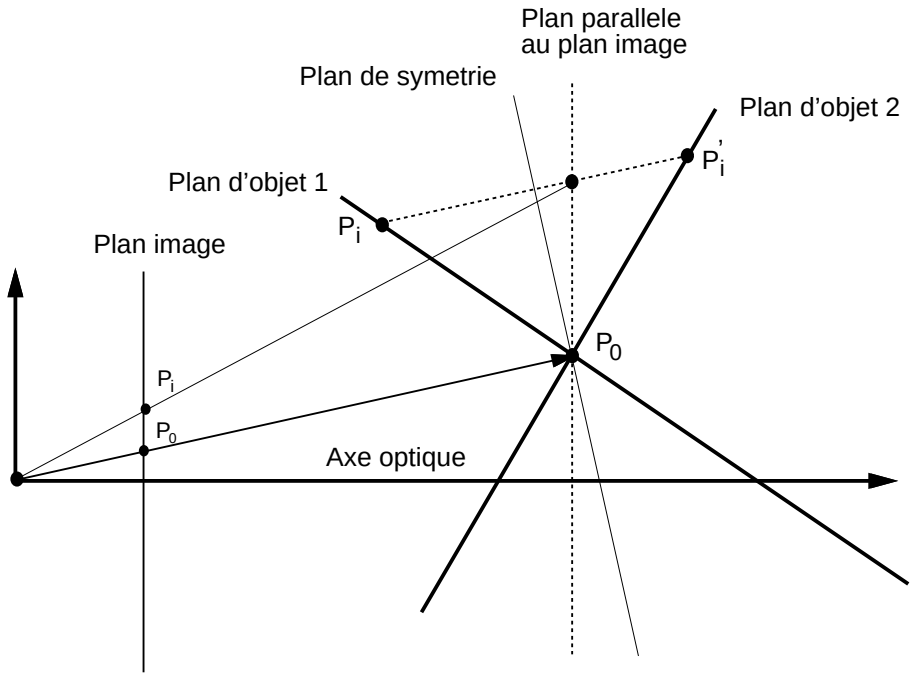


FIG. 4.7 - *Deux objets plans ayant la même projection para-perspective.*

Ainsi, on obtient deux poses qui ont le même vecteur de translation (le vecteur de translation est inversement proportionnel aux modules de $\mathbf{I}_p$ et $\mathbf{J}_p$). Ces deux solutions correspondent à deux orientations du plan qui sont symétriques par rapport à un plan perpendiculaire au vecteur de translation et passant par le point de référence (voir Figure 4.7). Ainsi, l'algorithme itératif qu'on vient de décrire dans le paragraphe 4.5.2 produit deux poses à chaque itération, nous obtenons 2^n solutions après n itérations. Afin d'éviter cette redondance, nous procédons de la manière suivante (voir Figure 4.8) : on garde les deux solutions du premier pas d'itération, mais pour chaque pas suivant on ne garde que la meilleure des deux solutions. Pour choisir la meilleure des deux solutions pour chaque itération après la première itération on utilise comme mesure de qualité la distance moyenne entre les points d'image actuels et les points projetés pour les solutions calculées. Cette démarche donne naissance à deux branches de recherche. Ainsi, à la convergence, on obtient deux poses. D'une manière générale, on pourrait éliminer celle qui n'est pas cohérente avec les données images. Cette stratégie se justifie par le fait que l'exploration d'une branche de moins bonne qualité n'est pas utile : en explorant une telle branche, soit on retrouve l'une des deux poses, soit le processus itératif diverge. Toutefois, dans certains cas, (orientation proche du plan de symétrie, bruit d'image important, profondeur importante) l'ambiguïté de ces deux poses n'est pas levée. Pour cette stratégie, le nombre d'itérations sera égal à $N_a + N_b - 1$. N_a (respectivement N_b) représente le nombre d'itérations de la branche a (branche b).

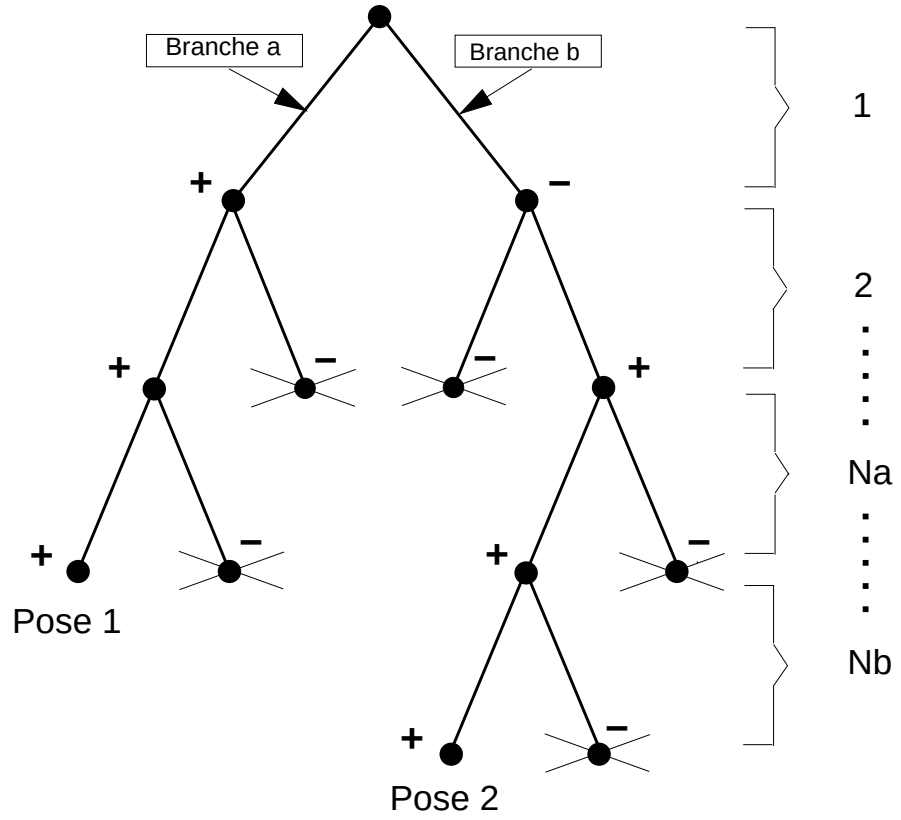


FIG. 4.8 - *Pour un objet plan l'algorithme de pose produit deux solutions à chaque niveau d'itération (+ : pose de meilleur qualité, − : pose de moindre qualité).*

4.5.4 Extension de l'algorithme au cas de droites

Nous venons de présenter comment on peut estimer itérativement la pose d'un objet à partir des correspondances points/points. L'algorithme itératif est basé sur l'approximation de la projection perspective par une projection para-perspective. Ici, nous proposons une approche similaire qui, à partir de l'image de droites, va déterminer la pose de l'objet correspondant au modèle perspectif de la caméra. Pour ce faire, considérons la Figure 4.4. Les coordonnées de la projection para-perspective de tout point appartenant à la droite $\mathcal{D}_i$ sont données par :

$$\begin{aligned} x^p &= \mathbf{P}_i \cdot \mathbf{I}_p + x_0 \\ y^p &= \mathbf{P}_i \cdot \mathbf{J}_p + y_0 \end{aligned}$$

Les équations (4.20) et (4.21), qui relient les coordonnées de deux projections (para-perspective et perspective), peuvent s'écrire :

$$\begin{aligned} x &= \frac{x^p + x_0\epsilon_i}{1+\epsilon_i} \\ y &= \frac{y^p + y_0\epsilon_i}{1+\epsilon_i} \end{aligned}$$

En se rappelant que l'équation de la projection perspective de $\mathcal{D}_i$ est donnée par :

$$a_i\, x + b_i\, y + c_i = 0$$

et en raisonnant de façon analogue à ce qui a été fait dans le paragraphe 4.4.3, on peut trouver, pour chaque droite, les deux égalités suivantes :

$$\begin{aligned} a_i\, \mathbf{W}_i \cdot \mathbf{I}_p + b_i\, \mathbf{W}_i \cdot \mathbf{J}_p + (a_i\, x_0 + b_i\, y_0 + c_i)\,(1 + \eta_i) &= 0 \qquad (4.37) \\ a_i\, \mathbf{v}_i \cdot \mathbf{I}_p + b_i\, \mathbf{v}_i \cdot \mathbf{J}_p + (a_i\, x_0 + b_i\, y_0 + c_i)\,\xi_i &= 0 \qquad (4.38) \end{aligned}$$

où η_i et ξ_i sont donnés par :

$$\begin{aligned} \eta_i &= \mathbf{k} \cdot \mathbf{W}_i / t_z \\ \xi_i &= \mathbf{k} \cdot \mathbf{V}_i / t_z \end{aligned}$$

Nous avons donc $2n$ équations pour n droites et nous pouvons écrire ces équations sous la forme matricielle suivante ($n \geq 3$) :

$$\underbrace{G}_{2n \times 6} \begin{bmatrix} \mathbf{I}_p \\ \mathbf{J}_p \end{bmatrix} = \underbrace{\mathbf{z}_p}_{2n \times 1} \qquad (4.39)$$

où la matrice G est donnée dans le paragraphe 4.4.3 et $\mathbf{z}_p$ est un vecteur de dimension $2n$ et est donné par :

$$\mathbf{z}_p = \begin{bmatrix} -(a_1\, x_0 + b_1\, y_0 + c_1)\,(1 + \eta_1) \\ \vdots \\ -(a_n\, x_0 + b_n\, y_0 + c_n)\,(1 + \eta_n) \\ -(a_1\, x_0 + b_1\, y_0 + c_1)\,\xi_1 \\ \vdots \\ -(a_n\, x_0 + b_n\, y_0 + c_n)\,\xi_n \end{bmatrix}$$

Une fois de plus, les vecteurs $\mathbf{I}_p$ et $\mathbf{J}_p$ sont calculés par approximations successives du système linéaire (4.39).

Notons que si l'objet est plan alors on procède de façon analogue à ce qui a été fait dans le cas des points coplanaires.

4.6 Méthodes non linéaires

Les équations reliant les coordonnées 3D d'un point avec les coordonnées 2D de son image montrent bien le caractère non linéaire du modèle perspectif de la caméra. Dans ces équations, nous trouvons les 9 éléments de la matrice de rotation ainsi que les 3 composantes du vecteur de translation. Les éléments de la matrice de rotation vérifient six contraintes d'orthogonalité. Afin de réduire le nombre des inconnues et des contraintes, nous représentons la rotation par un quaternion unitaire. Il en résulte que le nombre de

paramètres de la pose passe de 12 paramètres à 7 paramètres. Le quaternion vérifie la contrainte non linéaire suivante : la somme des carrés de ses quatre composantes est égale à l'unité. Nous pouvons également représenter la rotation et la translation par un quaternion dual formé de huit paramètres vérifiant deux contraintes non linéaires [PHYP95]. Une technique non linéaire va chercher les paramètres de la pose qui minimisent un certain critère. Le plus souvent, ce critère représente la distance entre les caractéristiques images et les caractéristiques obtenues par reprojection du modèle à la pose calculée.

4.6.1 Correspondances de points

Considérons la Figure 4.1 et supposons qu'on a n correspondances points/points $(P_1, \ldots, P_n)$. Chaque correspondance point objet/point image fournit deux équations précisant que le point objet, le point image (normalisé) et le centre de projection sont alignés :

$$\begin{cases} x_i &= \frac{r_{11}X_i + r_{12}Y_i + r_{13}Z_i + t_x}{r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_z} \\ y_i &= \frac{r_{21}X_i + r_{22}Y_i + r_{23}Z_i + t_y}{r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_z} \end{cases}$$

Ce qui donne les deux contraintes suivantes :

$$r_{11}X_i + r_{12}Y_i + r_{13}Z_i + t_x - x_i\,(r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_z) = 0 \qquad (4.40)$$

$$r_{21}X_i + r_{22}Y_i + r_{23}Z_i + t_y - y_i\,(r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_z) = 0 \qquad (4.41)$$

La relation entre les termes r_{ij} de la matrice de rotation et les composants du quaternion associé est la suivante :

$$R = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 2(q_x q_y + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}$$

En remplaçant les termes r_{ij} par leurs expressions dans les deux équations (4.40) et (4.41), nous obtenons deux équations polynomiales de degré deux. Comme nous avons n correspondances, le critère à minimiser s'écrit en fonction du quaternion et du vecteur de translation de la façon suivante :

$$\begin{aligned} f(\mathbf{q}, \mathbf{t}) &= \sum_{i=1}^{n} (r_{11}X_i + r_{12}Y_i + r_{13}Z_i + t_x - x_i\,(r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_z))^2 + \\ &\quad \sum_{i=1}^{n} (r_{21}X_i + r_{22}Y_i + r_{23}Z_i + t_y - y_i\,(r_{31}X_i + r_{32}Y_i + r_{33}Z_i + t_z))^2 + \\ &\quad \lambda\,(1 - ||\mathbf{q}||^2)^2 \end{aligned} \qquad (4.42)$$

Le dernier terme de cette fonction d'erreur est un terme de pénalisation avec $\lambda > 0$.

Plus λ est grand et mieux on garantira que le quaternion sera unitaire. La minimisation de la fonction d'erreur est équivalente au critère quadratique non linéaire suivant :

$$\min\{f(\mathbf{x}) = \frac{1}{2}\sum_{j=1}^{m} \Phi_j^2(\mathbf{x}) : \mathbf{x} \in I\!R^k\} \qquad (4.43)$$

Afin de minimiser la fonction d'erreur décrite par cette équation, plusieurs méthodes d'optimisation ont été proposées : méthode de Levenberg-Marquardt [PFTW88], méthode de région de confiance. Nous avons choisi celle de Levenberg-Marquardt.

4.6.2 Correspondances de droites

Considérons la Figure 4.4. La droite objet $\mathcal{D}_i$, se projette sur la droite δ_i dont l'équation est donnée par :

$$a_i\, x + b_i\, y + c_i = 0$$

Les coordonnées x et y sont données par :

$$\begin{cases} x &= \frac{X}{Z} \\ y &= \frac{Y}{Z} \end{cases}$$

où X, Y et Z sont les coordonnées dans le repère caméra d'un point P appartenant à la droite $\mathcal{D}_i$. En remplaçant cette dernière équation dans l'équation de la droite 2D, nous obtenons :

$$a_i\, X + b_i\, Y + c_i\, Z = \mathbf{N}_i \cdot \overrightarrow{CP} = 0 \quad (P \in \mathcal{D}_i) \tag{4.44}$$

L'équation (4.44) n'est autre que l'équation du plan d'interprétation contenant le centre de projection, la droite de l'image et le point P appartenant à la droite objet $\mathcal{D}_i$; $\mathbf{N}_i$ est le vecteur normal à ce plan (voir Figure 4.4).

Dans le référentiel de l'objet, la droite objet $\mathcal{D}_i$ est décrite par sa position $\mathbf{W}_i$ et sa direction $\mathbf{V}_i$. Cette droite peut également être exprimée dans le repère de la caméra :

$$\mathbf{W}'_i = R\,\mathbf{W}_i + \mathbf{t} \tag{4.45}$$
$$\mathbf{V}'_i = R\,\mathbf{V}_i \tag{4.46}$$

où R et $\mathbf{t}$ décrivent la rotation et la translation entre le référentiel objet et le référentiel caméra et contiennent justement les paramètres extrinsèques que l'on se propose de déterminer.

Dans le repère de la caméra, le vecteur $\vec{CP}$ s'écrit :

$$\overrightarrow{CP} = \mathbf{W}'_i + \lambda_i\, \mathbf{V}'_i$$

En remplaçant cette équation dans l'équation (4.44), nous obtenons :

$$\mathbf{N}_i \cdot \mathbf{W}'_i + \lambda_i\, \mathbf{N}_i \cdot \mathbf{V}'_i = 0$$

Comme cette dernière équation est vérifiée quelle que soit la valeur de λ_i, nous obtenons :

$$\mathbf{N}_i \cdot \mathbf{W}'_i = 0 \tag{4.47}$$
$$\mathbf{N}_i \cdot \mathbf{V}'_i = 0 \tag{4.48}$$

Par substitution des équations (4.45) et (4.46) dans les équations (4.47) et (4.48), nous pouvons écrire :

$$\begin{aligned} \mathbf{N}_i \cdot (R\,\mathbf{W}_i + \mathbf{t}) &= 0 \\ \mathbf{N}_i \cdot (R\,\mathbf{V}_i) &= 0 \end{aligned}$$

Par conséquent, chaque correspondance droite objet/droite image fournit 2 contraintes. Puisque le nombre de paramètres inconnus qu'il faut déterminer est 6 (3 pour la rotation et 3 pour la translation), le problème peut être résolu si au moins 3 correspondances sont disponibles. Dans le cas général, lorsque n correspondances droite/droite sont disponibles, le problème est de résoudre $2n$ contraintes non linéaires, ce qui est équivalent au problème de minimiser la fonction d'erreur suivante :

$$f(R,\mathbf{t}) = \sum_{i=1}^{n} [(\mathbf{N}_i \cdot (R\,\mathbf{V}_i))^2 + (\mathbf{N}_i \cdot (R\,\mathbf{W}_i + \mathbf{t}))^2]$$

Grâce à la représentation des rotations par des quaternions unitaires nous pouvons transformer la fonction d'erreur qu'on vient d'établir en une somme de carrés de contraintes quadratiques. Rappelons qu'on peut également représenter la rotation et la translation par un quaternion dual.

4.7 Contrainte d'orthogonalité

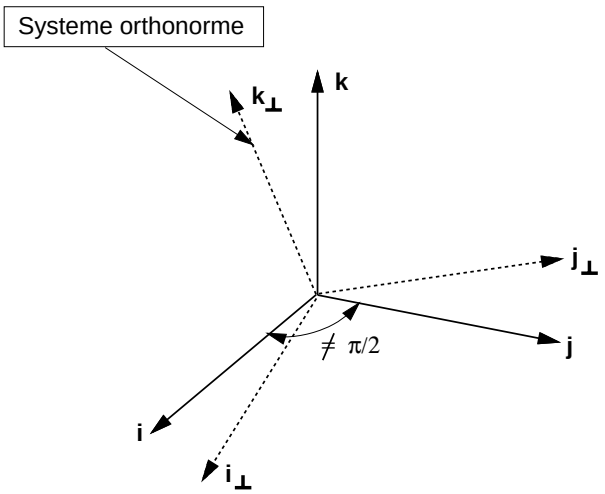


FIG. 4.9 - *Obtention d'une matrice de rotation orthogonale $R_\perp$ à partir des vecteurs $\boldsymbol{i}$, $\boldsymbol{j}$ et $\boldsymbol{k}$.*

Dans le cas général où les points caractéristiques sont non coplanaires, les deux algorithmes linéaires que nous avons présentés (voir sections 4.4 et 4.5) ne garantissent pas l'orthogonalité de la matrice de rotation. En effet, cette matrice est formée par les trois vecteurs lignes : $\mathbf{i}^T$, $\mathbf{j}^T$ et $\mathbf{k}^T$ qui sont mis à jour à chaque itération de l'algorithme de pose. Cet algorithme itératif trouve les deux vecteurs $\mathbf{i}^T$ et $\mathbf{j}^T$ sans avoir recours aux conditions

d'orthogonalité et d'égalité de normes entre ces deux vecteurs. Par conséquent, ces deux vecteurs ne sont pas perpendiculaires et la matrice de rotation estimée n'est pas orthogonale. Afin d'y remédier, nous proposons, à chaque itération, de calculer à partir de cette matrice une vraie matrice de rotation (Figure 4.9). Autrement dit, nous cherchons une matrice de rotation orthogonale $R_\perp$ qui vérifie :

$$R_\perp \begin{bmatrix} \mathbf{i}^T \\ \mathbf{j}^T \\ \mathbf{k}^T \end{bmatrix}^T = I_{3\times 3}$$

Cette expression peut être écrite sous la forme suivante :

$$\begin{aligned} R_\perp \mathbf{i} &= \mathbf{e}_1 \\ R_\perp \mathbf{j} &= \mathbf{e}_2 \\ R_\perp \mathbf{k} &= \mathbf{e}_3 \end{aligned}$$

où $\mathbf{e}_i$ est la $i^{ème}$ colonne de la matrice d'identité.

La solution est donnée par une matrice de rotation qui minimise le critère suivant :

$$\min_{R_\perp} \sum_{i=1}^{3} \| R_\perp \mathbf{v}_i - \mathbf{e}_i \|^2$$

où $\mathbf{v}_i$ peut être $\mathbf{i}$, $\mathbf{j}$ ou $\mathbf{k}$. Il est bien connu que ce problème a une solution analytique [Fau93]. En effet, si nous représentons la rotation inconnue par un quaternion unitaire $\mathbf{q}$, alors le problème de minimisation peut être écrit sous la forme quadratique suivante :

$$\min_{\mathbf{q}} (\mathbf{q}^T B \, \mathbf{q})$$

où B est une matrice de dimension 4×4, symétrique et positive. Ainsi, le quaternion qui minimise l'expression quadratique est le vecteur propre unitaire de la matrice B associé à la plus petite valeur propre de cette matrice. Bien que cette stratégie augmente la complexité du calcul (à chaque pas d'itération on doit calculer la matrice $R_\perp$), la rotation trouvée à la convergence serait plus précise que celle trouvée sans cette stratégie. Notons enfin que l'utilisation des deux vecteurs $\mathbf{i}$ et $\mathbf{j}$ peut suffire pour la détermination de la matrice orthogonale $R_\perp$.

4.8 Sensibilité au calibrage de la caméra

Dans ce chapitre, nous avons supposé que la caméra est calibrée, c'est-à-dire que ses paramètres intrinsèques sont connus [FT87] [LT87]. Ces paramètres sont plus ou moins entachés des incertitudes. Dans ce paragraphe, nous allons analyser la sensibilité des deux algorithmes itératifs en présence d'incertitudes affectant les paramètres intrinsèques de la caméra.

Considérons les équations associées à la projection perspective. En combinant les équations (4.3) et (4.4) avec les équations (4.1) et (4.2), nous obtenons :

$$u_i\,(1+\epsilon_i) - u_0 - u_c\epsilon_i = \alpha_u\,\mathbf{I}\cdot\mathbf{P}_i \tag{4.49}$$
$$v_i\,(1+\epsilon_i) - v_0 - v_c\epsilon_i = \alpha_v\,\mathbf{J}\cdot\mathbf{P}_i \tag{4.50}$$

Lors du calibrage d'une caméra, les facteurs d'échelle verticale et horizontale α_v et α_u sont stables tandis que les coordonnées images du centre optique, u_c et v_c, sont instables et entachées d'une erreur qui peut dépasser 10% de leur valeur nominale. Dans [WS93], on peut trouver plusieurs définitions du centre optique ainsi qu'une étude expérimentale liant les coordonnées de ce centre avec les paramètres des lentilles de la caméra. L'analyse des équations précédentes montre que l'influence de u_c et v_c est pondérée par ϵ_i. Pour chaque point P_i, ϵ_i est la projection du vecteur $\vec{P_0P_i}$ sur l'axe optique divisé par la composante sur cet axe du vecteur de translation $\mathbf{t}$.

Si l'objet est très loin de la caméra, alors l'influence de u_c et v_c serait minimale. Dans ce cas, les points images doivent être extraits avec une grande précision. Dans le cas où l'objet est proche de la caméra cette influence serait importante. Ainsi, en pratique, l'objet doit être placé à une distance qui sera entre 5 et 10 fois la taille de l'objet. Dans ce cas, l'ordre de grandeur des ϵ_i varie entre 0.1 et 0.2 ; et l'erreur sur u_c et v_c serait divisée par un facteur allant de 5 à 10.

4.9 Analyse de la convergence

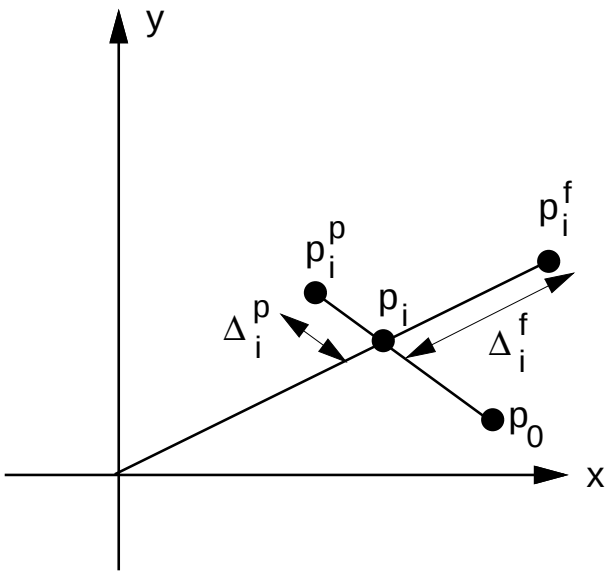


FIG. 4.10 - *Les erreurs de projection Δ_i^f et Δ_i^p.*

Afin d'analyser la convergence des deux algorithmes itératifs présentés dans les paragraphes 4.4 et 4.5, nous considérons les deux systèmes d'équations :

$$\begin{aligned}\mathbf{P}_i\cdot\mathbf{I} &= x_i(1+\epsilon_i) - x_0\\ \mathbf{P}_i\cdot\mathbf{J} &= y_i(1+\epsilon_i) - y_0\end{aligned}$$

$$\begin{aligned} \mathbf{P}_i \cdot \mathbf{I}_p &= (x_i - x_0)(1 + \epsilon_i) \\ \mathbf{P}_i \cdot \mathbf{J}_p &= (y_i - y_0)(1 + \epsilon_i) \end{aligned}$$

Ces deux systèmes décrivent la projection perspective. Le premier système permet de transformer le problème du calcul de la pose en un algorithme itératif utilisant la projection perspective faible tandis que le second système permet de transformer ce problème en un algorithme itératif utilisant la projection para-perspective. Considérons la Figure 4.10 qui représente la projection perspective de P_0 et les trois types de projection de P_i :

- p_0, ayant les coordonnées x_0 et y_0, est la projection perspective du point de référence P_0 ;
- p_i, ayant les coordonnées x_i et y_i, est la projection perspective de P_i ;
- p_i^f, ayant les coordonnées $x_i(1 + \epsilon_i)$ et $y_i(1 + \epsilon_i)$, est la projection perspective faible de P_i ;
- p_i^p, ayant les coordonnées $x_i(1 + \epsilon_i) - x_0\epsilon_i$ et $y_i(1 + \epsilon_i) - y_0\epsilon_i$, est la projection para-perspective de P_i.

Nous avons vu qu'à la première itération des deux algorithmes itératifs tous les ϵ_i sont initialisés avec la valeur zéro. Considérons le premier algorithme et désignons par $\epsilon_i^\star$ les valeurs réelles des ϵ_i que l'algorithme doit calculer. A la première itération ($\epsilon_i = 0$), l'algorithme fournit une solution approximative de la vraie solution puisque nous avons utilisé les points p_i au lieu des points p_i^f. Cette approximation est d'autant plus grossière que le distance entre le point p_i et le point p_i^f est importante. Cette distance représente l'erreur de projection et est égale à :

$$\begin{aligned} \Delta_i^f &= ||p_i - p_i^f|| \\ &= \sqrt{x_i^2 + y_i^2}\ ||\epsilon_i^\star|| \end{aligned}$$

L'erreur de projection Δ_i^f dépend à la fois de $\epsilon_i^\star$ et de la distance entre le point p_i et le centre de l'image. Si ces erreurs sont importantes, alors la convergence de l'algorithme ne sera pas garantie, soit on obtient une convergence après un très grand nombre d'itérations, soit on obtient une divergence. Cependant, DeMenthon et Davis ont remarqué que l'algorithme peut converger même si les $\epsilon_i^\star$ sont très grands pourvu que les points de l'objet soient assez proches de l'axe optique. Dans ce cas, les quantités x_i et y_i sont très petites et peuvent compenser les grandes valeurs de $\epsilon_i^\star$ dans l'expression de Δ_i^f.

Dans le cas du deuxième algorithme où l'on utilise l'approximation para-perspective, l'erreur de projection Δ_i^p est donnée par l'équation suivante :

$$\begin{aligned} \Delta_i^p &= ||p_i - p_i^p|| \\ &= \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}\ ||\epsilon_i^\star|| \\ &= ||p_i - p_0||\ ||\epsilon_i^\star|| \end{aligned}$$

Cette erreur dépend de $\epsilon_i^\star$ et de la distance entre le point p_i et le point p_0. Nous pouvons constater que si le point de référence P_0 n'appartient pas à l'axe optique, on aura l'inéquation suivante :

$$\Delta_i^p < \Delta_i^f$$

Si le point de référence P_0 est adéquatement choisi, alors la distance $||p_i - p_0||$ sera très petite et pourra compenser les grandes valeurs de $\epsilon_i^\star$ même si l'objet est loin de l'axe optique.

Un choix adéquat du point de référence peut être effectué de la manière suivante : On calcule le barycentre des points images p_i, ensuite on choisit, parmi ces points, celui qui est le plus proche du barycentre. Le point objet 3D correspondant à ce point sera pris comme point de référence. Ce choix du point de référence revient à sélectionner, parmi les points images p_i, un point image (x_0, y_0) qui minimise la quantité suivante :

$$\sum_{i=1}^{n} (x_i - x_0)^2 + (y_i - y_0)^2$$

En résumé, l'approximation para-perspective introduit une erreur de projection plus petite que celle introduite par l'approximation perspective faible, de plus cette erreur peut être indépendante de la translation latérale de l'objet par rapport à l'axe optique. Du coup, le comportement de la convergence de l'algorithme itératif utilisant l'approximation para-perspective est meilleur que celui de l'algorithme utilisant l'approximation perspective faible.

4.10 Etude de performance

Dans ce paragraphe nous allons présenter une étude comparative des méthodes de localisation à partir d'une image. Nous évaluons les performances des trois méthodes évoquées dans ce chapitre. Deux types de performances sont étudiés : la stabilité et la convergence.

Pour le premier type nous comparons les trois méthodes, pour le second type nous comparons les deux méthodes itératives linéaires.

Nous venons de voir que les paramètres de la pose dépendent des paramètres intrinsèques de la caméra, des coordonnées 3D des caractéristiques de la scène (points, droites) et des coordonnées 2D de leurs images. Il en résulte que la précision de la pose est directement liée à la précision de ces trois types de données. D'une manière générale, les coordonnées 3D sont parfaitement connues (disposition d'un modèle CAO de l'objet). Cependant, les paramètres intrinsèques de la caméra et les coordonnées images sont entachés d'erreur. Les paramètres intrinsèques de la caméra sont déterminés par un étalonnage préalable qui n'est pas parfait. La précision avec laquelle on extrait les caractéristiques 2D de l'image (points, droites, segments, ...) dépend de plusieurs facteurs: la discrétisation, la distorsion et la précision de l'opérateur d'extraction. Nous pouvons évoquer deux phénomènes liés à la caméra : la discrétisation et la distorsion qui ont pour effet un déplacement du point dans l'image, de l'ordre du pixel dans le cas de la discrétisation et pouvant atteindre plusieurs pixels dans le cas de la distorsion.

Notons que les trois méthodes présentées dans ce chapitre sont toutes des méthodes itératives. Les deux algorithmes linéaires itératifs n'ont pas besoin d'une initialisation, en revanche l'algorithme non linéaire en a besoin. La convergence de l'algorithme non linéaire dépend, entre autres, de la solution initiale, alors que le comportement de la convergence des deux algorithmes linéaires itératifs dépend de la pose de l'objet par rapport à la caméra

et en particulier de la profondeur qui joue un rôle important dans la détermination de nombre d'itérations nécessaires à la convergence.

Dans la suite, nous allons évaluer la stabilité des trois méthodes présentées en présence des erreurs affectant les points dans l'image et/ou les paramètres intrinsèques de la caméra. Cette étude permet de quantifier la précision d'une méthode donnée ainsi que la comparaison des plusieurs méthodes. Nous évaluons également le comportement de la convergence des deux algorithmes linéaires itératifs en fonction de la pose de l'objet (profondeur et translation latérale par rapport à l'axe optique).

