Nouvelle approche pour le calcul du jacobien inverse en asservissement visuel 2D
How to Compute the Inverse jacobian Matrix in the Visual Servoing context

J.T. Lapresté[1] F. Jurie[1] M. Dhome[1] F. Chaumette[2]

[1] LASMEA, UMR 6602 du CNRS,Université Blaise Pascal, 63177 Aubière Cedex, France
[2] IRISA/INRIA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France

Résumé

Les auteurs présentent une méthode numérique de calcul d'une jacobienne inverse d'une fonction, qui ne fait pas intervenir le calcul de la jacobienne elle-même. Cette sorte de jacobienne inverse apprise se montre bien supérieure dans sa capacité à modéliser localement la relation $\theta = f^{-1}(x)$ que l'approximation traditionnelle par J_f^+ (pseudo-inverse de Moore-Penrose).
Une approche théorique aussi bien que des comparaisons avec des méthodes classiques dans le cadre de l'asservissement visuel prouvent le bien fondé de cette assertion.

Mots Clef

jacobienne inverse, perturbations, asservissement visuel, apprentissage.

Abstract

The authors present a method to estimate the inverse Jacobian matrix of a function, without computing the direct Jacobian matrix. This kind of inverse Jacobian matrix proves to perform much better in modeling a relation $\theta = f^{-1}(x)$ than the traditionnal computation of the Moore-Penrose inverse J_f^+
Theoretical insight as well as comparisons in the domain of visual servoing are provided to prove the correctness of the assertion.

Keywords

jacobian matrix, visual servoing, perturbations, learning.

1 Introduction

La cinématique inverse joue un rôle clé dans de nombreuses applications liées à la vision par ordinateur, l'infographie ou la robotique. Dans tous les cas, il s'agit de systèmes possédant un certains nombre de degrés de libertés (que nous nommerons $\theta_t \in \mathbb{R}^n$ dans le reste de cet article) qui permettent d'agir sur le système ; à chaque valeur de θ_t correspond un état observable du système (nommé $x_t \in \mathbb{R}^m$ par la suite). La variable t désigne le temps, et souligne le fait que x et θ varient avec le temps.

Par exemple dans le cas de l'infographie [5], l'utilisation de la cinématique inverse permet d'animer des personnages synthétiques possédant plusieurs centaines de degrés de liberté (θ_t) difficiles à contrôler un par un. L'animateur fixe les trajectoires de certains points clés du corps (x_t) puis se repose sur un algorithme qui calcule les vitesses articulaires à appliquer de manière à ce que l'avatar réalise les trajectoires souhaitées, tout en satisfaisant au mieux des contraintes biomécaniques liées à la posture.
En vision par ordinateur, l'objectif peut être, par exemple, de déterminer la pose d'un objet (θ_t) de manière à ce que son apparence (x_t) soit conforme à celle observée dans l'image [6, 10].
Dans le cas de la robotique [13], il peut s'agir de contrôler l'effecteur d'un robot au moyen des variables articulaires, de manière à ce qu'il suive une trajectoire, définie par la succession des poses de l'effecteur. Enfin, si l'on s'intéresse à l'asservissement visuel [7, 8, 2], il convient de contrôler la pose relative d'une caméra embarquée (θ_t) au moyen de mesures extraites de l'image (x_t).
Pour l'ensemble de ces cas, l'idée de base est la même. Il existe une relation géométrique directe entre θ_t et x_t, que nous écrirons par la suite :

$$x_t = f(\theta_t) \tag{1}$$

L'objectif est d'inverser cette équation, c'est-à-dire d'obtenir une expression de θ (qui représente ce que l'on souhaite contrôler) en fonction de x.
Cette relation peut être inversée de deux manières différentes. La première possibilité est d'utiliser une méthode globale, consistant à trouver le chemin optimal de θ sur la trajectoire complète, ce qui est généralement très coûteux et ne peut être fait en ligne.
L'autre possibilité est d'utiliser des méthodes locales, plus adaptées au calcul en ligne. Cela consiste à écrire une relation cinématique, par dérivation de l'équation (1) :

$$\dot{x}_t = \frac{\partial f}{\partial \theta_t}\dot{\theta}_t \tag{2}$$

Cette équation permet de mesurer comment de petites variations sur θ se répercutent sur x.

La cinématique inverse consiste à inverser cette relation de manière à obtenir $\dot{\theta}_t$ comme une fonction de $\dot{x}_t$. Ce calcul repose généralement sur la notion d'inverse généralisée, notée J_f^+ où $J_f = \frac{\partial f}{\partial \theta_t}$.
Comme nous le verrons plus loin, ce calcul de cinématique inverse est au cœur de nombreux systèmes tels que ceux évoqués au début de cette introduction, et en particulier au cœur des systèmes d'asservissement en robotique.
Dans cet article, une nouvelle définition de cette inverse généralisée est proposée. Elle reprend, dans un contexte différent une approche déjà proposée dans [10] et la justifie. Après l'avoir justifiée sur le plan théorique, son intérêt dans le contexte de l'asservissement visuel est ensuite présenté.
Le plan de cet article reprend cette chronologie : la section 2 pose le problème et justifie théoriquement l'approche proposée ; la section 3 présente son application à l'asservissement visuel. L'article se termine par une discussion de la méthode proposée.

2 La problématique générale

Supposons que nous sommes dans la situation commune où des paramètres $\theta = (\theta_i)_{1 \leq i \leq n}$ sont en mesure d'expliquer des mesures $x = (x_j)_{1 \leq j \leq m}$ par une formule ou un processus calculable

$$x = f(\theta).$$

Le problème inverse général qui se pose est de pouvoir remonter de l'observable x vers le vecteur de paramètres θ, ceci par une formule aussi peu coûteuse que possible en calculs et valide dans un voisinage aussi large que possible.

Remarques sur les notations Dans la suite de l'article, dans le but de simplifier les notations, le temps (t) ne figure plus dans les équations. Ainsi, chaque fois que x apparaît, il faut lire x_t. Il en va de même pour θ_t. D'autre part, étant donné que les algorithmes rencontrés sont de nature discrète (sur le plan temporel), aux vitesses $\dot{x}$ et $\dot{\theta}$ sont substitués des incréments $\Delta\theta$ et Δx qui correspondent à $\dot{x}\Delta t$ et $\dot{\theta}\Delta t$, où Δt représente la durée d'un pas temporel.

