REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
ECOLE NATIONALE POLYTECHNIQUE

Projet De Fin d'Etude Pour l'Obtention Du Diplôme d'Ingénieur d'Etat En Automatique

Thème :

Asservissement Visuel d'Un Bras De Robot

Réalisé par :

- *ZAIDI Nedjma*
- *TAIDIRT Hassen*

Encadré Par :

Mr. H. CHEKIREB

Année Universitaire 2005 - 2006

ملخص :

العمل المقدم من خلال هذه المذكرة يتمثل في دراسة التحكم البصري على ذراع آلية، قمنا بعرض مختلف الوسائل الضرورية لتحديد الموقع باستخدام التصوير أحادي الكاميرا. قمنا بدراسة الأنواع الثلاثة للتحكم البصري المتمثلة في $2D1/2,3D,2D$ مع استخدام أنظمة تحكم مختلفة للتحكم البصري من جهة والتحكم على ذراع الآلية من جهة أخرى. مختلف أنظمة التحكم المقترحة مثلت على ذراع آلية من نوع PUMA 560

كلمات مفتاحية :
التحكم البصري، ذراع آلية (PUMA 560) - تحديد الموقع- تصوير أحادي الكاميرا- تحكم بصري $2D1/2,3D,2D$ - التحكم على الذراع.

RESUME :

Le travail présenté dans ce mémoire est une étude de l'asservissement visuel d'un bras de robot. Nous présentons les outils nécessaires pour la localisation par vision monoculaire. Nous étudions les trois types d'asservissement visuel 2D, 3D et 2D1/2 où appliquons différents types de commandes pour les deux boucles de vision et du robot. Les simulations ont été appliquées sur le robot PUMA 560.

Mots clés :
Asservissement visuel, bras manipulateur (PUMA 560), localisation, vision monoculaire, asservissement 2D, 3D et 2D1/2, commande visuelle, commande du robot.

ABSTRACT :

The work presented in this thesis is a study of the visual servoing of an arm manipulator. We have presented the necessary tools for the monocular vision localization. We also study the three types of visual servoing 2D, 3D and 2D1/2 where we apply several types of controls for robot and vision. The simulations were made on the arm of robot PUMA 560.

Key words:
Visual servoing, arm manipulator (PUMA 560), localization, monocular vision, visual servoing 2D, 3D and 2D1/2, vision control, robot control.

РЕЗЮМЕ :

Настоящая работа является исследованием визуального привода руки робота. Изучается необходимый инструментарий для местоопределения путем монокулярного видения. Представлены и изучены три типа визуального привода 2Д, 3Д и 2½Д. Мы применили различные типы команд для двух контуров видения и робота. Испытания проводились на роботе «PUMA 560».

Ключевые слова:
Визуальный привод, рука-манипулятор (PUMA 560), местонахождение, монокулярное видение, приводы 2Д, 3Д и 2½ Д, визуальная команда, команда робота.

R E M E R C I E M E N T S :

Nous tenons à exprimer nos vifs remerciements à notre promoteur, monsieur H. CHEKIREB, pour nous avoir proposé ce sujet, et conseillé tout au long de notre travail, pour sa patience et sa confiance.

Nous remercions très chaleureusement les autres membres du jury pour l'honneur qu'ils nous ont fait en acceptant d'être rapporteurs de notre mémoire.

Nous souhaitons aussi remercier tous les enseignants de l'Ecole Nationale Polytechnique d'Alger, et en particulier nos enseignants d'Automatique, pour l'enseignement qu'ils nous ont transmis, pour leur disponibilité et leur gentillesse, ainsi que pour tous leurs sacrifices afin de nous offrir les meilleures conditions d'étude.

Nous tenons aussi à remercier tous les chercheurs du CDTA pour leurs accueils, en particulier monsieur M. BELHOCINE pour sa gentillesse et le temps qu'il nous a consacré.

Nous remercions aussi toute notre famille ,nos frères et nos amis pour leurs soutiens, ainsi que le personnel de l'Ecole Polytechnique.

Enfin, nous aimerions adresser nos plus fervents remerciements à nos parents, car nul autre qu'eux se sont plus sacrifiés pour notre bien et l'accomplissement de ce travail. Ils ont fait de nous ce que nous sommes aujourd'hui, et pour cela, nous leur dédions ce travail.

Que tous ceux qui ont contribués de prés ou de loin à la réalisation de ce modeste travail trouvent ici l'expression de notre sincère gratitude.

S O M M A I R E

Introduction générale

Dépourvus de moyens de percevoir leur environnement, les robots ont été cantonnés, pendant très longtemps, à réaliser des tâches simples, répétitives et *apprises à l'avance*. Dans ces conditions, des modifications imprévues dans la configuration géométrique de l'environnement peuvent entraîner de graves conséquences dans la tâche à réaliser.

Grâce à l'arrivée de capteurs qui leurs permettent de mieux appréhender l'environnement en leurs procurant de nouveaux sens tel que la vue, les robots sont passés en quelques années du stade de machine automatique programmable à celui d'assistant intelligent. En particulier, l'utilisation d'une caméra et d'algorithmes de vision associés, permet au robot de « voir » son environnement et de se localiser par rapport aux objets qui le composent. Ainsi, les asservissements visuels permettent de commander directement les déplacements du robot à partir des mesures effectuées par la caméra. Leurs exploitations ont ouvert des champs d'applications nouveaux et très variés.

Par exemple [ZAN 03], les asservissements visuels sont utilisés par EDF pour assister un téléopérateur intervenant en milieu nucléaire.

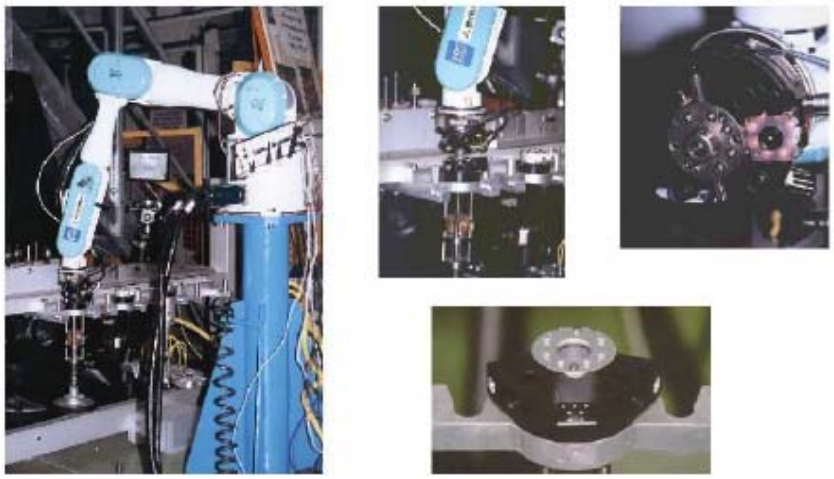


Figure 0.1. *Asservissement visuel pour la saisie automatique d'outils.*

Le système représenté ci-dessus est capable de saisir de façon automatique les outils nécessaires à l'intervention grâce à la caméra placée sur son effecteur. Par rapport à l'intervention classique sans caméra, les temps d'exécution sont réduits et le positionnement est beaucoup plus précis. En outre cette application de téléopérateur, l'utilisation des

asservissements visuels ne nécessitent pas l'installation des asservissements coûteux, dans la mesure où des caméras sont déjà présentes sur le site.

Les mêmes principes ont été récemment appliqués pour l'assistance chirurgicale [ZAN 03], pour cette application, on commande les déplacements d'un robot manipulant à partir de l'image fournie par l'endoscope. Ceci permet au chirurgien de se concentrer sur les gestes les plus délicats.

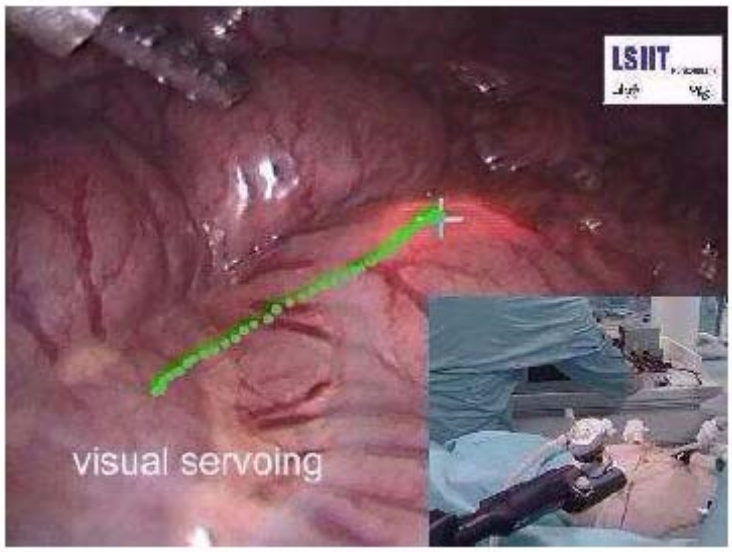


Figure 0.2. *Asservissement visuel en milieu chirurgical.*

Bien que les méthodes d'asservissement visuel existantes soient, pour certaines applications, très performantes, beaucoup de problèmes théoriques sont encore ouverts. En effet, en l'état actuel, aucune technique d'asservissement visuel ne réunit à la fois toutes les caractéristiques de précision, simplicité, de mise en œuvre et de fiabilité.

Notre mémoire a été organisé en cinq chapitres de la manière suivante :

- Un premier chapitre décrivant l'état de l'art en asservissement visuel où on décrit brièvement les différents types d'asservissement visuel ;
- Dans le second chapitre, on étudie le problème d'intégration de la vision à la robotique où on traitera le calibrage ainsi que le calcul de pose ;
- Les chapitres 3, 4 et 5, traiteront les différents types d'asservissements visuels 2D, 3D et 2D1/2 respectivement.

Les différentes simulations ont été appliquées au robot PUMA 560, afin de valider les résultats théoriques.

Chapitre

1

Introduction à l'asservissement Visuel

Préambule :

La plupart des manipulateurs industriels sont des machines ne possédants aucune capacité de perception intrinsèque. En environnement manufacturier, un programme de commande fait l'hypothèse que les pièces à manipuler arrivent à l'endroit désiré avec une orientation correcte selon un ordre donné et en un temps voulu. C'est ce qui se passe encore dans la majorité des usines où la plupart des applications fonctionnent avec succès sans aucune référence sensorielle extéroceptive. Alors quel intérêt peut-on trouver à l'utilisation des capteurs et en particulier à *la vision artificielle* ?

En effet les industriels cherchent des systèmes *versatiles* permettant de réunir en une machine un instrument relativement polyvalent et évitant l'intervention humaine hormis à niveau haut telle que la programmation des tâches … Ceci réclame l'emploi des capteurs extéroceptifs permettant de renseigner le robot sur son environnement. La caméra vidéo semble être un des moyens le plus adapté pour répondre à ce problème.

1.1. Etat de l'art en asservissement visuel :

1.1.1. Introduction :

Depuis leur apparition, les robots industriels se sont concentrés de réaliser des tâches classiques qui peuvent toujours se ramener à des interactions précises avec leur environnement. De ce fait, une modification de ce dernier peut entraîner de graves conséquences sur le déroulement de la tâche à accomplir. C'est ce qui a poussé les chercheurs à munir le robot de capteurs extéroceptifs, afin de lui permettre d'appréhender cet environnement et de réagir à d'éventuels changements.

Plusieurs capteurs ont été utilisés, on cite les capteurs à ultrason, les télémètres laser, les caméras, etc. Des études récentes ont montré que l'adjonction de la faculté de vision aux robots, permet d'améliorer de manière significative leurs performances et aide à développer de nouvelles applications [GAN 99]. Ce domaine de recherche a pris le nom d'*Asservissement visuel.*

Grâce à un modèle géométrique de la scène, on peut déterminer avec une bonne précision la position de la caméra par rapport à celle-ci [CHA 90]. Cette position peut être reconstituée sans connaissance a priori de la scène si on utilise plusieurs caméras.

1.1.2. L'asservissement visuel :

Le principe de l'asservissement visuel consiste à prendre en compte des informations visuelles issues d'une ou plusieurs caméras dans la boucle de commande d'un robot afin d'en contrôler le mouvement. Les techniques d'asservissement visuel utilisent la perception afin d'améliorer l'action et augmentent donc la flexibilité et la précision des systèmes robotiques [HAS 93]. La boucle perception-vision peut être complétée par la vision active où le mouvement du robot est contrôlé afin d'améliorer la perception.

1.1.3. Les avantages de la vision :

L'utilisation de la caméra comme capteur possède plusieurs avantages :

- La mesure est passive ;
- Il n'existe aucune interaction entre la caméra et l'environnement ;
- Les caractéristiques de la mesure sont proches de l'œil humain ;
- La caméra est un capteur très précis, rapide et possédant une grande résolution ;
- La mesure utilisant la vision n'est pas coûteuse ;
- La portée est de 50cm à 10 mètres.

1.2. Configuration robot-système de vision :

On distingue généralement trois types de configurations entre le robot et le système de vision :

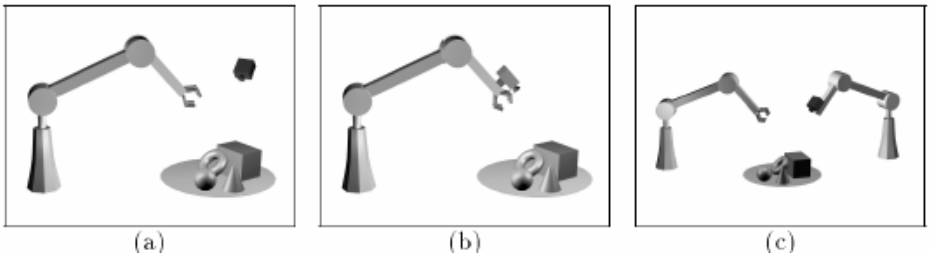


Figure 1.1. *Les différents types de configurations robot-caméra*

- La première consiste à utiliser une *caméra fixe qui observe la scène* ; les informations visuelles permettent de commander le robot mais le déplacement de celui-ci n'interagit pas sur la position de la caméra ; l'avantage de cette configuration est de voir la scène de l'*extérieur* mais son inconvénient c'est qu'elle ne propose qu'*une vue unique* sur la scène considérée.
- La seconde possibilité consiste à *embarquer la caméra sur le manipulateur* donc évoluer dans la scène avec l'organe qu'elle contrôle dont le mouvement définit l'exécution d'une tâche alors la possibilité d'*extraire* de la caméra les *données nécessaires à la régulation* de la tâche envisagée, cette configuration est dite « *eye in hand* » ou l'information visuelle permet non seulement la mesure de l'attitude ou du changement d'attitude d'un objet situé dans l'espace de travail du robot, mais également la mesure de l'attitude ou du changement d'attitude de l'organe terminal.
- La troisième possibilité rassemble les avantages des deux précédentes possibilités en *embarquant la caméra sur un second manipulateur* ; la caméra observe alors la scène de l'*extérieur* tout ayant la possibilité de *se mouvoir*.