4.10.1 Cas non coplanaire

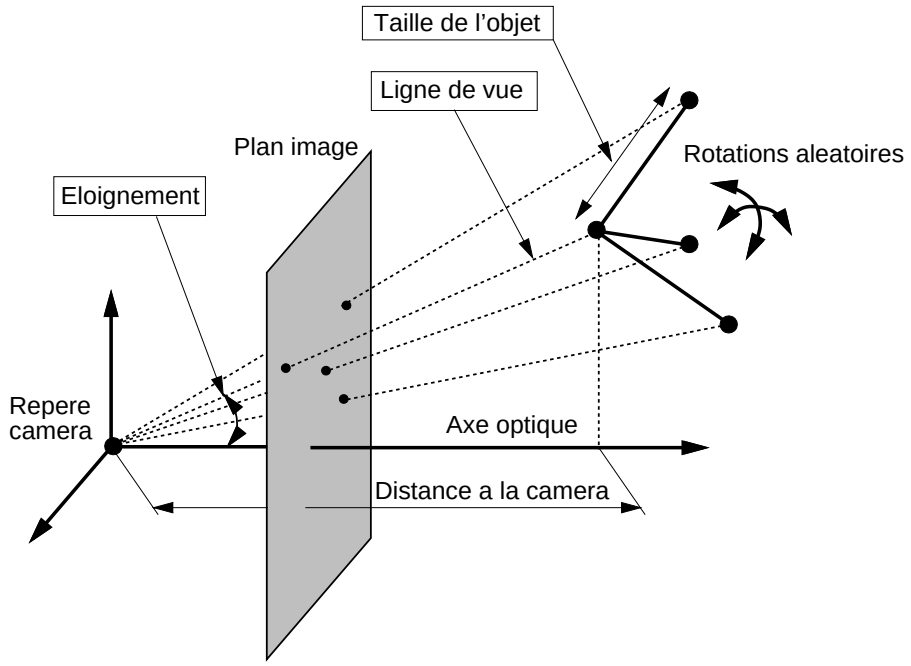


FIG. 4.11 - *Définition des paramètres utilisés dans la mesure des performances.*

Objets

Nous considérons deux objets :

1. Une configuration de 4 points : un tétraèdre dont les trois arêtes sont égaux (10 cm) et perpendiculaires. L'intersection de ces trois arêtes constitue le point de référence (voir Figure 4.11).

2. Une configuration de 8 points : un cube de 10 cm.

Le tétraèdre sera utilisé dans toutes les simulations suivantes (sauf indication contraire).

Positions des objets

La position de l'objet est donnée par celle du point de référence puisque celui-ci représente l'extrémité du vecteur de translation caméra-objet. La distance de l'objet à la

caméra est donnée par la composante sur l'axe optique du vecteur de translation. Elle est exprimée par rapport à la dimension caractéristique de l'objet (cette dimension est égale à 10 cm). Treize distances sont considérées, de deux à vingt-six fois la dimension de l'objet. Ces distances relatives sont représentées en abscisses dans tous les diagrammes d'erreurs. Leur inverse représente la valeur maximale de ϵ. Pour ces 13 distances, le point de référence est contraint d'appartenir à une ligne de vue fixe dans le repère de la caméra. L'angle que fait cette ligne de vue avec l'axe optique pourrait donner l'éloignement de l'objet par rapport à cet axe. Autour de chacune des positions du point de référence, le long de la ligne de vue choisie, l'objet est orienté dans 1000 rotations aléatoires. La matrice de chaque rotation est calculée à partir de trois angles d'Euler choisis par un générateur de nombres aléatoires dans l'intervalle $[0, 2\pi]$. Les erreurs de pose pour ces 1000 rotations sont moyennées. Cette moyenne est représentée en ordonnées dans tous les diagrammes d'erreurs.

Génération des images

Nous obtenons les images des points caractéristiques par projection perspective avec une distance focale de 800 pixels. Nous ne coupons pas l'image, afin d'observer l'effet de grands décentrages des points d'images qui correspondent à d'importants éloignements de l'objet par rapport à l'axe optique. Nous spécifions deux types de bruit. Pour le premier type, nous ajoutons aux coordonnées 2D réelles un bruit uniforme dans l'intervalle $[-0.5,\ +0.5]$ pixel. Pour le deuxième type, le bruit ajouté est un bruit gaussien dont l'écart type est égal à un pixel.

Calcul des erreurs d'orientation et de position

Pour chacune des images synthétiques, l'orientation et la position sont calculées par les trois algorithmes de pose. Ces orientations et positions sont comparées à l'orientation et à la position de l'objet utilisées pour obtenir l'image. Pour la comparaison des orientations nous calculons l'axe et l'angle de rotation autour de cet axe nécessaires pour aligner le repère de l'objet dans son orientation estimée avec le repère dans son orientation réelle. L'erreur d'orientation est définie par cet angle exprimé en degrés. L'erreur de position est définie par le module du vecteur obtenu en faisant la différence vectorielle entre les deux vecteurs de translation : l'estimé et le réel, divisé par le module de ce dernier.

Résultats de simulation

Les Figures 4.12.a et 4.12.b montrent respectivement l'erreur d'orientation et l'erreur relative de position en fonction de la distance de l'objet (tétraèdre) par rapport à la caméra. Ces deux figures ont été obtenues en ajoutant aux coordonnées 2D un bruit gaussien centré dont l'écart type est égal à un pixel. Sur la Figure 4.12.a, la courbe continue (—) correspond à l'algorithme *para-perspective*, la courbe pointillée (...) correspond à ce même algorithme mais avec orthogonalisation de la matrice de rotation (décrite dans le paragraphe 4.7), la courbe interrompue (- - -) correspond à l'algorithme non linéaire. Sur la Figure 4.12.b, la courbe continue correspond à l'algorithme itératif *para-perspective*, la courbe interrompue correspond à l'algorithme non linéaire. L'algorithme *perspective faible*

fournit les mêmes courbes d'erreur que l'algorithme itératif *para-perspective*. Par exemple, sur la Figure (4.12.b), lorsque la distance est égale à 6 fois la taille de l'objet alors l'erreur de position fournie par les deux méthodes linéaires itératives est de $0.6\% = 0.37$ cm ; celle fournie par la méthode non linéaire est de $0.53\% = 0.33$ cm.

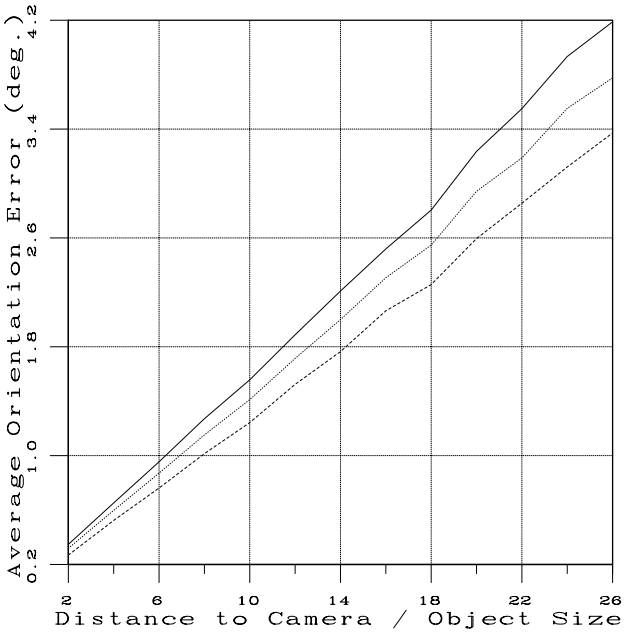
Les Figures 4.13.a et 4.13.b montrent respectivement l'erreur d'orientation et l'erreur relative de position en fonction de la distance de l'objet (cube) par rapport à la caméra.

Sur la Figure 4.14 est présentée l'erreur d'orientation en fonction de la distance de l'objet par rapport à la caméra en présence d'un bruit affectant la caméra et/ou l'image. Les courbes continues correspondent à l'algorithme *para-perspective* alors que les courbes interrompues correspondent à l'algorithme non linéaire. Les courbes de la Figure 4.14.a ont été obtenues en ajoutant aux coordonnées du centre de l'image, u_c et v_c, un bruit uniforme centré dans l'intervalle $[-15\%,\ +15\%]$ de leur valeur réelle ; celles de la Figure 4.14.b ont été obtenues en ajoutant aux deux paramètres, α_u et α_v, un bruit uniforme centré dans l'intervalle $[-10\%,\ +10\%]$ de leur valeur réelle ; celles de la Figure 4.14.c ont été obtenues en ajoutant i) un bruit uniforme, dans l'intervalle $[-0.5,\ +0.5]$ pixel, sur les coordonnées images et ii) un bruit uniforme, dans l'intervalle $[-15\%,\ +15\%]$, sur u_c et v_c ; celles de la Figure 4.14.d ont été obtenues en ajoutant i) un bruit uniforme, dans l'intervalle $[-1,\ +1]$ pixel, sur les coordonnées images et ii) un bruit uniforme sur les quatre paramètres intrinsèques de la caméra - même bruit utilisé dans les deux figures a) et b).

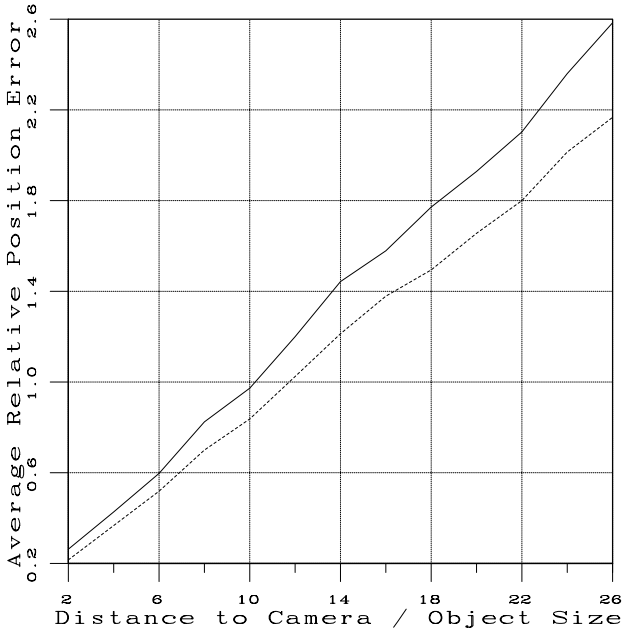
Les Figures 4.15.a et 4.15.b présentent le nombre d'itérations nécessaires à la convergence des deux algorithmes linéaires itératifs. Les courbes marquées par des carrés ($\Box$) ont été obtenues par l'algorithme *perspective faible*, celles marquées par des triangles ($\triangle$) ont été obtenues par l'algorithme *para-perspective*. La Figure 4.15.a correspond à un éloignement de 23 degrés par rapport à l'axe optique, la Figure 4.15.b correspond à un éloignement de 30 degrés. Pour chaque profondeur le nombre présenté sur ces deux figures est obtenu en moyennant les résultats de 1000 orientations aléatoires.

Les Figures 4.16.a et 4.16.b présentent le pourcentage de convergence des deux algorithmes itératifs lorsque l'objet est très proche de la caméra. La Figure 4.16.a correspond à un éloignement de 30 degrés, tandis que la Figure 4.16.b correspond à un éloignement de 35 degrés.

En résumé, les deux méthodes linéaires itératives ont la même précision de localisation ; cela pourrait être dû à ce que chacune de ces deux méthodes résolue itérativement des systèmes linéaires surcontraints similaires. Nous pouvons également constater (dans le cas d'un objet compact) que les propriétés de la convergence de l'algorithme itératif basé sur l'approximation para-perspective ne dépendent pas de la position du point de référence par rapport à l'axe optique, tandis que les propriétés de la convergence de l'algorithme itératif basé sur l'approximation perspective faible se dégradent au fur et à mesure que le point de référence s'éloigne de l'axe optique.

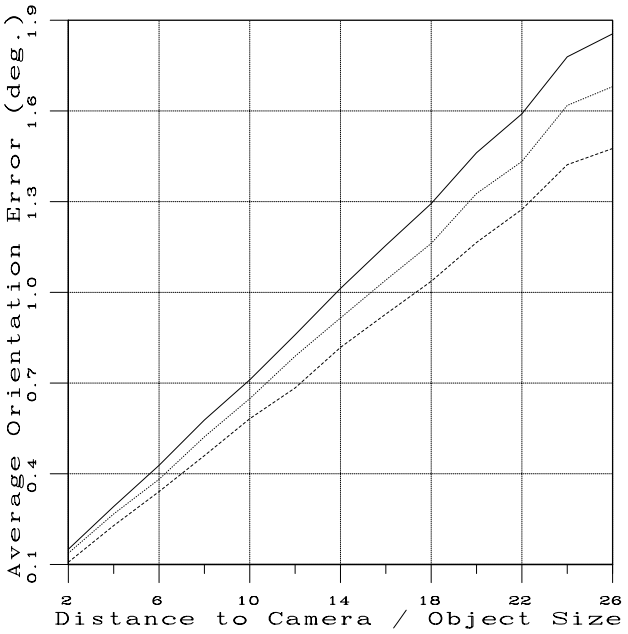


a) Erreur d'orientation.

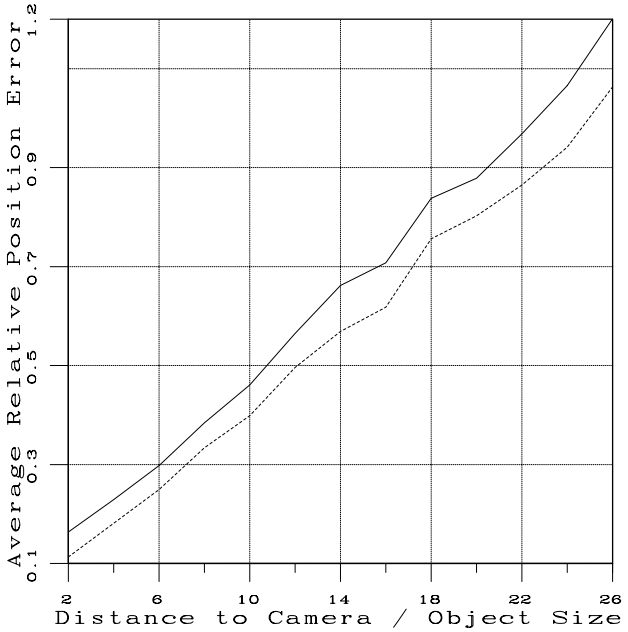


b) Erreur relative de position.

FIG. 4.12 - *Erreurs de pose pour un tétraèdre à 13 distances de la caméra, en présence d'un bruit d'image gaussien d'écart type égal à un pixel. Les courbes (—) correspondent aux algorithmes linéaires itératifs, la courbe (...) correspond à ces mêmes algorithmes mais avec orthogonalisation de la matrice de rotation, les courbes (- - -) correspondent à l'algorithme non linéaire.*



a) Erreur d'orientation.



b) Erreur relative de position.

FIG. 4.13 - *Erreurs de pose pour un cube à 13 distances de la caméra, en présence d'un bruit d'image gaussien d'écart type égal à un pixel. Les courbes (—) correspondent aux algorithmes linéaires itératifs, la courbe (...) correspond à ces mêmes algorithmes mais avec orthogonalisation de la matrice de rotation, les courbes (- - -) correspondent à l'algorithme non linéaire.*

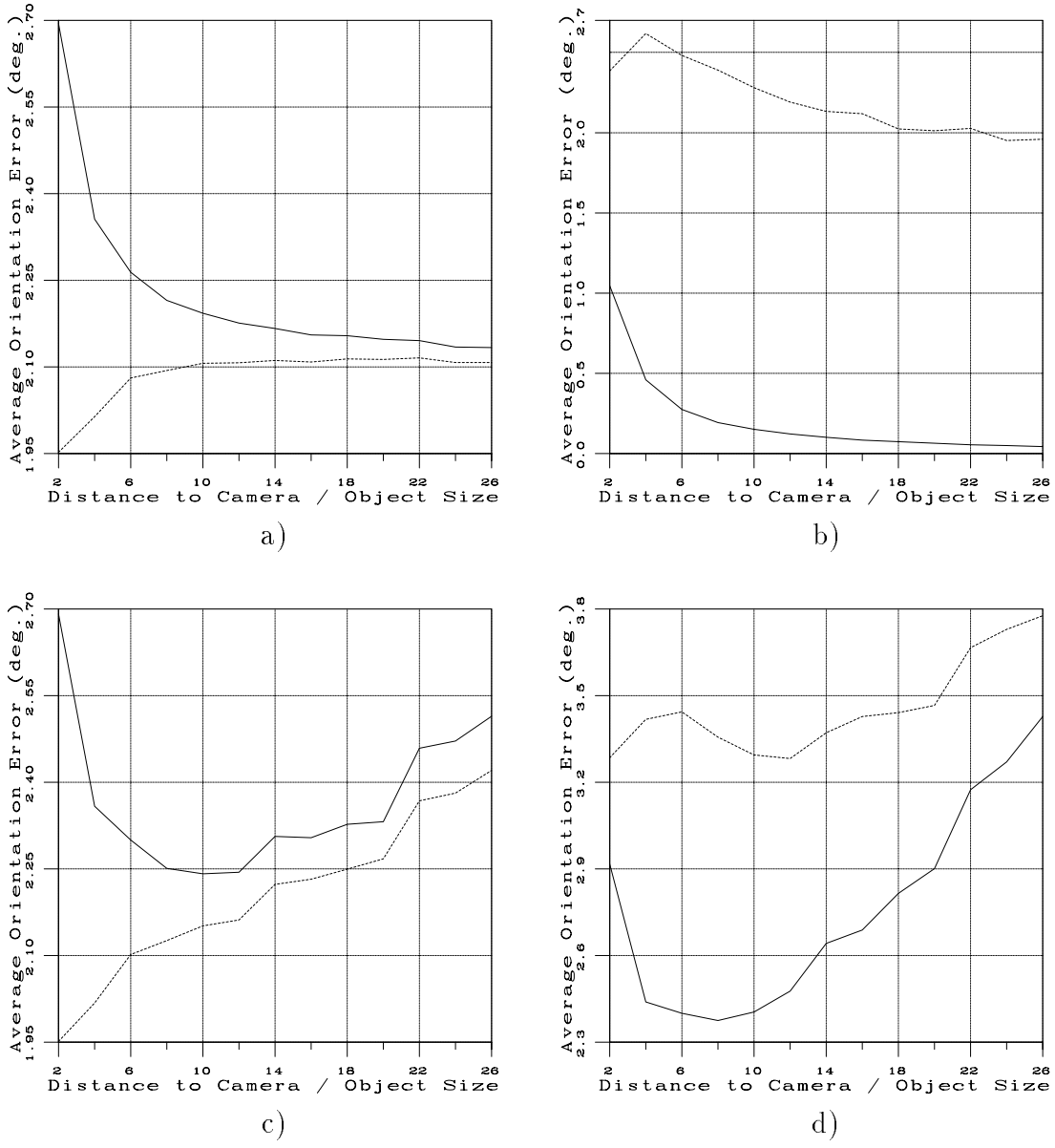
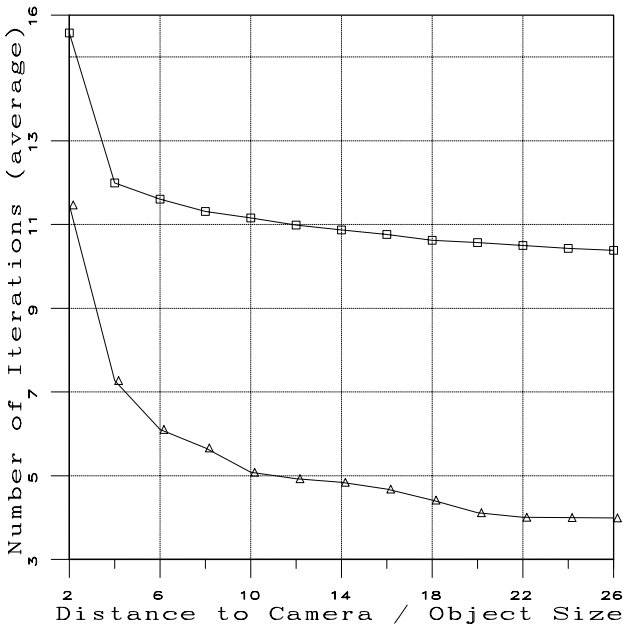
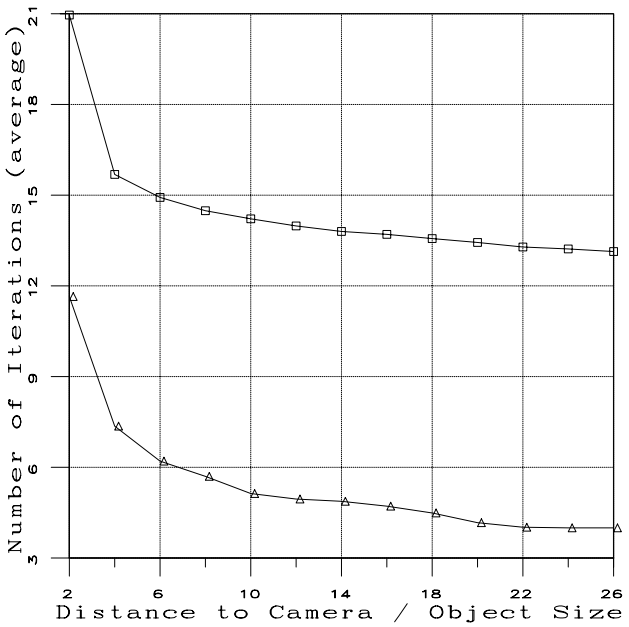


FIG. 4.14 - *Erreurs angulaires d'orientation en présence d'un bruit affectant la caméra et/ou l'image. Les courbes (—) correspondent aux algorithmes linéaires itératifs, les courbes (- - -) correspondent à l'algorithme non linéaire. a) bruit uniforme affectant les coordonnées du centre de l'image (u_c et v_c) ; b) bruit uniforme affectant les deux coefficients α_u et α_v ; c) bruit uniforme affectant les points images et les coordonnées (u_c et v_c) ; d) bruit uniforme affectant les points images et les 4 paramètres intrinsèques de la caméra.*

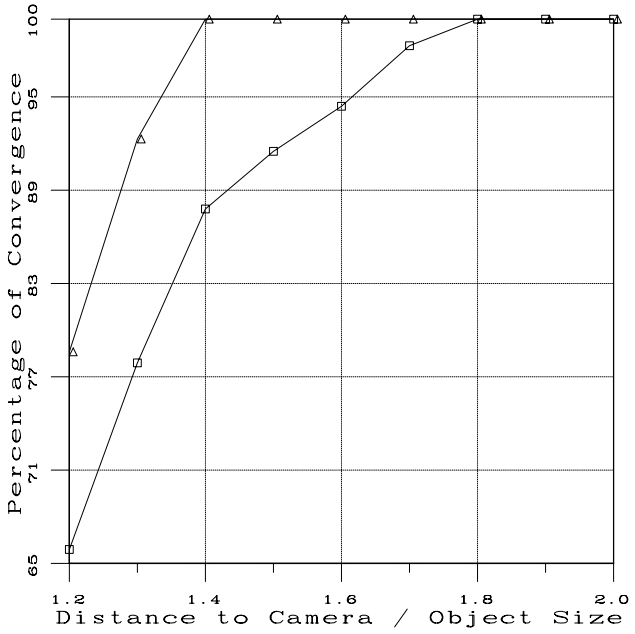


a) Eloignement de 23 degrés.

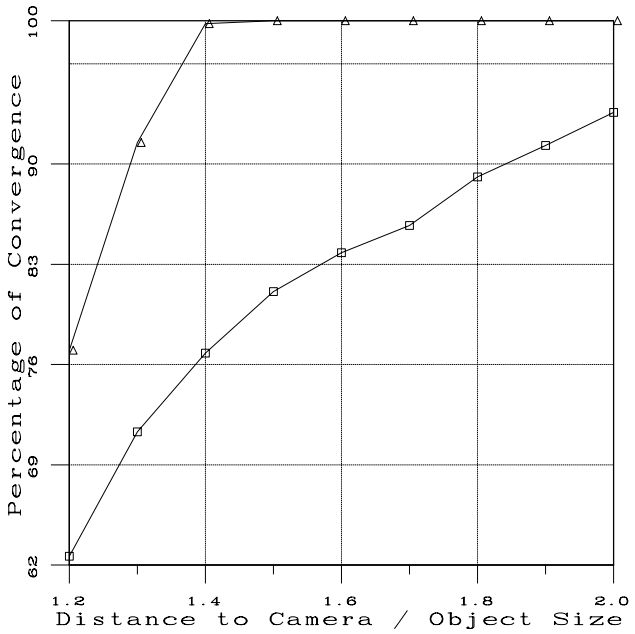


b) Eloignement de 30 degrés.

FIG. 4.15 - *Nombre d'itérations en fonction de la profondeur : a) pour un éloignement de 23 degrés, b) pour un éloignement de 30 degrés. Les courbes marquées par des carrés (□) correspondent à l'algorithme perspective faible et les courbes marquées par des triangles (△) correspondent à l'algorithme para-perspective.*



a) Eloignement de 30 degrés.



b) Eloignement de 35 degrés.

FIG. 4.16 - *Pourcentages de convergence en fonction de la profondeur: a) pour un éloignement de 30 degrés, b) pour un éloignement de 35 degrés. Les courbes marquées par des carrés (□) correspondent à l'algorithme perspective faible et les courbes marquées par des triangles (△) à l'algorithme para-perspective.*

4.10.2 Cas coplanaire

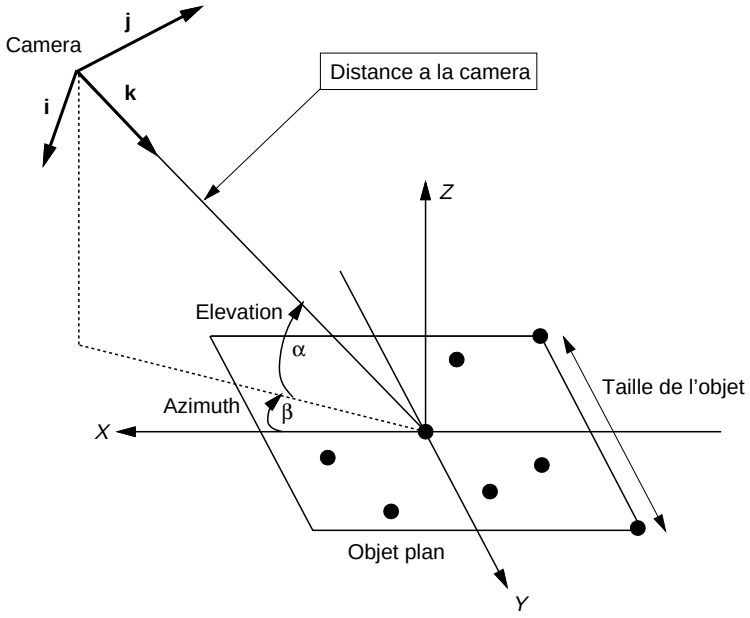


FIG. 4.17 - *élévation α et azimut β de la caméra dans les expériences.*

Dans ce paragraphe, nous évaluons la stabilité de l'algorithme itératif basé sur l'approximation *para-perspective* ainsi que celle de l'algorithme non linéaire (décrit dans le paragraphe 4.6) lorsque les points caractéristiques sont coplanaires. Par une démarche similaire à celle du paragraphe précédent, des images synthétiques sont créées, et les poses calculées sont comparées avec les poses réelles qui ont servi à calculer ces images. Nous faisons ces calculs pour un objet comprenant 8 points coplanaires distribués dans un carré de 10 cm, à trois distances de la caméra, moyennant pour chaque distance les erreurs obtenues pour 120 azimuts à intervalle régulier par rapport au plan de l'objet. Pour un objet plan, il est logique de définir les orientations de la caméra par rapport à l'objet en termes d' azimuts et d'élévations, et de comparer les erreurs des vues de face et rasantes. La caméra pointe toujours vers l'origine de l'objet qui constitue le point de référence. Elle est positionnée successivement à trois distances du point de référence: 4, 8 et 16 fois la dimension du carré contenant les points coplanaires. Pour chacune de ces distances, on considère 17 angles d'élévation, de 10 à 90 degrés. Ces élévations sont notées α sur la Figure 4.17. Pour la dernière élévation, la caméra fait face au plan de l'objet. Toutes les erreurs de pose sont représentées en fonction de ces élévations dans les diagrammes. Cette représentation permet de comparer les erreurs à différents angles de la caméra. La Figure 4.17 illustre aussi la définition de l'azimut β de la caméra.

Sur la Figure 4.18 sont regroupés les diagrammes des erreurs d'orientation et de position obtenues en moyennant les erreurs pour 120 angles d'azimut à intervalles réguliers autour de l'objet (3 degrés). Le bruit d'image est un bruit uniforme appartenant à l'intervalle $[-0.5, +0.5]$ pixel. Les courbes continues (—) correspondent à l'algorithme *para-perspective* alors que les courbes interrompues (- - -) correspondent à l'algorithme non linéaire. Les diagrammes de la Figure 4.18.a (respectivement 4.18.b et 4.18.c) corres-

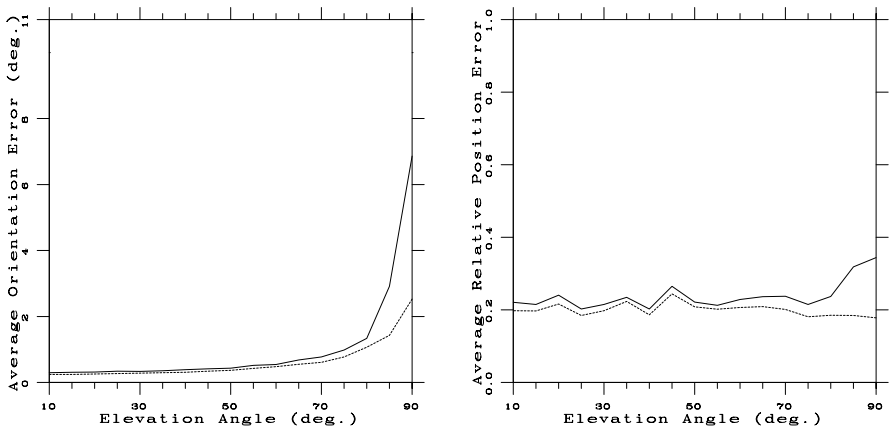
pondent à une distance qui est égale à 4 (8 et 16) fois la dimension de l'objet plan. Les diagrammes de gauche présentent les erreurs d'orientation, ceux de droite présentent les erreurs relatives de position.

On peut constater que les erreurs d'orientation peuvent être très importantes (10 degrés) dans le cas où la caméra fait face au plan de l'objet. En effet, toutes les rotations autour d'axes dans le plan de l'objet déplacent les points de l'objet dans des directions proches des directions des lignes de vue, et sont ainsi difficilement détectées puisqu'elles produisent peu de changements dans l'image. Par contre, en vue rasante, la plupart des rotations déplacent les points de telle sorte que leurs images produisent un grand changement de position dans le plan de l'image. On peut également constater que les calculs de position sont moins sensibles au bruit d'image que les erreurs d'orientation quand la caméra fait face au plan de l'objet. Notons que la précision obtenue avec l'algorithme linéaire itératif est comparable à celle de l'algorithme non linéaire. Ceci est dû au fait qu'à chaque pas, les inconnues du problème sont estimées en tenant compte des contraintes d'orthogonalité.

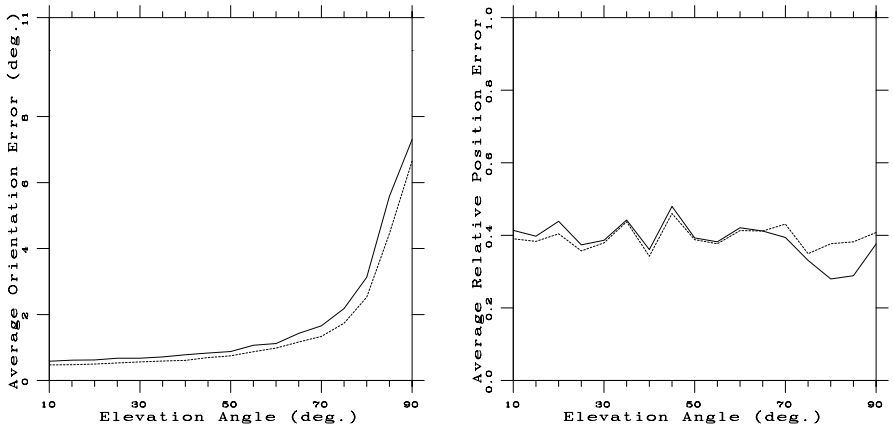
Nous venons de montrer que le calcul de la pose d'un objet plan fournit deux solutions. La distinction entre ces deux poses met en jeu la position relative entre cet objet et la caméra aussi bien que l'amplitude du bruit affectant les points images. On dit qu'une pose est acceptable si elle satisfait la définition suivante : on considère les écarts entre les points d'image réels et les projections des points caractéristiques pour cette pose ; la pose est acceptable si tous ces écarts sont inférieurs à l'amplitude du bruit d'image. Les diagrammes de la Figure 4.19 présentent la probabilité de trouver deux poses acceptables en fonction de l'angle de l'élévation. L'objet plan utilisé comprend 4 points coplanaires. Le protocole de calcul de ces pourcentages est le suivant : pour chacune des élévations on génère 120 images bruitées qui correspondent aux 120 azimuts et on calcule le nombre de fois pour lesquelles on obtient deux poses acceptables, et on divise ce nombre par le nombre total d'expériences (120). Ces pourcentages correspondent aux deux algorithmes linéaires itératifs puisque le point de référence appartient à l'axe optique.