2.1 Les méthodes

Bien évidemment f n'est ni linéaire ni inversible. Au vu d'un simple développement de Taylor de f, l'approche traditionnelle du problème, qui est une approche locale, est de trouver une solution à

$$\Delta f(\theta) = J_f(\theta)\Delta\theta$$

où $J_f(\theta)$ est la matrice jacobienne de f en θ. La solution au problème est alors déduite de la pseudo-inverse de Moore-Penrose $J_f^+(\theta)$.
Il est cependant possible d'envisager une autre approche par apprentissage permettant d'obtenir directement la jacobienne d'une supposée fonction f^{-1}.
L'apprentissage s'effectue simplement, en générant une famille aléatoire de N_r incréments $\Delta\theta$, produisant une matrice de dimension $n \times N_r$ contenant les perturbations $\theta + \Delta\theta$, et en considérant les variations correspondantes $\Delta x = f(\theta + \Delta\theta) - f(\theta)$ des mesures, qui sont calculables à l'aide de f et fournissent la matrice Δx de taille $m \times N_r$.
Il ne reste plus qu'à résoudre au sens des moindres carrés le système $A\Delta x = \Delta\theta$ pour calculer A que nous noterons dans la suite par $J_f^{\oplus}$.
Il est à noter, qu'il est indispensable dans ce calcul d'utiliser une résolution stable (en utilisant la SVD par exemple) car la matrice Δx peut ne pas être de rang plein et que N_r doit être supérieur ou égal à m.

2.2 Pourquoi cette dernière approche est-elle plus performante ?

Notre approche n'est (déjà) pas moins performante que la classique, dans le sens que le calcul de J_f^+ se fait numériquement selon un schéma simple de différences divisées (dès que le calcul analytique n'est plus disponible ou même plus coûteux), non probabiliste, mais analogue à celui de l'apprentissage, et lui aussi coûteux.
De plus, si f a un comportement assez linéaire autour de θ, il est clair que la solution classique apporte des résultats satisfaisants. Mais ce n'est pas toujours le cas. En général, l'approximation linéaire de $f(\theta)$ n'est valide que dans un voisinage très restreint de θ.
Si on évalue la dimension N de l'espace vectoriel engendré par l'image par f d'un petit voisinage de θ en observant le rang approximatif de matrices d'échantillons, on trouvera évidemment que ce rang est supérieur à la dimension de l'espace des paramètres, et qu'en fait plus la différence des dimensions est importante, plus l'approximation est de piètre qualité.
Ce saut dimensionnel peut être interprété comme la preuve de l'existence de paramètres supplémentaires «cachés» dont la connaissance pourrait entraîner une bien meilleure approximation linéaire, mais qui ne sont (et ne peuvent) être pris en compte.
Par exemple supposons que nous rajoutions à nos paramètres θ_i les $\nu_{ij} = \theta_i\theta_j$, fournissant ainsi une expression linéaire d'un modèle en fait quadratique. Bien sûr le calcul de la jacobienne inverse de cette fonction de $n(n+1)/2$ variables améliorera le domaine de validité de la méthode, au prix d'un important sur-coût de calcul.
Notre approche nous autorise en fait à prendre en compte ces paramètres «cachés» sans jamais avoir à les utiliser explicitement, ni même tenter de les définir et sans augmenter la charge de calcul en ligne (seul le temps de calcul lié à l'apprentissage est augmenté).
Comme il a déjà été dit, le calcul de la matrice Δx correspondant aux perturbations de nos paramètres nous permet (si nous le désirons) d'estimer le nombre des paramètres cachés nécessaires à une approximation linéaire correcte, et même de nous renseigner sur le nombre de paramètres manquants.
Le calcul de $J_f^{\oplus}$ prend implicitement ces résultats en compte. Nous prétendons que $J_f^{\oplus}$ est formée d'une approximation des n premières lignes d'une pseudo-inverse

(pas nécessairement celle de Moore-Penrose) de la matrice jacobienne d'une fonction supposée $\tilde{f}(\theta, \nu)$ où les ν_i sont cachés.
Cela s'explique dans la mesure où la génération aléatoire de l'ensemble des perturbations autour de θ conduit à l'estimation d'un ensemble de Δx qui intègrent aussi les variations correspondant aux paramètres cachés, cela étant du à la non linéarité de f.

Bien sûr le haut de la matrice jacobienne inverse est tout ce que nous désirons, puisque ce ne sont que les variations de nos paramètres originaux (θ) que nous voulons évaluer, et non celles des paramètres cachés.

Il est également clair que l'information véhiculée par $J_f^{\oplus}$ est largement plus importante que celle contenue dans J_f^{+} dès que f n'est pas parfaitement affine, puisque la pseudo-inverse de J_f n'est pas formée des n premières lignes de J_f^{+} !

2.3 Deux exemples «illustratifs »

Une fonction quadratique. Supposons que nous possédions une fonction $f(\theta_1, \theta_2)$ de $\mathbb{R}^2$ dans $\mathbb{R}^p$, $f = (f_i)_{1 \leq i \leq p}$ définie par :

$$x_i = f_i(\theta_1, \theta_2) = a_{i1}\theta_1 + a_{i2}\theta_2 + a_{i3}\theta_1\theta_2 + a_{i4}\theta_1^2 + a_{i5}\theta_2^2$$

J_f vaut en $(0, 0)$:

$$J_f(\theta_1, \theta_2) = ((a_{ij})_{1 \leq i \leq p, 1 \leq j \leq 2})$$

De même, posons :

$$x_i = \tilde{f}(\theta_1, \theta_2, ..., \theta_5) = a_{i1}\theta_1 + a_{i2}\theta_2 + a_{i3}\theta_3 + a_{i4}\theta_4 + a_{i5}\theta_5$$

$\tilde{f}$ est une fonction linéaire de 5 paramètres qui coïncide avec f sur la variété engendrée par $(\theta_1, \theta_2, \theta_1\theta_2, \theta_1^2, \theta_2^2)$
$J_{\tilde{f}}$ est constante et on a :

$$J_{\tilde{f}}(\theta_1, \theta_2, ..., \theta_5) = ((a_{ij})_{1 \leq i \leq p, 1 \leq j \leq 5})$$

Supposons de plus que le point d'intérêt soit l'origine. Nous avons à présent au moins trois possibilités pour résoudre notre problème :

1. Calculer J_f et en déduire J_f^{+}
2. Calculer $J_{\tilde{f}}$ et la partie pertinente de $J_{\tilde{f}}^{+}$
3. Calculer $J_f^{\oplus}$

Les tables qui suivent montrent l'adéquation de l'approximation. A partir d'un choix de $\tilde{f}$, on présente les normes $||K\Delta x - \Delta\theta||$ (où K représente une des jacobiennes). Dans la première, les calculs ont été effectués avec les matrices Δx et $\Delta\theta$ qui ont servi à générer $J_f^{\oplus}$. La première colonne contient le nombre de perturbations utilisées.