1.3. Les différents types de tâches robotiques :

Avant d'effectuer la commande de robot, il est généralement nécessaire de déterminer les tâches que l'on désire lui faire accomplir ; celles-ci sont classées suivant trois grandes catégories : le *positionnement,* le *suivi* et la *navigation.*

1.3.1. Positionnement :

Le but d'une tâche de positionnement est d'amener le robot dans une situation d'équilibre donnée. Ce type de tâche trouve de nombreuses applications en particulier dans le secteur industriel.

Figure1.2. *Exemple de positionnement : l'instant initial (a) et l'instant final (b)*

1.3.2. Suivi de cible :

Dans le cadre d'une tâche de suivi, on considère que la cible est mobile. Le problème ne revient plus simplement à amener le robot dans une situation d'équilibre, mais à estimer le mouvement de la cible de façon à pouvoir maintenir l'interaction entre la cible et le capteur.

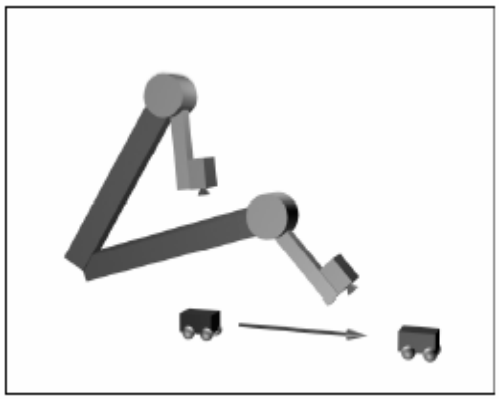


Figure1.3. *Exemple de suivi de cible mobile*

1.3.3. Navigation :

L'objectif d'une tâche de navigation est de répondre au problème fondamental :
A partir d'une situation A, aller dans une situation B suivant un chemin donné.
Pour cela, le robot doit réaliser quatre actions :

- Modéliser et interpréter l'environnement ;
- Se localiser dans cet environnement ;
- Planifier des déplacements ;
- Assurer la bonne exécution des déplacements planifiés.

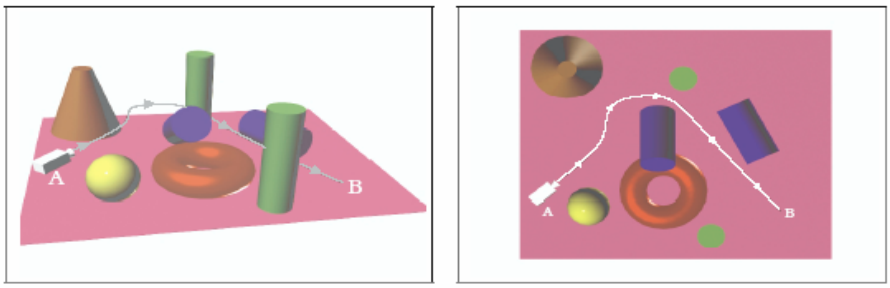


Figure1.4. *Navigation de A vers B.*

1.4. Les principaux types d'asservissement visuel :

Une classification des asservissements par vision a été établie en prenant en compte la grandeur asservie :

1.4.1. Asservissement visuel 3D :

Dans un asservissement 3D, la référence est exprimée sous la forme d'une attitude dans l'espace cartésien. L'attitude d'un repère par rapport à un autre est définie par une translation et une rotation. Dans la configuration *eye in hand*, il s'agit de l'attitude r^* d'un

repère lié à un objet vu par la caméra par rapport à un repère lié à l'organe terminal. La mesure utilisée dans l'asservissement est une estimation $\hat{r}$ de l'attitude courante r entre l'organe terminal et l'objet.

Cette mesure s'obtient grâce aux primitives extraites de l'image et à un modèle géométrique de la cible. Mais cette mesure est très sensible aux erreurs de calibrage de la caméra. Grâce à l'asservissement, $\hat{r} = r^*$ en régime établi.

Le traitement d'image se limite alors à une détection des contours des disques et à un calcul du centre de gravité des points de contour afin de déterminer le centre de chaque disque.

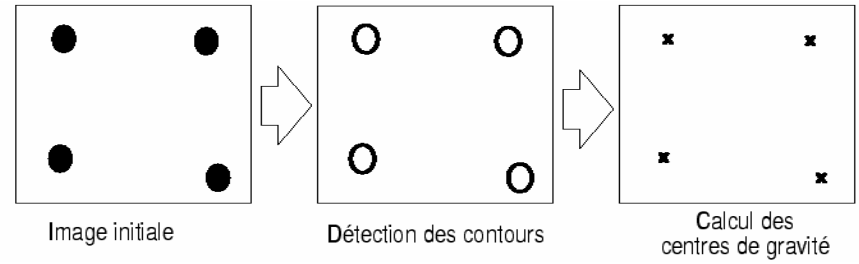


Figure 1.5 *Traitement d'image d'une cible constituée de disques coplanaires*

Le modèle de la cible, à savoir la position relative de tous les centres des disques, est connu par le système de vision qui peut donc calculer une estimation de l'attitude de l'objet. Il a été montré qu'il est nécessaire de connaître la projection d'au moins 4 primitives de type points pour pouvoir obtenir cette attitude de manière univoque.

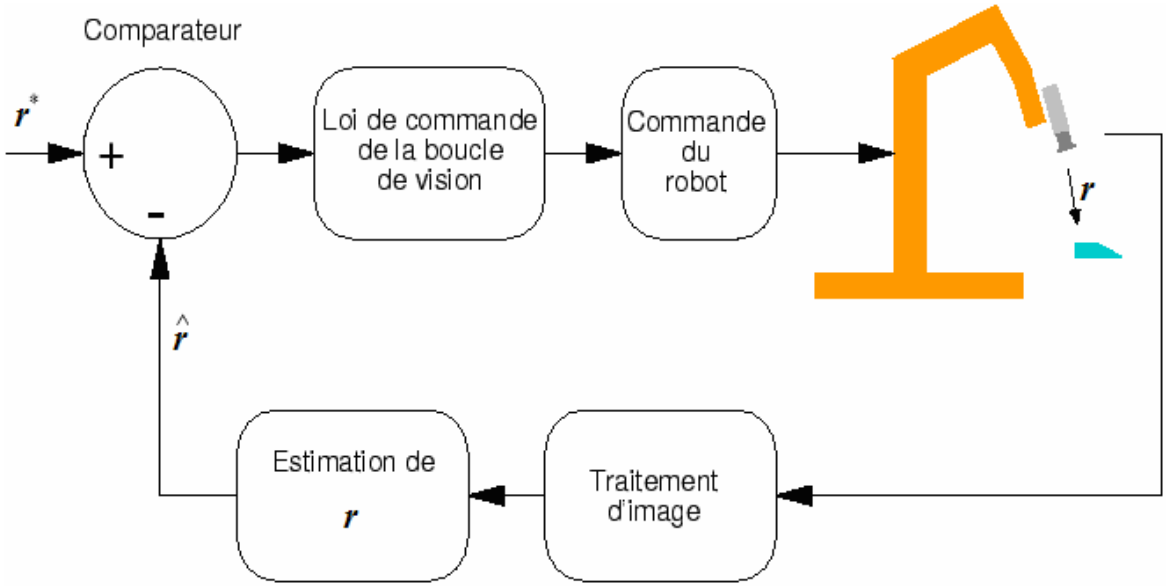


Figure 1.6. Structure d'un asservissement visuel 3D.

1.4.2. Asservissement visuel 2D :

Dans un asservissement 2D, le signal de référence est exprimé sous la forme de primitives visuelles dans l'image. Une primitive est une forme géométrique élémentaire

(point, segment de droite, portion d'ellipse,...). Elle sert à modéliser la projection d'un objet dans le plan image.

La plupart des travaux qui traitent de l'asservissement 2D utilisent des primitives constituées de points. Ces points peuvent par exemple être situés à l'intersection de segments dans l'image ou encore être extraits du centre de gravité de la projection de disques. En fait, les cibles les plus couramment rencontrées sont simplement constituées de plusieurs disques coplanaires. De telles cibles ont l'avantage de nécessiter un temps de traitement d'image faible.

Un des principaux avantages de l'asservissement 2D est qu'il n'est pas nécessaire de connaître un modèle de la cible. Dans le cas où les primitives sont des points, si on dispose d'une estimation de la profondeur de la cible (c'est à dire son éloignement par rapport à la caméra suivant l'axe optique), les informations contenues dans l'image suffisent à déterminer le déplacement de la cible par rapport à la caméra.

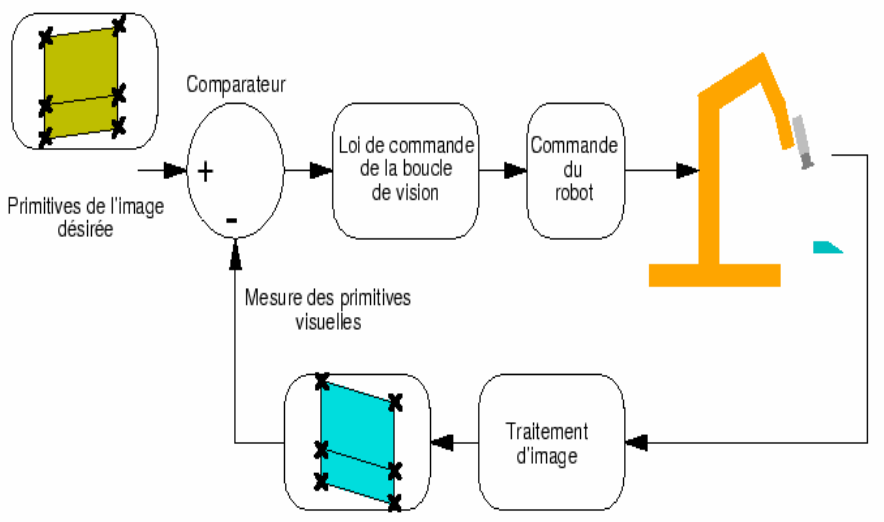


Figure 1.7 *Structure d'un asservissement visuel 2D.*

1.4.3. Asservissement visuel 2D ½ :

L'asservissement 2D½ utilise une combinaison d'informations exprimées pour certaines d'entre elles dans l'image et pour d'autres dans le repère caméra ; avec cette technique, il est possible de contraindre la trajectoire d'un point de la cible à se projeter suivant une droite dans l'image. L'objectif est un meilleur contrôle de la position de la cible dans l'image lors de régimes transitoires pour des écarts importants entre la position courante de la caméra et sa position de référence (éviter entre autres qu'un des points ne sorte de l'image).

1.4.4. Asservissement visuel d2D/dt :

Dans ce type d'asservissement, la grandeur asservie est une vitesse relative entre la caméra et la cible. La référence est définie dans le plan image par un champ de vitesse des points. La mesure peut être établie par des techniques classiques de flot optique.

1.5. Approche fonction de tâche en asservissement visuel :

Une tâche robotique peut généralement être exprimée comme le régulation, la réalisation d'une fonction de tâche à l'aide d'asservissement visuel nécessite la sélection d'informations adéquates ; ceci revient à définir une fonction de tâche ayant les propriétés assurant la capacité de réaliser la tâche suivie .L'élaboration d'une loi de commande en boucle fermée repose sur ces informations ; elle a pour but de réguler cette tâche ; une fois que la fonction de tâche a été définie, on doit concevoir une loi de commande afin de la réguler à zéro ; les fonctions de tâches classiques sont [MAL 98] :

- $e(q,t) = r(s(q,t)) - r(s^*)$. La position *r* de l'effecteur du robot est estimée à partir des informations visuelles *s(t)* ; ce type de fonction de tâche est utilisé dans les *asservissement visuels 3D* car le contrôle de la caméra s'effectue dans l'*espace cartésien* [WIL 96] ;

- $e(q,t) = C(s(q,t) - s^*)$ Ou C est une matrice qui permet de tenir compte d'une éventuelle redondance d'informations ; la fonction de tâche est utilisée dans les *asservissements visuels 2D* car le contrôle s'effectue dans l'*image* [CHA 90] ;

Ces deux méthodes sont les plus répandues dans la littérature et seront détaillées davantage par la suite.

1.6. Obtention de la consigne s^* [MAL 98] :

Une des difficultés majeures pour la construction de la fonction de tâche est la définition de la trajectoire désirée s^*. En effet, il est nécessaire de construire la fonction de tâche de telle manière qu'elle soit nulle si les informations visuelles courantes coïncident avec les informations visuelles de référence.

Une première méthode pour définir les informations visuelles désirées est basée sur leurs *modélisation* en utilisant des connaissances à priori sur la scène observée ; cette méthode demande, entre autre, une connaissance parfaite de la géométrie du système de vision pour que s^* puisse physiquement exister. Une technique pour éviter cette phase de modélisation consiste à effectuer un *apprentissage expérimental* de l'image de référence.

1.7. Conclusion :

Les systèmes d'asservissement visuel sont désormais opérationnels sur les sites industriels. Ces systèmes présentent des caractéristiques différentes et le choix d'une technique plutôt qu'une autre n'est pas toujours facile.

L'objectif n'étant pas de déterminer la meilleure technique mais la technique la plus adaptée pour une application donnée suivant cinq critères : les informations nécessaires pour la modélisation, les performances, la sensibilité au bruit de mesure, le comportement dynamique et la robustesse aux erreurs de calibrage.

Les différentes approches liées aux problèmes de modélisation de la caméra, son calibrage et l'approche calcul de poses seront détaillés dans le prochain chapitre.

Chapitre

2

Intégration vision/robotique : Modélisation, calibrage et calcul de pose

Préambule :

Les espaces de travail des robots disposent actuellement d'un grand nombre de capteurs externes, fournissant ainsi les informations nécessaires à la commande du robot. Le capteur le plus performant du point de vue volume d'informations est peut être le système de vision. C'est un outil capable de fournir toutes les informations nécessaires pour le contrôle d'une cellule robotisée, ainsi que les informations sur l'orientation et la position de tous les objets qu'il observe.

La vision peut se faire par une caméra (elle est dite dans ce cas monoculaire) ou par plusieurs caméras (binoculaire dans le cas de deux caméras).

Le calcul de pose a suscité beaucoup d'intérêts dans le domaine de la vision par ordinateur dont on peut citer la réalité augmentée ou encore la reconnaissance des formes.

Dans ce chapitre, nous proposons d'étudier les outils nécessaires pour la localisation par vision monoculaire.

2.1. Calibrage Capteur/Robot :

2.1.1. Introduction :

Le principal objectif des systèmes de vision par ordinateur est l'analyse et la compréhension de l'environnement 3D qui nous entoure en vue d'application comme le contrôle industriel ou la navigation autonome des robots.

La vision active est caractérisée par sa capacité à contrôler ses paramètres pour extraire au mieux les informations visuelles de la scène : la position du capteur dans son environnement, par exemple. 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 une caméra. Il est donc nécessaire de connaître la relation géométrique entre la caméra et l'effecteur du robot. 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 de ce dernier.