Les diagrammes de gauche correspondent à un bruit uniforme de $\pm$ 0.5 pixel, ceux de droite à un bruit uniforme de $\pm$ 1 pixel. Les diagrammes du haut, du milieu et du bas correspondent aux distances égales à 4, 8 et 16 fois la dimension de l'objet plan.

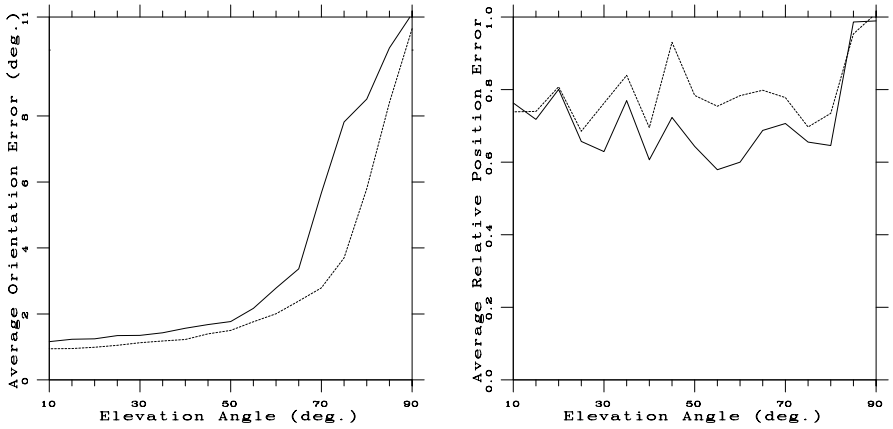
On voit sur ces diagrammes qu'on a plus de chances de trouver deux poses acceptables quand le rapport de la distance de la caméra à l'objet divisée par la projection de l'objet sur l'axe optique (l'inverse des ϵ_i) est élevé. En pratique, si l'objet plan a une position quelconque par rapport à la caméra et si l'on veut réduire la probabilité d'obtenir deux poses acceptables, on adopte la stratégie suivante : soit on augmente la précision avec laquelle on extrait les points images, soit on augmente le nombre de ces points pour compenser le bruit d'image. Notons enfin qu'aux distances moyennes la probabilité d'obtenir deux poses acceptables devient importante lorsque le plan de l'objet a une orientation presque parallèle au plan de l'image (pour l'algorithme *perspective faible*) et une orientation presque perpendiculaire au vecteur de translation (pour l'algorithme *para-perspective*).



a) Distance à la caméra = 4 × Taille de l'objet.

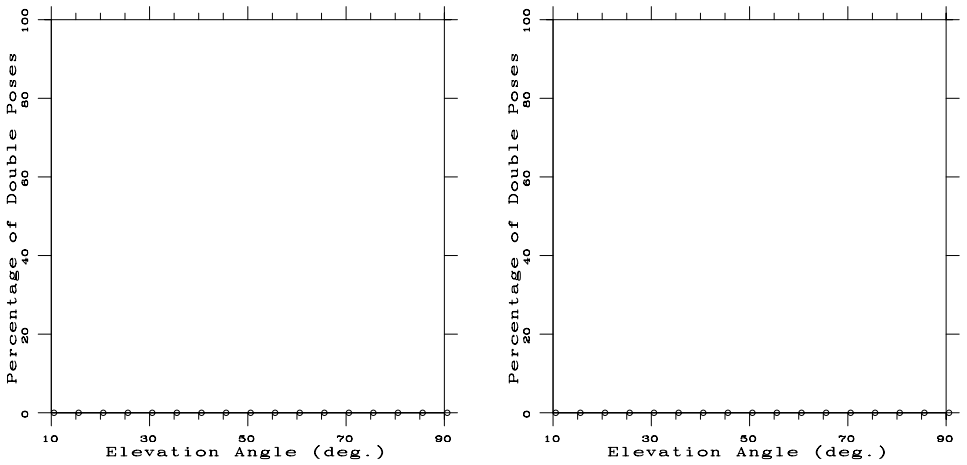


b) Distance à la caméra = 8 × Taille de l'objet.

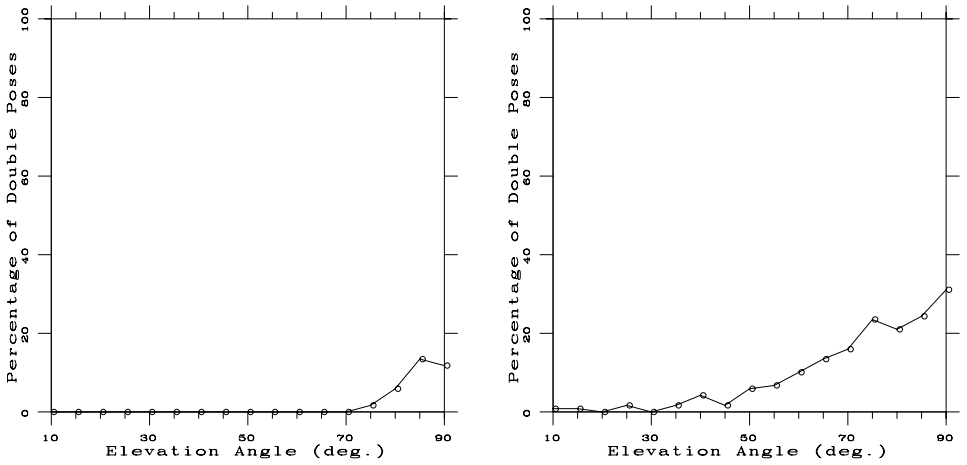


c) Distance à la caméra = 16 × Taille de l'objet.

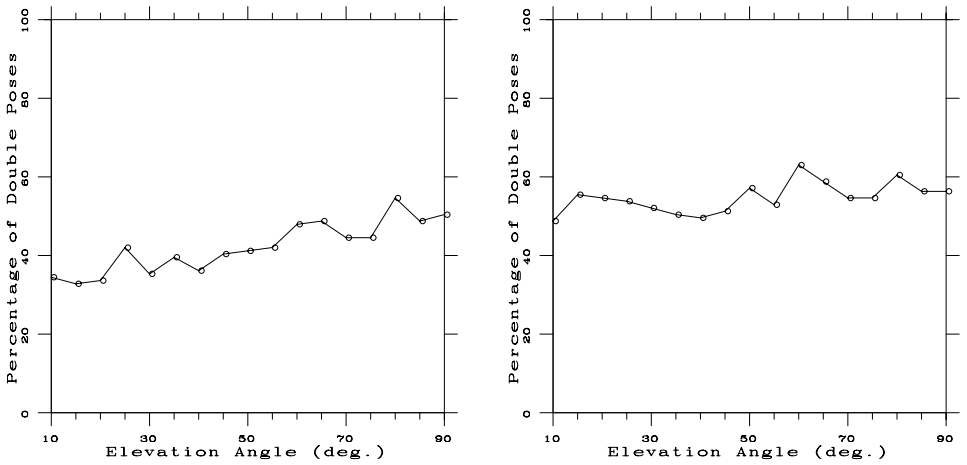
FIG. 4.18 - *Erreurs d'orientation et de position en fonction de l'angle d'élévation pour 8 points coplanaires avec un bruit d'image uniforme dans l'intervalle* $[-0.5, +0.5]$ *pixel. Les courbes continues (—) correspondent à l'algorithme para-perspective, les courbes interrompues (- - -) à l'algorithme non linéaire. Ces résultats sont des moyennes pour 120 azimuts autour de l'objet.*



a) Distance à la caméra = 4 × Taille de l'objet.



b) Distance à la caméra = 8 × Taille de l'objet.



c) Distance à la caméra = 16 × Taille de l'objet.

FIG. 4.19 - *Probabilités d'obtenir deux poses acceptables en fonction des angles d'élévation, pour 4 points coplanaires. Les diagrammes de gauche correspondent à un bruit uniforme de ± 0.5 pixel, ceux de droite à un bruit uniforme de ± 1 pixel.*

4.11 Résultats expérimentaux

Nous allons à présent tester les trois méthodes de localisation caméra/objet sur des scènes réelles.

1. **Scène 1 :** sur la Figure 4.20 sont présentés les résultats de localisation d'un objet polyédrique à partir des correspondances points/points. Les points de l'image sont les images des sommets de l'objet dont on veut déterminer la pose. La Figure 4.20.a illustre l'image fournie par la caméra. La Figure 4.20.b illustre les jonctions qui ont été obtenues par les intersections des contours droites. La Figure 4.20.c montre les 11 points caractéristiques utilisés par les algorithmes de pose. Ces 11 points sont marqués par des croix. La Figure 4.20.d représente le modèle CAO de l'objet polyédrique (représentation fil de fer) ainsi que les points objets correspondant aux points images. Les Figures 4.20.e et 4.20.f représentent la projection du modèle à la pose calculée à la première itération et à la troisième itération respectivement (algorithme *para-perspective*). Le tableau 4.1 présente le nombre d'itérations et le temps de calcul en fonction du nombre de correspondances pour les deux méthodes (*para-perspective* et non linéaire). La méthode non linéaire est initialisée par la solution de la méthode *para-perspective* obtenue à la première itération. Bien que la méthode non linéaire soit initialisée par une solution affine, nous obtenons un gain en temps de calcul qui peut atteindre 100 si nous utilisons la méthode linéaire itérative.

2. **Scène 2 :** sur la Figure 4.21 sont présentés les résultats de localisation d'un autre objet polyédrique à partir de 10 correspondances points/points. Les Figures 4.21.d et 4.21.e représentent la projection du modèle à la pose calculée à la première itération et à la troisième itération respectivement (algorithme *para-perspective*).

3. **Scène 3 :** la Figure 4.22 et les tableaux 4.2 et 4.3 illustrent les résultats de localisation d'un objet plan par les trois méthodes de localisation (présentées dans ce chapitre). Les points caractéristiques de l'objet sont les centres de gravité de quatre cibles circulaires et coplanaires. Les points images sont donnés par les centres de gravité des tâches elliptiques représentant les images des cibles circulaires (voir Figure 4.22.b). Notons que la projection du centre d'une cible circulaire ne correspond pas au centre de l'ellipse image de ce cercle. Cependant, on peut considérer que cette erreur est négligeable puisqu'elle est de l'ordre du dixième de pixel et tend vers zéro avec la croissance de la distance par rapport à la caméra. L'extraction des primitives images (4 tâches elliptiques) a été réalisée d'une façon automatique en utilisant les caractéristiques décrivant les composants convexes 2D. Nous avons utilisé l'excentricité [BB82]). Comme les quatre tâches circulaires ont le même rayon et appartiennent au même plan, les excentricités de leurs images doivent être proches. Ce qui peut constituer un outil permettant la reconnaissance de ces quatre tâches dans l'image. De même, la mise en correspondance points images/points objets a été effectuée d'une façon automatique en utilisant des connaissances préalables sur les positions relatives des cibles. Le tableau 4.2 présente les erreurs résiduelles de projection pour chaque point. Cette erreur représente la distance euclidienne en pixels entre le centre de l'ellipse et le point image projeté à la pose estimée. Le

	Para-perspective		Non linéaire	
N. de points	N. d'itérations	temps CPU (ms)	N. d'itérations	temps CPU (ms)
4	5	0.87	114	87.5
5	5	0.95	67	58.7
6	6	1.02	66	68.0
7	5	1.06	66	72.5
8	6	1.12	58	70.5
9	6	1.33	82	108.6
10	6	1.38	78	117.4
11	6	1.47	78	123.6

TAB. 4.1 - *Une comparaison entre la méthode linéaire itérative (para-perspective) et la méthode non linéaire en fonction du nombre de correspondances point/point. On peut remarquer le grand nombre d'itérations nécessaires à la méthode non linéaire lorsqu'on a 4 correspondances. L'ordinateur utilisé est un Sparc 10/51.*

tableau 4.3 présente le comportement de la convergence de chaque algorithme. La première ligne représente le nombre d'itérations, la seconde ligne fournit le temps de calcul. La méthode non linéaire a été initialisée par la solution de l'algorithme *para-perspective* calculée à la première itération.

Comme l'extraction des coordonnées 2D est assez précise, les trois méthodes ont quasiment la même précision. En ce qui concerne la convergence, le meilleur comportement a été obtenu avec l'algorithme itératif basé sur l'approximation para-perspective.

4. **Scène 4 :** sur la Figure 4.23 sont présentés les résultats de localisation du même objet polyédrique associé à la scène 2. La localisation a été obtenue en utilisant 13 correspondances droite 2D/droite 3D et d'une correspondance point/point. La Figure 4.23.b représente la projection du modèle à la pose calculée à la cinquième itération de l'algorithme *para-perspective*.

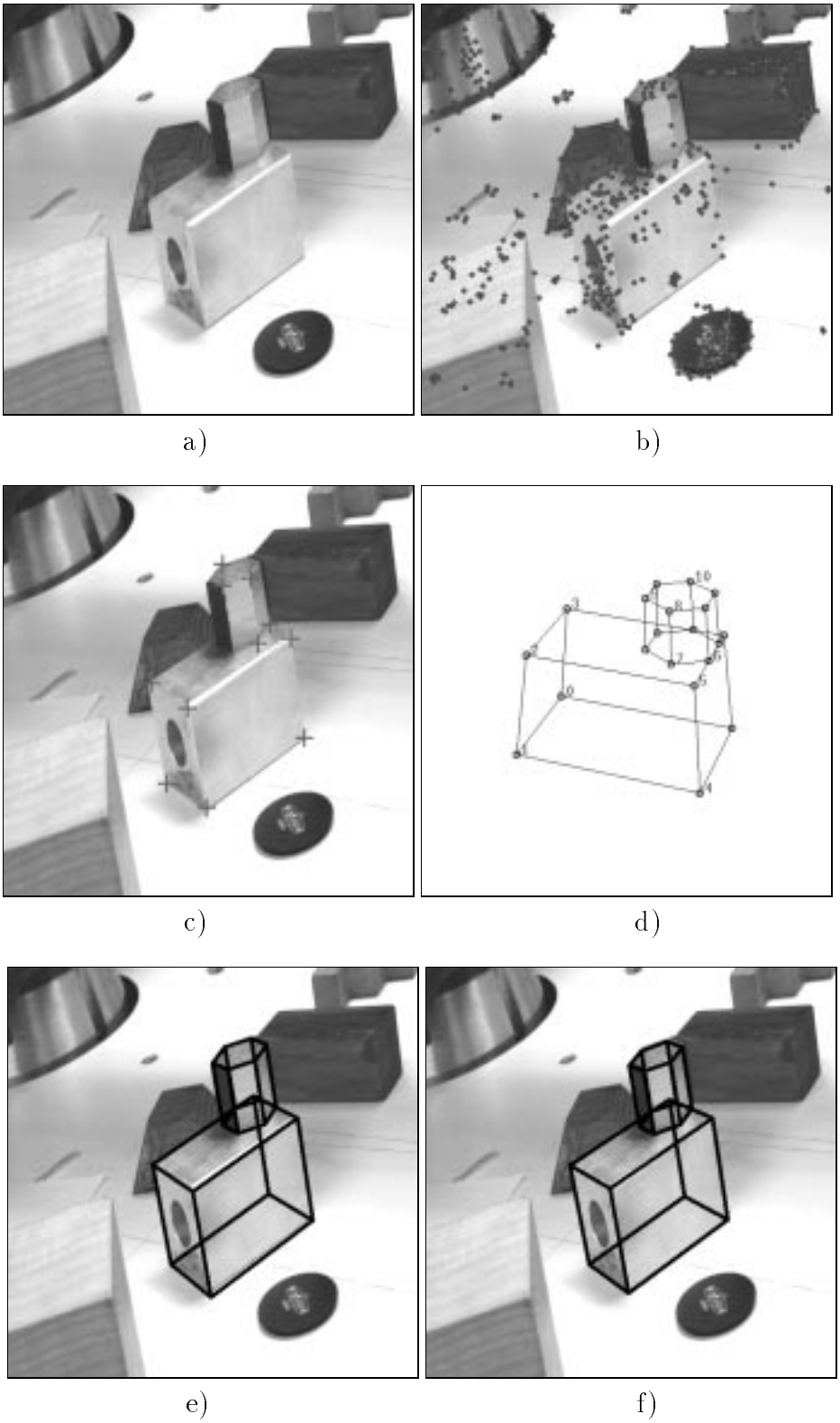
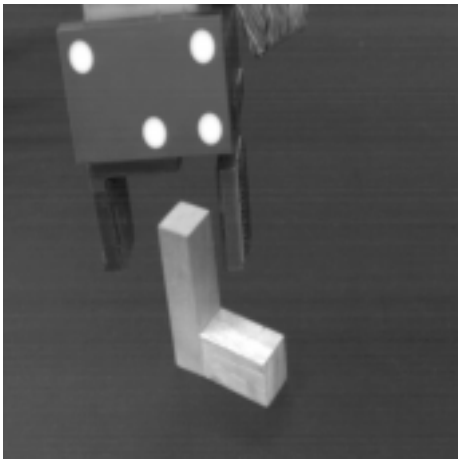
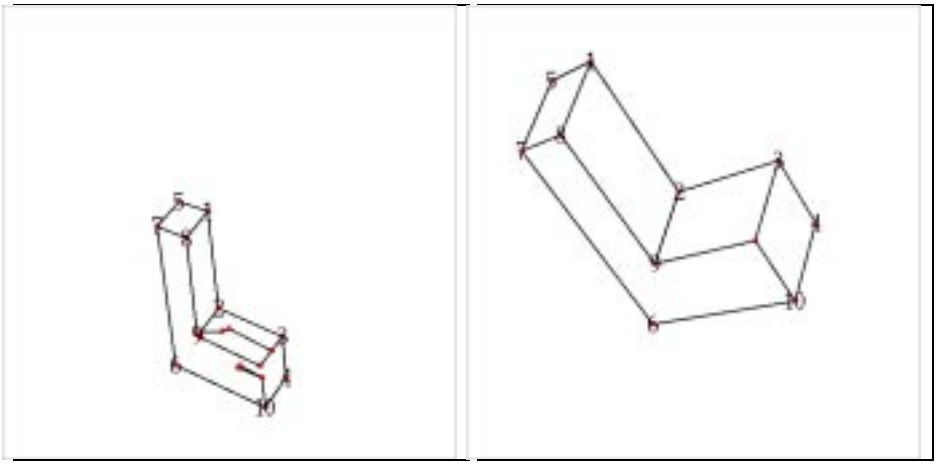


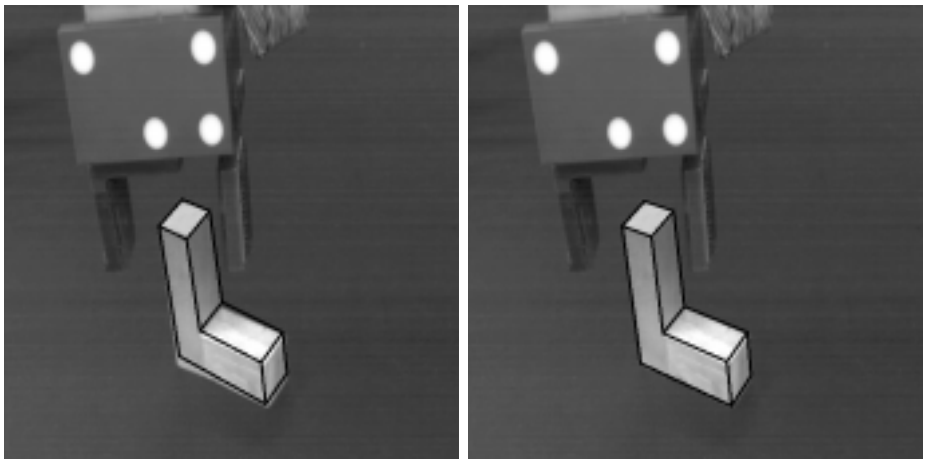
FIG. 4.20 - *Scène 1 : Localisation d'un objet polyédrique à partir de 11 correspondances point/point.*



a)

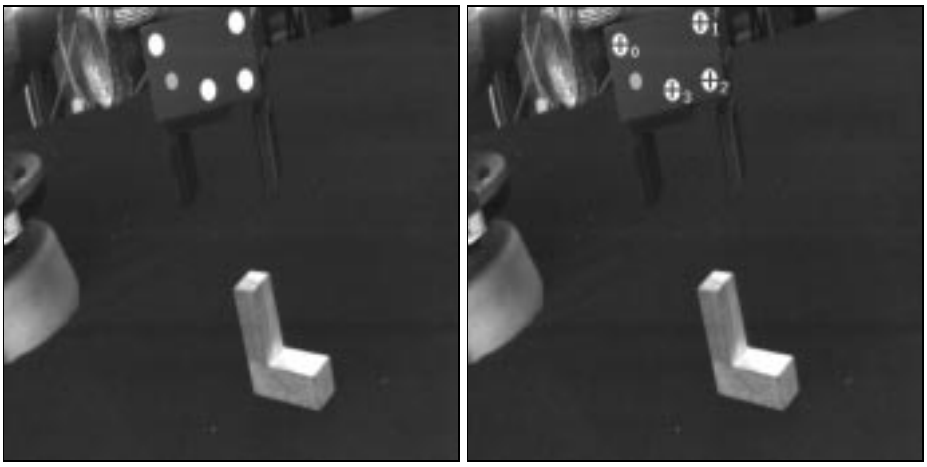


b) c)



d) e)

FIG. 4.21 - *Scène 2 : Localisation d'un objet polyédrique à partir de 10 correspondances point/point. La mise en correspondance s'effectue d'une manière automatique.*



a) b)

FIG. 4.22 - *Scène 3 : localisation d'un objet plan dont les points caractéristiques sont les centres de 4 cibles circulaires. La mise en correspondance s'effectue d'une manière automatique.*

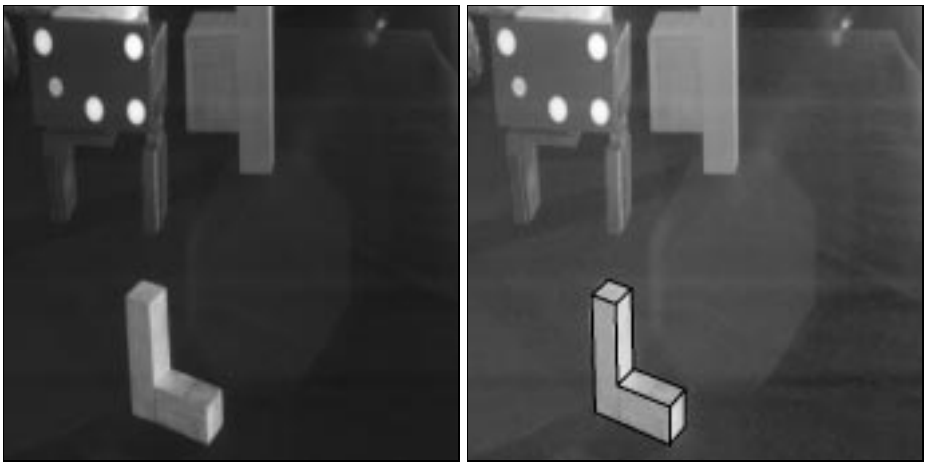
	Erreur résiduelle (pixels)		
point	*perspective faible*	*para-perspective*	*non linéaire*
point 0	0.0	0.0	0.018
point 1	0.019	0.019	0.011
point 2	0.040	0.040	0.034
point 3	0.040	0.040	0.040
moyenne	0.025	0.025	0.026

TAB. 4.2 - *Comparaison des erreurs résiduelles des trois méthodes de localisation d'un objet plan (4 points coplanaires).*

	Convergence		
	perspective faible	*para-perspective*	*non linéaire*
nombre d'itérations	20	10	91
temps CPU[a] (ms)	4.3	2.9	65.3

TAB. 4.3 - *La convergence des trois méthodes de localisation (4 points coplanaires).*

[a] Sparc 10/51



a) b)

FIG. 4.23 - *Scène 4: localisation d'un objet polyédrique à partir de 13 correspondances droite 2D/droite 3D et d'une correspondance point/point.*

4.12 Conclusion

Ce chapitre a été consacré à la localisation caméra/objet. De nombreuses méthodes existent pour résoudre ce problème. Deux critères orientent les recherches associées à ce problème : i) la robustesse de la solution en présence du bruit de mesures et ii) le temps de calcul. Nous avons vu qu'on peut utiliser les techniques de l'algèbre linéaire pour déterminer, d'une manière itérative, les paramètres de la pose correspondant au modèle perspectif associé à la caméra. Deux méthodes sont possibles : la première est basée sur l'approximation de la projection par une projection perspective faible (proposée par DeMenthon), la deuxième (notre méthode) suit cette idée de base mais elle utilise la projection paraperspective. Cette deuxième méthode permet de s'affranchir des limitations rencontrées dans la première. Autrement dit, elle permet d'obtenir un comportement de convergence indépendant de la position de l'objet par rapport à l'axe optique. De plus, la convergence de la deuxième méthode est mieux garantie et plus rapide que celle de la première.

Nous avons démontré que la précision de ces deux approches est comparable à celle des méthodes numériques (surtout lorsque les scènes sont planes). La différence de précision peut s'amenuiser si le nombre des primitives est assez grand ou leur extraction est robuste. Les deux approches sont bien adaptées aux applications temps réel (4 à 6 itérations pour des distances réalistes). Toutefois et d'une manière générale, la deuxième méthode l'est davantage.

Il est à noter que dans les applications où l'aspect temps réel n'existe pas, on peut envisager une coopération entre les deux approches linéaires itératives et les approches numériques. Ainsi, la solution fournie par les approches linéaires itératives est utilisée pour initialiser l'approche numérique (non linéaire).

Chapitre 5

Asservissement visuel

Dans ce chapitre, nous allons aborder certains aspects de l'utilisation de la vision dans la commande des robots en boucle fermée. Nous montrons l'intérêt des capteurs *extéroceptifs* qui sont capables de fournir des informations sur l'environnement d'un robot. Nous présentons quelques aspects de l'état de l'art dans le domaine du contrôle visuel des robots. Nous nous focalisons ensuite sur la notion de l'interaction capteur/environnement. Cette interaction relie les variations des informations visuelles aux mouvements de la caméra (ou du robot). Ensuite, nous proposons une approche basée sur la méthode *commande référencée capteur* et qui permet à une caméra fixe de contrôler en temps réel le mouvement 3D d'un robot manipulateur dans son environnement. Nous montrons que les deux aspects *calibrage capteur/robot* et *localisation caméra/objet*, tels qu'ils sont présentés dans les chapitres 2 et 4, peuvent intervenir dans cette approche. Nous montrons que l'utilisation de la *localisation caméra/objet* dans la boucle de commande va améliorer la performance de celle-ci. A la fin de ce chapitre, nous présentons les résultats qui ont été obtenus, soit en simulation, soit sur site expérimental, et qui portent sur la réalisation de la tâche de positionnement de l'effecteur d'un robot par rapport à son environnement. Dans tous les cas, nous nous plaçons dans le cadre le plus général où le robot a 6 degrés de liberté.

5.1 Introduction

Un robot est un système mécanique capable de se mouvoir (pour un robot mobile) ou de déplacer son effecteur (pour un robot manipulateur) afin de réaliser une tâche qui lui est assignée. Une cellule robotisée a deux composantes essentielles :

- La perception, qui permet de gérer les relations entre le robot et son environnement. Les organes de perception sont des capteurs dits *capteurs proprioceptifs* lorsqu'ils mesurent l'état interne du robot (positions et vitesses des articulations) et *extéroceptifs* lorsqu'ils recueillent des informations sur l'environnement (détection de présence, mesure de distance, vision artificielle). Leur sont associés des systèmes de traitement spécifiques.
- La commande, qui synthétise les consignes des asservissements pilotant les actionneurs. A partir de la fonction de perception et des ordres de l'utilisateur, elle permet

d'engendrer les mouvements du robot.

Cependant, les capteurs *proprioceptifs* fournissent des renseignements sur l'état interne du robot et uniquement sur cet état. Ils ne tiennent pas compte des problèmes dûs à l'interaction avec l'environnement. Ceux-ci ne permettent donc pas de situer précisément le robot. L'utilisation exclusive des capteurs *propriocetifs* paraît extrêmement limitative : en effet, les tâches ne peuvent alors s'exprimer que dans l'espace articulaire ou opérationnel. Ainsi, seules les tâches de suivi de trajectoires prédéfinies ou de positionnement sont réalisables.

Il faut cependant souligner les problèmes de précision que l'on peut rencontrer en pratique pour passer de l'espace articulaire dans lequel sont exprimées les informations fournies par les capteurs *proprioceptifs*, à l'espace opérationnel. En effet, cette étape nécessite l'emploi du modèle géométrique du robot. Ce modèle est généralement entaché d'erreurs en raison des problèmes complexes liés à son identification. Par conséquent, la situation de l'effecteur du robot est une situation estimée et même souvent biaisée. Les tâches exprimées dans l'espace opérationnel peuvent ainsi être correctement réalisées du point de vue "commande" (c'est-à-dire que la situation mesurée est égale à la situation souhaitée) sans être réalisées effectivement : aucun réel retour sur la situation exacte de l'effecteur n'est possible en utilisant les capteurs *proprioceptifs*.

De plus, les problèmes de frottement et de roulement avec ou sans glissement interdisent à un robot mobile d'effectuer précisément une trajectoire prédéfinie, et même d'effectuer plusieurs fois la même trajectoire. Ces problèmes de répétabilité sont beaucoup moins sensibles pour les robots manipulateurs. Ils subsistent cependant car ils sont directement liés au bon fonctionnement et à la précision plus ou moins grande des capteurs *proprioceptifs*.

Ainsi, quelle que soit la performance de la loi de commande utilisée, il est impossible de connaître exactement la position d'un robot mobile ou la situation de l'effecteur d'un robot manipulateur. Même dans le cas où le robot évolue dans un univers parfaitement connu, les capteurs *proprioceptifs* ne permettent pas de le situer précisément dans cet univers. Il paraît alors important d'utiliser des capteurs *extéroceptifs* capables de fournir des informations, soit sur l'environnement du robot, soit sur la situation du robot dans cet environnement. Le champ d'application des robots s'en trouve d'ailleurs grandement élargi. Ainsi, l'autonomie et l'efficacité d'un robot vont être augmentées grâce à l'emploi des capteurs *extéroceptifs*.

On distingue plusieurs types de capteurs *extéroceptifs* :

- les capteurs tactiles qui préviennent d'un contact avec un élément extérieur,
- les capteurs proximétriques qui détectent la présence d'un objet situé à une distance inférieure au mètre,
- les capteurs télémétriques pour des objets situés à une distance plus importante,
- les capteurs visuels qui permettent de percevoir l'environnement.

Il est très intéressant d'intégrer les informations fournies par les capteurs *extéroceptifs* dans des lois de commande en boucle fermée sur ces informations. Ainsi, les fonctions de tâches peuvent s'exprimer directement dans l'espace du capteur et non plus dans les espaces articulaire ou opérationnel. Elles sont spécifiées sous la forme d'une relation entre le robot et son environnement. La régulation de ces fonctions de tâches permet d'obtenir la situation souhaitée du robot dans cet environnement. Il faut noter qu'on peut faire coopérer plusieurs types de capteurs à la fois. Par exemple, dans [NH93] on trouve une utilisation d'une caméra et d'un capteur ultrasonique pour la localisation et la reconnaissance d'objets.