#	$\|\|J_f^{+}\Delta x - \Delta\theta\|\|$	$\|\|J_f^{\oplus}\Delta x - \Delta\theta\|\|$	$\|\|J_{\tilde{f}}^{+}\Delta x - \Delta\theta\|\|$
1	4.1633e-17	0.0000e+00	2.0977e-14
2	1.5701e-16	1.8916e-16	1.1932e-15
3	9.4907e-03	6.4980e-16	8.2088e-16
4	6.6804e-02	8.4902e-16	2.4521e-15
5	1.7664e-01	1.2310e-15	1.2910e-15
10	9.7743e-02	5.0797e-16	2.7565e-15
50	6.0087e-01	3.6708e-15	2.3398e-14
100	6.0682e-01	4.2373e-15	3.5011e-15
500	1.8890e+00	1.3453e-14	1.8441e-14

Le tableau qui suit propose des résultats similaires, mais une fois les trois jacobiennes calculées, de nouvelles matrices $\Delta\vartheta$ et Δx ont été tirées aléatoirement.

#	$\|\|J_f^{+}\Delta\mathrm{x} - \Delta\vartheta\|\|$	$\|\|J_f^{\oplus}\Delta\mathrm{x} - \Delta\vartheta\|\|$	$\|\|J_{\tilde{f}}^{+}\Delta\mathrm{x} - \Delta\vartheta\|\|$
1	3.1735e+00	3.1735e+00	1.1941e-13
2	1.0325e+00	1.0325e+00	1.2159e-14
3	7.9791e-01	7.1834e-01	9.7157e-15
4	7.1033e-01	1.5500e-01	1.3281e-14
5	8.1970e-01	2.1996e-14	4.8466e-15
10	4.4527e-01	2.7380e-15	8.2650e-15
50	8.1853e-01	5.0798e-15	2.9975e-14
100	6.4036e-01	4.2410e-15	2.9141e-15
500	7.7098e-01	5.6362e-15	7.6079e-15

Il apparaît qu'en dehors des points d'apprentissage il est nécessaire d'avoir 5 perturbations au moins pour avoir une bonne estimée. Pour $\tilde{f}$ qui est linéaire, le résultat est toujours évidemment parfait.
On peut aussi remarquer que dans ce cas, le domaine de validité de l'approximation qui était pour la jacobienne inverse J_f^{+} un petit voisinage de l'origine est passé pour $J_f^{\oplus}$ à l'espace tout entier, puisque dans ce cas une jacobienne constante est estimée.

Trouver le minimum de la fonction de Rosenbrock. La fonction de Rosenbrock est souvent appelée «la banane». Si on dénote par $P = (u, v)$ un point du plan :

$$r(P) = r(u, v) = (u - 1)^2 + 10 * (u^2 - v)^2;$$

On peut voir sur la Figure 1 les lignes de niveau d'une translation par le vecteur $(0.5, 0)$ de cette fonction qui admet un minimum global en $(1, 1)$, de valeur 0.
Elle est appelée banane à cause de la manière dont sa courbure s'incline autour de l'origine. Elle est fréquemment utilisée comme exemple de fonction difficile à optimiser à cause de la lenteur de la convergence de la plupart des méthodes classiques d'optimisation à son égard.
Notre méthode ne permet évidemment pas de remplacer une minimisation en toute circonstance, cependant dans certains cas, son usage peut être fort intéressant.
En particulier, quand on cherche non le minimum d'une fonction, mais celui d'une translation inconnue de celle-ci. Supposons que la fonction à minimiser est :

$$r_0(P) = r_0(u, v) = (u - u_0 - 1)^2 + 10 * ((u - u_0)^2 - v + v0)^2,$$

avec u_0 et v_0 inconnus. On sait que le minimum de r est en $(1,1)$ et on peut donc apprendre la forme de la fonction r_0 autour de son propre minimum en choisissant :
– quelques points au hasard $(P_e^i)_{1\leq i\leq p}$ autour du minimum de r ;
– quelques perturbations aléatoires $(\Delta r_j)_{1\leq j\leq q}$.
On calcule alors $J_{r_0}^{\oplus}$ à l'aide des

$$\Delta x_{ij} = r(P_e^i + \Delta r_j) - r(P_e^i).$$

qui ne nécessitent pas la connaissance de r_0. La figure ci-dessous présente la trajectoire vers la solution. Une unique itération suffit à atteindre le minimum.

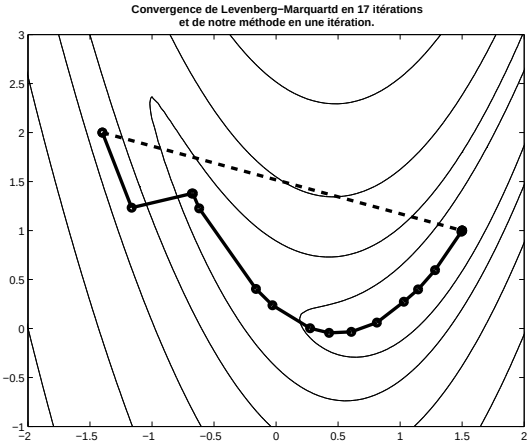

FIG. 1 – Convergence en une seule itération

On doit utiliser au moins $p = 5$ et $q = 5$ pour assurer le fonctionnement du programme en exactement une itération. En fait si on développe la banane on a :

$$r(u+1, v+1) = 41u^2 - 40uv + 10v^2 + 10u^4 + 40u^3 - 20u^2v;$$

A priori 6 paramètres sont nécessaires pour linéariser r, mais il est aisé d'imaginer qu'un changement de base orthogonale diagonalisera la partie quadratique, supprimant un paramètre. Ainsi un rang 5 est nécessaire et d'ailleurs observé.
Dans les mêmes conditions un algorithme comme celui de Levenberg-Marquardt utilise typiquement entre 10 et 30 itérations et ne converge pas toujours si le point de départ est trop mal choisi.
Sur la figure on a décalé la banane de 0.5 en x et le minimum se trouve donc en $(1.5, 1)$. La convergence de notre algorithme a lieu en une itération, indépendamment du point de départ car la modélisation linéaire est parfaite. Ce ne serait pas le cas si par exemple on avait fait subir une rotation à la banane dans le plan des (u, v). La méthode basée sur la jacobienne inverse (méthode de Gauss-Newton) n'est en général pas convergente, en un nombre raisonnable d'itérations.