Un autre domaine intéressant où ce calibrage s'avère très important est celui de l'asservissement visuel. On cherche à commander le mouvement 3D de la caméra à partir des informations visuelles. La connaissance de la relation géométrique caméra-effecteur (ou *caméra - dernière articulation du robot*) 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 ou sa dernière articulation.

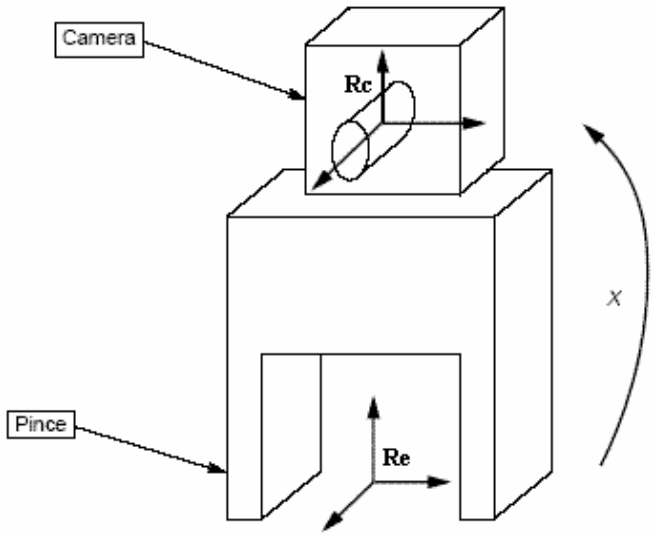


Figure 2.1. - *Le système caméra-pince est formé par la caméra qui est, soit saisie par la pince, soit montée sur cette pince. Le calibrage caméra-pince consiste à estimer la transformation rigide (rotation et translation) entre un repère attaché à la caméra et un repère attaché à la pince.*

2.1.2. Formulation du problème de calibrage [COL 98] :

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. En terme d'équations, c'est la recherche d'une matrice homogène de passage X entre la base de l'effecteur (représenté par le repère R_e) et la base de caméra (représenté par le repère R_c), (Figure 2.2).

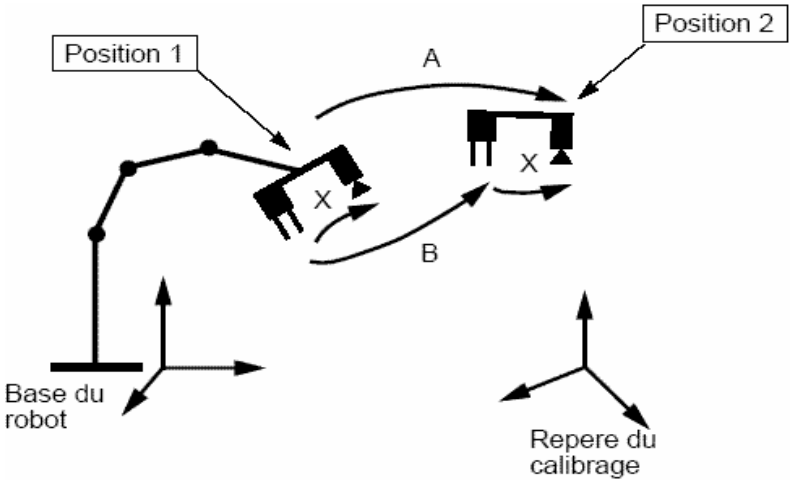


Figure 2.2. *Formulation du problème de calibrage.*

X représente une rotation R_x et une translation t_x :

$$X = \begin{bmatrix} R_X & P_X \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

L'ensemble effecteur-caméra est supposé rigidement fixé.

2.1.3. Solution au problème de calibrage [DOR 95] :

Il existe une multitude de solutions possibles pour le problème de calibrage ; faisant même l'objet de thèses. Cependant, presque toutes ces méthodes reposent sur le même principe. Ce dernier est exposé ci-dessous.

On place l'ensemble effecteur-caméra en deux positions de l'espace. Nous désignons par A (resp. B) la matrice homogène de passage de R_C^1 (de la position 1) à R_C^2 (position 2), (resp. de R_e^1 à R_e^2), comme décrit figure 2.2. Ainsi, la relation matricielle qui doit être résolue est la suivante :

$$XA = BX \qquad \Rightarrow \begin{bmatrix} R_X & P_X \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} R_A & P_A \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_B & P_B \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} R_X & P_X \\ 0 & 1 \end{bmatrix} \tag{2.2}$$

En représentant A (resp. B) par une matrice de rotation R_A (resp. R_B) et une matrice de translation P_A (resp. P_B), la relation précédente se met sous la forme :

$$\begin{cases} R_B R_X = R_X R_A & (a) \\ (R_B - I)P_X = R_X P_A - P_B & (b) \end{cases} \quad (2.3)$$

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 (2.3.b). Quand à la résolution de l'équation (2.3.a), elle est loin d'être triviale, mais les propriétés algébriques et géométriques des matrices de rotation ($R^{-1} = R^T$) nous permettent d'aboutir à l'équation suivante :

$$R_B = R_X R_A R_X^T \quad (2.4)$$

Puisque la matrice R_X est orthogonale (propriété des matrices de rotation), les matrices R_A et R_B ont les mêmes valeurs propres puisqu'elles sont semblables (propriété des matrices).

On sait qu'une matrice de rotation possède « 1 » comme une de ses valeurs propres ; soit n_A (resp. n_B) le vecteur propre associé à la valeur propre « 1 » de R_A (resp. R_B). En multipliant à droite l'équation (2.2) par n_A, on obtient :

$$\begin{aligned} R_B R_X n_A &= R_X R_A n_A \\ &= R_X n_A \end{aligned} \quad (2.5)$$

On en déduit que le vecteur $R_X n_A$ est le vecteur propre de R_B associé à la valeur propre « 1 ».

$$n_B = R_X n_A \quad (2.6)$$

La solution pour la rotation de la caméra est donnée en résolvant le système linéaire (2.6) puisque les vecteurs n_A et n_B ont été extraits de R_A et R_B.

2.2. Modélisation de la caméra :

2.2.1. Notions de bases :

Quelques notions fondamentales propres aux asservissements par vision sont décrites dans cette section. Elles sont communes à tous les asservissements de l'attitude de l'organe terminal du robot manipulateur à partir de mesures fournies par une caméra.

a/ Définition de l'attitude d'un repère [GAN 99, DOR 95] :

En robotique, il est souvent nécessaire de définir la position relative d'un corps par rapport à un autre. Lorsqu'il s'agit d'une position relative suivant 6 degrés de liberté, on parle alors *d'attitude* (*pose* en anglais). Une attitude est définie mathématiquement par une translation et une rotation qui permettent de passer d'un repère à un autre. L'attitude peut également être définie de manière unique par un vecteur de 6 coordonnées. On parle alors de coordonnées opérationnelles. Parmi ces 6 coordonnées, 3 définissent la translation et 3 autres

définissent la rotation. S'il n'y a pas d'ambiguïté pour la définition de la translation, il n'en est pas de même pour la rotation. Il existe en effet plusieurs conventions permettant de décomposer cette rotation en 3 rotations élémentaires. Celle qui est la plus largement utilisée consiste en la représentation minimale θu.

b/ Définition des repères [GAN 99] :

On appelle R_c le repère lié à la caméra. Par convention, R_c est défini comme suit (voir figure 2.3) :

- Son origine est au centre optique de la caméra.
- L'axe z_c est confondu avec l'axe optique et est dirigé vers la scène.
- L'axe x_c est vertical dans l'image et dirigé vers le haut.
- L'axe y_c est horizontal et dirigé tel que R_c soit direct.

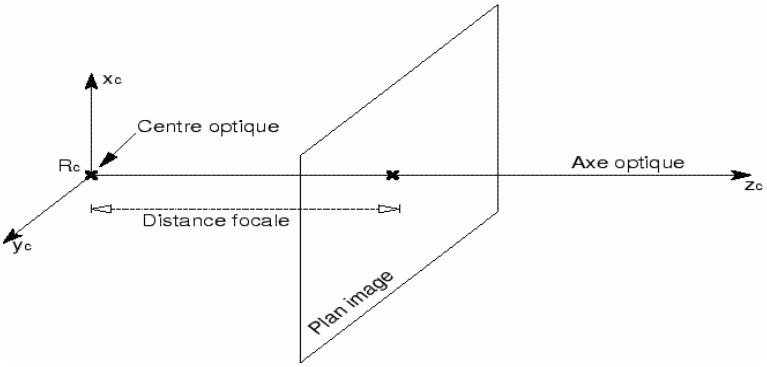


Figure 2.3. *Définition du repère caméra.*

On désigne par R_o un repère lié à un objet situé dans l'espace de travail du robot. La matrice homogène ${}^cT_o$ définit l'attitude de R_o par rapport à R_c. Il est souvent utile d'utiliser une *image virtuelle* symétrique à l'image réelle par rapport au centre de la caméra, et ceci afin d'éviter d'inverser l'image en utilisant une distance focale f négative (figure 2.3).

Dans un asservissement par vision d'un robot, la grandeur à asservir est souvent l'attitude de R_o (ou d'un autre repère lié à l'objet) par rapport à un repère R_t lié à l'organe terminal dans le cas d'une caméra déportée (*cf.* figure 2.4). Dans ce cas l'estimation des transformations ${}^cT_t$ et ${}^cT_o$ permet de déterminer la position de l'objet par rapport à l'organe terminal.

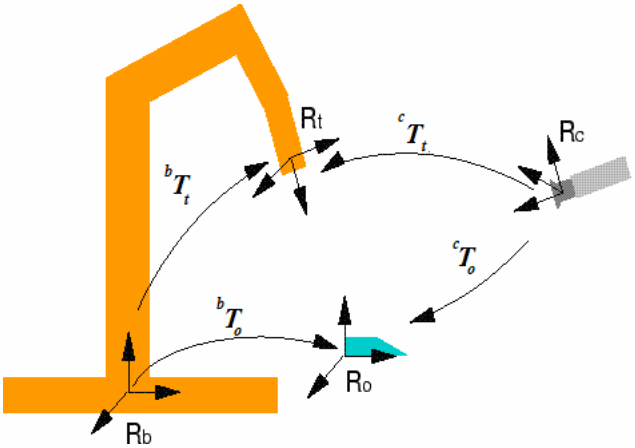


Figure 2.4. *Description de la configuration où la caméra est extérieure.*

Dans le cas d'une caméra embarquée, la recherche de ${}^{c}T_{t}$ constitue le problème de calibrage.

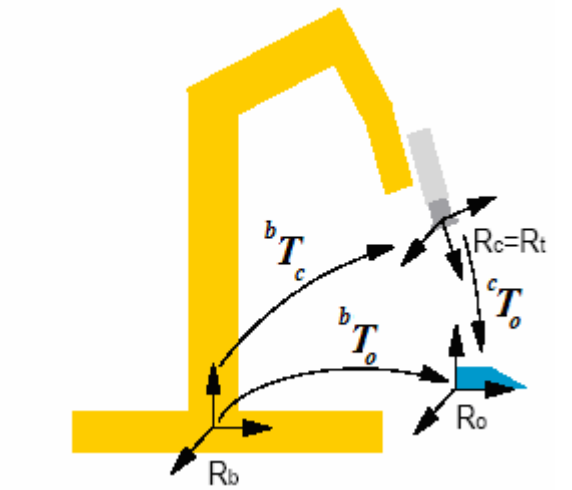


Figure 2.5. *Description de la configuration eye in hand.*

2.2.2. Modélisation de la caméra [MAL 98]:

La modélisation de la caméra permet de réaliser une projection perspective d'un point quelconque dans l'espace cartésien en un point dans l'image exprimé en pixels $p = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$, d'équivalence métrique $m = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$.

La transformation perspective entre les coordonnées métriques et les coordonnées pixels d'un *point image* est :

$$m = A^{-1} p \tag{2.7}$$

Où A est la matrice des paramètres intrinsèques de la caméra propre à celle-ci :

$$A = \begin{bmatrix} fk_u & -fk_v \cos(\phi) & u_0 \\ 0 & fk_v \sin(\phi) & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & \alpha_{uv} & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

Où u_0 et v_0 sont les coordonnées en pixels du point principal ; k_u et k_v sont des facteurs d'échelle suivant l'axe $\vec{u}$ et l'axe $\vec{v}$ (en pixels /mètres), et ϕ l'angle entre ces axes, qui dans le cas réel est de 90° (voir figure 2.6).

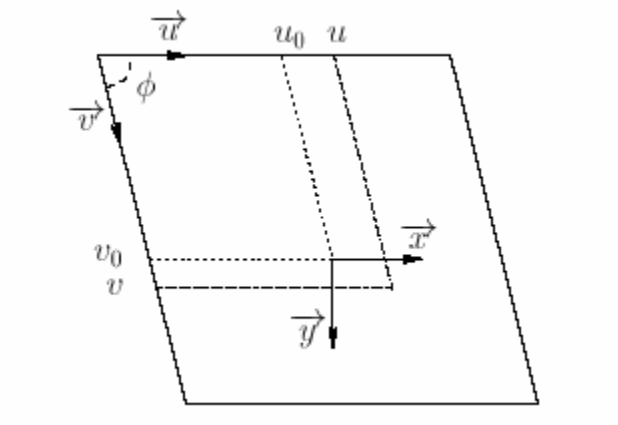


Figure 2.6. *Transformation mètre pixel.*

2.3. Calcul de pose d'un objet constitué de n points coplanaires [GAN 99] :

Le calcul de pose consiste à déterminer l'attitude (position et orientation) d'un objet par rapport à la caméra en utilisant une image de cet objet et la connaissance de son modèle 3D. Ce calcul sert de base à tout asservissement 3D. Soient :

R_0 Le repère lié à un objet plan ;
R_c Le repère lié à la caméra ;
P Un point de l'objet ;
P_p Le projeté du point P dans le plan image ;
f La longueur focale de la caméra.