5.2 Etat de l'art

Dans la suite, nous nous intéressons aux capteurs visuels qui ont l'avantage de fournir des informations très riches sur l'environnement. L'emploi de la vision avec des robots a une longue histoire. Toutefois, les récents développements dans le domaine des capteurs de vision et du traitement d'images permet d'espérer que l'utilisation des données visuelles dans une commande à boucle fermée n'est plus utopique. Actuellement, les systèmes de vision sont intégrés avec les systèmes de programmation du robot. Les possibilités vont d'un simple traitement d'image binaire aux systèmes d'extraction de contours et de primitives. Typiquement, le temps de traitement d'image est de l'ordre : 0.1–1 seconde. Ainsi, le contrôle visuel est la fusion des résultats de nombreux domaines de recherche : traitement d'images, la cinématique, la dynamique, la théorie de contrôle et le calcul temps-réel.

Le contrôle visuel de robots a plusieurs points en commun avec la vision active [AWB88] [NK94] [NPK93] [DAO94] [YAMU94]. Cette dernière propose d'établir une action réflexe au niveau du capteur visuel à partir des informations visuelles comme le contrôle du regard [Mes94] et la télésurveillance [SBC94]. Ainsi, l'approche fondamentale de la vision active n'est pas l'interprétation d'une scène mais elle consiste à diriger l'attention vers une partie de la scène. Ainsi, grâce à la vision active on peut lever les ambiguïtés et augmenter la précision du système de vision. De plus, le contrôle visuel peut être impliqué dans la reconstruction d'une scène 3D à partir d'une séquence d'images. Ainsi, avec plusieurs images, on peut reconstruire une scène en 3 dimensions grâce à des techniques telles que :

- la stéréoscopie avec deux ou trois caméras [AL87] [LTM91],
- ou la vision active en utilisant une caméra mobile [PK94] [SK94] [Bou94] [Bou93].

L'asservissement sur des informations visuelles peut se mettre en œuvre de deux façons :

- la première est basée sur un asservissement en position de la caméra ou du robot par rapport à son environnement,
- la seconde, plus récente, est basée sur une régulation dans l'image (*asservissement visuel*).

Les premiers travaux ayant abouti à ce formalisme sont dûs à Weiss et Sanderson [SW80].

5.2.1 Asservissement en position

Cette approche consiste à positionner la caméra ou le robot par rapport à un objet. La situation courante vis à vis de l'objet est obtenue en interprétant les informations visuelles de l'objet qui sont extraites de l'image. Cette estimation est indispensable afin d'atteindre la position voulue. Le déplacement de la caméra (ou du robot) est ensuite assuré par un classique asservissement en position dans l'espace cartésien (voir Figures 5.1 et 5.2).

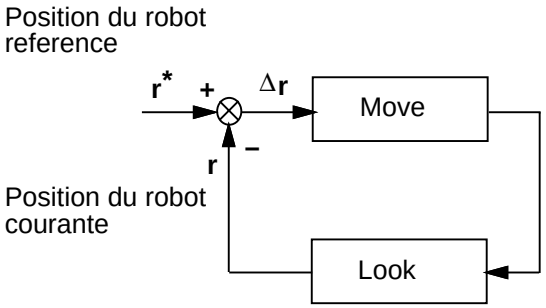


FIG. 5.1 - *Asservissement en position.*

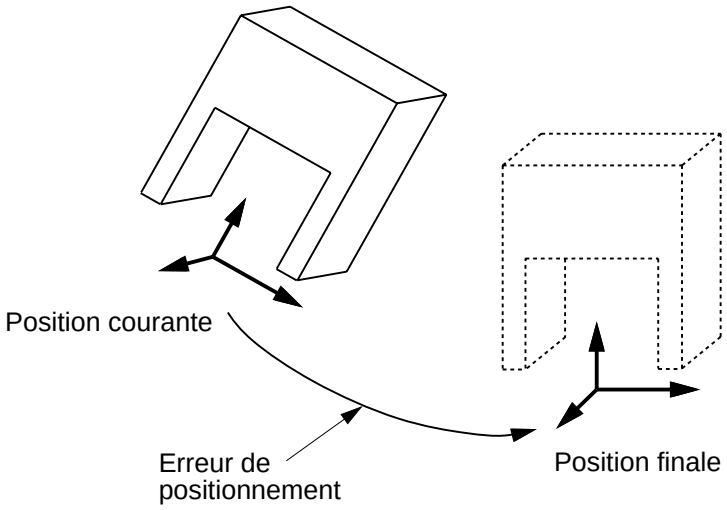


FIG. 5.2 - *Un exemple des positions courante et finale d'un objet mobile.*

Il existe deux manières de réaliser cet asservissement en position :

1. "*Static Look and Move*" : la phase d'acquisition d'images avec déduction de la situation courante est enchaînée séquentiellement avec le déplacement du robot. La simplicité de ce processus a permis son utilisation dès la fin des années 1970 dans les premières applications robotiques utilisant la vision.

2. "*Dynamic Look and Move*" : les deux étapes sont effectuées simultanément. Cette technique a l'avantage de pouvoir suivre des objets en mouvement puisque les informations visuelles sont prises en compte dès qu'elles sont opérationnelles.

Théoriquement, pour atteindre une position désirée, une seule itération est nécessaire, mais dans la pratique il faut tenir compte des erreurs dûes à :

- l'extraction de primitives dans l'image,
- la modélisation de la caméra,
- la modélisation du robot.

Des exemples de l'utilisation de cette approche sont les suivants :

- Schrott [Sch92] utilise un système caméra-pince pour saisir un objet. La caméra embarquée s'approche de l'objet par des étapes séquentielles. Chaque étape consiste en un asservissement en position de la caméra. Une fois ces étapes réalisées, la pince du robot est commandée de façon à occuper la position de la caméra.
- Hollinghurst & Cipolla [HC93] utilisent deux caméras fixes pour positionner la pince d'un robot par rapport à un objet. Ils adoptent le modèle perspectif faible pour la tête stéréoscopique ainsi obtenue. Dans la commande, ils utilisent l'intégrale de l'erreur des position et orientation.
- Wijesoma et al. [WWR93] emploient une caméra fixe pour positionner un robot planaire. Les auteurs utilisent la vision en temps-réel comme retour au contrôleur articulaire ou cartésien. Ils évitent l'utilisation des capteurs articulaires.
- Zheng et al. [ZFL93] utilisent une caméra mobile, qui est indépendante du robot asservi, pour contrôler les 4 degrés de liberté de celui-ci afin de saisir et déplacer un objet 3D.
- Allen et al. [ATYM93] emploient deux caméras fixes pour visualiser le mouvement d'un train en modèle réduit. L'estimation de sa trajectoire et de sa vitesse est réalisée à l'aide d'une architecture pipeline et dans le but de saisir le train en mouvement à l'aide d'un bras manipulateur.
- Kay & Lee [KL91] placent deux caméras CCD sur le bras manipulateur. L'estimation de la position 3D de l'objet tient compte du bruit sur la mesure mais aussi du bruit sur le positionnement du robot.
- Dans les travaux de Skofteland [Sko91], l'effecteur est muni de deux caméras pour faire de la stéréoscopie, des capteurs de force, et des capteurs télémétriques laser, ceci afin d'effectuer la saisie d'un objet polyédrique dans l'espace 3D. Ce travail n'en est qu'à l'étape de la simulation.

5.2.2 Asservissement visuel

Avec cette approche, on cherche à atteindre un motif dans l'image et non plus à contrôler une situation entre la caméra et l'objet (voir Figures 5.3 et 5.4). Ainsi on supprime la phase de reconstruction d'un modèle 3D et on élimine du même coup les erreurs sur l'estimé de la situation. Notons que grâce à cette approche on peut supprimer les erreurs dûes au modèle de la caméra. Seule l'extraction de primitives dans l'image est une source d'erreur (en supposant que le motif final est correctement calculé). Les tâches réalisables par l'approche asservissement en position sont également réalisables par l'approche asservissement visuel. Des exemples de l'utilisation de cette approche sont les suivants :

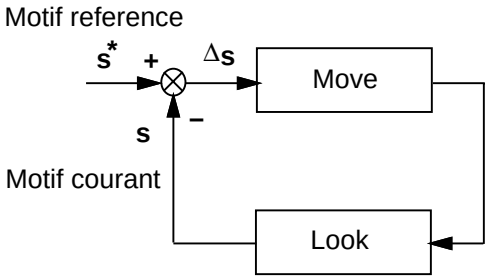


FIG. 5.3 - *Asservissement visuel.*

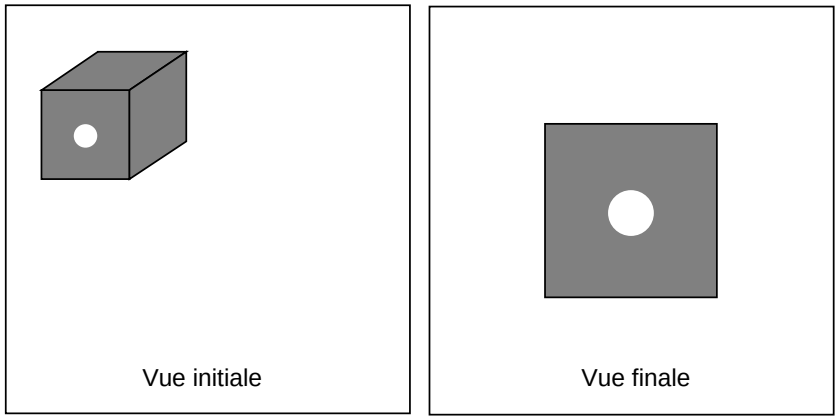


FIG. 5.4 - *Exemple des vues initiale et désirée d'un cube.*

– Les travaux présentés par Espiau et al. [ECR92] [Cha90] [CRE91] ont été portés sur :

1. la modélisation entre les variations d'un signal et les mouvements de la caméra. Cette modélisation est faite sous la forme d'un torseur d'interaction calculé pour des primitives géométriques paramétrables dans l'image,
2. la classification des tâches spécifiées comme des liaisons virtuelles entre la caméra et la scène,
3. l'intégration du problème dans la commande référencée capteur.

A titre de démonstration, un manipulateur grâce à une caméra située sur son bout utilise les techniques d'asservissement visuel pour pointer un objet en mouvement et réaliser des tâches de positionnement vis à vis d'une cible.

- Maru et al. [MKY+93] utilisent deux caméras embarquées sur un bras manipulateur. La principale différence avec les travaux présentés dans [ECR92] est que la matrice d'interaction est mise à jour grâce aux informations fournies par la tête stéréoscopique sous forme d'une disparité. Ils supposent que les deux caméras ont la même orientation et que leurs axes optiques sont perpendiculaires à la ligne de base.

- Hager [Hag95] propose une méthode de calcul des consignes 2D par utilisation des invariants projectifs 2D et 3D. Dans [HCM95] [HGH94] et [Hag94], les auteurs utilisent une tête stéréoscopique fixe pour contrôler la position relative entre les objets. Ils appliquent leur méthode à la tâche d'insertion d'une disquette dans le driver d'un ordinateur.

- Hosoda & Asada [HA94] proposent une méthode d'estimation de la matrice d'interaction qui relie les variations des signaux 2D aux mouvements articulaires d'un robot. Cette estimation ne nécessite pas une connaissance *a priori* du modèle du système caméra-robot. Le test de cette étude reste limité à l'asservissement de trois degrés de liberté du robot.

- Les travaux de Feddema et al. [FLM93] ont porté sur la poursuite d'objets réalisée par asservissement visuel avec une caméra embarquée sur l'effecteur du robot. Les mouvements du robot sont déduits à partir des trajectoires 2D désirées. On y trouve également une discussion sur le choix de primitives 2D permettant une poursuite robuste.

- Papanikolopoulos et al. [PKK93] [PKK91] [PNK94] ont effectué une étude comparative de différentes modes de contrôle qui sont :

  1. régulation proportionnelle-intégrale (PI),
  2. boucle fermée à placement de pôles,
  3. version modifiée du contrôle LQG (*Linear Quadratic Gaussien*) où le vecteur de perturbation et le vecteur d'état sont estimés par un filtre de Kalman.

- Hashimoto et al. [HKEK91] utilisent aussi les équations du flot optique pour calculer la matrice d'interaction pour n points de l'image. La commande est faite par un retour d'état. L'expérimentation consiste à suivre la trajectoire 2D d'un objet. L'inconvénient majeur d'utiliser des points comme amer dans l'image est qu'ils fournissent, dans certains cas, des informations bruitées en comparaison à des primitives géométriques plus structurées telles que les droites.

- Dans les travaux de Bien et al. [BJP93], l'information visuelle extraite dans l'image peut être la surface, les moments centraux ... Dans le cas où le nombre d'informations visuelles est égal au nombre de degrés de liberté du robot la matrice d'interaction peut posséder des configurations singulières. Les auteurs proposent d'utiliser des informations redondantes pour résoudre ce problème.

– Corke [Cor93a] décrit quelques aspects matériels et logiciels d'un système capable de réaliser un asservissement visuel à une cadence de 50 Hz. Cette cadence élevée simplifie quelques stratégies de contrôle (par exemple la prédiction et la génération des trajectoires explicites peuvent être éliminées). Dans cette étude on trouve également une analyse des effets dynamiques de la boucle fermée.

Enfin, notons que pour les deux types d'asservissement, la nature de la tâche robotique impose, soit l'utilisation d'une caméra embarquée sur le robot asservi, soit l'utilisation d'une caméra indépendante de ce robot.

5.3 Interaction capteur/environnement

5.3.1 Définitions

Soit e l'espace euclidien à 3 dimensions, l'espace vectoriel associé étant $\mathbb{R}^3$. L'espace de configuration des corps rigides et des repères est le groupe de Lie des déplacements de e, appelé SE_3 (*Special Euclidian Group*), isomorphe à $\mathbb{R}^3 \times SO_3$ où SO_3 est le groupe des rotations. Un élément de SE_3 définit une situation $\mathbf{r}$. L'espace tangent à SE_3 en l'identité, noté se_3, est une algèbre de Lie isomorphe à l'algèbre de Lie des champs équiprojectifs de e dans $\mathbb{R}^3$, ce qui signifie que tout élément de se_3 n'est autre que le traditionnel torseur cinématique.

Classiquement, un torseur $\mathbf{T}$ est aussi défini par son vecteur Ω et son champ antisymétrique $\mathbf{V}$. La connaissance du vecteur Ω et du champs $\mathbf{V}$ en un point O détermine le champ en tout point M par :

$$\mathbf{V}(M) = \mathbf{V}(O) + \Omega \times \overrightarrow{OM}$$

Exprimé dans un repère un torseur est un vecteur de $\mathbb{R}^6$ et nous appelons coordonnées en O du torseur $\mathbf{T}$ le couple de vecteurs $[\Omega,\ \mathbf{V}(O)]$.

Le produit scalaire de deux torseurs $\mathbf{T}_1$ et $\mathbf{T}_2$ en un point O est donné par :

$$\mathbf{T}_1 \bullet \mathbf{T}_2 = \Omega_1 \mathbf{V}_2(O) + \Omega_2 \mathbf{V}_1(O)$$

Notons que ce produit ne dépend pas du choix de O.

5.3.2 Le torseur d'interaction

Dans ce paragraphe nous présentons la notion du Jacobien d'image. Cette entité quantifie la relation différentielle entre les variations des informations visuelles fournies par un ou plusieurs capteurs et les mouvements du robot engendrant ces variations.

On définit une information visuelle $\mathbf{s}$, également appelée signal capteur, par la donnée d'une application de SE_3 dans $\mathbb{R}^k$ où k représente la dimension du vecteur $\mathbf{s}$. Cette définition implique que la valeur de $\mathbf{s}$ ne dépend que de la situation entre la caméra et son environnement. Ainsi on considère que seules des modifications de nature géométrique sont susceptibles de faire varier la valeur d'une information visuelle. On sait que la caméra est indépendante du robot qui se trouve dans le champ visuel de cette caméra. Les

déplacements de la pince du robot ou des objets embarqués sur cette pince sont réalisés à l'aide des différents axes constituant le robot. La situation de la pince ne dépend que de la valeur des coordonnées articulaires $\mathbf{q}$. De plus, si la caméra est elle-même mobile, alors $\mathbf{s}$ peut s'écrire :

$$\mathbf{s} = \mathbf{s}(\mathbf{q}, t) \tag{5.1}$$

où le paramètre temps t représente la contribution du mouvement de la caméra.

La différentielle de $\mathbf{s}$ permet de relier les variations des informations visuelles dans l'image aux mouvements de la caméra et de la scène. A partir de l'équation (5.1), on obtient :

$$\dot{\mathbf{s}} = \frac{\partial \mathbf{s}}{\partial \mathbf{q}} \dot{\mathbf{q}} + \frac{\partial \mathbf{s}}{\partial t} \tag{5.2}$$

Le terme $\frac{\partial \mathbf{s}}{\partial \mathbf{q}}$ peut se décomposer sous la forme :

$$\frac{\partial \mathbf{s}}{\partial \mathbf{q}} = \frac{\partial \mathbf{s}}{\partial \mathbf{r}} \frac{\partial \mathbf{r}}{\partial \mathbf{q}} \tag{5.3}$$

où l'on reconnaît le classique Jacobien du robot $\frac{\partial \mathbf{r}}{\partial \mathbf{q}}$ si l'on choisit pour $\mathbf{r}$ la situation de la pince. Cette décomposition permet d'exhiber :

- un terme ne dépendant que de la configuration du robot et en aucun cas de la tâche : le Jacobien du robot,
- un terme ne dépendant que de la tâche choisie, représentée par $\mathbf{s}$, et en aucun cas de la configuration du robot : le terme $\frac{\partial \mathbf{s}}{\partial \mathbf{r}}$ que l'on peut appeler Jacobien de la tâche.

Il est donc essentiel de caractériser le Jacobien de la tâche pour conserver une approche générale applicable à tout robot et notamment quel que soit son nombre de degrés de liberté. Cette caractérisation permet :

- la réalisation de la commande du robot en boucle fermée à partir de la valeur des informations visuelles (commande référencée capteur),
- l'estimation de la vitesse 3D des objets à partir de la vitesse de leurs primitives 2D fournies par la caméra,
- la quantification de la qualité d'observation des objets mobiles [SH94].

Considérons maintenant une composante $\mathbf{s}_j$ de $\mathbf{s}$. Nous savons que la différentielle de $\mathbf{s}_j$ est une application linéaire de $se_3|_{\mathbf{r}}$ dans $I\!R$. Par conséquent, la différentielle de $\mathbf{s}_j$ en $\mathbf{r}$ n'est autre qu'un torseur. Rappelant qu'un élément de se_3 est aussi un torseur cinématique $\mathbf{T}$, nous pouvons écrire au point $\mathbf{r}$ de SE_3 la relation fondamentale :

$$\dot{\mathbf{s}}_j = H_j \bullet \mathbf{T} \tag{5.4}$$

où:

- $\mathbf{T}$ est le torseur cinématique représentant la vitesse relative de la scène par rapport à la caméra,

- $\bullet$ est le produit de torseurs défini plus haut,

- H_j est un torseur dont l'expression dépend à la fois des caractéristiques de l'environnement et du capteur lui-même. Il caractérise complètement les interactions entre le capteur et son environnement et nous l'appellerons torseur d'interaction.

La relation (5.4) peut s'écrire sous la forme matricielle suivante :

$$\dot{\mathbf{s}}_j = J_j \, \mathbf{T} \quad \text{où} \quad J_j = H_j \begin{bmatrix} 0 & \mathbf{I}_3 \\ \mathbf{I}_3 & 0 \end{bmatrix} \tag{5.5}$$

J_j est la représentation matricielle du torseur d'interaction H_j exprimé dans un repère et en point donnés. La représentation matricielle de l'ensemble $\{H_1 \dots H_n\}$ associé à $\mathbf{s} = (s_1 \dots s_n)^T$ est appelé Jacobien d'image ou matrice d'interaction et notée J. Avec cette notation, nous aurons :

$$\dot{\mathbf{s}} = J \, \mathbf{T} \tag{5.6}$$

Le choix des informations visuelles doit tenir compte de deux conditions essentielles : tout d'abord, une condition évidemment nécessaire est qu'il existe des algorithmes de traitement d'images capables de mesurer cette information, et ce à un rythme le plus proche possible du rythme d'échantillonnage de la commande du robot. La seconde condition est qu'il soit possible de calculer de manière explicite le Jacobien de cette information visuelle.

5.4 Modélisation du Jacobien d'image

L'estimation du Jacobien peut être réalisée d'une façon analytique si l'on connaît le modèle de l'objet poursuivi [Cha90] [ECR92]. Elle peut être également réalisée d'une manière expérimentale, dans ce cas on imprime au robot des mouvements incrémentaux connus (autour d'une position nominale) et on mesure dans l'image l'effet de ces mouvements sur les signaux 2D. Ensuite, la valeur numérique du Jacobien à la position nominale sera déterminée en résolvant un système linéaire en les coefficients de ce Jacobien. Deux exemples de cette approche sont rapportés dans [YA94] et [BJP93]. Le premier correspond à un mouvement de deux degrés de liberté, le second à un mouvement de trois degrés de liberté. Toutefois, cette approche est mal adaptée à l'asservissement temps réel et la précision du Jacobien dépend du calibrage du robot.

Dans la suite, nous allons donner la forme explicite du Jacobien lorsque les primitives visuelles sont des points, des segments et des droites.

Supposons que la caméra soit immobile et que la pince soit animée d'une vitesse de translation $\mathbf{V}_O$ et d'une vitesse de rotation Ω_O (voir Figure 5.5). Le torseur cinématique, appliqué à la pince du robot, s'écrit donc :

$$\mathbf{T}_O = \begin{bmatrix} \mathbf{V}_O \\ \Omega_O \end{bmatrix}$$

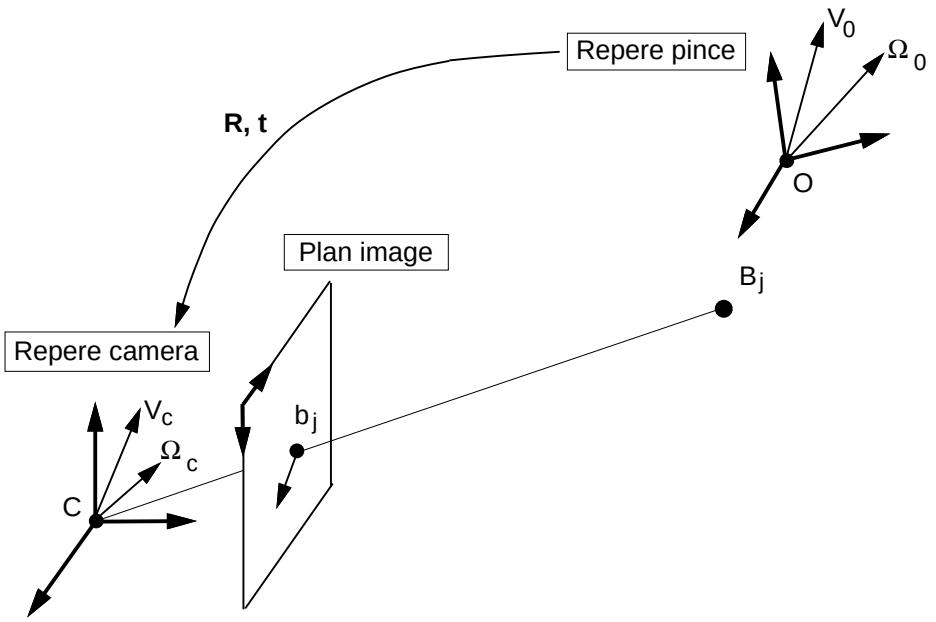


FIG. 5.5 - *Cette figure représente la relation entre le torseur cinématique appliqué à la pince du robot et la vitesse 2D de l'image d'un point (B_j) de cette pince.*

Si l'on note par $\mathbf{T}_C$ le mouvement de la caméra exprimé dans son repère et qui représente le même mouvement que $\mathbf{T}_O$, on obtient la relation suivante :

$$\begin{bmatrix} \mathbf{V}_C \\ \Omega_C \end{bmatrix} = \Theta \begin{bmatrix} \mathbf{V}_O \\ \Omega_O \end{bmatrix} \tag{5.7}$$

avec :

$$\Theta = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\, S(-\mathbf{R}^T\mathbf{t}) \\ 0 & \mathbf{R} \end{bmatrix} \tag{5.8}$$

Dans cette équation, $\mathbf{R}$ et $\mathbf{t}$ représentent la matrice de rotation et le vecteur de translation associés à la transformation rigide entre la caméra et la pince, cette transformation permet le changement de coordonnées *du* repère pince *vers* le repère caméra ; $S(\mathbf{a})$ représente la matrice antisymétrique associée au vecteur tridimensionnel $\mathbf{a}$.

5.4.1 Les points

On considère un point de la pince du robot désigné par $\mathbf{B}_j$ (voir Figure 5.5). Les coordonnées de ce point $(x_j,\ y_j,\ z_j)$ sont exprimées dans le repère de la caméra. La projection de ce point sur le plan image a pour coordonnées :

$$u_j = \alpha_u \frac{x_j}{z_j} + u_c \tag{5.9}$$

$$v_j = \alpha_v \frac{y_j}{z_j} + v_c \tag{5.10}$$

où α_u, α_v, u_c et v_c sont les paramètres intrinsèques de la caméra associés au modèle sténopé.

En différentiant les deux équations (5.9) et (5.10), on obtient les variations dans l'image des coordonnées u_j et v_j de $\mathbf{b}_j$ par rapport à la vitesse du point $\mathbf{B}_j$:

$$\begin{bmatrix} \dot{u}_j \\ \dot{v}_j \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{bmatrix} \begin{bmatrix} 1/z_j & 0 & -x_j/z_j^2 \\ 0 & 1/z_j & -y_j/z_j^2 \end{bmatrix} \begin{bmatrix} \dot{x}_j \\ \dot{y}_j \\ \dot{z}_j \end{bmatrix} \tag{5.11}$$

En appliquant la loi des mouvements rigides, on peut écrire la relation qui donne la vitesse $\mathbf{V}_{B_j} = \begin{pmatrix} \dot{x}_j & \dot{y}_j & \dot{z}_j \end{pmatrix}^T$ du point $\mathbf{B}_j$ en fonction des éléments de réduction du torseur $\mathbf{T}_c$, soit :

$$\mathbf{V}_{B_j} = \mathbf{V}_C + \Omega_C \times \overrightarrow{CB_j}$$

En remplaçant cette équation dans l'équation (5.11), on obtient :

$$\begin{bmatrix} \dot{u}_j \\ \dot{v}_j \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{bmatrix} \begin{bmatrix} 1/z_j & 0 & -x_j/z_j^2 & -x_jy_j/z_j^2 & 1+x_j^2/z_j^2 & -y_j/z_j \\ 0 & 1/z_j & -y_j/z_j^2 & -(1+y_j^2/z_j^2) & x_jy_j/z_j^2 & x_j/z_j \end{bmatrix} \begin{bmatrix} \mathbf{V}_C \\ \Omega_C \end{bmatrix} \tag{5.12}$$

En combinant les deux équations (5.12) et (5.7), nous obtenons :

$$\underbrace{\begin{bmatrix} \dot{u}_j \\ \dot{v}_j \end{bmatrix}}_{2\times 1} = J(\mathbf{B}_j) \underbrace{\begin{bmatrix} \mathbf{V}_O \\ \Omega_O \end{bmatrix}}_{6\times 1} \tag{5.13}$$

Dans cette formule, le Jacobien $J(\mathbf{B}_j)$, associé au point $\mathbf{B}_j$ de la pince, est donné par :

$$\begin{aligned} J(\mathbf{B}_j) \;=\; & \begin{bmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{bmatrix} \begin{bmatrix} 1/z_j & 0 & -x_j/z_j^2 & -x_jy_j/z_j^2 & 1+x_j^2/z_j^2 & -y_j/z_j \\ 0 & 1/z_j & -y_j/z_j^2 & -(1+y_j^2/z_j^2) & x_jy_j/z_j^2 & x_j/z_j \end{bmatrix} \\ & \begin{bmatrix} \mathbf{R} & -\mathbf{R}\, S(-\mathbf{R}^T\mathbf{t}) \\ 0 & \mathbf{R} \end{bmatrix} \end{aligned} \tag{5.14}$$

Il est à noter que dans certaines conditions le champ des vitesses, représenté par le vecteur $(\dot{u}_j, \dot{v}_j)$, et le flot optique ont la même composante suivant la direction du gradient spatial [HA91].

Supposons maintenant que l'on veut déterminer le Jacobien qui relie le torseur cinématique $\mathbf{T}_O$ et les variations des coordonnées normalisées X_j et Y_j de $\mathbf{b}_j$ qui sont données par :

$$X_j = \frac{x_j}{z_j} = \frac{u_j - u_c}{\alpha_u} \tag{5.15}$$

$$Y_j = \frac{y_j}{z_j} = \frac{v_j - v_c}{\alpha_v} \tag{5.16}$$

En différentiant ces deux équations, nous obtenons :

$$\begin{bmatrix} \dot{X}_j \\ \dot{Y}_j \end{bmatrix} = \begin{bmatrix} 1/\alpha_u & 0 \\ 0 & 1/\alpha_v \end{bmatrix} \begin{bmatrix} \dot{u}_j \\ \dot{v}_j \end{bmatrix} \tag{5.17}$$

En ulilisant l'équation (5.13), on peut écrire :

$$\begin{bmatrix} \dot{X}_j \\ \dot{Y}_j \end{bmatrix} = \begin{bmatrix} 1/\alpha_u & 0 \\ 0 & 1/\alpha_v \end{bmatrix} J(\mathbf{B}_j) \begin{bmatrix} \mathbf{V}_O \\ \Omega_O \end{bmatrix}$$

On en déduit l'expression du Jacobien associé aux coordonnées normalisées du point $\mathbf{b}_j$:

$$J_N(\mathbf{B}_j) = \begin{bmatrix} 1/z_j & 0 & -x_j/z_j^2 & -x_jy_j/z_j^2 & 1+x_j^2/z_j^2 & -y_j/z_j \\ 0 & 1/z_j & -y_j/z_j^2 & -(1+y_j^2/z_j^2) & x_jy_j/z_j^2 & x_j/z_j \end{bmatrix} \begin{bmatrix} \mathbf{R} & -\mathbf{R}\, S(-\mathbf{R}^T\mathbf{t}) \\ 0 & \mathbf{R} \end{bmatrix}$$

En utilisant les deux équations (5.15) et (5.16), on peut réécrire le Jacobien $J_N(\mathbf{B}_j)$ sous la forme suivante :

$$J_N(\mathbf{B}_j) = \begin{bmatrix} 1/z_j & 0 & -X_j/z_j & -X_jY_j & 1+X_j^2 & -Y_j \\ 0 & 1/z_j & -Y_j/z_j & -(1+Y_j^2) & X_jY_j & X_j \end{bmatrix} \begin{bmatrix} \mathbf{R} & -\mathbf{R}\, S(-\mathbf{R}^T\mathbf{t}) \\ 0 & \mathbf{R} \end{bmatrix} \tag{5.18}$$

Dans le cas général où l'on a n points, les expressions du Jacobien deviennent :

$$J = \begin{bmatrix} J(\mathbf{B}_1) \\ \vdots \\ J(\mathbf{B}_n) \end{bmatrix}$$

et

$$J_N = \begin{bmatrix} J_N(\mathbf{B}_1) \\ \vdots \\ J_N(\mathbf{B}_n) \end{bmatrix}$$

5.4.2 Les segments

Considérons un segment 3D d'extrémités les points B_1 et B_2. Ce segment peut être représenté dans l'image par les coordonnées u_1, v_1, u_2 et v_2 de ses extrémités b_1 et b_2. Le Jacobien est alors obtenu à partir de l'équation (5.14) et on a :

$$\begin{array}{rcl} J_{u_1} & = & \alpha_u \begin{bmatrix} 1/z_1 & 0 & -x_1/z_1^2 & -x_1y_1/z_1^2 & 1+x_1^2/z_1^2 & -y_1/z_1 \end{bmatrix} \Theta \\ J_{v_1} & = & \alpha_v \begin{bmatrix} 0 & 1/z_1 & -y_1/z_1^2 & -(1+y_1^2/z_1^2) & x_1y_1/z_1^2 & x_1/z_1 \end{bmatrix} \Theta \\ J_{u_2} & = & \alpha_u \begin{bmatrix} 1/z_2 & 0 & -x_2/z_2^2 & -x_2y_2/z_2^2 & 1+x_2^2/z_2^2 & -y_2/z_2 \end{bmatrix} \Theta \\ J_{v_2} & = & \alpha_v \begin{bmatrix} 0 & 1/z_2 & -y_2/z_2^2 & -(1+y_2^2/z_2^2) & x_2y_2/z_2^2 & x_2/z_2 \end{bmatrix} \Theta \end{array}$$

Une autre représentation des segments est souvent utilisée : la longueur l, l'orientation α et les coordonnées u_g et v_g du milieu du segment. Le passage à cette représentation est donnée par :

$$\begin{array}{rcl} u_g & = & (u_1+u_2)/2 \\ v_g & = & (v_1+v_2)/2 \\ l & = & \sqrt{(u_1-u_2)^2+(v_1-v_2)^2} \\ \alpha & = & \arctan(u_1-u_2)/(v_1-v_2) \end{array}$$

A l'aide de ces relations, on déduit :

$$\begin{bmatrix} J_{u_g} \\ J_{v_g} \\ J_l \\ J_\alpha \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ \Delta u/l & \Delta v/l & -\Delta u/l & -\Delta v/l \\ \Delta v/l^2 & -\Delta u/l^2 & -\Delta v/l^2 & \Delta u/l^2 \end{bmatrix} \begin{bmatrix} J_{u_1} \\ J_{v_1} \\ J_{u_2} \\ J_{v_2} \end{bmatrix}$$

où $\Delta u = u_1 - u_2$ et $\Delta v = v_1 - v_2$.