3 Application à l'asservissement visuel

3.1 Introduction

Nous montrons dans cette section comment la technique précédemment proposée permet d'amener des résultats intéressants dans le contexte de l'asservissement visuel.
Les techniques d'asservissement visuel 2D sont connues pour donner un comportement satisfaisant lorsque l'erreur entre les mesures x et les consignes x_c sont faibles. Dès que l'erreur est importante, il n'est plus possible de démontrer la stabilité du système et son comportement peut devenir totalement insatisfaisant [1] : convergence vers un minimum local, trajectoire inadéquate du robot en raison de forts couplages, etc. Pour pallier ces problèmes, une première solution consiste à sélectionner des informations visuelles présentant de bonnes propriétés [11, 3, 9, 15], mais il reste encore beaucoup à faire dans ce domaine. Une seconde solution consiste à faire précéder l'asservissement par une phase de planification, l'erreur au cours du suivi des trajectoires planifées étant alors toujours faible [12, 17]. L'approche que nous proposons est plus simple dans son principe : elle revient à effectuer un asservissement visuel classique, mais en annihilant autant que possible les fortes non linéarités du système (qui sont la cause des problèmes évoqués ci-dessus) en les prenant en compte dans l'apprentissage direct de la pseudo-inverse de la jacobienne.
On peut enfin noter que des techniques d'apprentissage ont déjà été employées en asservissement visuel, notamment pour appréhender des objets de formes et textures inconnues et complexes [4]. Des réseaux de neurones ont également été utilisés [16]. Mais, dans tous les cas, les auteurs de ces travaux ont malheureusement préféré estimer la jacobienne, plutôt que directement sa pseudo inverse.
Nous nous intéressons au cas où l'on souhaite contrôler la position relative d'un effecteur de robot par rapport à un objet. L'objet est observé au moyen d'une caméra embarquée sur l'effecteur. C'est donc en réalité la position de la caméra qui sera contrôlée.
Nous notons $x = (x_1, \ldots, x_m) \in \mathbb{R}^{\mathrm{m}}$ un ensemble de primitives visuelles mesurées dans l'image (repère 2D lié à la caméra), issues de la projection d'une scène 3D. Nous ne faisons pas à ce stade de restriction sur la nature des primitives possibles. Ce peuvent être par exemple les coordonnées de points 3D projetés dans les images, les niveaux de gris d'une région de l'image issue de la projection d'un objet 3D, etc.
Le repère de la caméra est localisé par rapport à celui dans lequel sont décrits les objets 3D au moyen de 6 paramètres, les trois coordonnées de la translation et les trois angles d'Euler paramétrant la rotation, réunis dans le vecteur

$$\theta = (\alpha, \beta, \gamma, t_x, t_y, t_z)$$

de $\mathbb{R}^6$.

Nous pouvons décrire la relation liant la projection des primitives visuelles et la localisation relative des deux repères sous la forme

$$x = f(\theta).$$

Les coordonnées des primitives dans le repère 3D lié à l'objet sont supposées fixes (objet rigide). Elles font implicitement partie de la définition de f. Seule la position relative caméra/objet peut varier.
L'objectif de l'asservissement visuel est de contrôler l'effecteur de manière à ce que les primitives visuelles x observées par la caméra coïncident avec une consigne x_c définie dans le repère image.
Si l'on s'intéresse aux aspects cinématiques, la relation entre vitesse dans l'espace des primitives visuelles et vitesse dans l'espace opérationnel peut être décrite par la relation :

$$\dot{x} = J(\dot{\theta}),$$

où J est la matrice des dérivées partielles de f : $J = \frac{\partial x}{\partial \theta}$. Cette matrice est communément appelée *matrice d'interaction* associée à x, ou encore *jacobienne image*.
Un des problèmes clé est le calcul de la matrice J. Un certain nombre d'approches ont été proposées pour traiter ce problème. Nous renvoyons le lecteur à [2] pour plus de détails.
A partir de là, la commande du robot peut être envisagée comme dans Samson *et al.* [14] par une approche basée sur la régulation à zéro d'une *fonction de tâche*, la tâche étant basée sur la minimisation de $\|e(\theta)\|$. Dans le cas de l'asservissement visuel, la fonction de tâche est écrite comme fonction de primitives de l'image, elles-mêmes fonction de la pose de la caméra :

$$e = C(x - x_c)$$

où $x = f(\theta)$. La matrice C est classiquement choisie égale à $C = J^+(x)$ ou $C = J^+(x_c)$ (où A^+ désigne la pseudo-inverse de A). Dans le premier cas, la matrice C est calculée à chaque itération de la loi de commande, en utilisant les valeurs courantes de x. Dans le second cas, C est constante et calculée hors ligne en utilisant les valeurs désirées x_c.
Une loi de commande très simple peut être établie en tentant s'assurer une décroissance exponentielle de la fonction de tâche ($\dot{e} = -\lambda e$ avec $\lambda > 0$). On obtient [2] :

$$\Delta\theta = -\lambda\, C(x - x_c) \tag{3}$$

L'approche que nous proposons se situe simplement au niveau du calcul de la pseudo-inverse de la matrice d'interaction et dans le cas où la dimension de l'espace de représentation des primitives (m) est plus grande que l'espace de représentation de la pose de la caméra ($m > 6$).
Bien que cela ne soit pas discuté dans cet article, il est évident que la méthode proposée peut aussi être appliquée au cas où moins de 6 degrés de libertés doivent être commandés.

3.2 Asservissement avec des primitives de type points

Nous allons présenter ici des simulations visant à montrer l'intérêt de la méthode proposée pour le calcul de la jacobienne inverse en asservissement visuel. L'idée consiste à remplacer le calcul de la pseudo-inverse de la matrice d'interaction (C) par la matrice $J^{\oplus}$ décrite dans la section 2.
Nous avons choisi, à titre d'illustration, le cas spécifique de l'asservissement à partir de quatre points formant un carré, mais la méthode peut être utilisée avec n'importe quel type de primitives visuelles (droites, sphères, motifs, etc.).
Dans le cas de point, la matrice d'interaction J peut s'écrire [2] :

$$\begin{pmatrix} -1/z & 0 & X/z & XY & -(1+X^2) & Y \\ 0 & -1/z & Y/z & 1+Y^2 & -XY & -X \end{pmatrix} \tag{4}$$

où (x, y, z) représentent les coordonnées d'un point dans la scène, et (X, Y) les coordonnées de sa projection dans l'image. La focale, sans perte de généralité, est supposée être égale à 1 dans cette formule.
Nous nous plaçons dans le cas où l'on souhaite contrôler entièrement la position de la caméra $\theta = (\alpha, \beta, \gamma, t_x, t_y, t_z)$ à partir des projections de quatre points formant un carré. Le vecteur $x = (X_1, Y_1 \ldots, X_4, Y_4)$ représente les 8 coordonnées de ces quatre points dans l'image.
Pour chaque expérience, le carré mesure 1 mètre de largeur, et il est placé au centre du repère lié à la scène. La position à atteindre est définie par la position du carré dans l'image lorsque la caméra se trouve en $\theta_d = (0, 0, 0, 0, 0, 3)$ (carré centré dans l'image, éloigné de 3m). La focale est supposée être de 500 pixels, et le point principal de la caméra a pour coordonnées $(0, 0)$. Bien que la valeur de la focale n'influe pas sur la convergence, fixer sa grandeur permet de mieux évaluer les performances (en particulier l'erreur dans l'image) des différents algorithmes proposés.
Enfin, pour toutes les simulations, la valeur de λ (définie dans l'équation 3) est de 0.2.
Trois simulations sont proposées, correspondant à trois positions de départ de la caméra. Comme détaillé dans [1], les deux dernières sont réputées difficiles en utilisant des coordonnées de points en asservissement visuel 2D classique en raison du fort couplage dans la matrice d'interaction entre les translations le long de l'axe optique et les rotations autour de cet axe.