Le repère R_0 est placé de manière à ce que tous les points de l'objet soient dans le plan (xy) de ce repère.

$$P = \begin{pmatrix} x_o \\ y_o \\ 0 \end{pmatrix}_{Ro} \quad , \quad P_p = \begin{pmatrix} x_p \\ y_p \\ f \end{pmatrix}_{Rc} \tag{2.9}$$

Soit ${}^cT_o$ La matrice homogène de transformation entre R_c et R_0.

$${}^cT_o = \begin{pmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{3\times 3} & P_{3\times 1} \\ 0_{1\times 3} & 1 \end{pmatrix}_{Rc} \tag{2.10}$$

D'après la loi de projection perspective faible [DOR 95] (qui considère une faible cassure due à la lentille), on a donc :

$$\begin{cases} x_p = f\dfrac{r_{11}x_0 + r_{12}y_0 + P_x}{r_{31}x_0 + r_{32}y_0 + P_z} \\ y_p = f\dfrac{r_{21}x_0 + r_{22}y_0 + P_y}{r_{31}x_0 + r_{32}y_0 + P_z} \end{cases} \tag{2.11}$$

Mais la caméra fournit les valeurs u et v de x_p et y_p en pixels, il faut donc convertir les pixels en longueurs ; pour cela on définit G_x et G_y respectivement les *grandissements* suivant x et y de la caméra [GAN 99] :

$$\begin{cases} x_p = \dfrac{f}{G_x}u \\ y_p = \dfrac{f}{G_y}v \end{cases} \tag{2.12}$$

Ceci, en considérant l'axe optique passant par l'origine du repère image (u_0=v_0=0). En divisant (2.11) par T_z et en exprimant les coordonnées des projections en pixels, on aura :

$$u = \frac{\theta_1 x_o + \theta_2 y_o + \theta_3}{\theta_7 x_o + \theta_8 y_o + 1} \qquad (a) \tag{2.13}$$

$$v = \frac{\theta_4 x_o + \theta_5 y_o + \theta_6}{\theta_7 x_o + \theta_8 y_o + 1} \qquad (b)$$

Avec :

$$\begin{cases} \theta_1 = \dfrac{r_{11} G_x}{P_z} & \theta_2 = \dfrac{r_{12} G_x}{P_z} & \theta_3 = \dfrac{P_x G_x}{P_z} & \theta_4 = \dfrac{r_{21} G_y}{P_z} \\ \theta_5 = \dfrac{r_{22} G_y}{P_z} & \theta_6 = \dfrac{P_y G_y}{P_z} & \theta_7 = \dfrac{r_{31}}{P_z} & \theta_8 = \dfrac{r_{32}}{P_z} \end{cases} \qquad (2.14)$$

En divisant (2.13.*a*) par (2.13.*b*) on aura :

$$-v\, y_o \theta_2 + u\, x_o \theta_4 + u\, y_o \theta_5 + u\, \theta_6 = v(x_o \theta_1 + \theta_3) \qquad (2.15)$$

Cette équation peut s'écrire sous forme matricielle :

$$\begin{bmatrix} -vy_o & ux_o & uy_o & u \end{bmatrix} \begin{bmatrix} \theta_2 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} = \begin{bmatrix} vx_o & v \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_3 \end{bmatrix} \qquad (2.16)$$

Si le nombre de points est $n \geqslant 4$, l'équation (2.16) admet une solution exacte pour α_{21}, α_{23}, α_{41}, α_{43}, α_{51}, α_{53}, α_{61} et α_{63} :

$$\begin{cases} \theta_2 = \alpha_{21}\theta_1 + \alpha_{23}\theta_3 \\ \theta_4 = \alpha_{41}\theta_1 + \alpha_{43}\theta_3 \\ \theta_5 = \alpha_{51}\theta_1 + \alpha_{53}\theta_3 \\ \theta_6 = \alpha_{61}\theta_1 + \alpha_{63}\theta_3 \end{cases} \qquad (2.17)$$

En substituant ces résultats de (2.13.*a*) on aura :

$$ux_o\theta_7 + uy_o\theta_8 + u - \theta_1 x_o - \theta_3 - y_o(\alpha_{21}\theta_1 + \alpha_{23}\theta_3) = 0 \qquad (2.18)$$

Cette équation mise sous forme matricielle devient :

$$\begin{bmatrix} (x_o + \alpha_{21} y_o) & (1 + \alpha_{23} y_o) & -ux_o & -uy_o \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_3 \\ \theta_7 \\ \theta_8 \end{bmatrix} = u \qquad (2.19)$$

La résolution de cette équation pour $n \geqslant 4$ permet de déterminer les valeurs de $\theta_1, \theta_3, \theta_7$ *et* θ_8. Finalement, en remplaçant θ_1 et θ_3 dans (2.17) par leurs valeurs, on déduit les valeurs de θ_2, θ_4, θ_5 et θ_6.

Afin de reconstituer tous les paramètres de la matrice homogène ${}^{c}T_{o}$, il faut déterminer la profondeur P_z. Celle-ci peut se déduire en utilisant la propriété d'orthogonalité de la matrice de rotation R. Ainsi en exprimant la normalité du premier vecteur colonne de la matrice de rotation on aura :

$$r_{11}{}^{2} + r_{21}{}^{2} + r_{31}{}^{2} = 1 \tag{2.20}$$

En utilisant les relations (2.14) on aura :

$$\left(\frac{\theta_1 P_z}{G_x}\right)^2 + \left(\frac{\theta_4 P_z}{G_y}\right)^2 + (\theta_7 P_z)^2 = 1 \tag{2.21}$$

Sachant que T_z est toujours positif (l'objet est toujours situé devant la caméra), on aura :

$$P_z = \frac{1}{\sqrt{\left(\left(\frac{\theta_1}{G_x}\right)^2 + \left(\frac{\theta_4}{G_y}\right)^2 + \theta_7{}^2\right)}} \tag{2.22}$$

2.4. Axe et angle de rotation équivalents à une rotation donnée :

On est souvent amené à devoir convertir une matrice homogène en 6 coordonnées opérationnelles ; soit R une matrice de rotation quelconque dont on voudrait extraire sa représentation minimale θu représentant un vecteur de rotation d'angle θ autour d'un axe u.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.23}$$

La représentation θu s'obtient de manière unique à partir des coefficients $r_{ij}\,(i = 1,....3,\ j = 1...3)$ de la matrice de rotation R à l'aide de l'équation suivante :

$$\theta u = \frac{\theta}{2\sin\theta}\begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix} \tag{2.24}$$

Où :

$$\theta = \arccos\left(\frac{r_{11} + r_{22} + r_{33} + 1}{2}\right) \tag{2.25}$$

D'après la relation suivante [MAL 98] :

$$[u]_x \sin\theta = \frac{R - R^T}{2} \tag{2.26}$$

Avec $[u]_x$ la matrice antisymétrique du pré produit vectoriel du vecteur *u* défini par :

$$u = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} \Rightarrow [u]_x = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \tag{2.27}$$

Le vecteur *u* est alors défini par [KHA 99] :

$$\begin{cases} u_x = sign(r_{32} - r_{23})\sqrt{\dfrac{r_{11} - \cos\theta}{1 - \cos\theta}} \\ u_y = sign(r_{13} - r_{31})\sqrt{\dfrac{r_{22} - \cos\theta}{1 - \cos\theta}} \\ u_z = sign(r_{21} - r_{12})\sqrt{\dfrac{r_{33} - \cos\theta}{1 - \cos\theta}} \end{cases} \tag{2.28}$$

2.5. Conclusion :

La vision est un des sens les plus puissants ; elle fournit une quantité importante d'informations qui nous permettent d'interagir intelligemment avec notre environnement.

La vision et la robotique nécessitent l'intégration des fonctionnalités de perception et d'action au sein d'un système robotique. Deux sujets sont liés à cette intégration : le calibrage capteur/robot et la localisation d'objets à partir des images.

La localisation d'objets à partir d'images n'est pas nécessaire dans les asservissements 2D car le contrôle s'effectue dans l'image, contrairement aux asservissements 3D qui nécessitent à chaque itération un calcul de pose, car le contrôle cette fois s'effectue dans l'espace cartésien.

Chapitre

3

Asservissement visuel 2D.

Préambule :

Une fois l'organe de vision placé et calibré sur le robot, il est nécessaire de concevoir une loi de commande régissant le système robot-vision. Différents types de méthodes existent. Dans ce chapitre, nous exposons l'asservissement visuel dit 2D, dans lequel le signal de référence est exprimé sous la forme de primitives visuelles dans l'image.

Nous commencerons par décrire la notion fondamentale de *matrice d'interaction*, essentielle pour la conception de loi de commande visuelle. Ce n'est qu'à partir de là qu'il nous sera possible d'étudier l'asservissement visuel dans des tâches de positionnement et de poursuite.

Nous commenterons ensuite les résultats obtenus, et nous tenterons de concevoir d'autres méthodes d'asservissement visuel 2D, en décrivant les avantages et les inconvénients de chacune, à travers diverses simulations.

3.1. Introduction :

Comme il a été vu auparavant, il existe différents types d'asservissement visuel. Dans ce chapitre, nous décrivons une méthode d'asservissement visuel, dite 2D, dans laquelle l'objectif est d'atteindre un motif dans l'image et non pas de contrôler une situation entre la caméra et l'objet. Ainsi on supprime la phase de reconstruction d'un modèle 3D et on élimine du même coup les erreurs sur l'estimée de la situation. Notons que grâce à cette approche on peut supprimer les erreurs dues 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).

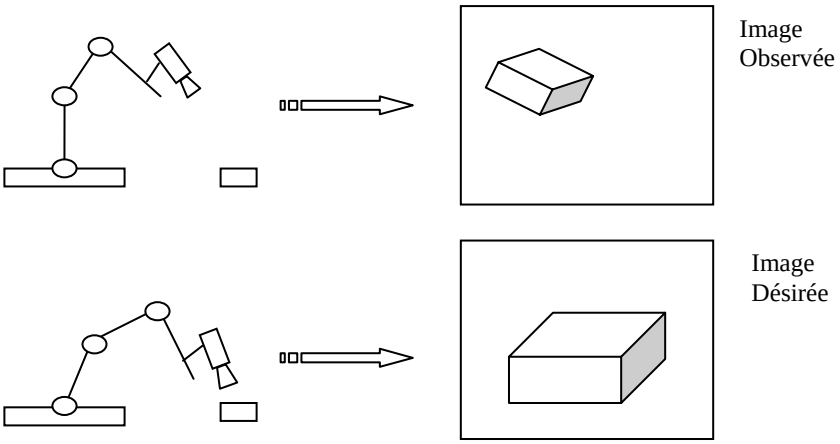


Figure 3.1. *Contrôle du robot de manière à tendre vers l'image désirée.*

3.2. Modélisation des informations visuelles :

Nous définissons l'information visuelle comme une forme géométrique particulière contenant une information *utile* sur l'objet et/ou le robot. L'ensemble de ces informations visuelles permet de définir l'objet en terme de position et orientation de ce dernier. Ces informations visuelles peuvent être des points, droites ou des cercles par exemples [CHA 90]. Dans le reste de notre travail, nous ne considérons que les informations visuelles de types *points*. Donc pour un point *i*, l'information visuelle $\mathbf{s_i}$ est définit par le vecteur suivant, dont l'unité est le *pixel* :

$$s_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} \tag{3.1}$$

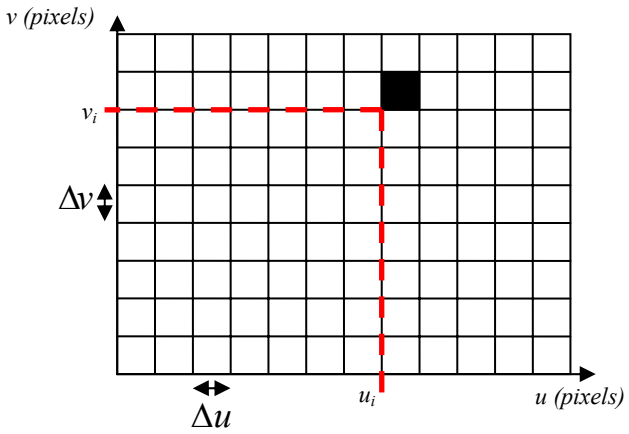


Figure 3.2. *Coordonnées d'une information visuelle de type point, dans le repère image.*

Un ensemble **s** de *n* informations visuelles peut être pris en considération dans un schéma d'asservissement visuel à partir du moment où il est définit par la donnée d'une application différentiable de SE_3 dans $\Re^n$ [KHA 02] :

$$s = s(r(t), t) \tag{3.2}$$

Où $r(t)$, élément de l'espace SE_3 de représentation des repères et des corps rigides, décrit la situation à l'instant t entre la caméra et son environnement. On considère donc que seuls les mouvements de la caméra et/ou des objets qu'elle perçoit sont susceptibles de faire varier la valeur d'une information visuelle.

La différentielle de **s** permet de relier les variations des informations visuelles au mouvement relatif entre la caméra et la scène. En dérivant (3.2) on obtient :

$$\dot{s} = \frac{\partial s}{\partial r}\dot{r} + \frac{\partial s}{\partial t} \tag{3.3}$$

$\frac{\partial s}{\partial t}$ Représente la variation de **s** due au mouvement propre (et généralement inconnu) de l'objet. Si l'on considère une caméra embarquée sur l'effecteur d'un robot observant un objet (configuration *eye-in-hand*), le lien entre $\dot{s}$ et la vitesse des variables articulaires $\dot{q}$ du robot s'obtient aisément depuis (3.3) :

$$\dot{s} = L_s \cdot W \cdot {}^n J_n(q) \cdot \dot{q} + \frac{\partial s}{\partial t} \tag{3.4}$$

Où ${}^n J_n(q)$ est le *Jacobien du robot* exprimé dans le repère R_n de son organe terminal [KHA 99]. L_s est une matrice Jacobienne de dimension $2n \times 6$, où n est le nombre d'informations visuelles. On l'appelle aussi *Jacobien image* ou *matrice d'interaction* associée à **s** ; et W la matrice de transformation du torseur cinématique pour passer de son expression dans le repère de la caméra R_c au repère de la dernière articulation R_n. Cette matrice, constante si la caméra est rigidement liée à l'organe terminal du robot, est donnée par [KHA 99] :

$$W = \begin{pmatrix} {}^c R_n & {}^c \tilde{P}_n \, {}^c R_n \\ 0_3 & {}^c R_n \end{pmatrix} \tag{3.5}$$

Nous allons à présent donner la forme analytique de la matrice d'interaction, pour des formes géométriques de types points. D'autres formes géométriques sont étudiées dans [KHA 02] et [DOR 95]. Par la suite, toutes les grandeurs nécessaires (coordonnées et vitesse de points, torseur cinématique, etc.) sont exprimées dans le repère de la caméra illustré sur la figure (3.3) :

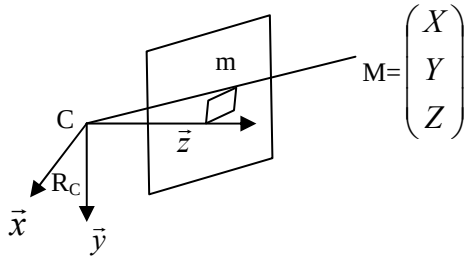


Figure 3.3. *Modèle de la caméra.*

Dans un cas très classique, le modèle mathématique d'une caméra est définit par une projection perspective telle que pour tout point $M = \begin{pmatrix} X & Y & Z \end{pmatrix}^T$ de l'espace opérationnel, se projette sur le plan image en un point $m = \begin{pmatrix} x & y & f \end{pmatrix}^T$ tel que schématisé sur la figure (3.3), avec :

$$x = fX/Z, \quad y = fY/Z \tag{3.6}$$

En différentiant cette équation, on obtient les variations dans l'image des coordonnées x et y de m par rapport à la vitesse $\dot{M}$ des coordonnées du point M :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} f/Z & 0 & -X \cdot f/Z^2 \\ 0 & f/Z & -Y \cdot f/Z^2 \end{pmatrix} \dot{M} \tag{3.7}$$