On obtient ainsi les torseurs d'interaction associés à la nouvelle représentation. De la même manière, si l'on considère un triangle, on peut le représenter :

- soit par les coordonnées dans l'image de ses trois sommets,
- soit par les coordonnées de son centre de gravité, la longueur de ses arêtes et l'orientation d'une de ses arêtes.

Rappelons qu'il est possible de calculer le Jacobien de toute autre information visuelle $\mathbf{s}$ basée sur les points (surface, distance, centre de gravité d'un polygone, ...) du moment que cette information puisse s'exprimer sous la forme $\mathbf{s} = \mathbf{s}(u_1, v_1, \ldots, u_n, v_n)$.

5.4.3 Les droites

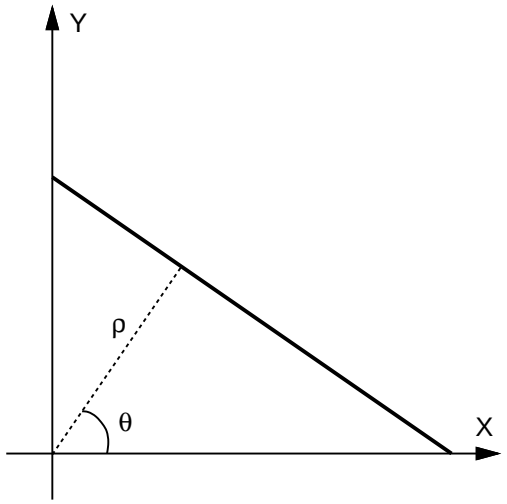


FIG. 5.6 - *Représentation (ρ, θ) des droites 2D.*

Une ligne droite peut être représentée par l'intersection de deux plans :

$$\mathcal{D} : \begin{cases} a_1 x + b_1 y + c_1 z + d_1 = 0 \\ a_2 x + b_2 y + c_2 z + d_2 = 0 \end{cases} \tag{5.19}$$

L'image 2D de cette droite, notée δ, a pour équation :

$$A\,X + B\,Y + C = 0 \tag{5.20}$$

Si nous adoptons la représentation (ρ, θ) (voir Figure 5.6), l'équation (5.20) s'écrit donc :

$$X \cos\theta + Y \sin\theta - \rho = 0 \tag{5.21}$$

où $\theta = \arctan(B/A)$ et $\rho = -C/\sqrt{A^2 + B^2}$.

L'ambiguïté de cette représentation (la même droite peut être paramétrée indifféremment par $(\rho, \theta + 2k\pi)$ et $(-\rho, \theta + (2k+1)\pi)$) est levée en partie en choisissant des sens pour les droites. Autrement dit, cela correspond à fixer le signe de ρ. De plus, l'ambiguïté dûe aux multiples choix possibles de θ, $(\theta + 2k\pi)$ d'une part, n'a aucune influence sur la valeur du Jacobien associé à ρ et θ et, d'autre part, peut facilement être levée en modulant à 2π l'erreur $\theta - \theta*$.

La rigidité des primitives géométriques permet d'écrire la différentielle de l'équation (5.21) sous la forme suivante :

$$\dot{\rho} + (X \sin\theta - Y \cos\theta)\, \dot{\theta} = \cos\theta\, \dot{X} + \sin\theta\, \dot{Y}, \quad \forall\, (X, Y) \in \delta \tag{5.22}$$

A partir des équations (5.19), (5.15) et (5.16), on peut écrire z en fonction de X et Y :

$$1/z = -(a_i X + b_i Y + c_i)/d_i \tag{5.23}$$

avec $i = 1$ si $d_1 \neq 0$ ou $i = 2$ si $d_2 \neq 0$.

En utilisant l'équation (5.18), nous pouvons écrire $\dot{X}$ et $\dot{Y}$ en fonction du torseur cinématique $\mathbf{T}_O$:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} 1/z & 0 & -X/z & -XY & 1+X^2 & -Y \\ 0 & 1/z & -Y/z & -(1+Y^2) & XY & X \end{bmatrix} \Theta\, \mathbf{T}_O \tag{5.24}$$

A partir de l'équation (5.21), on écrit X en fonction de Y si $\cos\theta \neq 0$ (ou Y en fonction de X sinon) et l'équation (5.22) peut s'écrire, en utilisant (5.23) et (5.24) :

$$(-\dot{\theta} / \cos\theta)\, Y + (\dot{\rho} + \rho \tan\theta\ \dot{\theta}) = Y\ K_1\ \Theta\ \mathbf{T}_O + K_2\ \Theta\ \mathbf{T}_O, \quad \forall\, Y \in I\!R. \tag{5.25}$$

Les deux vecteurs lignes K_1 et K_2 sont donnés par :

$$\begin{cases} K_1 = [\ -\lambda_1 \cos\theta & -\lambda_1 \sin\theta & \lambda_1 \rho & -\rho & -\rho \tan\theta & -1/\cos\theta\ ] \\ K_2 = [\ -\lambda_2 \cos\theta & -\lambda_2 \sin\theta & \lambda_2 \rho & -\sin\theta & \cos\theta + \rho^2/\cos\theta & \rho \tan\theta\ ] \end{cases}$$

avec :

$$\begin{aligned} \lambda_1 &= (-a_i \tan\theta + b_i)/d_i \\ \lambda_2 &= (a_i \rho/\cos\theta + c_i)/d_i \end{aligned}$$

L'équation(5.25) peut s'écrire sous la forme suivante :

$$(-\dot{\theta} / \cos\theta - K_1\ \Theta\ \mathbf{T}_O)\, Y + \dot{\rho} + \rho \tan\theta\ \dot{\theta} - K_2\ \Theta\ \mathbf{T}_O = 0 \quad \forall\, Y \in I\!R.$$

On a alors :

$$\dot{\theta} = -K_1 \cos\theta \; \Theta \; \mathbf{T}_O \tag{5.26}$$

$$\dot{\rho} = (K_2 + \rho \sin\theta K_1) \; \Theta \; \mathbf{T}_O \tag{5.27}$$

On en déduit l'expression du Jacobien associé à ρ et θ :

$$J(\delta) = \begin{bmatrix} -\lambda_\theta \cos\theta & -\lambda_\theta \sin\theta & \lambda_\theta \rho & \rho\cos\theta & \rho\sin\theta & 1 \\ -\lambda_\rho \cos\theta & -\lambda_\rho \sin\theta & \lambda_\rho \rho & -(1+\rho^2)\sin\theta & (1+\rho^2)\cos\theta & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & -\mathbf{R}\, S(-\mathbf{R}^T\mathbf{t}) \\ 0 & \mathbf{R} \end{bmatrix}$$

avec :

$$\begin{aligned} \lambda_\theta &= (a_i \sin\theta - b_i \cos\theta)/d_i \\ \lambda_\rho &= (a_i \rho \cos\theta + b_i \rho \sin\theta + c_i)/d_i \end{aligned}$$

5.5 Contrôle

Nous allons à présent montrer comment on peut construire une commande à partir des informations visuelles 2D. Pour ce faire, nous considérons le vecteur $\mathbf{s}$ qui représente les mesures extraites de l'image. Les composantes de ce vecteur caractérise d'une manière géométrique les primitives 2D qui peuvent être de nature différente (point, droite, ellipse, ...).

Soit maintenant une fonction scalaire $h(\mathbf{s})$ que l'on souhaite minimiser par rapport à $\mathbf{q}$. Si un tel minimum m existe, au moins localement, son gradient y sera nul :

$$\forall \mathbf{q} \in m : \frac{\partial h}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial \mathbf{r}} \frac{\partial \mathbf{r}}{\partial \mathbf{q}} = \mathbf{0} \tag{5.28}$$

Le Jacobien du robot $\frac{\partial \mathbf{r}}{\partial \mathbf{q}}$ étant supposé régulier, l'équation (5.28) devient :

$$\forall \mathbf{r} \in f(m) : \frac{\partial h}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial \mathbf{r}} = \mathbf{0} \tag{5.29}$$

$\frac{\partial \mathbf{s}}{\partial \mathbf{r}}$, de dimension $k \times 6$ et de rang $p \leq min(k, 6)$, est le Jacobien d'image J. La solution de (5.29) est donc m tel que :

$$\forall \mathbf{r} \in f(m) : \frac{\partial h}{\partial \mathbf{s}} \in Ker(J) \tag{5.30}$$

Soit par exemple $h = \frac{1}{2}(\mathbf{s} - \mathbf{s}^\star)^T(\mathbf{s} - \mathbf{s}^\star)$. On a alors $\frac{\partial h}{\partial \mathbf{s}} = \mathbf{s} - \mathbf{s}^\star$. Suivant la dimension k et le rang p il peut ou non exister plusieurs classes de solution à (5.30), dont :

$$\mathbf{s} - \mathbf{s}^\star = \mathbf{0} \tag{5.31}$$

seule situation que nous considérerons dans la suite.

Si les contraintes définies par (5.31) sont compatibles, alors (5.31) définit une liaison virtuelle. En dérivant (5.31), il vient :

$$\dot{\mathbf{s}} = \frac{\partial \mathbf{s}}{\partial \mathbf{r}} \mathbf{T} = \mathbf{0}$$

Comme l'asservissement a pour but le contrôle du mouvement du robot, nous pouvons supposer que la variable de commande n'est autre que le torseur cinématique imprimé à la pince du robot. On cherche à atteindre un motif dans l'image représenté par $\mathbf{s}^\star$ qui représente la valeur désirée de $\mathbf{s}$. Le choix des informations visuelles $\mathbf{s}$ va déterminer la nature de la tâche qu'on veut réaliser. Il faut à présent construire une commande qui permette de réaliser la contrainte (5.31). Supposons que la dimension du vecteur $\mathbf{s}$ soit plus grande ou égale à 6 et que les contraintes définies par (5.31) soient compatibles et indépendantes. En d'autres termes, dans un certain domaine de SE_3, la solution de (5.31) est unique et la matrice J est régulière.

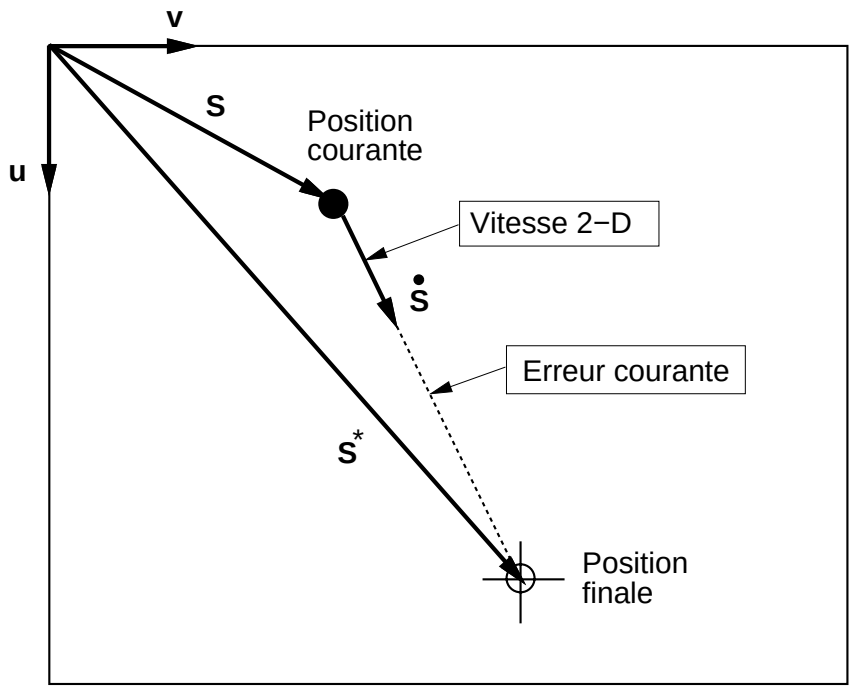


FIG. 5.7 - *Le vecteur vitesse d'un point image doit être proportionnel au vecteur d'erreur entre la position finale et la position courante.*

Sans la moindre perte de généralité, supposons que les informations visuelles sont les coordonnées images de n points attachés à la pince du robot. Dans ce cas, le vecteur de mesures $\mathbf{s}$, de dimension $2n$, s'écrit :

$$\mathbf{s} = (u_1\ v_1 \ldots u_j\ v_j \ldots u_n\ v_n)^T$$

Nous représentons par $\mathbf{s}^\star$ les coordonnées 2D de ces n points correspondant à la position finale de la pince. Ce vecteur peut être calculé par plusieurs méthodes : apprentissage, localisation 3D, alignement robot/objet tel qu'il est présenté dans le paragraphe suivant. L'apprentissage consiste à se placer explicitement et réellement dans la situation correspondant à la réalisation de la tâche, et à mesurer le vecteur $\mathbf{s}^\star$ associé. Il s'agit ici de en quelque sorte d'étalonner l'application plutôt la caméra. La deuxième solution consiste à calculer le $\mathbf{s}^\star$ à partir d'un modèle. L'inconvénient majeur de cette solution est que les

incertitudes sur les paramètres du modèle (paramètres intrinsèques de la caméra) vont introduire une erreur sur le positionnement 3D.

La manière idéale consiste à imposer à chaque point image une vitesse proportionnelle à la distance qui le sépare de sa position finale (voir Figure 5.7). Ce comportement idéal se traduit par :

$$\dot{\mathbf{s}} = g\,(\mathbf{s}^\star - \mathbf{s}) \tag{5.32}$$

où g est un scalaire positif qui influe sur le régime transitoire de l'asservissement et appelé gain de l'asservissement.

En se rappelant que :

$$\dot{\mathbf{s}} = J \begin{bmatrix} \mathbf{V}_O \\ \Omega_O \end{bmatrix}$$

on obtient :

$$J \begin{bmatrix} \mathbf{V}_O \\ \Omega_O \end{bmatrix} = g\,(\mathbf{s}^\star - \mathbf{s}) \tag{5.33}$$

Avec :

$$J = \begin{bmatrix} J(\mathbf{B}_1) \\ \vdots \\ J(\mathbf{B}_n) \end{bmatrix}$$

Le Jacobien $J(\mathbf{B}_j)$ associé au point $\mathbf{B}_j$ est donné par l'équation (5.14). La matrice J a pour dimension $2n \times 6$ et dépend des paramètres suivants :

- α_u et α_v qui sont les facteurs d'échelle vertical et horizontal associés à l'ensemble caméra/convertisseur analogique numérique. Nous pouvons noter que la connaissance du rapport α_u/α_v est suffisante pour garantir la colinéarité exigée par l'équation (5.32). Le plus souvent, ce rapport est connu avec une grande précision ;

- x_j, y_j et z_j qui sont les coordonnées des points de la pince exprimées dans le repère caméra. Ces coordonnées varient avec le temps puisque la pince se déplace ;

- $\mathbf{R}$ et $\mathbf{t}$ qui sont la rotation et la translation entre le repère pince et le repère caméra. Ces valeurs varient avec le temps pour la même raison citée plus haut.

Le Jacobien J n'est pas constant puisque la relation entre une scène 3D et sa projection 2D n'est pas linéaire. Il s'avère très important d'estimer la valeur de cette matrice à n'importe quelle position. Cette mise à jour sera effectuée par un calcul de la pose entre la pince et la caméra.

La solution de l'équation (5.33) au sens des moindres carrés va donner le torseur cinématique :

$$\mathbf{T}_O = \begin{bmatrix} \mathbf{V}_O \\ \Omega_O \end{bmatrix} = g\; J^\dagger\,(\mathbf{s}^\star - \mathbf{s}) \tag{5.34}$$

où $J^{\dagger}$ est la pseudo-inverse de la matrice J et est donnée par :

$$J^{\dagger} = \left(J^T J\right)^{-1} J^T$$

Dans tous les cas, la solution de l'équation (5.33), notée $\mathbf{T}_O$, vérifie les deux propriétés suivantes :

1. $\forall\ \mathbf{T}_O^{'}$ on a la relation suivante :

$$\| J\ \mathbf{T}_O - g\left(\mathbf{s}^{\star} - \mathbf{s}\right) \| \leq \| J\ \mathbf{T}_O^{'} - g\left(\mathbf{s}^{\star} - \mathbf{s}\right) \|$$

2. pour tous les mouvements $\mathbf{T}_O^{'} \neq \mathbf{T}_O$ tels que :

$$\| J\ \mathbf{T}_O - g\left(\mathbf{s}^{\star} - \mathbf{s}\right) \| = \| J\ \mathbf{T}_O^{'} - g\left(\mathbf{s}^{\star} - \mathbf{s}\right) \|$$

on a :

$$\|\mathbf{T}_O\| < \|\mathbf{T}_O^{'}\|$$

La condition sur le rang de la matrice J implique que le nombre minimal de points soit égal à 3. Cependant, dans [Cha90] on montre que dans le cas de trois points il existe une infinité de situations où la matrice J n'est pas de rang plein (son rang est inférieur à 6). Une solution efficace au problème du rang consiste à augmenter le nombre des informations visuelles de telle sorte que ce nombre soit plus grand que le nombre de degrés de liberté contrôlés.

L'algorithme de l'asservissement visuel est le suivant :

1. Prendre une image de la pince du robot.
2. Détecter les points images correspondant à certains points spécifiés de la pince.
3. Apparier ces points avec leurs correspondants 3D.
   Si la position courante des points est assez proche de leur position finale *alors* arrêter.
   Sinon aller à l'étape suivante.
4. Calculer la pose de la pince par rapport à la caméra.
5. Calculer la matrice J ainsi que sa pseudo-inverse.
6. Calculer le torseur cinématique de la pince tel qu'il est donné par l'équation (5.34) et imprimer ce torseur à la pince.
7. Aller à l'étape 1.

Nous allons à présent comparer le comportement dynamique de l'asservissement visuel dans deux cas de figure: (i) un Jacobien constant, (ii) un Jacobien qui est mis à jour d'une façon continue.

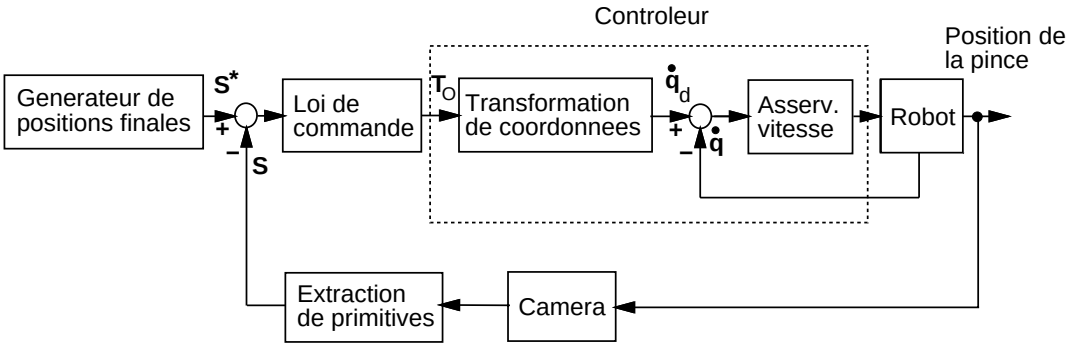


FIG. 5.8 - *Implantation de l'asservissement visuel de la pince du robot: matériels et logiciels.*

Pour ce faire, considérons la loi de commande (5.34). En pratique, on utilise un modèle $\hat{J}$ de la matrice J. La commande sera :

$$\mathbf{T}_O = g\ \hat{J}^{\dagger}\ (\mathbf{s}^{\star} - \mathbf{s})$$

L'évolution de l'erreur $\mathbf{e} = \mathbf{s}^{\star} - \mathbf{s}$ est donnée par :

$$\begin{aligned} \dot{\mathbf{e}} &= -\dot{\mathbf{s}} \\ &= -J\ \mathbf{T}_O \\ &= -g\ J\ \hat{J}^{\dagger}\ \mathbf{e} \end{aligned}$$

La dérivée temporelle du module de l'erreur $\mathbf{e}$ s'écrit :

$$\begin{aligned} \frac{d}{dt}\left(\frac{1}{2}\|\mathbf{e}\|^2\right) &= \mathbf{e}^T\ \dot{\mathbf{e}} \\ &= -g\ \mathbf{e}^T J\ \hat{J}^{\dagger}\ \mathbf{e} \end{aligned}$$

On voit qu'une condition suffisante de décroissance de $\|\mathbf{e}\|$ est que la matrice $J\hat{J}^{\dagger}$ soit positive.

Il est clair que :

- Si $\hat{J}$ est une bonne approximation de la matrice J, alors $\hat{J} \approx J$ et la condition de positivité sera garantie quelle que soit la position du robot.
- Si $\hat{J}$ est constant alors la condition de positivité ne sera pas garantie pour toutes les positions du robot.

En résumé, l'utilisation d'un Jacobien variable a les avantages suivants:

- On garantit la décroissance exponentielle de l'erreur 2D $\|\mathbf{s}^{\star} - \mathbf{s}\|$ et ceci indépendamment de l'écart entre la position initiale et la position finale.
- La poursuite des points dans l'image sera robuste surtout lorsqu'on utilise des fenêtres. Avec un Jacobien constant on risque de perdre la poursuite des points, soit ces points ne seront pas dans le champ de vue de la caméra, soit ces points n'appartiendront pas aux fenêtres de recherche.
- Le temps de l'asservissement est optimisé.

5.6 Alignement robot/objet

La caractérisation de l'alignement de deux objets (un objet et une pince, ...) nécessite une représentation de la relation géométrique existant entre ces deux objets. En robotique, cette relation est représenté par une transformation euclidienne qui peut être symbolisée par une liaison virtuelle entre les deux objets. La nature de cette liaison virtuelle est variée : (rigide, prismatique, rotoïde, pivot glissant, appui plan, rotule, linéaire rectiligne, linéaire annulaire, ponctuelle). Elle est imposée par la nature de l'alignement entre les objets. Ainsi, la saisie d'un objet 3D pourrait être un cas particulier d'une liaison virtuelle rigide.

Dans ce paragraphe, nous allons montrer comment on peut caractériser les alignements de telle sorte que cette caractérisation soit non euclidienne donc invariante par projection. Nous allons considérer les conditions d'alignement associées à une caméra non étalonnée dont le modèle de projection est celui de la projection perspective puisqu'il est plus fidèle à la réalité. Dans ce cas, le processus géométrique de la formation de l'image sera décrit par une transformation projective de l'espace projectif 3D dans l'espace projectif 2D.

Premièrement, Nous considérons 6 points de l'objet à saisir. Cette approche peut être généralisée à un nombre quelconque de points pourvu que leur nombre soit plus grand ou égal à 6 et qu'il existe 5 points parmi eux qui ne sont pas quatre à quatre coplanaires. Les coordonnées 3D de ces 6 points sont connus dans un repère orthonormé associé à l'objet à saisir. De même, les points de la pince du robot sont connus dans un repère orthonormé associé à cette pince. Supposons que la caractérisation de l'alignement robot/objet est donnée par le déplacement rigide D (transformation homogène 4×4) entre le repère de la pince et le repère de l'objet (voir Figure 5.9).

Deuxièment, nous considérons une homographie, représentée par la matrice inversible P de dimension 4×4 en coordonnées homogènes. Cette homographie correspond à un changement de base entre l'espace euclidien (repère de l'objet) et l'espace projectif 3D dont la base est formée par les 5 points. La matrice P est déterminée à un facteur multiplicatif près (elle contient 15 paramètres indépendants) et peut être calculée de la manière suivante :

Soient $\mathbf{A}_1, \ldots, \mathbf{A}_5$ les vecteurs de dimension 4 décrivant les coordonnées homogènes des 5 points de l'objet (non quatre à quatre coplanaires). Soit $\mathbf{A}_6$ le vecteur associé au sixième point de l'objet. On désigne par $\mathbf{A}_i^p$ le vecteur de dimension 4 associé aux coordonnées projectives du même point $\mathbf{A}_i$. Nous supposons que les 5 premiers points de l'objet sont la base canonique de l'espace projectif. Il en résulte que ces 5 points ont les coordonnées projectives suivantes :

$$(0\ 0\ 0\ 1)\ \ (1\ 0\ 0\ 1)\ \ (0\ 1\ 0\ 1)\ \ (0\ 0\ 1\ 1)\ \ (1\ 1\ 1\ 1)$$

La transformation projective de chaque point s'écrit :

$$\mathbf{A}_i^p = \lambda_i\ P\ \mathbf{A}_i \tag{5.35}$$

Comme cette équation fournit 3 contraintes linéaires en les coefficients de P, les 16 coefficients de cette matrice peuvent être déterminés en utilisant les 5 points $\mathbf{A}_1, \ldots, \mathbf{A}_5$ et leur transformée $\mathbf{A}_i^p$. Les 15 équations résultant sont indépendantes puisque les 5 points utilisés forment une base projective. Une contrainte supplémentaire que l'on ajoute à ces

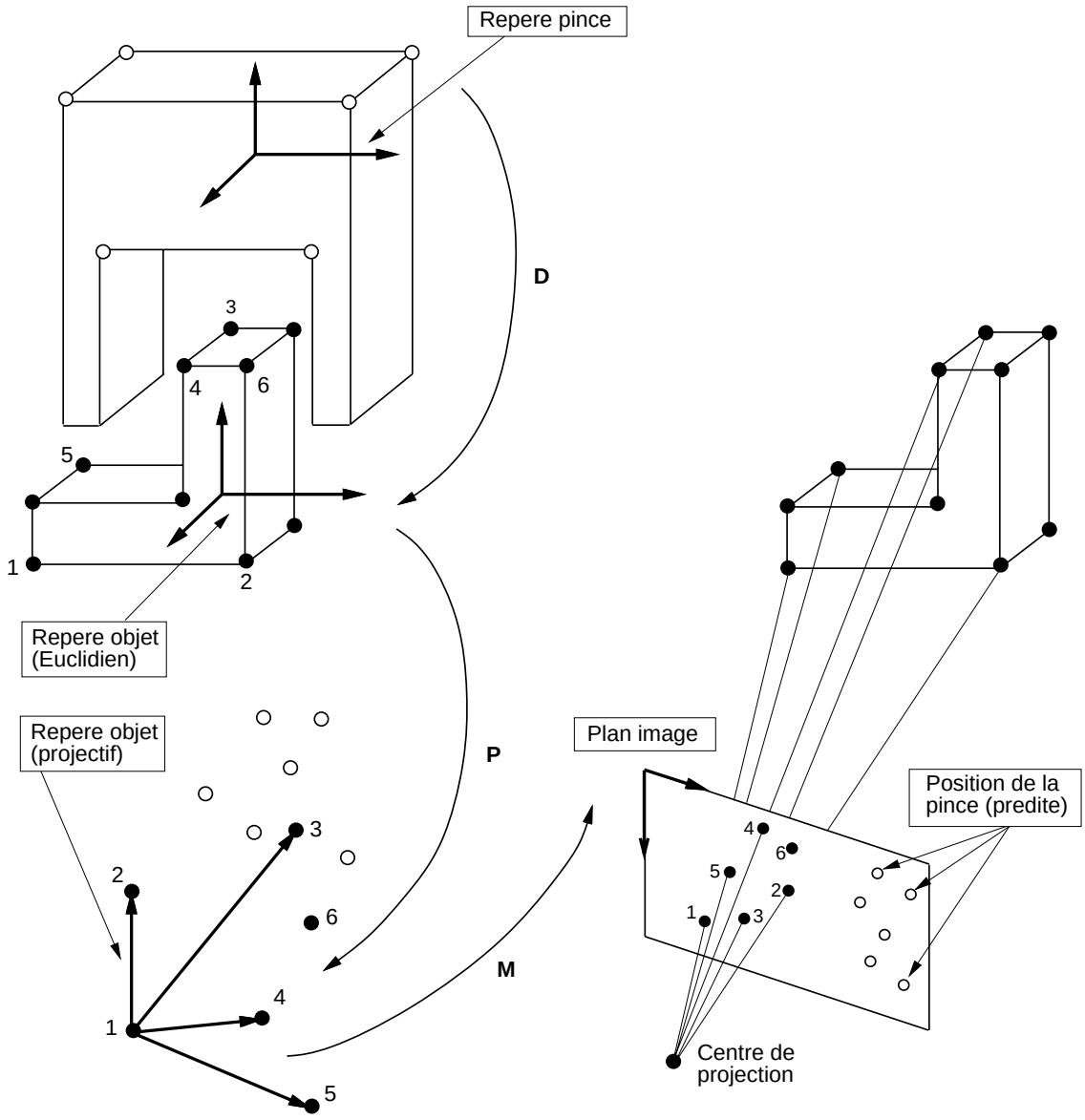


FIG. 5.9 - *Spécification de la consigne 2D $\boldsymbol{s}^\star$ correspondant à la tâche d'alignement robot/objet. On doit connaître: la transformation rigide pince-objet, au moins, 6 correspondances point 3D/point 2D de l'objet à saisir.*

15 équations va nous permettre de calculer les 16 coefficients de P. Une possibilité consiste à utiliser la contrainte suivante :

$$\sum_{i,j=1}^{4} p_{ij} = 1$$

Une fois la matrice P déterminée, on peut calculer, dans l'espace projectif ainsi défini, les coordonnées projectives de n'importe quel point exprimé dans le repère de l'objet.

Soit $\mathbf{B}_j$ le vecteur de dimension 4 associé à un point de la pince. Ce vecteur est exprimé dans un repère orthonormé attaché à la pince. La connaissance des deux matrices D et P va nous permettre de calculer le vecteur $\mathbf{B}_j^p$ qui représente les coordonnées projectives du point considéré dans la base projective canonique :

$$\mathbf{B}_j^p = \lambda_j \; P \; D \; \mathbf{B}_j$$

Considérons maintenant une caméra observant l'objet que la pince doit saisir. Nous n'avons aucune information sur les position et orientation entre cette caméra et son environnement (objet, pince). Soient $\mathbf{a}_1, \ldots, \mathbf{a}_6$ les images des 6 points objets. Comme le processus géométrique de la formation de l'image est décrit par une projection perspective, nous pouvons représenter cette projection par une matrice M de dimension 3×4 :

$$\mathbf{a}_i = M \; \mathbf{A}_i^p \tag{5.36}$$

avec $\mathbf{a}_i = (su_i \;\; sv_i \;\; s)^T$ où u_i et v_i sont les coordonnées images du point $\mathbf{a}_i$ et s est un facteur d'échelle. Il est évident qu'au moins 6 correspondances point 3D/point 2D vont permettre le calcul de la matrice M, puisque chaque correspondance fournit deux équations indépendantes en les coefficients de cette matrice.

Après l'estimation de la matrice de projection M, on peut prédire les positions dans l'image des points de la pince tels qu'ils apparaissent lors de la saisie de l'objet :

$$\mathbf{b}_j = M \; \mathbf{B}_j^p \tag{5.37}$$

En pratique, les calculs qu'on vient de présenter peuvent être décomposés en deux étapes. La première étape consiste à déterminer les vecteurs $\mathbf{A}_1^p \ldots \mathbf{A}_6^p$ et les vecteurs $\mathbf{B}_j^p$. Cette étape est effectuée hors ligne. La deuxième étape consiste à calculer la matrice M et les vecteurs $\mathbf{b}_j$. Cette deuxième étape est effectuée en ligne.

La majeure difficulté dans ce que nous venons de présenter est de trouver les 6 appariements points objets/points images. Ce problème d'appariement est un problème de reconnaissance d'objets à partir d'une image [HM93] [BB82].