1. Dans la première simulation, la caméra subit une rotation de 45 degrés autour de l'axe optique (axe γ).
2. Dans la seconde simulation, la caméra subit une rotation de 160 degrés autour de l'axe optique.
3. Enfin, dans la dernière simulation, sont combinées une rotation de 50 degrés en α et 50 degrés en γ.

Nous étudions, pour chaque simulation, trois cas différents :

1. utilisation de la pseudo-inverse de la matrice jacobienne, calculée à chaque itération. Nous appelons cette matrice J_i^+,

2. utilisation de la pseudo-inverse calculée à l'équilibre, notée J^+,
3. utilisation de la matrice $J^\oplus$ que nous proposons.

Dans les cas 1 et 2, la matrice d'interaction donnée par l'équation (4) est utilisée.
Dans le troisième cas, la matrice $J^\oplus$ est calculée numériquement comme indiqué section 2.1. Des déplacements $\Delta\theta$ sont générés autour de la position à la convergence, et des déplacements Δx des primitives sont mesurés. Dans les expériences présentées, 1000 déplacements sont utilisés, mais ce chiffre n'a pas une grande influence sur la qualité des résultats. La matrice $J^\oplus$ est ensuite calculée avec la relation $J^\oplus = \Delta\theta\Delta x^+$. L'amplitude maximale des déplacements pendant l'apprentissage est de 50 degrés en rotation, et de 1 mètre en translation. Cette matrice est calculée une fois pour toute ; elle n'est pas recalculée en ligne, contrairement à J_i^+
Remarque : afin que les comparaisons soient les plus complètes possibles, nous avons également testé le cas de l'utilisation d'une pseudo-inverse J_n calculée numériquement à partir des données utilisées pour le calcul de $J^\oplus$, au moyen de la relation $J_n = (\Delta x\Delta\theta^+)^+$. Les résultats étant systématiquement moins bons que ceux obtenus avec J_i^+, nous ne les présentons pas.

3.3 Résultats

Les figures 3 à 8 présentent les résultats obtenus pour les trois simulations réalisées. Pour chaque simulation, les informations suivantes sont données :
– la position du motif pour la première image de la séquence et la position du motif à atteindre,
– les vitesses de rotation et de translation de la caméra durant l'exécution de la tâche, pour chaque loi de commande étudiée (J_i^+, J^+ et $J^\oplus$),
– un critère d'erreur dans l'image (la somme des carrés des distances des sommets du carré par rapport à la position souhaitée), pour chaque loi de commande,
– une comparaison des courbes d'erreur dans l'image,
– les trajectoires des quatre points durant la réalisation de la trajectoire.

De manière à mieux comprendre le sens des vitesses de la caméra, il convient de préciser que t_x, t_y, t_z représentent les coordonnées de l'origine du repère de l'objet exprimées dans le repère de la caméra. La rotation relative des deux repères (objet/caméra) est décrite par un enchaînement de trois rotations axiales R_γ, R_β et R_α correspondant à des rotation successives du repère de la caméra par rapport à celui de l'objet. Ces rotations correspondent à des rotations respectivement selon les axes des Z, des Y et des X du repère de l'objet (l'axe des Z étant normal au plan support des quatre points).
Les figures 3 à 4 décrivent la première simulation. Dans cette simulation une rotation de 45 degrés de la caméra autour de l'axe optique de la caméra est réalisée. Dans cette configuration, les résultats de convergence obtenus avec les trois méthodes sont similaires. En terme de déplacement, la trajectoire réalisée avec $J^\oplus$ est la plus satisfaisante, car elle est réalisée en n'effectuant qu'une rotation pure, alors que les autres produisent également des translations, qui ne sont pas nécessaires.
Les seconde et troisième expériences imposent des conditions initiales nettement plus sévères. Dans ces conditions, la matrice J^+ calculée à l'équilibre est insuffisante pour obtenir des convergences satisfaisantes, en particulier dans la troisième expérience où la caméra rejoint le plan de l'objet ($z = 0$). Dans ces deux cas, les trajectoires dans l'image et les vitesses de la caméra obtenues avec $J^\oplus$ sont nettement meilleures que celle obtenue avec J_i^+, qui deviennent irréalistes dans la troisième expérimentation. Une analyse du rang de Δx montre que cette matrice est de rang 8, pour ces deux exemples. Nous sommes donc bien dans le cas décrit section 2.1, et ces résultats étaient donc bien prévisibles.
On notera en particulier la trajectoire obtenue pour la troisième simulation, pour laquelle la rotation de 160 degrés autour de l'axe optique est bien compensée par une rotation pure dans la loi de commande utilisant $J^\oplus$ alors que la loi utilisant J_i^+ produit une translation très importante du capteur.
Enfin, il est aisé d'observer que les vitesses obtenues avec la méthode utilisant $J^\oplus$ sont toujours équivalentes ou inférieures à celles obtenues avec les autres méthodes, bien que la convergence soit atteinte plus rapidement.
Des résultats encore meilleurs peuvent être obtenus si l'on augmente le nombre de points. En effet, cela permet, avec notre approche, de mieux prendre en compte les non linéarités de la fonction f. Ils ne sont pas présentés faute de place.

4 Discussion

Un mot sur les complexités comparées Des trois méthodes, seule celle faisant intervenir J_i^+ demande un calcul en ligne, c'est donc la plus coûteuse avec l'évaluation de la pseudo-inverse de la matrice Jacobienne à chaque itération. Dans le cas des expérimentations il s'agit d'une matrice 8×6.
Pour ce qui est des coût hors lignes, il est clair que notre méthode est légèrement plus coûteuse. Le calcul de $J^\oplus$ nécessite la résolution d'un système $J^\oplus\Delta x = \Delta\theta$ où, si N_p désigne le nombre de perturbations :
– $J^\oplus$ l'inconnue est de taille $6 \times 8 = 48$;
– Δx est de taille $8 \times N_p$;
– $\Delta\theta$ est de taille $6 \times N_p$;

Ce système équivaut à la recherche de la recherche de la solution au sens des moindres carré d'un système de N_p équations à 8 inconnues pour 6 seconds membres. Ce qui se fait efficacement avec une svd économique (un calcul de pseudo-inverse 8×8).
Le calcul de J^+ est lui celui de la pseudo-inverse d'une matrice 6×8

Singularités Un des inconvénients lié aux méthodes classiques est la présence de singularités qui provoquent des instabilités dans les lois de commande.