En utilisant la relation cinématique suivante :

$$\overrightarrow{V(M)} = \overrightarrow{V(C)} + \omega \times \overrightarrow{CM} \tag{3.8}$$

Nous aboutissons à la relation suivante :

$$\dot{M} = -V - \omega \times M = -V + \tilde{M}\omega = \begin{pmatrix} -I_3 & \tilde{M} \end{pmatrix} \cdot T \tag{3.9}$$

Où $\tilde{M}$ est la matrice traduisant le produit vectoriel, et décrite dans [KHA 99] et T le torseur cinématique entre la caméra et son environnement. L'équation (3.6) peut donc se mettre sous la forme [DOR 95] :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = L_{xy_j} T \tag{3.10}$$

Où,

$$L_{xy_j} = \begin{pmatrix} -f/Z & 0 & x \cdot f/Z & xy & -(1+x^2) & y \\ 0 & -f/Z & y \cdot f/Z & 1+y^2 & -xy & -x \end{pmatrix} \tag{3.11}$$

$L_{s\,j}$ est le Jacobien image associé aux coordonnées normalisés du point j, dans le repère caméra ;

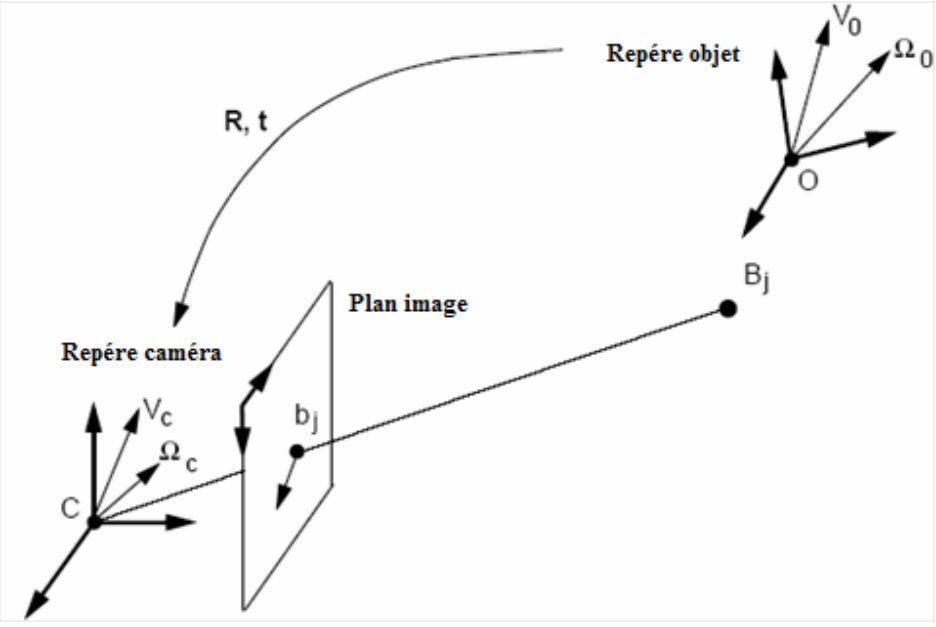


Figure 3.4. *Relation entre le torseur cinématique de la caméra et la vitesse 2D de l'image d'un point (b_j) de l'objet.*

Il est aussi possible d'exprimer la matrice d'interaction associée aux coordonnées d'un point exprimé directement en pixels [MAL 98]. Pour ce faire, nous emploierons la matrice contenant les paramètres intrinsèques de la caméra décrite au chapitre 2 et que nous rappelons ci-dessous, dans le cas où les vecteurs $\vec{u}$ et $\vec{v}$ du plan image sont perpendiculaires :

$$A = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

Ce qui permet d'écrire que :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = A^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.13}$$

et que :

$$L_{s\,j} = \begin{bmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{bmatrix} \cdot L_{xy\,j} \tag{3.14}$$

Pour un système de vision formé par *n* points, le jacobien image s'écrit :

$$L_s = \begin{pmatrix} L_{s1} \\ \vdots \\ L_{s\,n} \end{pmatrix} \tag{3.15}$$

Reste à définir le nombre d'informations visuelles à choisir. Il a été montré [CHA 02] que le nombre d'informations visuelles à choisir doit être de sorte que la dimension du vecteur **s** soit supérieur au nombre de degré de liberté du robot. D'autres recherches montrent que pour éviter d'avoir une perte de rang de la matrice d'interaction, le nombre minimal d'informations visuelles à choisir doit être supérieur ou égal à 4.

Ainsi, la matrice d'interaction joue un rôle important, dans le sens où elle lie les variations de l'image aux variations articulaires du robot. Nous abordons à présent la commande du robot en y introduisant l'information visuelle *s*.

3.3. Commande du bras manipulateur :

La forme générale du modèle dynamique d'un bras manipulateur est donnée sous la forme [KHA 99] :

$$\Gamma = A(q)\,\ddot{q} + H(q, \dot{q}) \tag{3. 16}$$

Où Γ est le couple articulaire, appliqué à chaque articulation ; q et $\dot{q}$ sont respectivement la variable articulaire et sa vitesse. Quant à la matrice *A(q)* et au vecteur *H(q, $\dot{q}$)*, ils sont caractéristiques au bras manipulateur employé.

Dans notre cas, nous emploierons le bras manipulateur PUMA 560, à 6 degrés de liberté. Son modèle dynamique explicite est donné par [ARM 86]. Mais dans tout les cas, nous décrirons dans cette section une méthode générale pour la commande d'un bras manipulateur, en nous basant sur la relation (3.16). Donnons à présent l'expression d'état du bras manipulateur :

$$\begin{pmatrix} \dot{q} \\ \ddot{q} \end{pmatrix} = \begin{pmatrix} \dot{q} \\ -A^{-1}(q) \cdot H(q,\dot{q}) \end{pmatrix} + \begin{pmatrix} 0 \\ A^{-1}(q) \end{pmatrix} \cdot \Gamma \tag{3.17}$$

Avec $X = \begin{pmatrix} q \\ \dot{q} \end{pmatrix}$ comme variable d'état et Γ le signal de commande.

La loi de commande choisie doit tenir compte du fait de la non linéarité du système. Il est donc inutile de commander le système en le linéarisant autour d'un point de fonctionnement. La commande par Backsteeping, la commande par mode de glissement, la commande linéarisante ou l'emploi d'un régulateur PI peuvent commander correctement le système. Nous ne les décrirons pas toutes dans ce mémoire mais nous en emploierons une qui :

- Stabilise le robot ;
- Permet de suivre correctement les variations de la consigne.

Evidement, toutes les lois de commande citées précédemment comme exemples permettent de vérifier ces deux contraintes. Nous choisissons d'employer un régulateur PI pour l'exemple ; et dans ce cas, la loi de commande s'écrit :

$$\Gamma = \ddot{q}_d + K_v(\dot{q} - \dot{q}_d) + K_p(q - q_d) \tag{3.18}$$

Il s'agit de retrouver les termes $\ddot{q}_d, \dot{q}, \dot{q}_d, q, q_d$ à partir de l'information visuelle *s*. Cependant, cette méthode ne peut s'appliquer que dans le cas particulier ou *l'objet à saisir est immobile*. En effet, la fonction *f* qui traduit le passage de l'espace des informations visuelles à l'espace articulaire *q=f(s)* n'est déterminable que dans ce seul cas particulier. La figure 3.5. illustre comment une même image observée peut résulter de différentes configurations du robot dans le cas où l'objet est mobile.

Dans le cas où l'objet à saisir est fixe, il est possible de retrouver à chaque instant la configuration du robot à partir des informations visuelles et de la *position initiale* du robot. Considérons pour cela les transformations de bases schématisées sur la figure 3.6. En considérant l'objet immobile, la transformation entre le repère de base et l'objet reste constante.

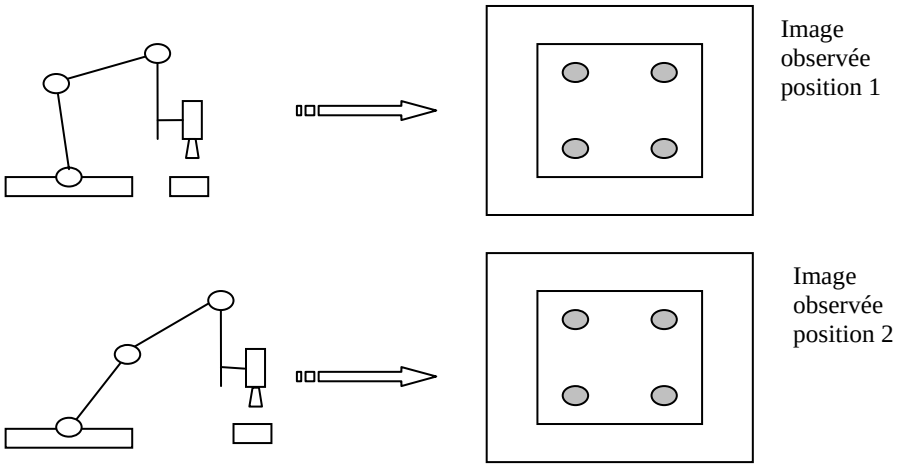


Figure 3.5. *Schéma de deux dispositions différentes du robot ayant la même image observée.*

De plus, la transformation ${}^{Caméra}T_{Objet}$ pouvant être estimée à partir d'un algorithme de calcul de pose (comme celui cité au chapitre 2), il est possible d'estimer la transformation ${}^{Base}T_{Objet}$ à partir de la connaissance de la configuration initiale du robot :

$$ {}^{Base}T_{Objet} = {}^{Base}T_{Effecteur}(q_0) \cdot {}^{Effecteur}T_{Caméra} \cdot {}^{Caméra}T_{Objet}(s_0), \qquad Cons\tan te \tag{3.19}$$

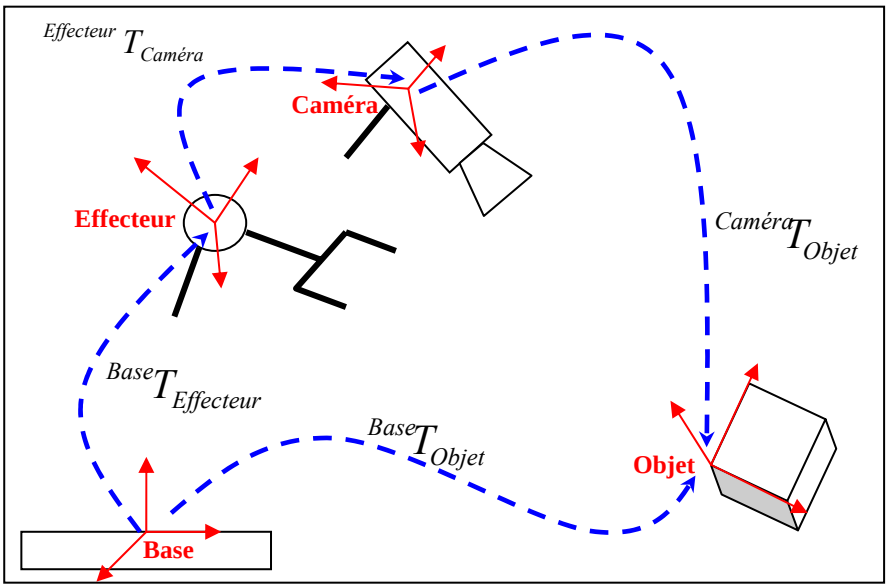


Figure 3.6. *Différentes transformations de bases présentant dans le système.*

Il est donc possible à partir de cette équation d'estimer la transformation entre la base du robot et celle de l'effecteur (ou dernière articulation) par la relation suivante :

$$ {}^{Base}T_{Effecteur}(s) = {}^{Base}T_{Objet} \cdot {}^{Caméra}T^{-1}_{Objet}(s) \cdot {}^{Effecteur}T^{-1}_{Caméra} \tag{3.20}$$

A partir du calcul de ${}^{Base}T_{Effecteur}(s)$, les articulations du robot q s'obtiennent par le calcul du modèle géométrique inverse. Représentons ces opérations comme une fonction de s permettant le calcul de q : $q=f(s)$. De même pour le calcul de l'articulation désirée : $q_d=f(s^*)$; où s^* représente l'image désirée, *constante même si l'objet est en mouvement*, du fait qu'elle représente la position relative entre la caméra et l'objet, nécessaire pour l'accomplissement de

la tâche souhaitée. Selon l'équation (3.4) et sans oublier que nous sommes dans le cas particulier ou l'objet ne dispose pas d'un mouvement propre, nous avons :

$$\dot{q} = \left(L_s W^n J_n\left(f(s)\right)\right)^+ \dot{s} \tag{3.21}$$

Avec l'utilisation de la pseudo inverse d'une matrice non carrée, tel que :

$$X^+ = \left(X^T X\right)^{-1} X^T \tag{3.22}$$

Ainsi, pour le calcul de la vitesse et accélération articulaire désirée, nous avons :

$$\dot{q}_d = \left(L_s\big|_{s=s^*} W^n J_n\left(f(s^*)\right)\right)^+ \dot{s}^* = \underline{0} \quad \text{car } \dot{s}^* = \underline{0} \tag{3.23}$$

Et de ce fait $\ddot{q}_d = \underline{0}$. Nous obtenons finalement la loi de commande, d'après (3.18) :

$$\Gamma = K_v \cdot \left(L_s \cdot W \cdot^n J_n\left(f(s)\right)\right)^+ \cdot \dot{s} + K_p \cdot \left(f(s) - f(s^*)\right) \tag{3.24}$$

La loi de commande finale ne dépend donc que des informations visuelles et de leurs vitesses. Cependant, cette commande souffre de trop fortes contraintes, à savoir :

- K_p et K_v doivent être déterminés de manière à stabiliser le système ;
- *f(s)* doit être calculé à chaque itération, ce qui alourdi le temps de calcul ;
- L'objet doit être immobile, faute de quoi, *f(s)* ne peut être estimée.

Cette dernière contrainte est la plus lourde en conséquence, car elle empêche l'utilisation de la commande pour une tâche de poursuite. D'autre part, le choix des constantes K_v et K_p pose problème au niveau de la stabilité globale du système et des performances. Pour remédier à tout ces problèmes, une idée consiste à séparer la partie commande du robot, et celle de l'organe de vision. Nous disposerons donc de deux boucles imbriquées ; la boucle interne ayant pour but principal d'assurer la stabilité du système, et la boucle externe, assurera la convergence du système vers le but spécifié, que ce soit une tâche de positionnement, ou de poursuite.