5.7 Etude de performance

L'objet de ce paragraphe est d'étudier, principalement expérimentalement, l'influence des incertitudes sur le comportement d'un asservissement visuel, en considérant ici uniquement comme primitives géométriques un ensemble de points. Dans ce cas, le vecteur de mesures extraites des images comporte les coordonnées images de ces points. Le Jacobien associé au vecteur $\mathbf{s}$ concentre plusieurs types d'erreur qui affectent le système

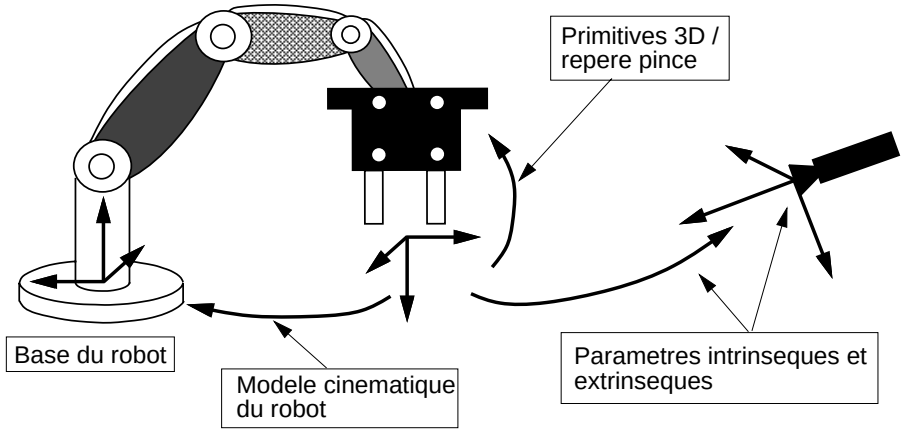


FIG. 5.10 - *Les diverses sources d'erreur affectant les paramètres du calibrage.*

caméra/robot. Notre optique sera celle de la régulation autour d'une valeur $\mathbf{s}^\star$ constante, supposée telle que l'utilisateur considère son objectif comme atteint dès que $\mathbf{s} = \mathbf{s}^\star$. Comme $\mathbf{s}^\star$ est correctement calculé, $\|\mathbf{s}^\star - \mathbf{s}\|$ peut tendre vers zéro même lorsque les paramètres de calibrage utilisés sont faux. La Figure 5.10 représente les diverses sources des incertitudes sur le calibrage du système caméra/robot.

On considère deux types de commande : l'une utilise une remise à jour du Jacobien à chaque pas en fonction de la situation 3D de la pince ; l'autre, généralement moins performante mais plus simple, utilise une valeur constante du Jacobien calculée à la situation finale. Pour chacune, on évalue la contribution de chaque erreur paramétrique dans diverses conditions initiales en rotation et en translation. On examine également leurs effets combinés dans quelques cas.

La commande est de la forme :

$$\mathbf{T}_O = \begin{bmatrix} \mathbf{V}_O \\ \Omega_O \end{bmatrix} = g\ J^\dagger\ (\mathbf{s}^\star - \mathbf{s})$$

Pour l'ensemble des simulations, ce torseur cinématique est appliqué à la pince sous forme de déplacement incrémental. En effet, si L_k est la matrice homogène représentant, à l'itération k, la situation de la pince par rapport à un repère fixe quelconque (le repère de la caméra) et si t_e représente la période d'échantillonnage, on a :

$$L_{k+1} = L_k \begin{bmatrix} R_d & \mathbf{V}_O\, t_e \\ \mathbf{0}^T & 1 \end{bmatrix}$$

avec :

$$R_d = \begin{bmatrix} \cos\theta + (1-\cos\theta)u_x^2 & -u_z s + (1-\cos\theta)u_x u_y & u_y s + (1-\cos\theta)u_x u_z \\ u_z s + (1-\cos\theta)u_x u_y & \cos\theta + (1-\cos\theta)u_y^2 & -u_x s + (1-\cos\theta)u_z u_y \\ -u_y s + (1-\cos\theta)u_x u_z & u_x s + (1-\cos\theta)u_z u_y & \cos\theta + (1-\cos\theta)u_z^2 \end{bmatrix}$$

où $\theta = \|\Omega_O\| t_e$ et $\mathbf{u}^T = [\ u_x \quad u_y \quad u_z\ ] = \Omega_O / \|\Omega_O\|$.

Dans les simulations suivantes, tous les paramètres du calibrage sont choisis de telle sorte qu'ils soient très proches de ceux du système réel qu'on a au (LIFIA). On considère 4 points coplanaires formant un carré de côté 10 cm. Ces 4 points sont attachés à la pince du robot. La position initiale du repère pince par rapport au repère caméra est donnée par [0, 0, 0] deg en orientation et [0, 0, 150] cm en position. Nous supposons que la période d'échantillonnage est de 100 ms et que le gain g vaut 0.5. Les paramètres intrinsèques de la caméra ont été fixés de la manière suivante :

$$\|\alpha_u\| = 1500, \|\alpha_v\| = 1000, u_c = v_c = 256$$

Expérience 1: petit écart initial

Dans cette expérience la position finale de la pince est donnée par [15, 15, 15] deg et [10, 10, 115] cm. Les diagrammes de la Figure 5.11 donnent le module de l'erreur, $\|\mathbf{s}^\star - \mathbf{s}\|$, obtenu avec les deux types de commande. Les courbes continues ont été obtenues avec un Jacobien variable, les courbes interrompues ont été obtenues avec un Jacobien constant. Le diagramme a) correspond au cas idéal où il n'y a pas d'erreur introduite sur les paramètres du Jacobien ; celui de b) a été obtenu en perturbant le modèle cinématique du robot : le torseur cinématique calculé par la loi de commande est multiplié par la matrice $Diag[\ 0.9,\ 1.2,\ 0.8,\ 1.1,\ 1.2,\ 0.7,\ ]$; celui de c) a été obtenu en perturbant la relation géométrique entre les primitives 3D et le repère de la pince : on a ajouté 2 cm aux composantes de la translation et 5 deg autour des trois axes ; celui de d) a été obtenu en perturbant le coefficient α_v de 20% ; celui de e) a été obtenu en ajoutant 8 deg de rotation et 10% de translation à la matrice de passage pince-caméra (les paramètres extrinsèques) ; celui de f) a été obtenu en combinant ces quatre types d'erreur.

La Figure 5.12 (respectivement 5.13) illustre les trajectoires des quatre points 2D dans le cas idéal associé au premier type de commande (au deuxième type de commande).

Les Figures 5.14 et 5.15 montrent les 8 composantes du vecteur $\mathbf{s}^\star - \mathbf{s}$ associé aux deux types de commande.

Expérience 2: grand écart initial

Dans ce cas, la position finale de la pince est donnée par [30, 30, 30] deg et [10, 10, 80] cm. Les résultats de simulation correspondant à cet écart sont présentés sur les Figures 5.16, 5.17, 5.18, 5.19 et 5.20, configurées comme les figures précédentes.

Expérience 3: écart initial moyen

Dans cette expérience la position finale de la pince est donnée par [20, 20, 20] deg en orientation et [0, 0, 100] cm en position. Dans ce cas, on a un point de la pince qui se trouve sur l'axe optique de la caméra dans les positions initiale et finale. Les Figures 5.21 et 5.22 présentent les trajectoires des 4 points associées aux deux types de commande.

Les Figures 5.23 et 5.24 présentent les 8 composantes du vecteur d'erreur $\mathbf{s}^\star - \mathbf{s}$.

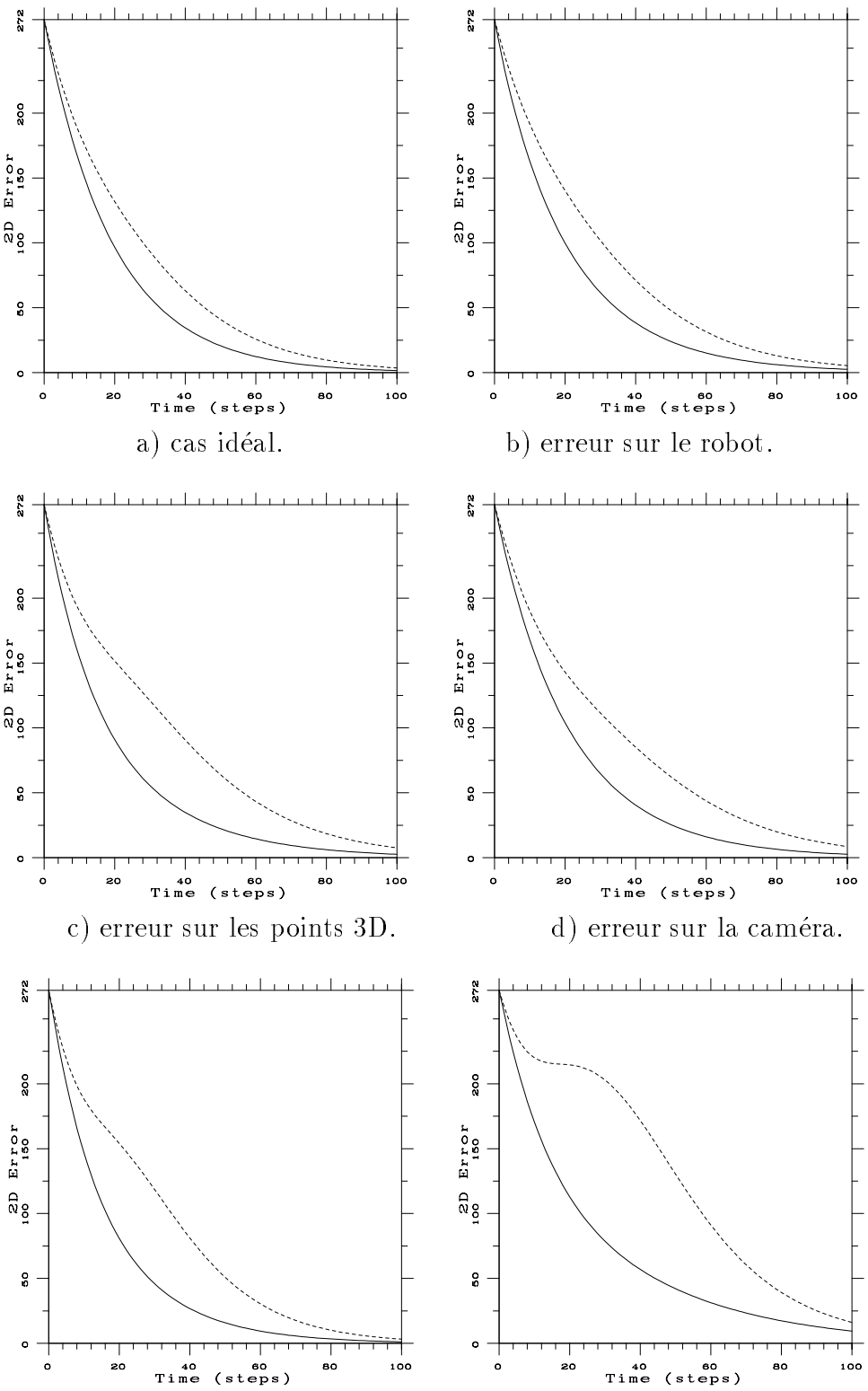
D'après ces résultats, on peut tirer les conclusions suivantes :

- Avec le premier type de commande (Jacobien variable), la convergence exponentielle de l'erreur est assurée même lorsque l'écart initial de positionnement est très grand. Avec le deuxième type de commande (Jacobien constant), le comportement transitoire de l'asservissement se dégrade au fur et à mesure que l'écart initial de positionnement augmente.

- La différence de comportement entre les deux types de commande est très sensible aux paramètres de calibrage de nature géométrique: i) la relation entre le repère pince et les primitives 3D et ii) la transformation caméra-pince (voir les Figures 5.11.c, 5.11.e et les Figures 5.16.c, 5.16.e).

- Le découplage est assuré par le premier type de commande. Dans la troisième expérience le point appartenant à l'axe optique et ayant son image initial confondu avec son image final le reste pendant toute la période de l'asservessiment, ce qui n'est pas le cas pour le deuxième type de commande (voir les coordonneés 2D données par les composantes 1 et 2 des Figures 5.23 et 5.24).

Notons enfin que le fait d'adopter un Jacobien constant dans la loi de la commande revient à commettre des erreurs sur les paramètres extrinsèques de la caméra (pose entre la caméra et la pince). Ces erreurs de calibrage deviennent très importantes si l'écart initial de positionnement est très grand.



a) cas idéal. b) erreur sur le robot.

c) erreur sur les points 3D. d) erreur sur la caméra.

e) erreur sur la pose. f) erreurs combinées.

FIG. 5.11 - *Expérience 1: le module de l'erreur, $\|\boldsymbol{s}^\star - \boldsymbol{s}\|$, correspondant à un petit écart initial.*

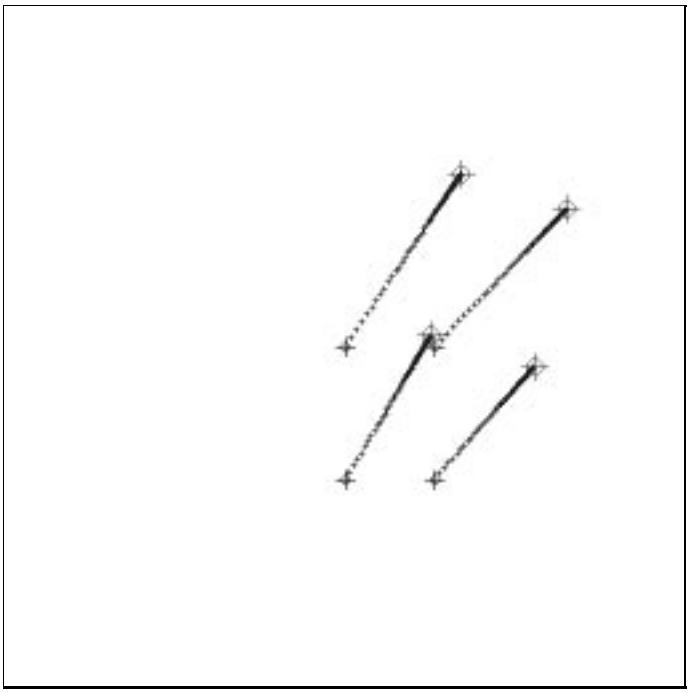


FIG. 5.12 - *Expérience 1: trajectoires 2D des 4 points obtenues avec un Jacobien variable (cas idéal correspondant à un petit écart initial).*

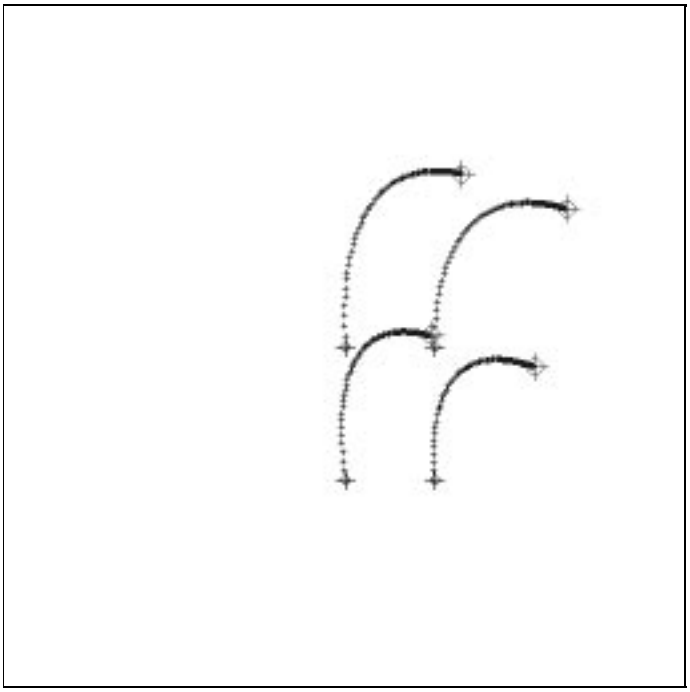


FIG. 5.13 - *Expérience 1: trajectoires 2D des 4 points obtenues avec un Jacobien constant (cas idéal correspondant à un petit écart initial).*

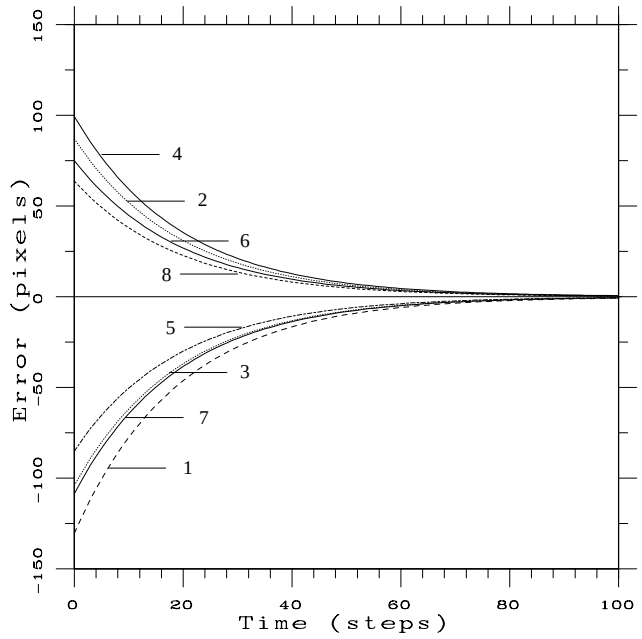


FIG. 5.14 - *Expérience 1: les 8 composantes du vecteur $\boldsymbol{s}^* - \boldsymbol{s}$ lors d'un asservissement avec un Jacobien variable (cas idéal correspondant à un petit écart initial).*

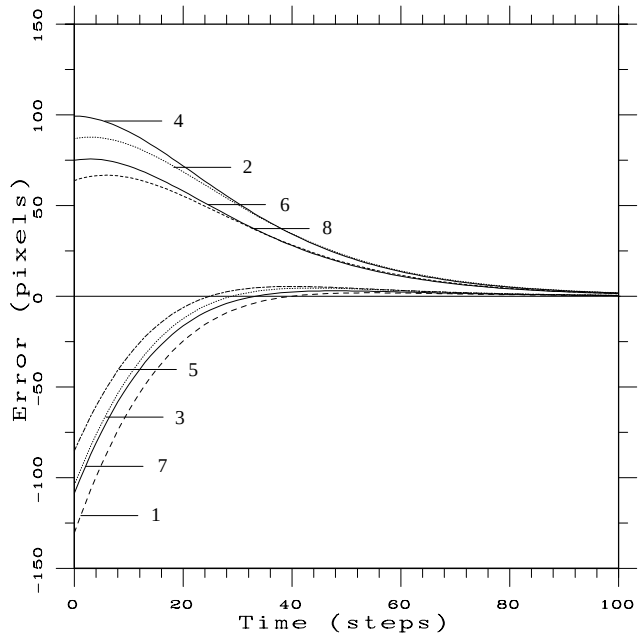
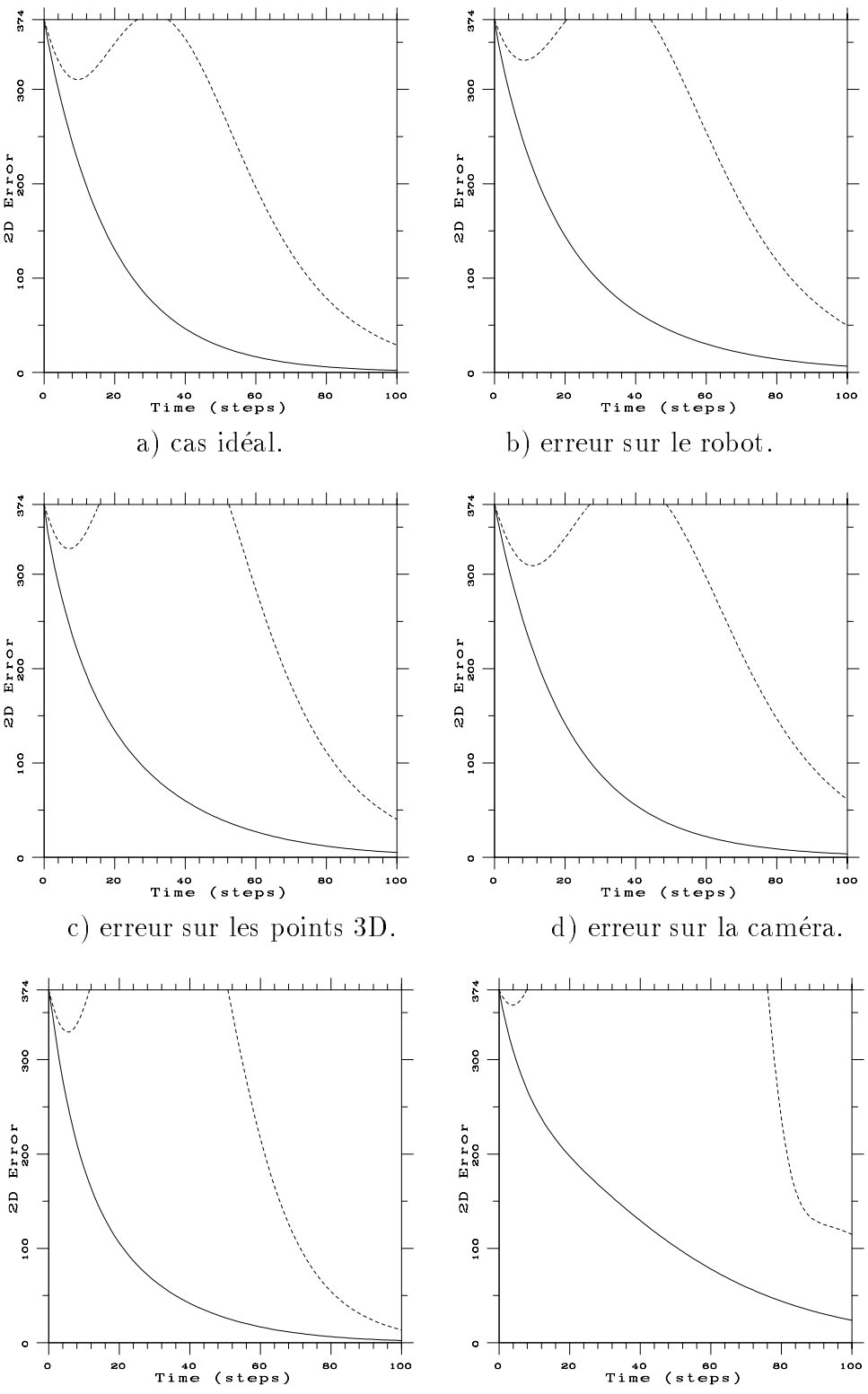


FIG. 5.15 - *Expérience 1: les 8 composantes du vecteur $\boldsymbol{s}^* - \boldsymbol{s}$ lors d'un asservissement avec un Jacobien constant (cas idéal correspondant à un petit écart initial).*



a) cas idéal. b) erreur sur le robot.

c) erreur sur les points 3D. d) erreur sur la caméra.

e) erreur sur la pose. f) erreurs combinées.

FIG. 5.16 - *Expérience 2: le module de l'erreur, $\|\boldsymbol{s}^\star - \boldsymbol{s}\|$, correspondant à un grand écart initial.*

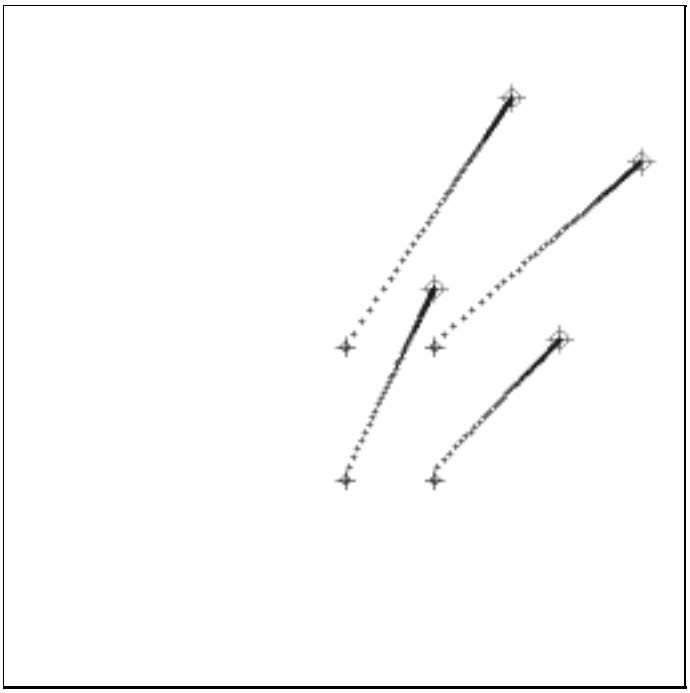


FIG. 5.17 - *Expérience 2: trajectoires 2D des 4 points obtenues avec un Jacobien variable (cas idéal correspondant à un grand écart initial).*

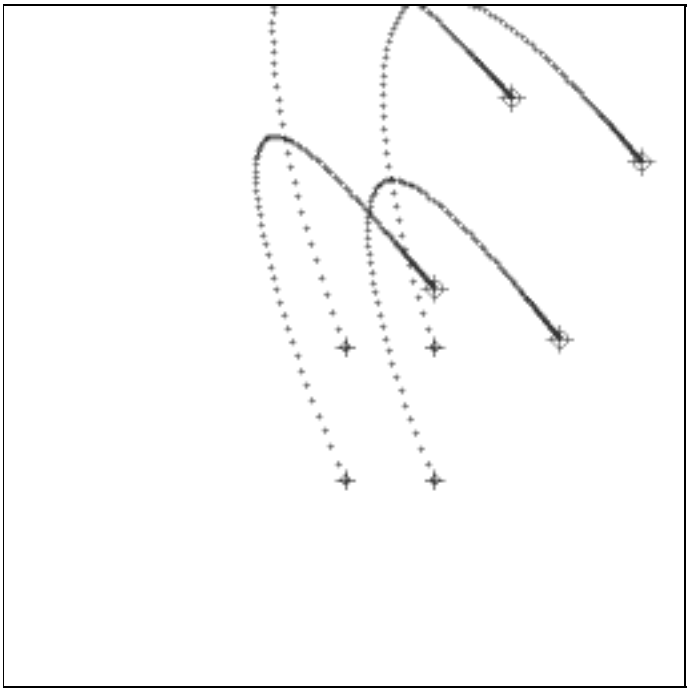


FIG. 5.18 - *Expérience 2: trajectoires 2D des 4 points obtenues avec un Jacobien constant (cas idéal correspondant à un grand écart initial).*

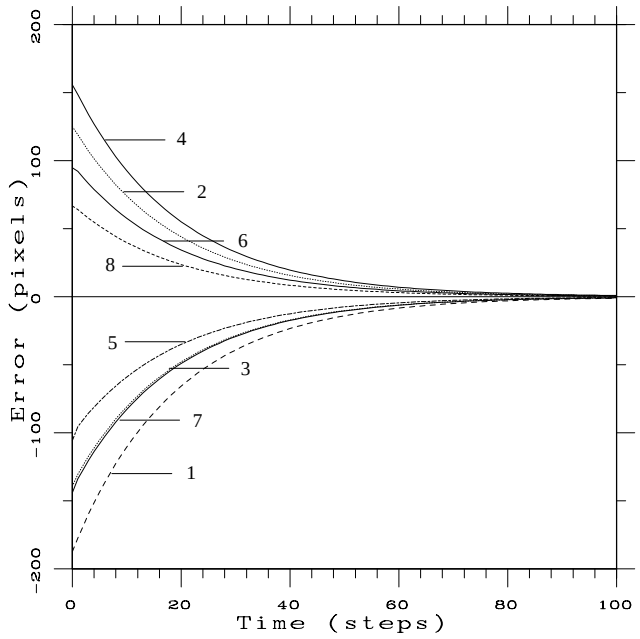


FIG. 5.19 - *Expérience 2: les 8 composantes du vecteur $\boldsymbol{s}^* - \boldsymbol{s}$ lors d'un asservissement avec un Jacobien variable (cas idéal correspondant à un grand écart initial).*

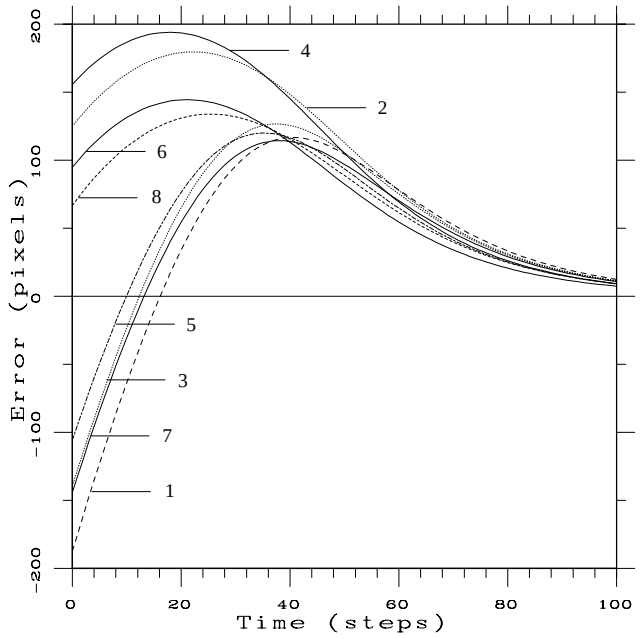


FIG. 5.20 - *Expérience 2: les 8 composantes du vecteur $\boldsymbol{s}^* - \boldsymbol{s}$ lors d'un asservissement avec un Jacobien constant (cas idéal correspondant à un grand écart initial).*

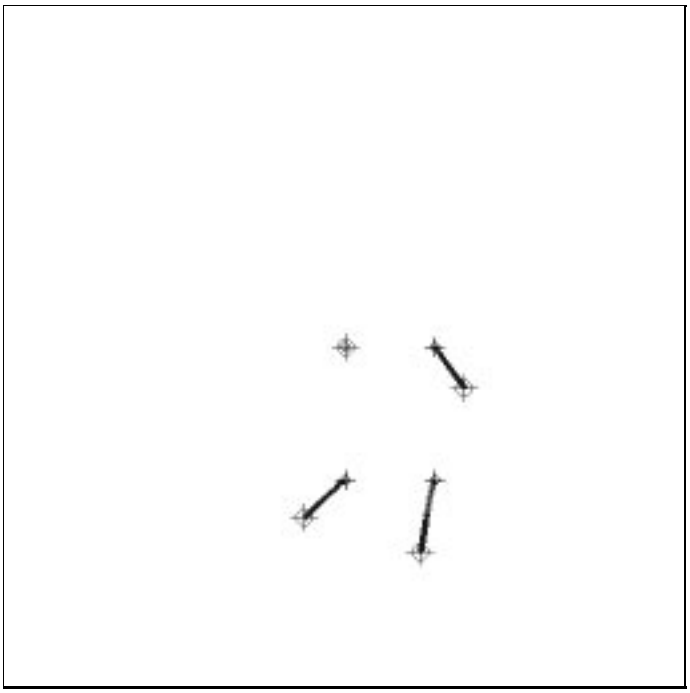


FIG. 5.21 - *Expérience 3: trajectoires 2D des 4 points obtenues avec un Jacobien variable (cas idéal correspondant à un écart initial moyen).*

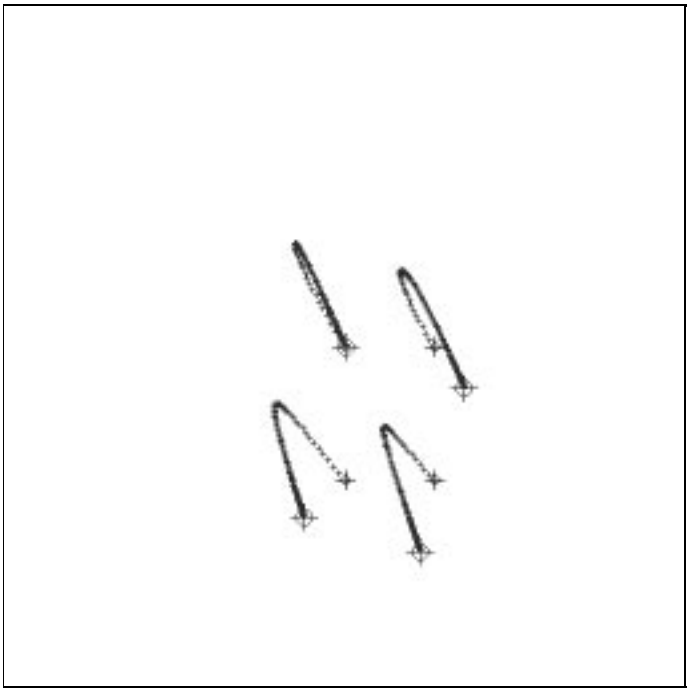


FIG. 5.22 - *Expérience 3: trajectoires 2D des 4 points obtenues avec un Jacobien constant (cas idéal correspondant à un écart initial moyen).*

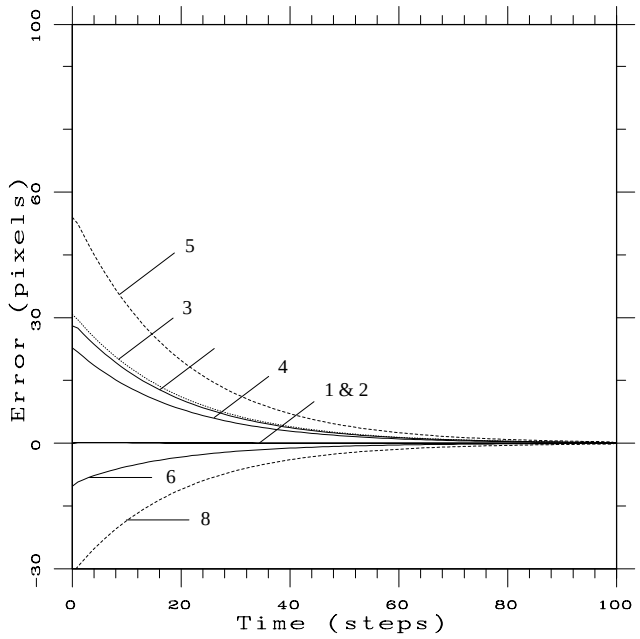


FIG. 5.23 - *Expérience 3: les 8 composantes du vecteur $\boldsymbol{s}^* - \boldsymbol{s}$ lors d'un asservissement avec un Jacobien variable (cas idéal correspondant à un écart initial moyen).*

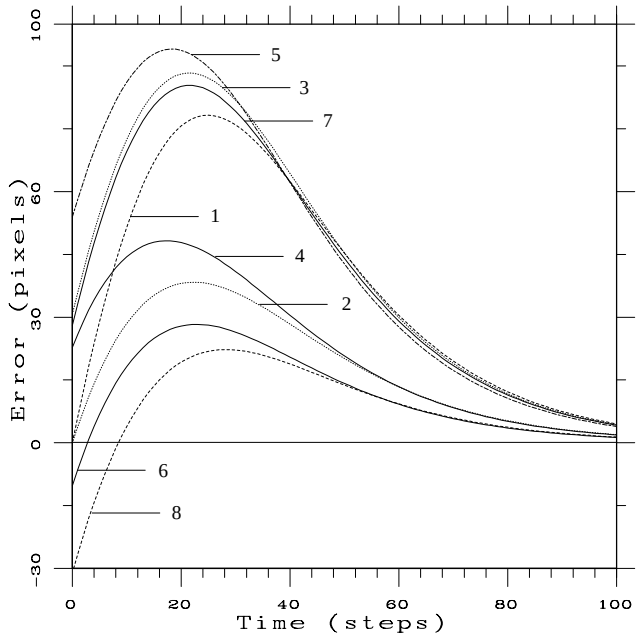


FIG. 5.24 - *Expérience 3: les 8 composantes du vecteur $\boldsymbol{s}^* - \boldsymbol{s}$ lors d'un asservissement avec un Jacobien constant (cas idéal correspondant à un écart initial moyen).*

5.8 Vitesse de convergence

Dans ce paragraphe, nous proposons de comparer le nombre d'itérations associé aux deux types de commande. Il est intéressant d'effectuer la tâche de positionnement en un temps minimum. Comme ce temps est plus ou moins proportionnel au nombre des itérations, celui-ci fournit le temps d'asservissement. Les divers paramètres du calibrage ont été choisi comme au paragraphe précédent. Nous supposons que la tâche est réalisée lorsque le module de l'erreur $\mathbf{s}^\star - \mathbf{s}$ est inférieur à 0.5 pixel. La position initiale de la pince, par rapport à la caméra, est donnée par [0, 0, 0] deg en orientation et [0, 0, 150] cm en position.