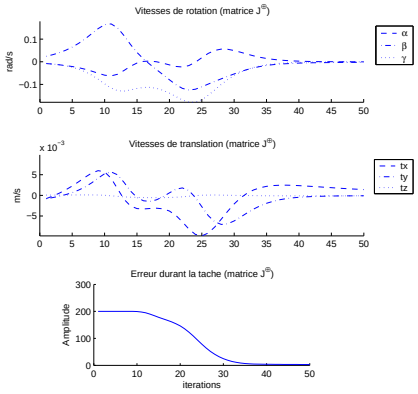

FIG. 2 – Convergence avec $J^{\oplus}$ (rotation de 180 degrés)

La nouvelle méthode proposée ne possède pas les mêmes singularités que les méthodes classiques.

Par exemple dans le cas d'une rotation de 180 degrés du motif autour de l'axe optique les lois de commande basées sur la pseudo-inverse de la jacobienne ne convergent plus.

A contrario comme nous pouvons le voir sur la figure 2, la méthode basée sur $J^{\oplus}$ converge, qui plus est avec des vitesses compatibles avec la commande d'un robot réel.

Bien sûr, cela ne préjuge pas de l'existence d'autres singularités pour cette méthode, mais l'étude reste à faire.

Connaissance du modèle de l'objet Avec la méthode basée sur $J^{\oplus}$ il n'est pas nécessaire de connaître le modèle 3D de l'objet. Mais dans ce cas un apprentissage qui peut être expérimentalement lourd à réaliser doit être effectué avec des images réelles de l'objet.

Cependant cette solution a aussi l'avantage de permettre une compensation implicite des erreurs de calibration du système. Si le modèle est connu on peut s'affranchir de la phase d'apprentissage sur les données réelles, en effectuant l'apprentissage via des simulations.

5 Conclusions et perspectives

Dans cet article est proposée une méthode de calcul du jacobien inverse dans un contexte d'asservissement visuel. Nous avons expliqué pourquoi et dans quelles conditions le calcul direct du jacobien inverse de la fonction peut être préférable au calcul de la pseudo-inverse du jacobien.

Nous avons ensuite montré des simulations informatiques dans lesquelles ce calcul direct est comparé avec le calcul qui est fait habituellement en asservissement visuel. Les résultats présentés semblent justifier pleinement l'utilisation de cette approche en asservissement visuel.

Il nous semble désormais nécessaire de conduire des expérimentation réelles, en implémentant les lois de commandes proposées sur des systèmes robotiques.

D'autre part, le calcul direct de la jacobienne inverse, tel qu'il est présenté ne permet pas directement de faire d'étude de stabilité, contrairement à un jacobien analytique qui lui, permet en théorie de le faire. C'est un point important sur lequel des avancées doivent être faites.

Enfin, on peut noter, qu'indépendamment du cadre de l'asservissement visuel, la méthode proposée consiste en un apprentissage de la matrice jacobienne de l'inverse autour de son optimum global, et qu'elle peut être utilisée en toute généralité pour optimiser des fonctions pour lesquelles la forme autour de l'optimum est connue par avance.

Références

[1] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The Confluence of Vision and Control*, pages 66–78. LNCIS 237, Springer Verlag, 1998.

[2] F. Chaumette. Asservissement visuel. In *La commande des robots manipulateurs*, pages 105–150. Traité IC2, Hermès, 2002.

[3] P. Corke and S. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Trans. on Robotics and Automation*, 17(4) :507–515, 2001.

[4] K. Deguchi. A direct interpretation of dynamic images with camera and object motions for vision guided robot control. *Int. Journal of Computer Vision*, 37(1) :7–20, 2000.

[5] M. Girard and A.A Maciejewski. Computational modeling for the computer animation of legged fi gures. *ACM Computer Graphics*, 19(3) :263–270, 1985.

[6] G.D. Hager and P.N. Belhumeur. Effi cient region tracking with parametric models of geometry and illumination. *IEEE Journal on Pattern Analysis and Machine Intelligence*, 20(10) :1025–1039, October 1998.

[7] K. Hashimoto. *Visual servoing*, volume 7 of *Series in Robotics and Automated Systems*. Worls Scientifi c, 1993.

[8] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5) :651–670, 1996.

[9] M. Iwatsuki and N. Okiyama. A new formulation of visual servoing based on cylindrical coordinate system with shiftable origin. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'02, Lausanne*, Octobre 2002.

[10] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *IEEE Journal on Pattern Analysis and Machine Intelligence*, 24(7) :996–1000, 2002.

[11] E. Malis, F. Chaumette, and S. Boudet. 2 1/2 d visual servoing. *IEEE Trans. on Robotics and Automation*, 15(2) :238–250, 1999.

[12] Y. Mezouar and F. Chaumette. Path planning for robust image-based control,. *IEEE Trans. on Robotics and Automation*, 18(4) :534–549, 2002.

[13] R.P. Paul. *Robot Manipulators : Mathematics, Programming, and Control*. MIT Press, Cambridge, 1981.

[14] C. Samson, B. Espiau, and M. Le Borgne. *Robot Control : the task function approach*. Oxford University Press, 1991.

[15] O. Tahri and F. Chaumette. Application of moment invariants to visual servoing. *IEEE Int. Conf. on Robotics and Automation, Taipeh*, Septembre 2003.

[16] G. Wells, C. Venaille, and C Torras. Vision-based robot positioning using neural networks. *Image and Vision Computing*, 14 :715–732, 1996.

[17] P. Zanne, G. Morel, and F. Plestan. Sensor-based control in the presence of uncertainties : bounding the task function tracking errors. *IEEE Int. Conf. on Robotics and Automation, Washington D.C.*, Mai 2002.