3.4. Régulation de la fonction de tâche :

La notion de fonction de tâche ayant déjà été abordée au premier chapitre, nous passerons donc à la régulation de cette dernière dans le cas d'une caméra embarquée observant l'objet. Rappelons la fonction de tâche dans le cas d'un asservissement visuel 2D :

$$e(r(t)) = C \cdot \left(s(r(t), t) - s^*\right), \quad C \in \Re^{6\times 2n} \text{ (matrice constante)} \tag{3.25}$$

La loi de commande élaborée doit donc faire tendre cette fonction vers zéro. Mais il faut souligner que, même si *e* est parfaitement régulée (c'est-à-dire *e=0*), cela n'implique pas forcément que la tâche visuelle soit réalisée (c'est-à-dire $s=s^*$). En effet, l'ensemble des configurations telles que :

$$\left(s - s^*\right) \in Ker\ C \tag{3.26}$$

aboutit à annuler *e* sans annuler (s-s^*). Il faut donc veiller, lors de la sélection des informations visuelles, à ne pas créer de minimums locaux potentiels dans l'espace de travail. A titre d'exemple, si l'on considère un carré centré parallèle au plan image, on peut montrer

qu'en choisissant pour informations visuelles les coordonnées dans l'image des quatre coins de ce carré, les configurations correspondant à des minimums locaux sont telles que la caméra observe le carré sur sa tranche. Les quatre points sont alors alignés dans l'image et l'on est alors dans le cas dégénérés situé bien évidemment en dehors de l'espace de travail. Nous nous placerons dans le reste de ce projet dans le cas ou il n'y a annulation de la fonction de tâche que si l'image courante et l'image désirée sont superposées.

Intéressons nous tout d'abord à réaliser simplement une décroissance exponentielle de la fonction de tâche, à savoir :

$$\dot{e} = -ge \quad , \quad g > 0 \tag{3.27}$$

Cette variation de l'erreur peut aussi d'après (3.4) avoir comme expression :

$$\dot{e} = C \cdot Ls \cdot W \cdot {}^{n}J_{n}(q) \cdot \dot{q} + \frac{\partial e}{\partial t} \tag{3.28}$$

Si l'on souhaite utiliser les variations articulaires comme signaux de commande, l'utilisation des relations (3.25), (3.27) et (3.28) en considérant l'image désirée fixe, nous permettent d'aboutir à la loi de commande suivante :

$$\dot{q}_d = {}^{n}J_{n}^{-1}(q) \cdot \left(CL_s W\right)^{-1} \left(-ge - \frac{\partial e}{\partial t} \right) \tag{3.29}$$

Il s'agit des variations articulaires que l'on souhaite appliquer au robot pour effectuer une tâche. Un inconvénient dans la loi de commande citée ci-dessus est le fait que des erreurs de mesure et d'estimation sont inévitables et la valeur exacte de L_s est donc inconnue. Seule une approximation $\hat{L}_s$ peut donc être considérée dans la loi de commande. De plus, le terme $\frac{\partial e}{\partial t}$ est en général inconnu. La loi de commande utilisée en pratique est donc :

$$\dot{q}_d = {}^{n}J_{n}^{-1}(q) \cdot \left(C\hat{L}_s W\right)^{-1} \left(-ge - \frac{\hat{\partial} e}{\partial t} \right) \tag{3.30}$$

Si l'on considère que cette consigne est parfaitement réalisée, l'utilisation de (3.30) dans (3.28) donne :

$$\dot{e} = -CL_s W\left(C\hat{L}_s W\right)^{-1} ge - CL_s W\left(C\hat{L}_s W\right)^{-1} \frac{\hat{\partial} e}{\partial t} + \frac{\partial e}{\partial t} \tag{3.31}$$

On constate alors, en supposant pour simplifier que $\frac{\hat{\partial} e}{\partial t} = \frac{\partial e}{\partial t} = \underline{0}$ [KHA 02], que la condition de positivité :

$$CL_s \left(C\hat{L}_s \right)^{-1} > 0 \tag{3.32}$$

est suffisante pour assurer la décroissance de $\|e\|$.

3.4.1. Choix de la matrice C :

Nous décrivons maintenant les choix possibles de la matrice *C*. Pour simplifier les calculs, considérons la condition $\frac{\hat{\partial e}}{\partial t} = \frac{\partial e}{\partial t} = \underline{0}$. Rappelons que *C* doit être une matrice constante, de dimension *6 x 2n*. Le choix le plus simple consiste à choisir pour *C* la pseudo inverse d'une approximation de la valeur de la matrice d'interaction à la position désirée :

$$C = \hat{L}^{+}_{s|s=s^*} \tag{3.33}$$

Donc, d'après la relation (3.30), la nouvelle loi de commande est :

$$\dot{q}_d = {}^{n}J_n^{-1}(q) \cdot \left(\hat{L}^{+}_{s|s=s^*}\hat{L}_s W\right)^{-1}\left(-ge - \frac{\hat{\partial e}}{\partial t}\right) \tag{3.34}$$

La condition de stabilité (3.32) se résume donc dans ce cas à :

$$\hat{L}^{+}_{s|s=s^*}L_s > 0 \tag{3.35}$$

Même si $\hat{L}^{+}_{s|s=s^*}$ est parfaitement estimée, cette condition de positivité n'est assurée que dans un voisinage autour de la position désirée. De même, le comportement exponentiel de *e* ne sera assuré qu'à cette position désirée. Il est donc possible, si la position initiale de la caméra est fortement éloignée de sa position désirée, que les trajectoires obtenues dans l'image soient peu satisfaisantes, voire ne permettent pas d'atteindre la convergence du système. En pratique, ces cas de figures ne se rencontrent que si de forts mouvements en rotation sont nécessaires [CHA 98]. D'autre part, remarquons qu'en choisissant *C* comme la pseudo inverse de la matrice d'interaction, la fonction de tâche *e* prend le sens d'erreur en position et orientation entre la situation actuelle de la caméra, et sa situation désirée.

D'autres choix de la matrice *C* sont possibles [HAS 93], mais leurs avantages par rapport à celles décrites précédemment ne semblent pas évidents, puisqu'elles ne présentent pas de bonne propriétés de découplage.

3.4.2. Estimation du paramètre $\frac{\partial e}{\partial t}$:

Une bonne partie des travaux en asservissement visuel traitant de la poursuite de cible considèrent le mouvement de l'objet comme une perturbation à rejeter le plus rapidement et le plus efficacement possible [GAN 98, GAN 99, PAP 93]. D'autres travaux utilisent des connaissances *a priori* sur la trajectoire de l'objet [ALL 93, HAS 95, RIZ 96]. D'autre part, l'utilisation d'un intégrateur est classique en automatique pour supprimer les erreurs de traînage. Notons $\mathbf{I_k}$ l'estimation de $\frac{\partial e}{\partial t}$ à l'itération *k*. On a alors :

$$I_{k+1} = I_k + \mu \cdot e_k \quad avec \quad I_0 = 0 \qquad (3.36)$$
$$= \mu \sum_{j=0}^{k} e_j$$

où μ est le gain de l'intégrateur. Cette technique ne permet de traiter correctement que les cas où l'objet est à vitesse constante. On a en effet $I_{k+1} = I_k$ si et seulement si $e_k=0$. Cela implique que les erreurs de traînage ne seront pas entièrement supprimées si le mouvement de l'objet est plus complexe.

D'autres méthodes permettent d'éliminer l'erreur de traînage due au mouvement propre de l'objet ; certaines se basent sur le fait de considérer ce mouvement comme une perturbation à éliminer le plus rapidement possible. La commande de la boucle de vision doit donc d'être robuste vis-à-vis des perturbations. Ce sont ces méthodes que nous testerons dans la suite de ce mémoire. En effet, en remarquant que toutes les perturbations agissant sur le système (couples résistants sur les actionneurs, erreurs de calibrage, erreurs de modélisation et même le mouvement propre de l'objet) ont tous des conséquences dans l'image, il serait plus judicieux de ne concevoir qu'un régulateur robuste vis-à-vis des perturbations dans la boucle de vision.

Il s'agira donc de commander le torseur cinématique, noté T, entre la caméra et son environnement. Elle est donnée selon (3.9) :

$$T = \begin{pmatrix} V \\ \omega \end{pmatrix} = W \cdot {}^{n}J_n(q) \cdot \dot{q} \qquad (3.37)$$

Selon (3.30) et en considérant le mouvement de l'objet comme une perturbation au système, la loi de commande de la boucle de vision s'écrit :

$$T_d = -\left(C\hat{L}_s\right)^{-1} \cdot g \cdot e \qquad (3.38)$$

Ce qui permet aussi de tracer le schéma de la boucle de commande :

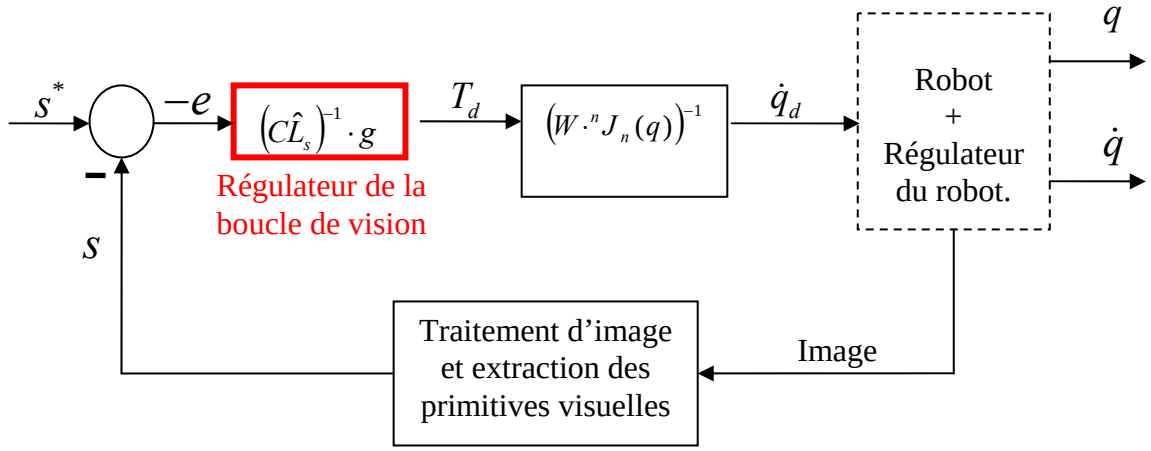


Figure 3.7. *Schéma de régulation de type 2D.*

Nous décrivons à présent le bloc associé au robot, ainsi que sa commande.

3.4.3. Commande du robot :

Comme il a été mentionné dans le paragraphe 3.3., la loi de commande choisie pour le robot doit assurer la stabilité de ce dernier, ainsi que ses performances dans le cas d'une consigne variable. Différentes méthodes de régulation existent, nous décidons d'appliquer la commande par Backsteeping, dont l'expression est donnée ci-dessous (voir annexe B) :

$$\Gamma = H(q,\dot{q}) + A(q)\cdot\left[\ddot{q}_d + (\dot{q}-\dot{q}_d)\cdot(\alpha+\beta) - (q-q_d)\cdot(1+\alpha\beta)\right] \quad (3.39)$$

$$Avec \quad \alpha,\beta < 0$$

Bien évidemment, le robot étant un système physique, la commande à appliquer est bornée. Les valeurs max des couples articulaires sont données par le tableau A.2.

Il ne s'agit donc pas seulement d'assurer la convergence des erreurs, mais à veiller à ce que les couples ne dépassent pas leurs valeurs max. Après avoir décrit la commande appliquée à la boucle du robot.

3.4.4. Simulations de l'asservissement visuel 2D :

Après avoir décrit la boucle de commande du robot et celle de l'asservissement visuel, passons à la simulation de ces derniers, disposés en cascade comme décrit dans la figure 3.7. Notons que la structure en cascade impose à la boucle de la commande du robot d'être plus *rapide* que celle de l'asservissement visuel. Ce qui impose de prendre des valeurs des paramètres α, β assez grands, mais pas trop pour éviter de dépasser les limites tolérés par les articulations.

Nous débuterons donc par décrire l'environnement de simulation, à savoir :

- Les paramètres intrinsèques de la caméra (se référer au chapitre 2) :
  - f = 1 cm, la distance focale ;
  - $u_0 = v_0 = 256$, les coordonnées en pixels du point principale dans l'image ;
  - $k_u = k_v = 5.000$, les facteurs d'échelle suivant l'axe $\vec{u}$ et l'axe $\vec{v}$ du plan image (en pixels /mètres) ;
  - ϕ = 90°, l'angle entre les axes $\vec{u}$ et $\vec{v}$ dans l'image ;
  - Le plan image est un carré de 512 pixels de cotés.
- La caméra est placée sur la dernière articulation, de manière à ce que les deux bases soient distantes de 10 cm selon l'axe x ;
- La configuration initiale du robot est $q_0 = \left[0° \quad 90° \quad 0° \quad 0° \quad 90° \quad 0°\right]^T$;
- Les primitives visuelles sont au nombre de 4 points, formant un carré. L'information visuelle à la position désirée est un carré centré dans l'image, avec $s^* = \left[306 \quad 306 \quad 206 \quad 306 \quad 206 \quad 206 \quad 306 \quad 206\right]^T$ pixels.

Ces caractéristiques seront identiques sur toutes les simulations effectués dans ce mémoire, quel qu'en soit la méthode d'asservissement choisie. Les seules modifications

apportées seront les paramètres de la commande, et l'image initiale de l'objet en termes de translation et rotation entre *l'image initiale* et *l'image désirée*.

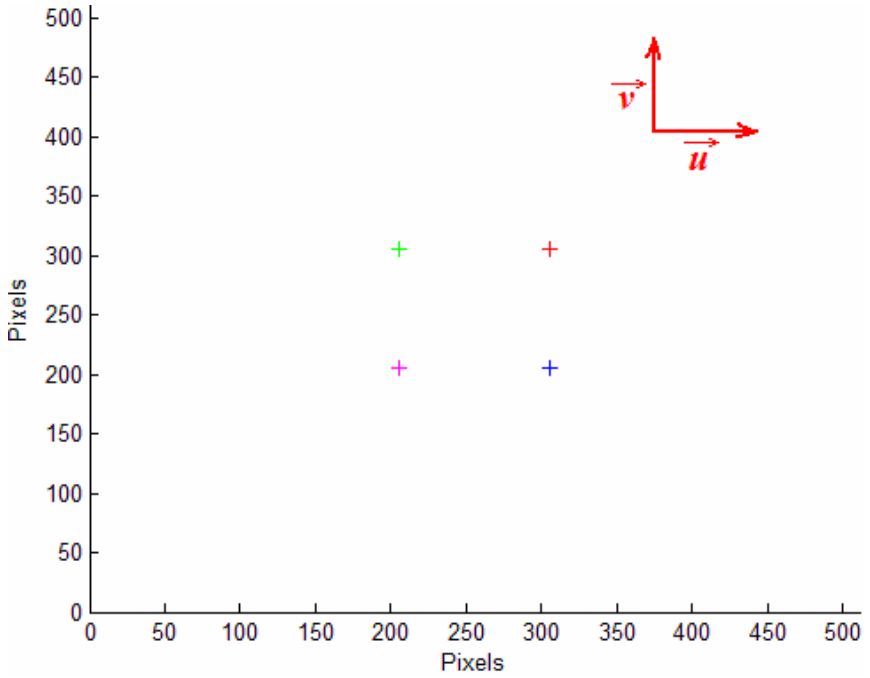


Figure 3.8. *Disposition des informations visuelles désirées.*

D'autre part, le calcul de la matrice d'interaction L_s impose l'estimation de la profondeur Z entre la caméra et l'objet à chaque itération, ce qui peut se faire par calcul de pose. Or, pour éviter la perte de temps due à ce calcul, il est courant d'employer non pas la profondeur estimée à chaque instant, mais la profondeur désirée Z_d, estimée une seule fois (lors de la phase d'apprentissage) [CHA 90]. Cette simplification a été validée par diverses expérimentations. Dans nos simulations, cette profondeur a été fixée à 15 cm. D'autre part, nous choisirons la matrice de conditionnement C comme étant la pseudo-inverse de la Jacobienne image, estimée à la position désirée $C = L_s^+|_{s=s^*}$.