Erreur de position

Les positions finales sont données par [0, 0, 0] deg en orientation et [0, 0, 150-e] cm en position. Ainsi, l'erreur de positionnement initiale est égale à e. Dans l'espace 3D, cette erreur est une translation pure suivant l'axe optique de la caméra. Sur la Figure 5.25 est donné le nombre des itérations, en fonction de e, pour les deux types de commande. On peut constater que si la position finale a une profondeur plus petite que celle de la position initiale (et c'est le cas de la plupart des asservissements visuels puisque le motif final doit être extrait avec une grande précision), alors la convergence correspondant à un Jacobien variable est plus rapide que celle correspondant à un Jacobien constant (évalué à la position finale). En effet, dans l'expression du Jacobien la profondeur de chaque point 3D joue le rôle du gain dans la loi de commande. Comme ces profondeurs sont estimées à chaque pas, on peut dire que la loi de commande utilisant un Jacobien variable a en quelque sorte un gain adaptatif.

Erreur d'orientation

Dans ce cas, les positions finales de la pince sont données par [β, β, β] deg et [0, 0, 150] cm en position. Dans l'espace 3D, l'erreur de positionnement initiale est une rotation pure. Sur la Figure 5.26 est présenté le nombre des itérations, en fonction de β, pour les deux types de commande.

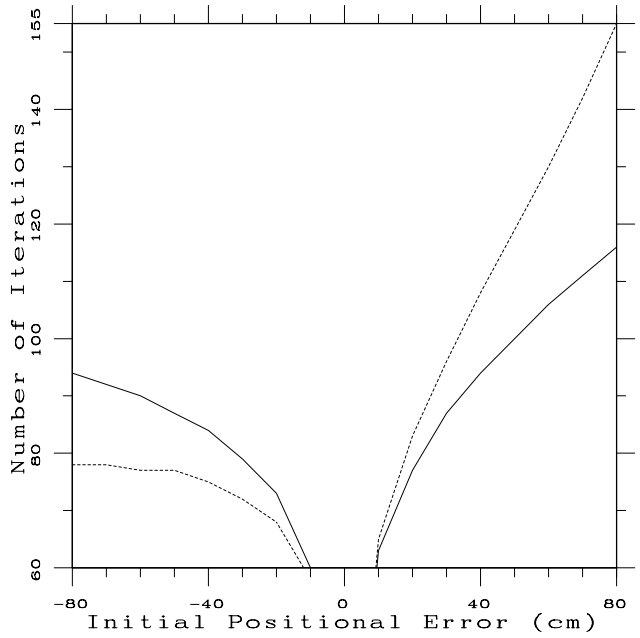


FIG. 5.25 - *Nombre d'itérations en fonction de l'erreur initiale (translation pure suivant l'axe optique) ; la courbe continue correspond à un Jacobien variable, la courbe interrompue à un Jacobien constant.*

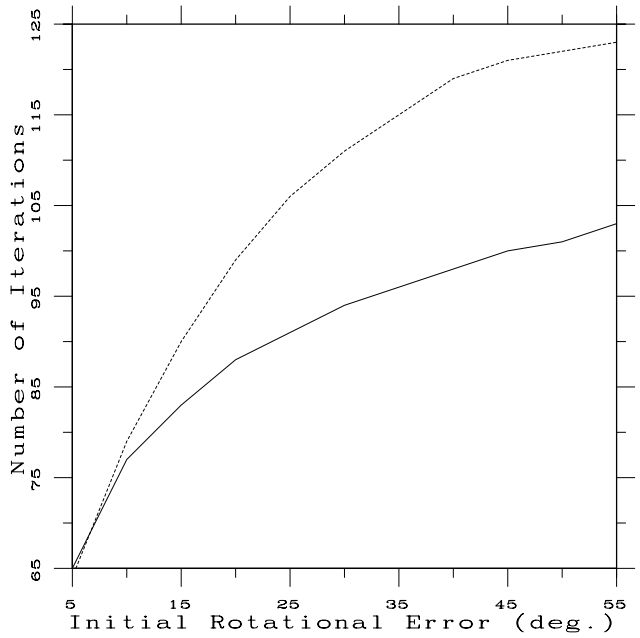


FIG. 5.26 - *Nombre d'itérations en fonction de l'erreur initiale (rotation pure) ; la courbe continue correspond à un Jacobien variable, la courbe interrompue à un Jacobien constant.*

5.9 Expérimentations

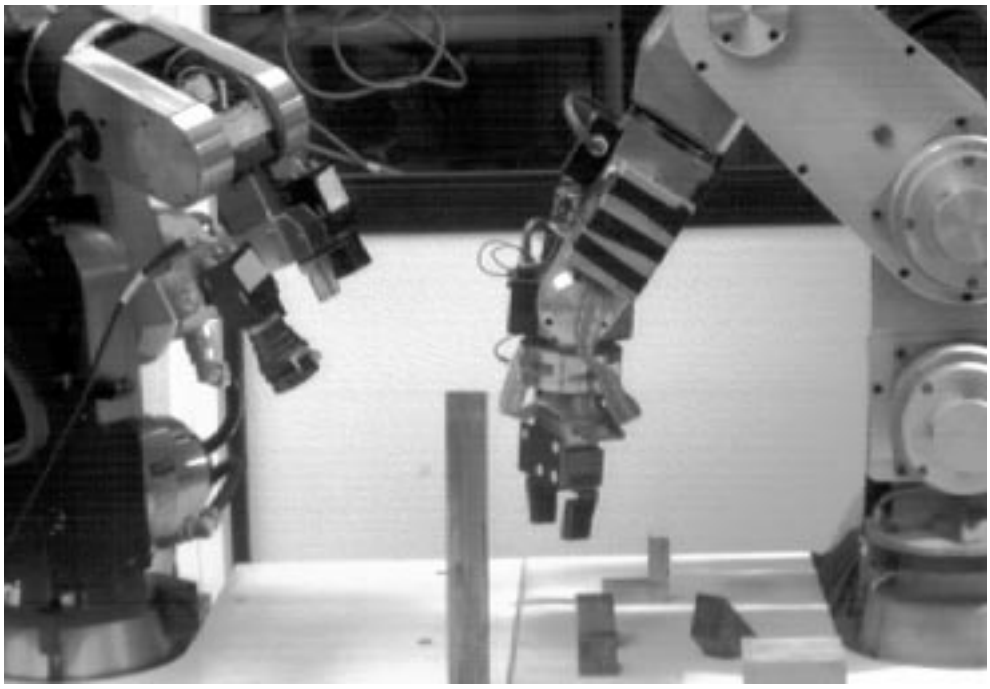


FIG. 5.27 - *Site expérimental.*

Nous disposons au LIFIA d'un système expérimental composé de deux robots manipulateurs (voir Figure 5.27). Un robot effectue les tâches d'asservissement et l'autre maintient une ou plusieurs caméras CCD. L'utilisation d'une caméra embarquée sur le bras d'un robot permet de positionner (d'une manière automatique ou pas) la caméra en un endroit optimal [TL95] [AAT93]: la pince du robot et les objets d'intérêt doivent être dans le champ visuel de la caméra. Nous avons vu que l'asservissement visuel nécessite l'utilisation des primitives 3D liées à la pince. Ces primitives peuvent être des amers naturels aussi bien qu'artificiels. Dans notre cas, ces amers sont constitués de quatre cibles blanches imprimées sur une plaque noire qui est collée à la pince du robot (Figure 5.27). Ainsi, les primitives géométriques sont les points 3D constitués par les centres de ces quatre cibles circulaires. La tâche de localisation de la pince dans l'image est devenue simple. Cependant, il faut que les coordonnées 3D de ces quatre points soient connues dans le repère de la pince. Autrement dit, il faut que la transformation rigide plaque-pince soit connue. Ce problème est résolu grâce à la technique du calibrage caméra-pince ($AX = XB$) (voir chapitre 2). Dans ce cas, une caméra fixe observe les déplacements du sytème plaque-pince.

5.9.1 Saisie d'objets

Bien que l'asservissement proposé dans ce chapitre puisse être appliqué à des différentes tâches robotiques, nous choisissons la tâche de la saisie d'objets. Cette tâche est un cas particulier des tâches nécessitant le positionnement relatif du robot par rapport à son

environnement. Comme la plupart des méthodes antérieures, notre approche est divisée en deux étapes :

1. ***étape hors ligne***: dans cette étape, la planification de la saisie automatique doit tenir compte de la morphologie de la pince du robot [FMN94], du modèle géométrique de l'objet, des modèles géométriques direct et inverse du robot, de la position relative pince-objet [KI91] et de la présence des obstacles [NH87]. De plus, cette planification doit prendre en considération la stabilité de la saisie [Per86].

2. ***étape en ligne***: cette étape comprend (i) le calcul de la position 2D finale de la pince dans l'image et (ii) l'asservissement visuel du robot.

Un exemple de cette deuxième étape est illustré sur la Figure 5.28. Le calcul de la position finale de la pince dans l'image s'effectue de la manière suivante:

- *traitement d'image*: On acquiert une image de l'objet qu'on veut saisir (Figure 5.28.a), on extrait les contours de cette image qu'on les approxime avec des segments de droite. Les extrémités de ces segments forment des jonctions (Figure 5.28.b). Ainsi, on peut construire un réseau de jonctions et de segments. Ce réseau est traité comme un graphe avec la possibilité de le diviser en plusieurs éléments connexes. La partie connexe qui ressemble le plus à un objet polyédrique sera choisie (Figure 5.28.d).

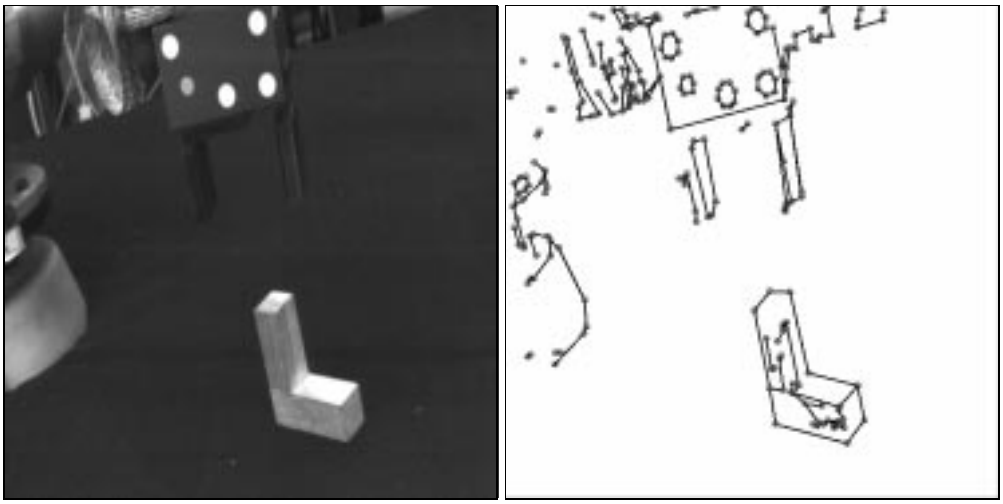
- *Prédiction du modèle de l'objet*: Le modèle 3D de l'objet (représentation fil de fer) est projeté dans une image en utilisant des paramètres intrinsèques et extrinsèques approximatives (Figure 5.28.c). Le choix de ces paramètres peut être interactif de façon que nous puissions obtenir une vue proche de celle de l'image réelle. La vue du modèle est aussi un réseau de jonctions et de segments.

- *Appariement image-modèle*: Afin de déterminer la matrice de projection objet-image, au moins 6 correspondances 3D/2D sont nécessaires. Pour ce faire, on compare le réseau associé à l'image réelle de l'objet avec le réseau associé à l'image du modèle. Nous pouvons noter que la présence du bruit et l'utilisation des paramètres approximatifs rendent très difficile la tâche de l'appariement image-modèle. La Figure 5.28.c présentent 9 correspondances jonction/jonction qui ont été obtenues en utilisant la méthode décrite dans [Gro93].

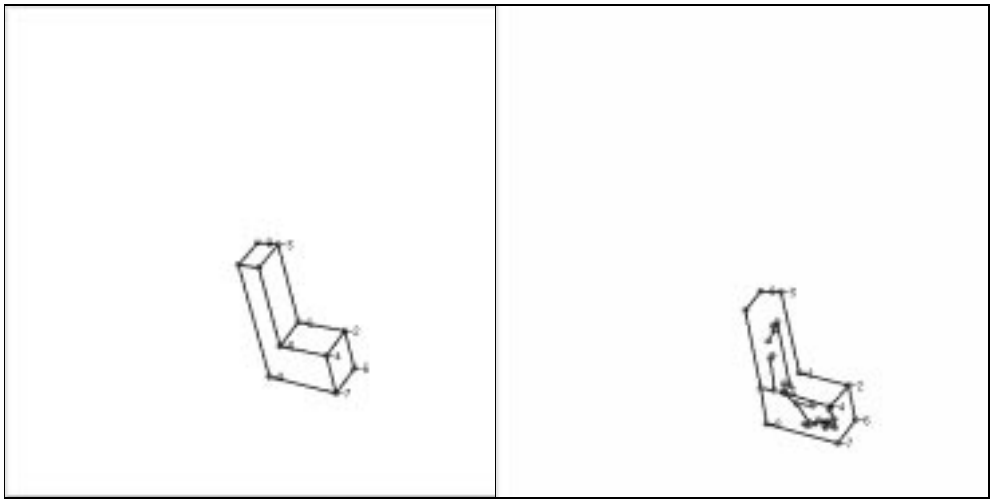
Enfin, en utilisant cette matrice, nous obtenons les valeurs désirées des coordonnées 2D de chaque cible (Figure 5.28.e). Le résultat de l'asservissement visuel tel qu'il est décrit dans la section 5.5 est présenté sur la (Figure 5.28.f).

5.9.2 Poursuite des cibles

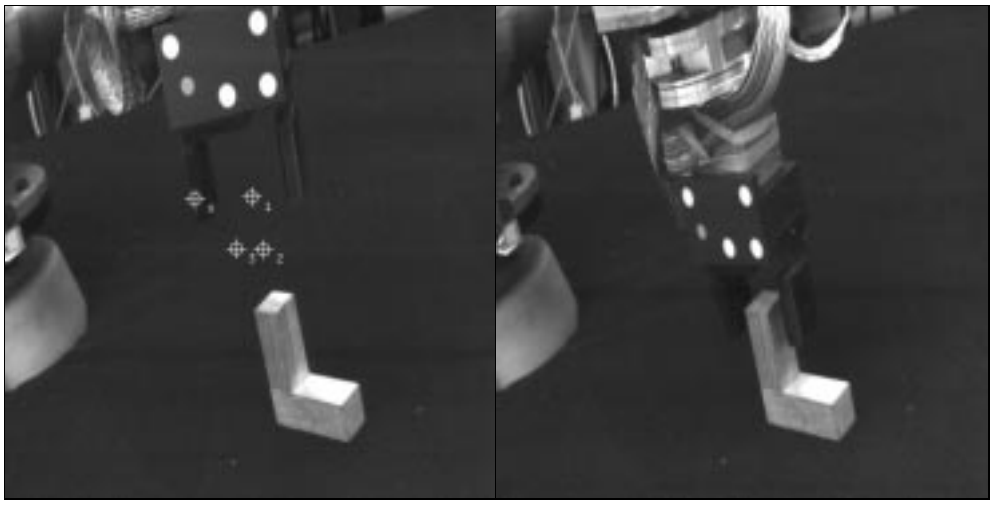
Nous n'utilisons pas de matériels spécialisés pour traiter le flot de données allant de la caméra vers l'ordinateur qui commande le robot. Le vecteur de mesures est constitué des coordonnées 2D des centres de gravité des quatre tâches claires. L'extraction des centres de gravité a une précision sub-pixel (de l'ordre du dixième de pixel). Ainsi, nous



a) b)



c) d)



e) f)

FIG. 5.28 - *Un exemple illustrant l'étape en ligne nécessaire à la saisie d'un objet polyédrique.*

devons calculer les moments d'ordre zéro et les moments d'ordre 1 des quatre tâches. Ce calcul peut être effectué soit par utilisation du matériel), soit par utilisation du logiciel (notre cas). Par conséquent, pour diminuer le temps de traitement nous devons adopter la technique de fenêtrage qui consiste à traiter uniquement les parties utiles de l'image acquise, c'est-à-dire, les parties qui encadrent les primitives 2D poursuivies. Avec cette technique, les nouvelles positions des fenêtres sont déduites à partir des centres de gravité calculés antérieurement. Cette technique réduit considérablement le temps de calcul et augmente la robustesse de la poursuite dans l'image. Deux architectures sont proposées:

1. la première est illustrée sur la Figure 5.29 et consiste à utiliser pour chaque tâche claire une fenêtre rectangulaire de taille constante. La taille de chaque fenêtre doit être choisie de telle sorte que la fenêtre puisse contenir la tâche claire dans deux images successives.

2. la deuxième est illustrée sur la Figure 5.30 et consiste à utiliser pour chaque tâche claire une fenêtre rectangulaire de taille variable. Cette taille est estimée à chaque itération en fonction de la profondeur des cibles 3D, obtenue par le calcul de la pose de ces cibles par rapport à la caméra. La nouvelle position de chaque fenêtre (son centre) est prédite de la façon suivante: on ajoute aux coordonnées des centres de gravité courants un déplacement $\mathbf{d}$ estimé à partir de la vitesse 2D souhaitée:

$$\mathbf{d} = g\,(\mathbf{s}^\star - \mathbf{s})\,t_e$$

où t_e représente le temps nécessaire à une itération.

Il est à noter que la deuxième architecture permet d'obtenir une cadence d'échantillonnage plus élevée que celle obtenue avec la première architecture. Expérimentalement, nous avons montré que la commande utilisant un Jacobien variable permet l'emploi de la deuxième architecture avec succès, alors qu'avec la loi de commande utilisant un Jacobien constant les fenêtres de recherche peuvent rater les cibles claires.

5.9.3 Résultats expérimentaux

Nous donnons sur la Figure 5.31 les résultats obtenus pour réaliser la tâche de la saisie. Ces résultats ont été obtenus avec un Jacobien variable. Sur la Figure 5.31.a sont présentées la position initiale et finale de la pince ainsi que les trajectoires 2D des quatre cibles claires. Sur la Figure 5.31.b sont présentées la courbe de l'erreur et la courbe de la vitesse de translation de la pince lorsque le gain est égal à 0.4. La Figure 5.31.c illustre ces mêmes courbes mais avec un gain croissant.

Sur la Figure 5.32 est présenté le comportement dynamique des deux types de commande pour un même écart initial. Les figures de gauche ont été obtenues lors d'un asservissement utilisant un Jacobien fixe, les figures de droite ont été obtenues lors d'un asservissement utilisant un Jacobien variable. Dans le premier cas, on peut s'apercevoir que toutes les composantes du vecteur d'erreur ont dépassé leur valeur initiale avant la convergence.

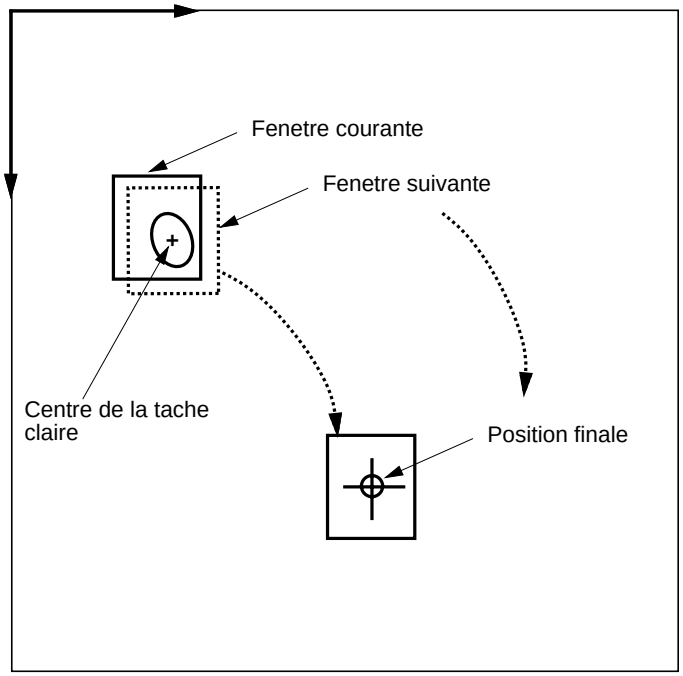


FIG. 5.29 - *Première technique de suivi des tâches claires: les fenêtres ont une taille constante, le centre d'une tâche claire est en même temps le centre de la fenêtre suivante.*

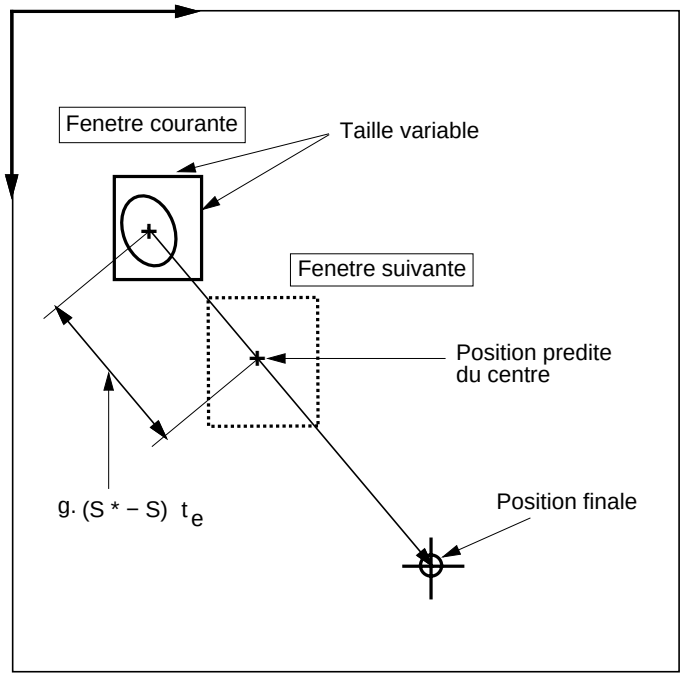
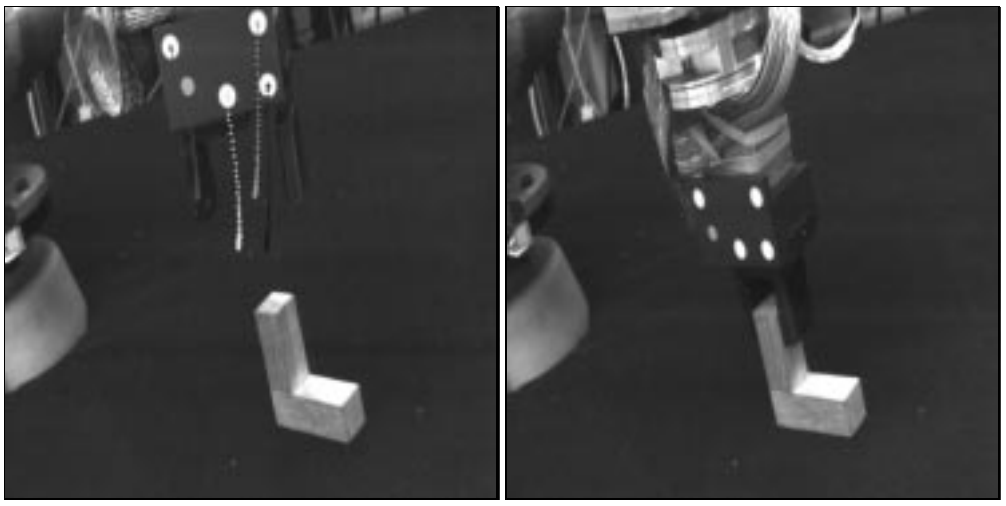
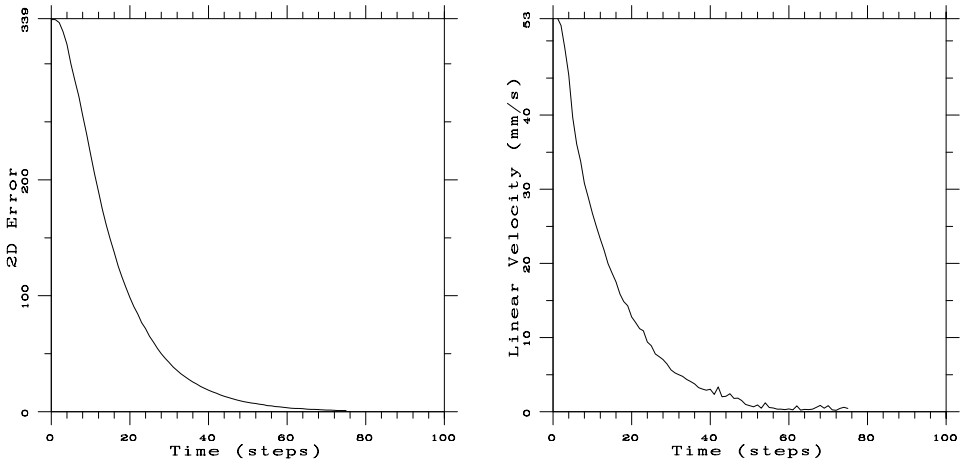


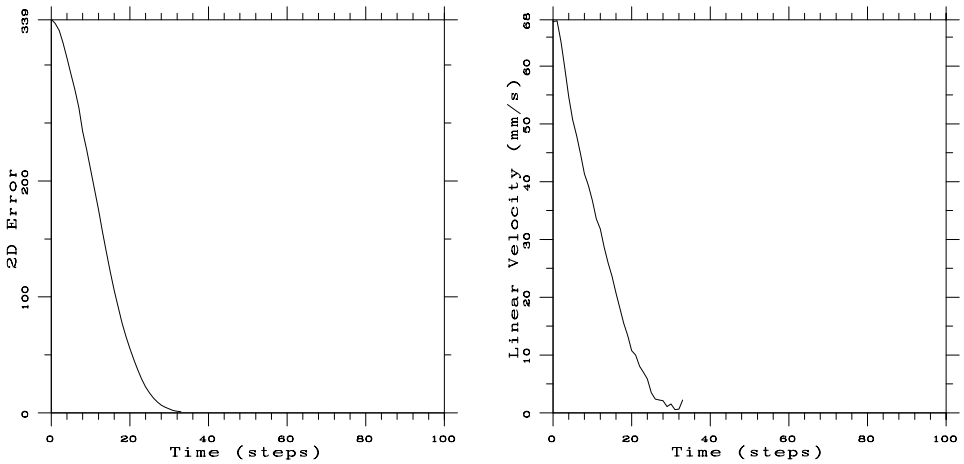
FIG. 5.30 - *Deuxième technique de suivi des tâches claires: les fenêtres ont une taille variable en fonction de la profondeur, le centre de la fenêtre suivante est prédite à partir de la vitesse 2D désirée des cibles qui a permis le calcul de la vitesse 3D.*



a) Trajectoires des cibles et la position finale de la pince.

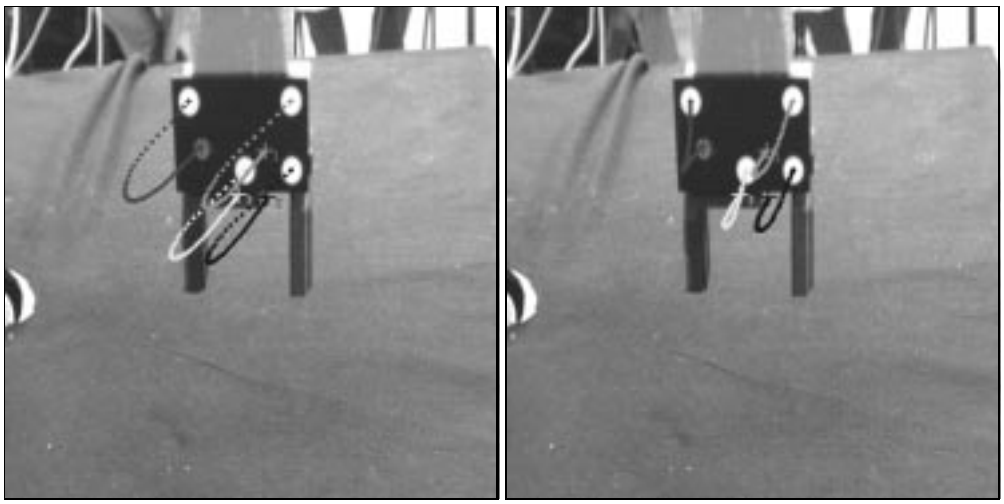


b) L'erreur $\|\mathbf{s}^\star - \mathbf{s}\|$ et la vitesse 3D de la pince associées à un gain fixe.