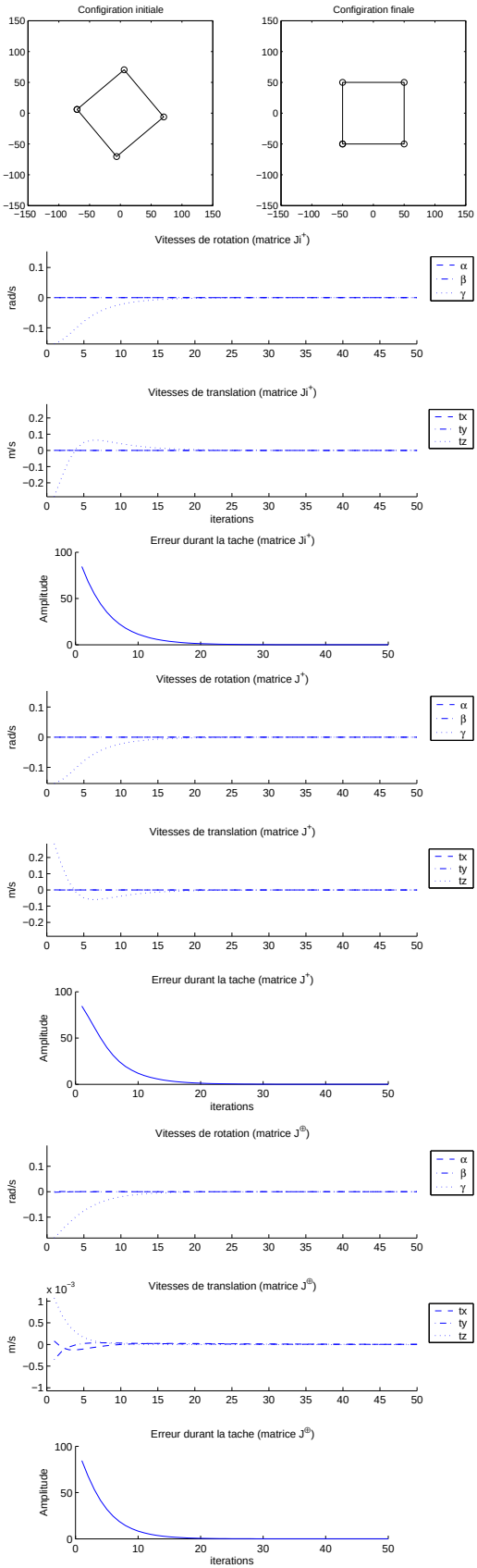

FIG. 3 – Première simulation : position de départ et d'arrivée, vitesses et erreurs dans l'image durant l'exécution de la tâche, avec les trois lois de commande.

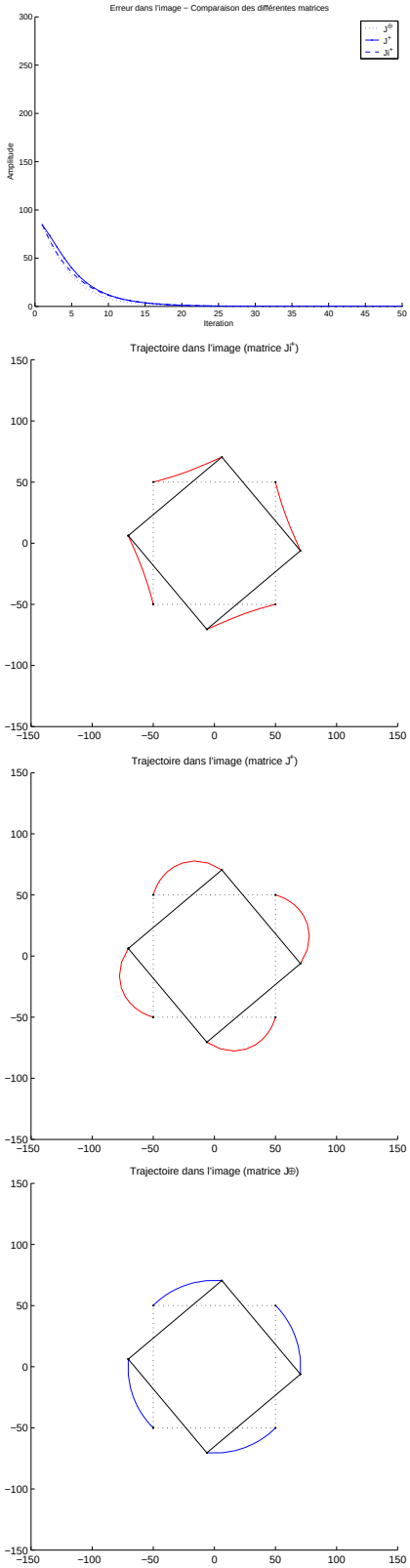

FIG. 4 – Première simulation : comparaison des erreurs, trajectoires dans l'image

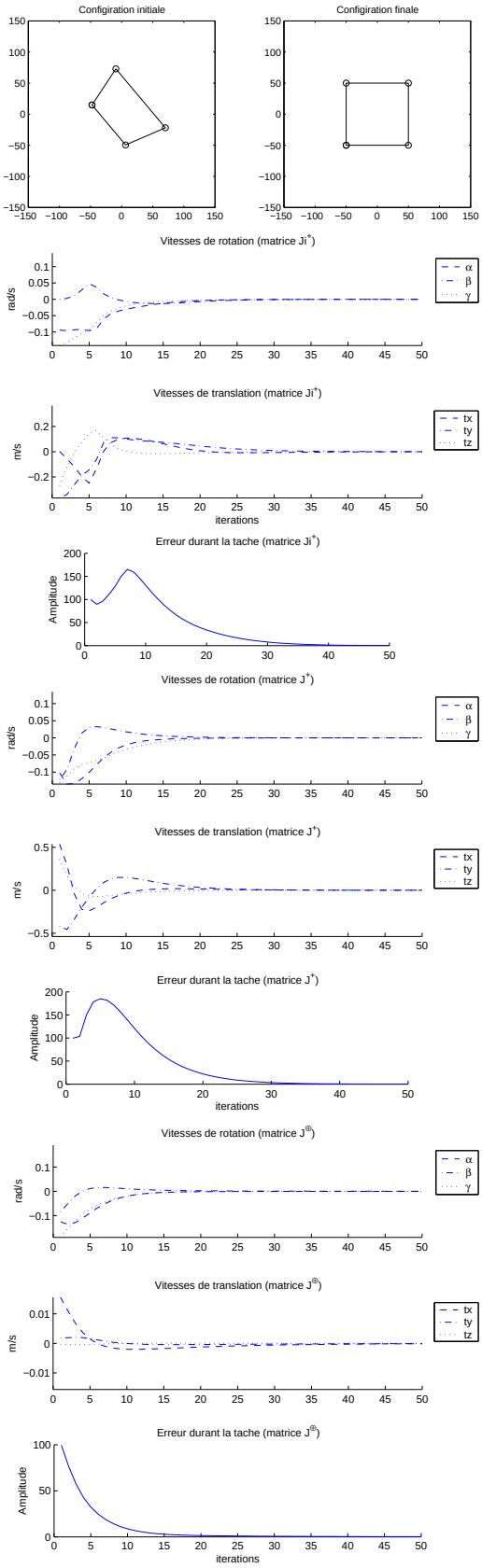

FIG. 5 – Seconde simulation : position de départ et d'arrivée, vitesses et erreurs dans l'image durant l'exécution de la tâche, avec les trois lois de commande.