Nous décrivons à présent l'asservissement visuel 2D dans le cas d'un robot *parfait* et du robot PUMA 560.

3.4.4.a. Cas d'un robot parfait :

Ce que nous appelons un « robot parfait » n'est d'autre qu'un robot exécutant parfaitement et instantanément la tâche désirée. Un tel robot est caractérisé par une fonction de transfert *identité*. Ce type de robot n'existe évidemment pas, mais permet, à travers les simulations, d'étudier certaines propriétés propres à la boucle de vision, à savoir, la trajectoire désirée dans l'image des primitives visuelles.

Appliquons la loi de commande (3.38), en considérant la constante $g=0.5$. Nous avons dans le cas d'une translation de -10 pixels en u et 39 pixels en v, et d'une rotation de 15°, la trajectoire des différentes primitives :

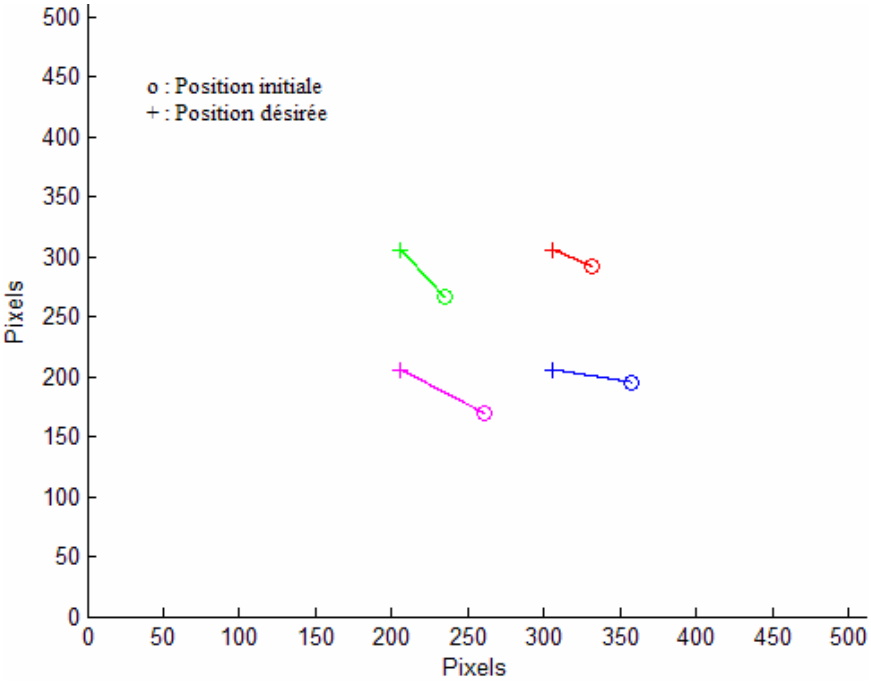


Figure 3.9.a.
Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=3 (s^{-1}) dans le cas d'un robot parfait.

Quand au tracé de l'erreur $e = C(s - s^*)$, elle est donnée par :

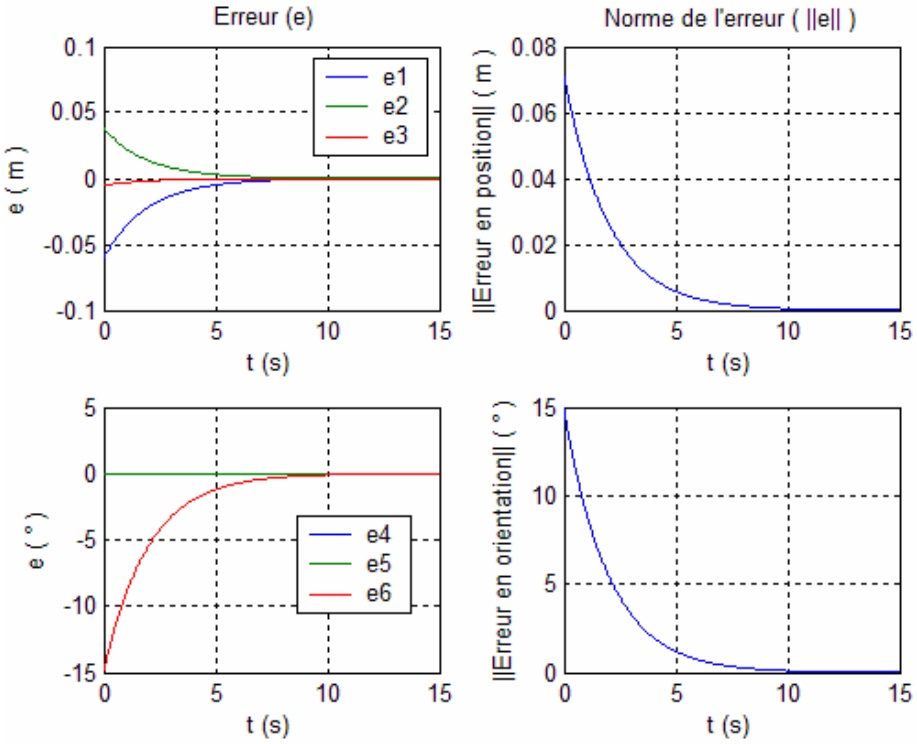


Figure 3.9.b.
Variation des erreurs pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=3 (s^{-1}) dans le cas d'un robot parfait.

Nous avons bel et bien convergence vers la position désirée, en 10 secondes. La trajectoire des points est dans ce cas optimale, représentée par des droites. Considérons maintenant un mouvement représenté par une rotation pure de 180° :

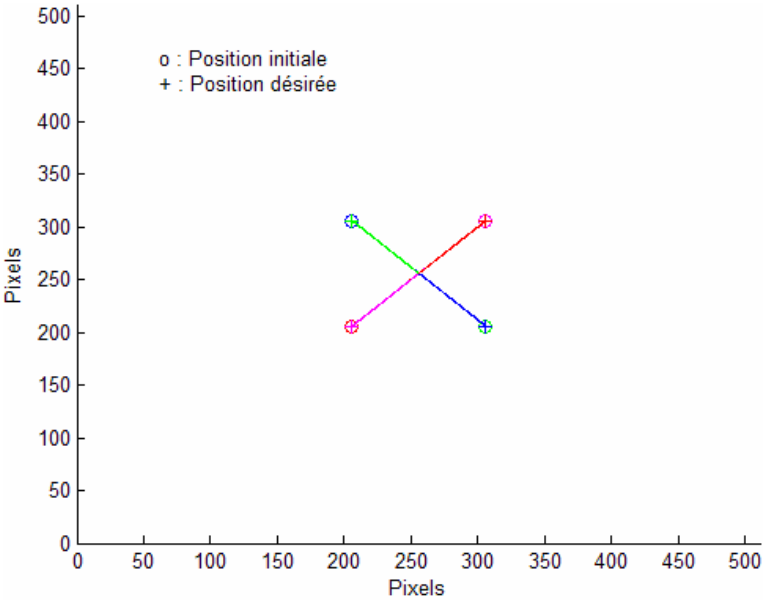


Figure 3.10. *Trajectoire dans l'image pour une rotation de 180°, avec un robot parfait.*

La trajectoire réelle à effectuer étant une rotation de 180°, notre robot *idéal*, recule jusqu'à *l'infini* (moment où les points se confondent), pour effectuer la rotation et revenir. La trajectoire décrite par le robot est *évidement* irréalisable. C'est le problème de *faisabilité*. Ceci s'explique par le fait qu'étant donnée l'idéalisme de notre robot, ce dernier n'est pas limité par la trajectoire imposée. Or dans le cas réel, la fonction de transfert du robot impose une dynamique au robot, permettant ainsi d'aboutir à des trajectoires *réalistes.*

3.4.4.b. Cas du robot PUMA 560 :

Nous considérons à présent le cas de la commande par asservissement visuel 2D d'un robot PUMA 560. Dans ce cas, nous prendrons pour la loi de commande du robot, les constantes $\alpha = \beta = -10$. Comme pour le cas précédent, simulons le comportement du robot pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec $g = 0.5$:

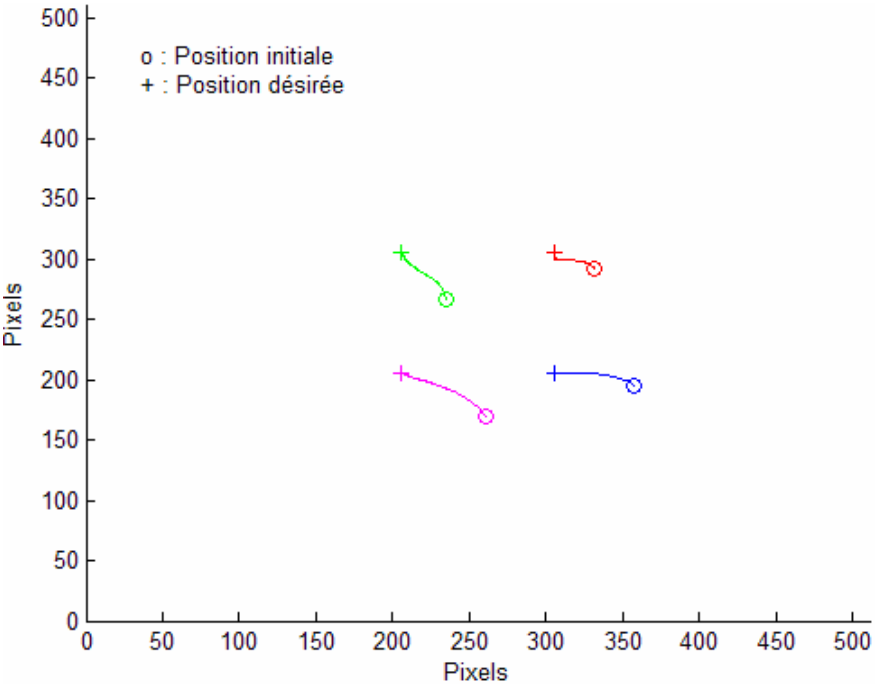


Figure 3.11.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=3 (s^{-1}) dans le cas d'un robot PUMA 560.*

Nous avons bel et bien convergence vers l'image désirée, mais plus avec des trajectoires rectilignes, mais plutôt courbés. La trajectoire n'est plus optimale du fait des limitations physiques du système. Nous donnons après le tracé des variables articulaires et vitesses articulaires, puis celles des erreurs en position et orientation ainsi que leurs normes :

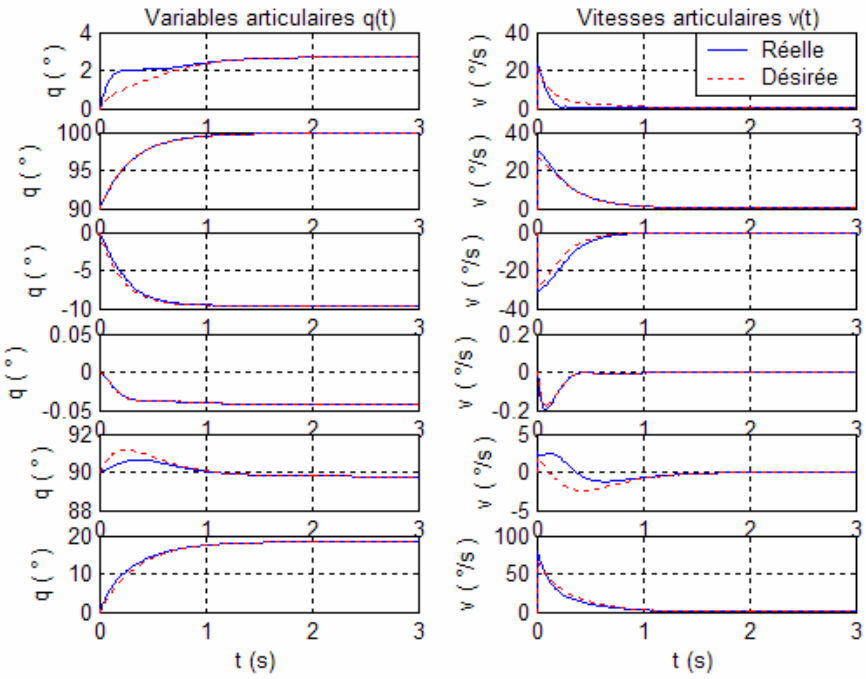


Figure 3.11.b. *Variables et vitesses articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=3 (s^{-1}) dans le cas d'un robot PUMA 560.*

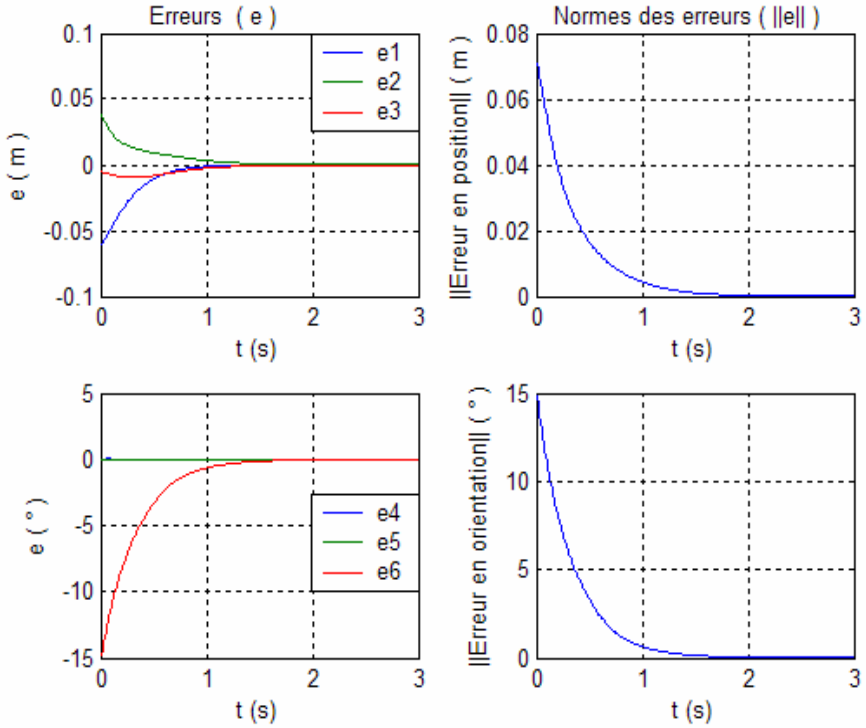


Figure 3.11.c. *Erreurs pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=3 (s^{-1}) dans le cas d'un robot PUMA 560.*