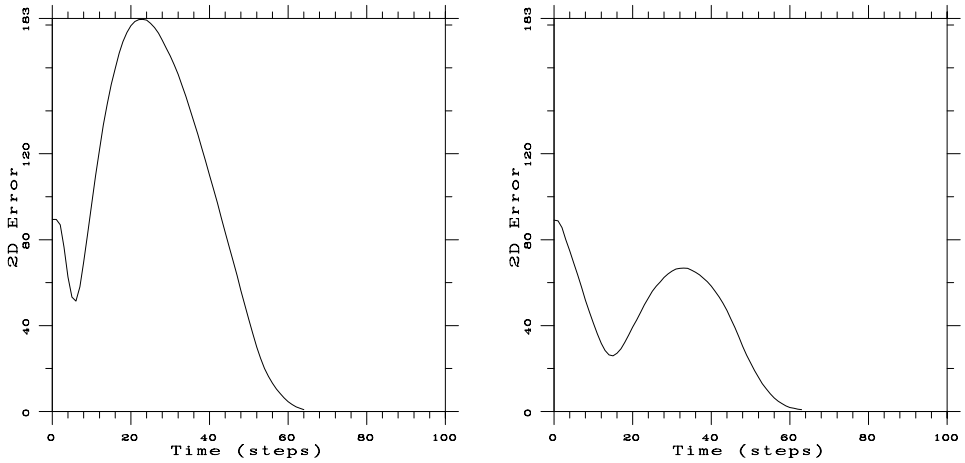


c) L'erreur $\|\mathbf{s}^\star - \mathbf{s}\|$ et la vitesse 3D de la pince associées à un gain croissant.

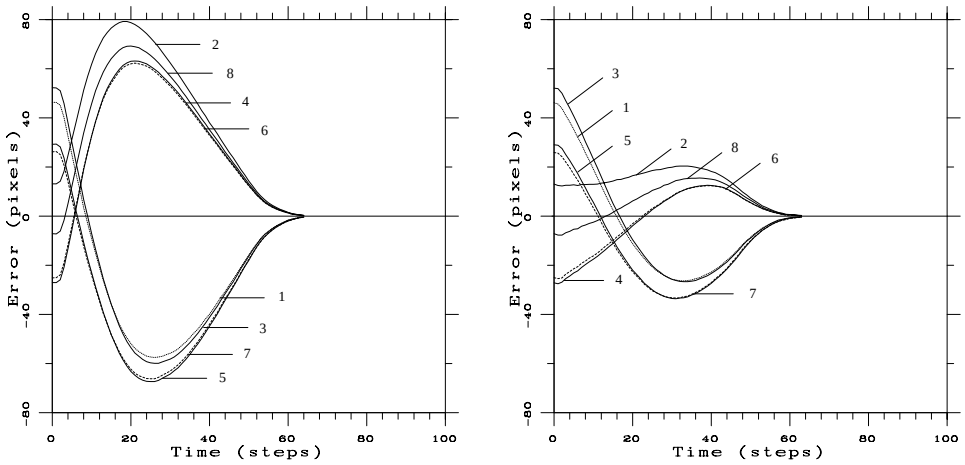
FIG. 5.31 - *Les résultats de l'asservissement appliquée à la tâche de la saisie.*



a) Trajectoires des cibles.



b) L'erreur $\|\mathbf{s}^\star - \mathbf{s}\|$.



c) L'erreur $\mathbf{s}^\star - \mathbf{s}$.

FIG. 5.32 - *Comparaison des deux types de commande : les figures de gauche correspondent à un Jacobien fixe alors que celles de droite correspondent à un Jacobien variable.*

5.10 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle approche permettant le contrôle du mouvement d'un robot manipulateur à l'aide d'une caméra. Cette approche s'apparente à la méthode *commande référencée capteur*. Avec cette approche, la précision du positionnement final du robot dans son environnement ne dépend ni du modèle de ce robot ni du modèle de la caméra jouant le rôle du capteur. Seule l'extraction des primitives 2D doit être précise. Un large éventail de tâches robotiques peut être ainsi réalisé grâce à cette approche. Cette méthode est souple en ce sens qu'elle peut être utilisée avec un robot et une caméra mal calibrés. De plus, la caméra est indépendante du robot asservi. Nous avons abordé l'aspect modélisation de l'interaction capteur/environnement qui n'est autre que la relation différentielle entre le mouvement relatif de la scène (la pince du robot) et le mouvement des primitives 2D images de cette scène. Nous avons montré que l'utilisation d'une relation différentielle qui varie en fonction de la position courante de la pince a des avantages par rapport à l'utilisation d'une relation constante. Comme le modèle 3D de la pince est connue, l'estimation temps réel des paramètres du Jacobien nécessite le calcul de la pose entre la pince et la caméra. Ce calcul est effectué grâce à l'algorithme de pose proposé dans le chapitre 4. Cet algorithme calcule la pose d'un objet (dans notre cas la pince du robot) par approximations successives avec un modèle de projection para-perspectif et ne requiert que 2 ms, qui est un temps négligeable si l'on se rappelle que le temps nécessaire à une itération de la boucle fermée est de l'ordre de 80-100 ms. Nous avons caractérisé la performance de l'approche proposée par un grand nombre d'expériences sur des données synthétiques avec ou sans erreurs introduites sur les paramètres de calibrage. Nous avons également comparé les deux types de commande : le premier type utilise un Jacobien qui varie à chaque pas en fonction de la situation 3D de la pince ; l'autre, utilise une valeur constante du Jacobien calculée à la situation finale. Nous avons également appliqué cette approche à la tâche de la saisie d'objets sur site expérimental.

Conclusion

Dans cette thèse nous avons traité principalement le problème de l'intégration vision/robotique. Les applications qui en découlent sont très diversifiées aussi bien pour les robots manipulateurs que pour les robots mobiles. Toutefois, cette intégration a posé plusieurs problèmes tant au niveau conceptuel qu'au niveau réalisation. On peut citer parmi eux, les phases de modélisation des éléments physiques constituant un système capteur/robot. L'identification des paramètres d'un modèle donné s'appelle *calibrage*. Ce calibrage peut rendre très utiles les informations fournies par le capteur visuel.

Ainsi, une caméra embarquée sur l'effecteur d'un robot peut imposer deux types de calibrage : i) identification de la relation géométrique caméra-effecteur et ii) identification des paramètres de la caméra elle-même.

Nous avons présenté une nouvelle formulation pour le calcul de la relation géométrique caméra-robot. Cette formulation ne nécessite pas une connaissance explicite des paramètres intrinsèques et extrinsèques de la caméra. De plus, nous avons proposé une nouvelle méthode de résolution qui est plus robuste que les méthodes algébriques. Cette robustesse est dûe au fait que les inconnues (rotation et translation) sont simultanément calculées.

Nous avons généralisé ce calibrage pour déterminer les relations géométriques entre la base du robot et une scène 3D. Nous avons proposé une nouvelle méthode de résolution qui est plus robuste que la méthode linéaire existante.

De plus nous avons proposé une approche originale permettant l'autocalibrage du système caméra-pince (paramètres de la caméra et la relation géométrique caméra-pince). Cet autocalibrage permet également la conversion d'une structure projective en une structure euclidienne. La majeure différence avec le calibrage caméra-pince réside dans le fait que la caméra observe une scène 3D dont le modèle est inconnu.

La deuxième partie de notre travail concerne la modélisation de la caméra et plus particulièrement le problème de la localisation d'objets à partir de mesures images. Nous avons vu qu'on peut utiliser les techniques de l'algèbre linéaire pour déterminer, d'une manière itérative, les paramètres de la pose correspondant au modèle perspectif associé à la caméra. Pendant le processus itératif le modèle utilisé pour la caméra est une projection para-perspective. A la convergence, ces approximations successives fournissent la pose de l'objet qui correspond au modèle perspectif de la caméra. Cette méthode constitue une variante d'une méthode récente qui utilise le modèle perspectif faible [DD95]. Les avantages de notre méthode par rapport à cette dernière sont :

– La convergence et sa vitesse ne dépendent pas de la translation latérale de l'objet

par rapport à l'axe optique de la caméra.

– En général, elle nécessite un temps de calcul plus petit.

Nous avons montré par une étude comparative que la précision de la méthode linéaire itérative est comparable à celle des méthodes numériques. Les deux avantages associés à la méthode linéaire itérative sont :

1. Elle n'a pas besoin d'une estimation initiale.
2. Elle converge rapidement et est adaptée aux applications temps réel.

Dans la troisième partie de notre travail, nous avons abordé le problème de la coordination entre une caméra et un robot. Cette coordination a suscité de nombreux travaux de recherche. Toutefois, ces travaux se heurtent à des problèmes dûs au fait que la caméra et le robot sont mal calibrés. Le contrôle visuel en boucle fermée se divise en deux catégories : i) asservissement en position et ii) asservissement visuel.

– La première catégorie permet de s'affranchir uniquement des incertitudes affectant le modèle du robot mais elle dépend de l'identification du modèle de la caméra. En effet, l'estimation de la position nécessite une interprétation des informations extraites de l'image.

– La deuxième catégorie peut s'affranchir des incertitudes affectant les modèles du robot et de la caméra puisqu'elle est basée sur une régulation dans l'image.

En robotique, l'information relative est souvent la seule information requise pour certaines tâches. Ainsi, les tâches de saisie et de suivi d'objets ne requièrent que la position relative entre le robot et les objets. Ceci nous a permis de calculer le motif final dans l'image indépendamment des paramètres de la caméra (intrinsèques et extrinsèques).

Nous avons proposé une approche de contrôle utilisable avec un système mal calibré. Dans cette approche, une seule caméra fixe observe le robot et son environnement. Cette approche appartient à la catégorie "*asservissement visuel*". Elle est basée sur la notion du Jacobien d'image qui représente la relation différentielle entre les mouvements 3D et les mouvements 2D résultants. La modélisation de ce Jacobien montre sa dépendance de la relation spatiale entre le capteur et la pince du robot. L'utilisation de la méthode linéaire itérative va permettre d'estimer ce Jacobien en temps réel. Nous avons démontré qu'une telle estimation améliore considérablement le comportement dynamique de l'asservissement et plus particulièrement la convergence et la poursuite de cibles. Dans cette approche, la précision du positionnement final dépend uniquement de l'extraction des primitives 2D. Cette approche a été testée sur un site expérimental dans l'objectif de la saisie automatique d'objets.

Ce travail ouvre plusieurs perspectives d'application :

– Intégration du calibrage des robots avec le calibrage caméra/pince.

– Utilisation des primitives naturelles comme amers.

– Exploitation de l'approche de contrôle dans la réalisation des tâches très complexes.

Bibliographie

[AAT93] S. Abrams, P. K. Allen, and K. A. Tarabanis. Dynamic sensor planning. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993.

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, September 1987.

[AL87] N. Ayache and F. Lustman. Fast and reliable passive strereovision using three cameras. In *Proceedings of the 1st International Conference on Computer Vision, London, England*, pages 422–427, 1987.

[Alo90] Y. Aloimonos. Perspective approximations. *Image and Vision Computing*, 8(3):177–192, August 1990.

[ATYM93] P. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, 9(2):152–165, April 1993.

[AWB88] J. Aloimonos, I. Weiss, and A. Bandyopadhay. Active vision. *International Journal of Computer Vision*, pages 333–356, 1988.

[Aya89] N. Ayache. *Vision Stéréoscopique et Perception Multisensorielle: Application à la Robotique Mobile.* Science Informatique. InterEditions, 1989.

[BB82] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice Hall, 1982.

[BJP93] Z. Bien, W. Jang, and J. Park. Characterization and use of feature-jacobian matrix for visual servoing. In K. Hashimoto, editor, *Visual Servoing*, pages 317–363. World Scientific, 1993.

[BMV93] B. Boufama, R. Mohr, and F. Veillon. Euclidian constraints for uncalibrated reconstruction. In *Proceedings Fourth International Conference on Computer Vision*, pages 466–470, Berlin, Germany, May 1993. IEEE Computer Society Press, Los Alamitos, Ca.

[BMZ92] P. Beardsley, D. Murray, and A. Zisserman. Camera calibration using multiple images. In G. Sandini, editor, *Computer Vision - ECCV 92, Proceedings Second European Conference on Computer Vision, Santa Margherita Ligure, May 1992*, pages 312–320. Springer Verlag, May 1992.

[Bou93] S. Boukir. *Reconstruction 3D d'un Environnement Statique par Vision Active*. PhD thesis, Université de Rennes I, IRISA Rennes, France, October 1993.

[Bou94] B. Boufama. *Reconstruction Tridimensionnelle en Vision par Ordinateur : cas des caméra non étalonnées.* Thèse de doctorat, Institut National Polytechnique de Grenoble, December 1994.

[Cas87] P. Casteljau. *Les quaternions*. Hermès, 1987.

[Cha90] F. Chaumette. *La relation vision-commande: théorie et applications à des tâches robotiques*. PhD thesis, Université de Rennes I, July 1990.

[Che91] H. Chen. A screw motion approach to uniqueness analysis of head-eye geometry. *In Proceedings Computer Vision and Pattern Recognition*, pages 145–151, June 1991.

[Cho92] J. Chou. Quaternion kinematic and dynamic differential equations. *IEEE Transactions on Robotics and Automation*, 8(1):53–64, February 1992.

[CK91] J. C. K. Chou and M. Kamel. Finding the position and orientation of a sensor on a robot manipulator using quaternions. *International Journal of Robotics Research*, 10(3):240–254, June 1991.

[CLA93] Y. Chang, X. Lebegue, and J. K. Aggarwal. Calibrating a mobile camera's parameters. *Pattern Recognition*, 26(1):75–88, 1993.

[CMCK93] E. Coste-Manière, P. Couvignou, and P. Khosla. Robotic contour following based on visual servoing. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 716–722, Yokohama, Japan, July 1993.

[Cor93a] P. I. Corke. Video-rate robot visual servoing. In K. Hashimoto, editor, *Visual Servoing*, pages 257–283. World Scientific, 1993.

[Cor93b] P. I. Corke. Visual control of robot manipulators – a review. In K. Hashimoto, editor, *Visual Servoing*, pages 1–32. World Scientific, 1993.

[CRE91] F. Chaumette, P. Rives, and B. Espiau. Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, volume 3, pages 2248–2253, Sacramento, California, April 1991.

[DAO94] C. Doignon, G. Abba, and E. Ostertag. Recognition and localization of solid objects by a monocular vision system for robotic tasks. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 3, pages 2007–2014, September 1994.

[DD92] D. F. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. In G. Sandini, editor, *Computer Vision – ECCV 92, Proceedings Second European Conference on Computer Vision, Santa Margherita Ligure, May 1992*, pages 335–343. Springer Verlag, May 1992.

[DD95] D. DeMenthon and L. Davis. Model-based object pose in 25 line of code. *International Journal of Computer Vision*, 15:123–141, June 1995.

[DeM93] D. F. DeMenthon. *De la Vision Artificielle à la Réalité Synthétique: Système d'interaction avec un ordinateur utilisant l'analyse d'images vidéo.* PhD thesis, Université Joseph Fourier - Grenoble I, Laboratoire TIMC/IMAG, October 1993.

[DK88] E. Dombre and W. Khalil. *Modélisation et Commande des Robots.* Hermès, Paris, 1988.

[DRLR89] M. Dhome, M. Richetin, J.T. Lapreste, and G. Rives. Determination of the Attitude of 3D Objects from a Single Perspective View. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989.

[DYL93] M. Dhome, A. Yassine, and J. M. Lavest. Determination of the pose of an articulated object from a single perspective view. In *Procceedings of the 4th British Machine Vision Conference*, volume 1, pages 95–104, September 1993.

[ECR92] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, June 1992.

[Fau88] O. D. Faugeras. Quelques pas vers la vision artificielle en trois dimensions. *Technique et Science Informatiques*, 7(6):547–590, 1988.

[Fau92] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig. In G. Sandini, editor, *Computer Vision – ECCV 92, Proceedings Second European Conference on Computer Vision, Santa Margherita Ligure, May 1992*, pages 563–578. Springer Verlag, May 1992.

[Fau93] O. D. Faugeras. *Three Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, Boston, 1993.

[FB81] M.A. Fischler and R.C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[FDD94] C. Fagerer, D. Dickmanns, and E. D. Dickmanns. Visual grasping with long delay time of a free floating object in orbit. *Autonomous Robots*, 1(1):53–68, 1994.

[FH86] O.D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-d objects. *International Journal of Robotics Research*, 5(3):27–52, Fall 1986.

[Fle90] R. Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, 1990.

[FLM92] O. D. Faugeras, Q. T. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In G. Sandini, editor, *Computer Vision – ECCV 92, Proceedings Second European Conference on Computer Vision, Santa Margherita Ligure, May 1992*, pages 321–334. Springer Verlag, May 1992.

[FLM93] J. T. Feddema, C. S. G. Lee, and O. R. Mitchell. Feature-based visual servoing of robotic systems. In K. Hashimoto, editor, *Visual Servoing*, pages 105–138. World Scientific, 1993.

[FMN94] O. Fuentes, H. F. Marengoni, and R. C. Nelson. Vision-based planning and execution of precision grasps. Technical Report 546, The University of Rochester, New York, 1994.

[FT86] O. D. Faugeras and G. Toscani. The calibration problem for stereo. In *Proc. Computer Vision and Pattern Recognition*, pages 15–20, Miami Beach, Florida, USA, June 1986.

[FT87] O.D. Faugeras and G. Toscani. Camera calibration for 3D computer vision. In *Proceedings of International Workshop on Machine Vision and Machine Intelligence, Tokyo, Japan*, 1987.

[GMW89] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, London, 1989.

[Gro93] P. Gros. Matching and clustering: two steps towards automatic model generation in computer vision. In *Proceedings of the AAAI Fall Symposium Series: Machine Learning in Computer Vision: What, Why, and How?, Raleigh, North Carolina, USA*, pages 40–44, October 1993.

[HA91] L. Huang and Y. Aloimonos. Relative depth from motion using normal flow: an active and purposive solution. In *Proceedings of the 1991 IEEE Workshop on Visual Motion*, pages 196–204, October 1991.

[HA94] K. Hosoda and M. Asada. Versatile visual servoing without knowledge of true jacobian. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, pages 186–193, September 1994.

[Hag94] G. D. Hager. Real-time feature tracking and projective invariance as a basis for hand-eye coordination. In *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 533–539, Seattle, Washington, June 1994.

[Hag95] G. Hager. Calibration-free visual control using projective invariance. In *Proceedings of the 1995 IEEE International Conference on Computer Vision*, June 1995.

[Har94a] R. I. Hartley. An algorithm for self calibration from several views. In *Proceedings Computer Vision and Pattern Recognition Conference*, pages 908–912, Seattle, Washington, June 1994. IEEE Computer Society Press.

[Har94b] R. I. Hartley. Euclidian reconstruction from uncalibrated views. In Mundy Zisserman Forsyth, editor, *Applications of Invariance in Computer Vision*, pages 237–256. Springer Verlag, Berlin Heidelberg, 1994.

[HC93] N. Hollinghurst and R. Cipolla. Uncalibrated stereo hand-eye coordination. In *Proceedings of the Fourth British Machine vision Conference (BMVC 93)*, 1993.

[HCLL89] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47(1):33–44, July 1989.

[HCM95] G. D. Hager, W. C. Chang, and A. S. Morse. Robot hand-eye coordination based on stereo vision. *Control Systems*, 15(1):30–39, February 1995.

[HD95] R. Horaud and F. Dornaika. Hand-eye calibration. *International Journal of Robotics Research*, 14(3):195–210, June 1995.

[HDBM94] R. Horaud, F. Dornaika, B. Boufama, and R. Mohr. Self calibration of a stereo head mounted onto a robot arm. In *Proc. Third European Conference on Computer Vision*, pages 455–462, Stockholm, Sweden, May 1994.

[HGH94] G. D. Hager, G. Grunwald, and G. Hirzinger. Feature-based visual servoing and its application to telerobotics. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, pages 164–171, September 1994.

[HJL+89] R. B. Haralick, H. Joo, C-N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1426–1445, 1989.

[HKEK91] K. Hashimoto, T. Kimoto, T. Ebine, and H. Kimura. Manipulator control with image-based visual servo. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, volume 3, pages 2267–2272, Sacramento, California, April 1991.

[HM93] R. Horaud and O. Monga. *Vision par ordinateur: outils fondamentaux*. Editions Hermès, Paris, 1993.

[HN91] R. J. Holt and A. N. Netravali. Camera calibration problem: some new results. *CGVIP-Image Understanding*, 54(3):368–383, November 1991.

[HN92] T. Horsch and H. Nolzen. Local motion planning avoiding obstacles with dual quaternions. In *Procceedings of the 1992 IEEE International Conference on Robotices and Automation*, volume 1, pages 241–245, May 1992.

[Hor87] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Amer. A.*, 4(4):629–642, 1987.

[HV93] H. Henocq and C. Venaille. Calibration dynamique d'une caméra CCD par transformée inverse. *Revue technique Thomson-CSF*, 25(1):65–82, Mars 1993.

[HYH85] Y. Hung, P.-S. Yeh, and D. Harwood. Passive ranging to known planar point sets. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 80–85, Saint-Louis, Missouri, USA, March 1985.

[KD94] K. Kinoshita and K. Deguchi. Simultaneous determination of camera pose and intrinsic parameters by visual servoing. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, pages 285–289, October 1994.

[KI91] S. B. Kang and K. Ikeuchi. A framework for recognizing grasps. Technical Report CMU-RI-TR-91-24, Carnegie Mellon University, Pennsylvania, 1991.

[KL91] Y. Kay and S. Lee. A robust 3-D motion estimation with stereo cameras on a robot manipulator. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1102–1107, Sacramento, California, April 1991.

[LHB+86] Z. Lin, T. Huang, S. Blostein, H. Lee, and E. Margerum. Motion estimation from 3-D point sets with and without correspondences. In *Procceedings Computer Vision and Pattern Recognition*, pages 194–201, Miami-Beach, Florida, USA, June 1986.

[LHF90] Y. Liu, T. S. Huang, and O. D. Faugeras. Determination of camera location from 2-d to 3-d line and point correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):28–37, January 1990.

[Low87] D. Lowe. Three-dimensional Object Recognition from Single Two-dimensional Images. *Artificial Intelligence*, 31:355–395, 1987.

[LT87] R.K. Lenz and R.Y. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3D machine vision metrology. In *Proceedings of* Ieee *International Conference on Robotics and Automation, Raleigh, North Carolina, USA*, pages 68–75, 1987.

[LTM91] S. Larabi, B. Thiesse, and P. Marthon. Utilisation de la géométrie projective en vision stéréoscopique binoculaire : appariement de stéréogrammes de segments. In *Actes du 8ème Congrès AFCET de Reconnaissance des Formes et Intelligence Artificielle, Lyon – Villeurbanne, France*, volume 1, pages 329–340. AFCET, November 1991.

[MBB94] R. Mohr, B. Boufama, and P. Brand. Accurate projective reconstruction. In Mundy Zisserman Forsyth, editor, *Applications of Invariance in Computer Vision*, pages 257–276. Springer Verlag, Berlin Heidelberg, 1994.

[Mes94] M. Mesrabi. *Contrôle du regard pour un système de vision active*. PhD thesis, Institut National Polytechnique de Grenoble, March 1994.

[MF92] S. J. Maybank and O. D. Faugeras. A theory of self calibration of a moving camera. *International Journal of Computer Vision*, 8(2):123–151, August 1992.

[MKY+93] N. Maru, H. Kase, S. Yamada, A. Nishikawa, and F. Miyazaki. Manipulator control by visual servoing with the stereo vision. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1866–1870, Yokohama, Japan, July 1993.

[NH87] W.S. Newman and N. Hogan. High speed robot control and obstacle avoidance using dynamic potential functions. In *Proceedings of* IEEE *International Conference on Robotics and Automation, Raleigh, North Carolina, USA*, pages 14–24, 1987.

[NH93] A. Nilsson and P. Holmberg. Robot-sensor system integration by means of 2-D vision and ultrasonic sensing for localisation and recognition of objects. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1793–1799, Yokohama, Japan, July 1993.

[NK94] B. J. Nelson and P. K. Khosla. The resolvability ellipsoid for visual servoing. In *Proceedings of the 1994 Conference on Computer Vision and Pattern Recognition (CVPR94)*, Seattle, Washington, June 1994.

[NPK93] B. Nelson, N. P. Papanikolopoulos, and P. K. Khosla. Dynamic sensor placement using controlled active vision. In *Proceedings of the International Federation of Automatic Control (IFAC) 1993 World Congress*, Sydney, Australia, July 1993.

[NTG92] A. Nagchaudhuri, M. Thint, and D. Garg. Camera-robot transform for vision-guided tracking in a manufacturing work cell. *Journal of Intelligent and Robotic Systems.*, 5:283–298, 1992.

[ODD93] D. Oberkampf, D. F. DeMenthon, and L. S. Davis. Iterative pose estimation using coplanar feature points. In *Proceedings Computer Vision and Pattern Recognition*, pages 626–627, New York, June 1993. IEEE Computer Society Press, Los Alamitos, Ca.

[Per86] J. Pertin. *Modélisation du Raisonnement Géométrique pour la Programmation des Robots*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, March 1986.

[PFTW88] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Wetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

[PHYP95] T. Q. Phong, R. Horaud, A. Yassine, and D. T. Pham. Object pose from 2-D to 3-D point and line correspondences. *International Journal of Computer Vision*, 15(3):225–243, July 1995.

[PK94] C. J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In Jan-Olof Eklundh, editor, *Computer Vision - ECCV 94, Proceedings Third European Conference on Computer Vision, Stockholm, May 1994*, volume 2, pages 97–108. Springer Verlag, May 1994.

[PKK91] N. Papanikolopoulos, P. K. Khosla, and T. Kanade. Vision and control techniques for robotic visual tracking. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, volume 1, pages 857–864, Sacramento, California, April 1991.

[PKK93] N. Papanikolopoulos, P. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision. *IEEE Transactions on Robotics and Automation*, 9(1):14–35, February 1993.

[PM94] F. Park and B. Martin. Robot sensor calibration: solving $AX = XB$ on the euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, October 1994.

[PNK94] N. P. Papanikolopoulos, B. Nelson, and P. K. Khosla. Six degree-of-freedom hand/eye visual tracking with uncertain parameters. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 1, pages 174–179, San Diego, California, May 1994.

[PS90] P. Puget and T. Skordas. An optimal solution for mobile camera calibration. In O. Faugeras, editor, *Proceedings of the 1st European Conference on Computer Vision, Antibes, France*, pages 187–198. Springer-Verlag, April 1990.

[PYM89] A. K. Pradeep, P. J. Yoder, and R. Mukundan. On the use of duals matrix exponentials in robotics kinematics. *International Journal of Robotics Research*, 8(5):57–66, October 1989.

[Rob94] L. Robert. Camera calibration without feature extraction. Technical Report 2204, INRIA Sophia-Antipolis, France, February 1994.

[SA89] Y. C. Shiu and S. Ahmad. Calibration of wrist mounted robotic sensors by solving homogeneous transform equations of the form $AX = XB$. *IEEE Journal of Robotics and Automation*, 5(1):16–29, February 1989.

[SBC94] V. Sundareswaran, P. Bouthemey, and F. Chaumette. Active camera self-orientation using dynamic image parameters. In Jan-Olof Eklundh, editor, *Computer Vision – ECCV 94, Proceedings Third European Conference on Computer Vision, Stockholm, May 1994*, volume 2, pages 111–116. Springer Verlag, May 1994.

[SBE91] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control (The Task Function Approach)*. 22. Clarendon Press, Oxford, 1991.

[Sch92] A. Schrott. Feature-based camera-guided grasping by an eye-in-hand robot. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, volume 2, pages 1832–1837, Nice, France, May 1992.

[SG95] T. Schnackertz and R. Grupen. A control basis for visual servoing tasks. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, 1995.

[SH94] R. Sharma and S. Hutchinson. On the observability of robot motion under active camera control. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 1, pages 162–167, San Diego, California, May 1994.

[SHK93] D. Simon, M. Hebert, and T. Kanade. Real-time 3D pose estimation using a high-speed range sensor. Technical Report CMU-RI-TR-93-24, Robotics Institute, Carnegie Mellon University, November 1993.

[SK89] T. Shakunaga and H. Kaneko. Perspective angle transform: Principle of shape from angles. *International Journal of Computer Vision*, 3(3):239–254, September 1989.

[SK94] R. Szeliski and S.B. Kang. Recovering 3D Shape and Motion from Image Streams using Nonlinear Least Squares. *Journal of Visual Communication and Image Representation*, 1(5):10–28, 1994.

[Sko91] G. Skofteland. Computing position and orientation of a freeflying polyhedron from 3D. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, volume 1, pages 150–156, Sacramento, California, April 1991.

[SMH91] A. E. Samuel, P. R. McAree, and K. H. Hunt. Unifiying screw geometry and matrix transformations. *International Journal of Robotics Research*, 10(5):454–472, October 1991.

[SW80] A. C. Sanderson and L. E. Weiss. Image-based visual servo control using relational graph error signals. In *Proceedings IEEE*, pages 1074–1077, 1980.

[TL88] R.Y. Tsai and R.K. Lenz. Real Time Versatile Robotics Hand/Eye Calibration using 3D Machine Vision. In *IEEE International Conference on Robotics and Automation*, pages 554–561, Philadelphia, Penn, USA, April 1988.

[TL89] R.Y. Tsai and R.K. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Journal of Robotics and Automation*, 5(3):345–358, June 1989.

[TL95] B. Triggs and C. Laugier. Automatic camera placement for robot vision tasks. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, 1995.

[Tsa87a] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[Tsa87b] R.Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE *Journal of Robotics and Automation*, 3(4):323–344, 1987.

[Ume91] S. Umeyama. Least-squares estimation of transformation parameters between two point pattern. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, April 1991.

[Wan92] C.-C. Wang. Extrinsic calibration of a robot sensor mounted on a robot. *IEEE Transactions on Robotics and Automation*, 8(2):161–175, April 1992.

[WMSM91] W. J. Wolfe, D. Mathis, C. W. Sklair, and M. Magee. The perspective view of three points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):66–73, January 1991.

[WS93] R. G. Willson and S. A. Shafer. What is the center of the image? Technical Report CMU-CS-93-122, School of Computer Science, Carnegie Mellon University, April 1993.

[WSV91] M. W. Walker, L. Shao, and R. A. Volz. Estimating 3-D location parameters using dual numbers quaternions. *CGVIP-Image Understanding*, 54(3):358–367, November 1991.

[WWR93] S. W. Wijesoma, D. Wolfe, and R. J. Richards. Eye-to-hand coordination for vision-guided robot control applications. *International Journal of Robotics Research*, 12(1):65–78, February 1993.

[YA94] B. H. Yoshimi and P. K. Allen. Active, uncalibrated visual servoing. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 1, pages 156–161, San Diego, California, May 1994.

[YAMU94] T. Yagi, N. Asano, S. Makita, and Y. Uchikawa. An anthropomorphic active vision system for a location task. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, pages 180–185, September 1994.

[Yas89] A. Yassine. *Etude Adaptatives et Comparatives de Certains Algorithmes en Optimisation. Implémentation Effectives et Applications.* PhD thesis, Université Joseph Fourier, Grenoble, 1989.

[Yua89] J. S.-C. Yuan. A general photogrammetric method for determining object position and orientation. *IEEE Transactions on Robotics and Automation*, 5(2):129–142, April 1989.

[ZBC94] X. Zhang, P. Burlina, and R. Chellappa. Semi- and fully-automatic techniques for image to site model registration. Technical Report CAR-TR-754, University of Maryland, College Park, MD, December 1994.

[ZFL93] N. Zheng, X. Fu, and H. Liu. Cad-based 3D robot vision. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1905–1910, Yokohama, Japan, July 1993.

[ZR91] H. Zhuang and Z. Roth. Comments on calibration of wrist-mounted robotic sensors by solving homogeneous transformation equation of the form $AX = XB$. *IEEE Transactions on Robotics and Automation*, 7:877–878, 1991.

[ZRS94] H. Zhuang, Z. Roth, and R. Sudhakar. Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation of the form $AX = YB$. *IEEE Transactions on Robotics and Automation*, 10(4):549–554, August 1994.