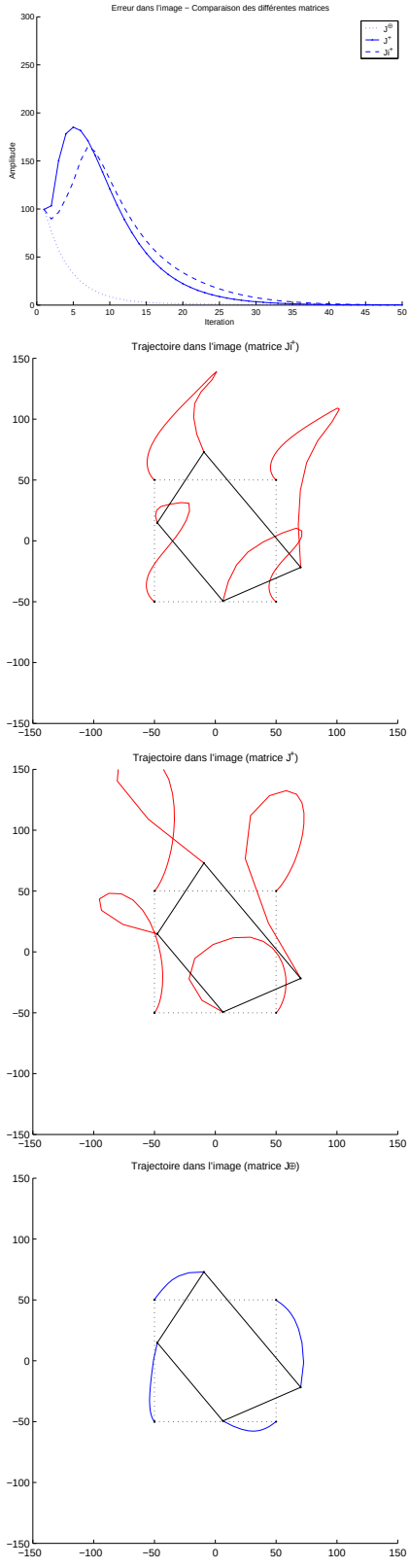

FIG. 6 – Seconde simulation : comparaison des erreurs, trajectoires dans l'image

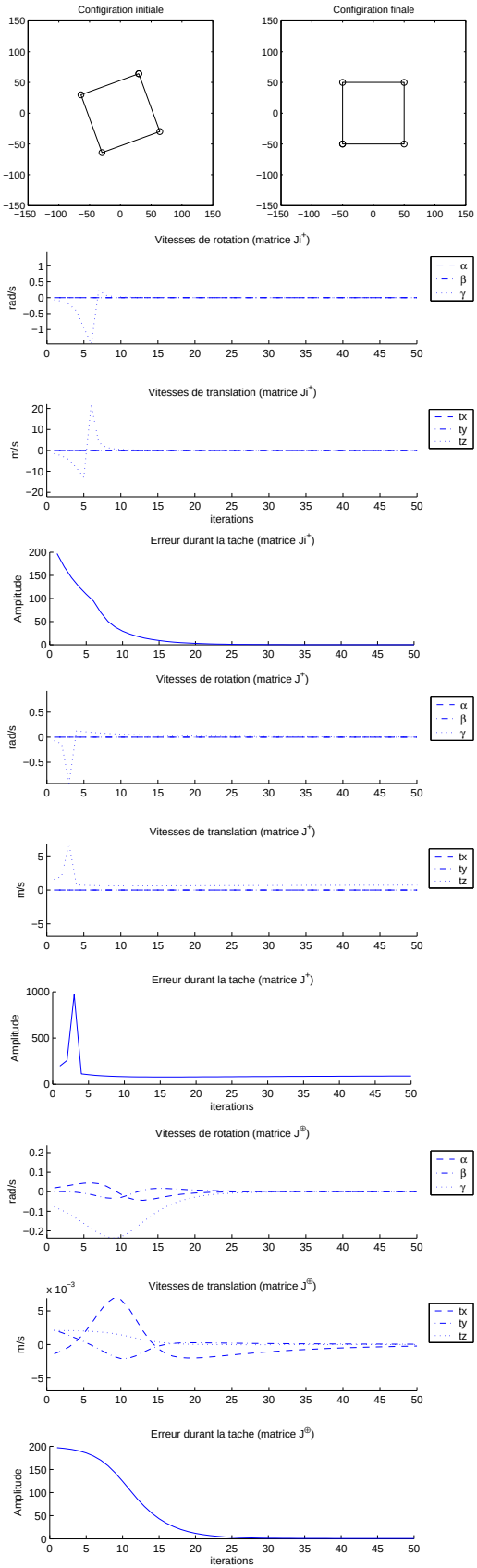

FIG. 7 – Troisième simulation : position de départ et d'arrivée, vitesses et erreurs dans l'image durant l'exécution de la tâche, avec les trois lois de commande.

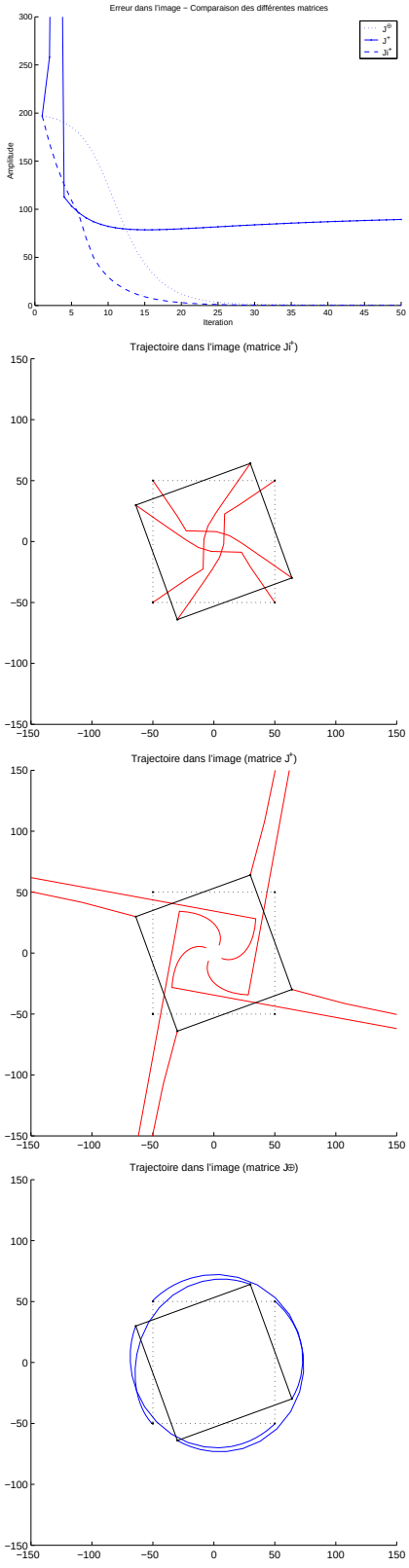

FIG. 8 – Troisième simulation : comparaison des erreurs, trajectoires dans l'image