La convergence s'effectue en 2 secondes. Il ne reste plus qu'à tracer les différents couples appliqués à chaque articulation :

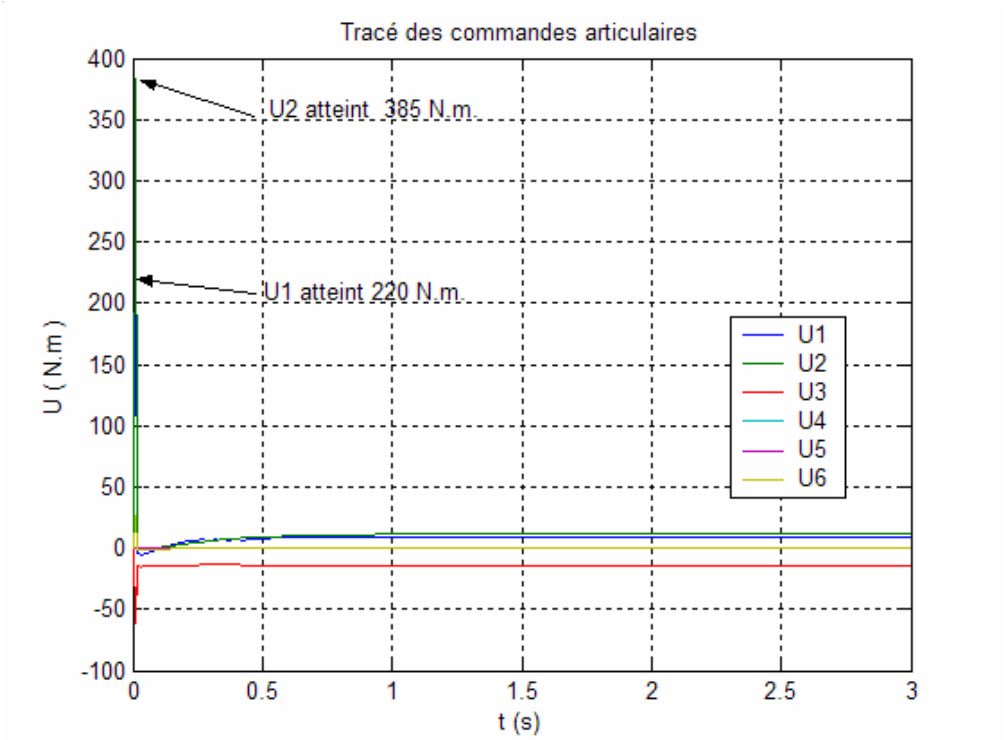


Figure 3.11.d. *Couples articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=3 (s^{-1}) dans le cas d'un robot PUMA 560.*

Le tracé des commandes articulaires révèle une forte commande à l'instant initial, sur la seconde articulation. Celle-ci atteint 385 N.m pour U_2 et 220 N.m pour U_1, alors qu'elle ne peut être supérieure à 186.4 N.m pour U_2 et 97.6 N.m pour U_1 selon le tableau 3.1. Ceci est dû au fait que la commande de la boucle de vision, représentant le torseur cinématique de la caméra a une forme exponentiellement décroissante selon l'équation (3.38) :

$$T_d(t) = -\left(C\hat{L}_s\right)^{-1} \cdot g \cdot (s_0 - s^*) \cdot e^{-gt} \tag{3.40}$$

Ce qui impose une augmentation brutale de ce dernier à l'instant initiale :

$$T_d(0) = -\left(C\hat{L}_s\right)^{-1} \cdot g \cdot (s_0 - s^*) \neq \underline{0} \tag{3.41}$$

D'où la forte impulsion à l'instant initial pour la commande. Mais il est possible de réduire ce pic, car selon (3.41), l'amplitude de ce pic a une relation directe avec le choix de g : plus g diminue, plus le pic initial diminue. Ceci est simulé à présent pour $g = 0.5(s^{-1})$, avec un temps de convergence de 8 secondes.

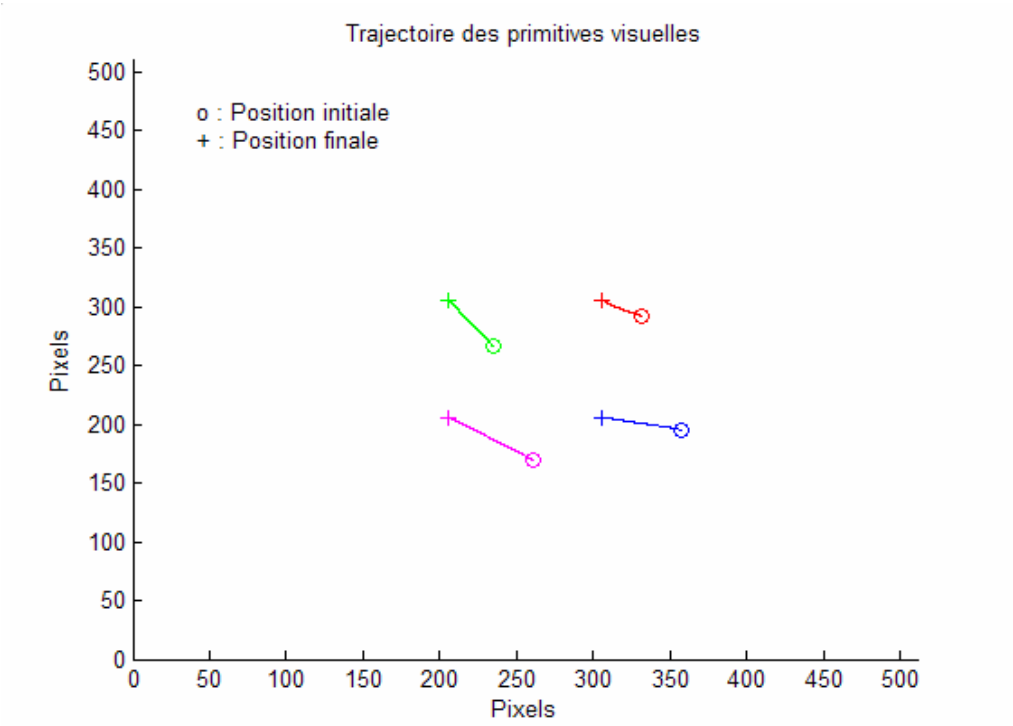


Figure 3.12.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=0.5 (s^{-1})*

Le tracé des trajectoires dans l'image permet d'aboutir à une première conclusion qui consiste en *l'indépendance des trajectoires dans l'image vis-à-vis de la modification du paramètre g*. Voici maintenant le reste des graphes obtenus.

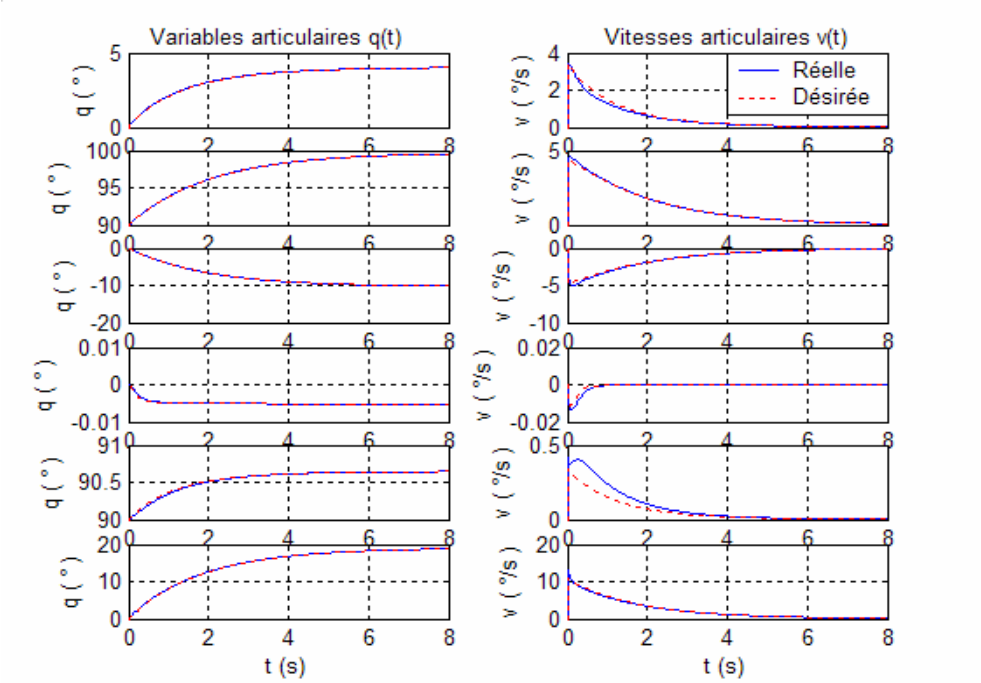


Figure 3.12.b. *Variables et vitesses articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=0.5 (s^{-1}) .*

Les variables articulaires ainsi que leurs vitesses sont sensibles aux variations du paramètre *g*. Cette sensibilité se voit sur le temps de convergence mais aussi sur l'amplitude.

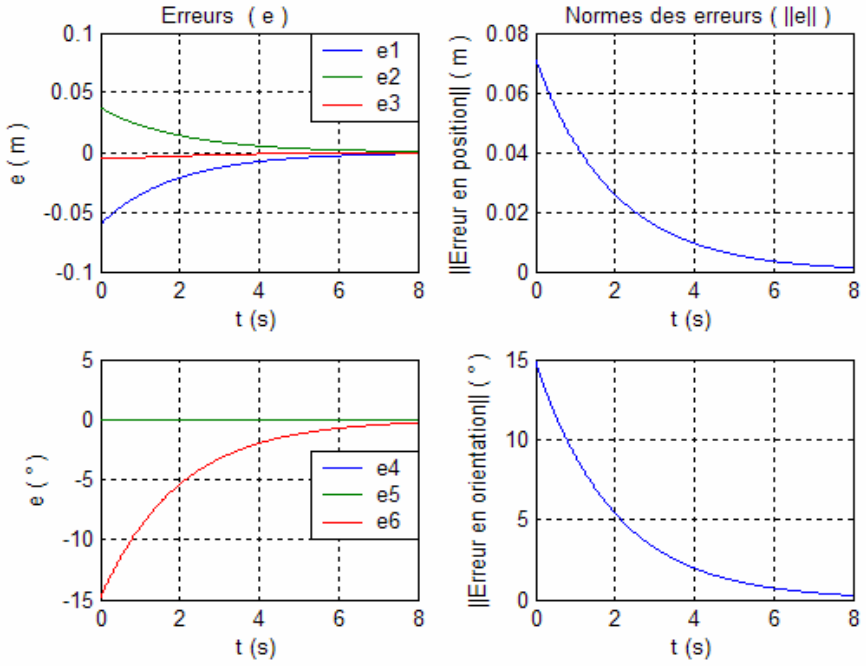


Figure 3.12.c.*Erreurs pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15° avec g=0.5(s^{-1})*

.

g n'influence les erreurs que sur le temps de convergence, et non pas sur l'amplitude. Voici finalement le tracé des signaux de commande :

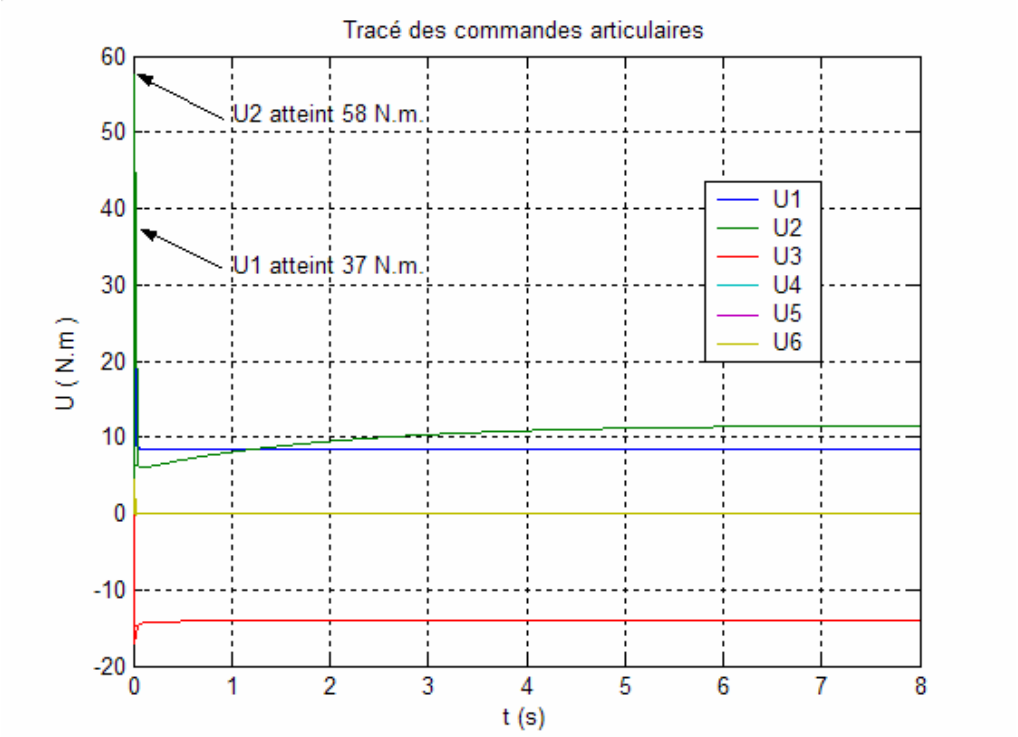


Figure 3.12.d. *Couples articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=0.5 (s^{-1}) dans le cas d'un robot PUMA 560.*

Contrairement au cas précédent, toutes les commandes sont admissibles par le système, malgré la présence du fort pic initial. Cependant, le temps de convergence du robot est nettement augmenté. Ceci est dû au fait que le torseur cinématique de la caméra décroît d'une vitesse dépendante de g selon (3.40). Plus g diminue, plus lente sera la convergence.

Nous sommes donc devant un dilemme *Energie-temps de convergence* : selon le fait que g augmente ou diminue, l'énergie nécessaire aux articulations augmente ou diminue (respectivement) et le temps de convergence diminue ou augmente (respectivement). Or, nous sommes principalement contrains de respecter les borgnitudes des différentes articulations du système.

Après avoir étudié le cas de positionnement, étudions à présent le cas de poursuite. Rappelons que la trajectoire et la vitesse de l'objet sont inconnues, et de ce fait, le terme $\frac{\partial e}{\partial t}$ est inconnu. Comme il a été dit précédemment, il est possible d'estimer ce terme de façon approximative et de l'injecter dans la loi de commande ; mais nous avons décidé de le considérer comme inconnu, voire même comme une perturbation à éliminer le plus rapidement possible. Nous donnons donc les résultats d'un cas de poursuite où l'objet est placé sur une bande convoyeur à vitesse constante :

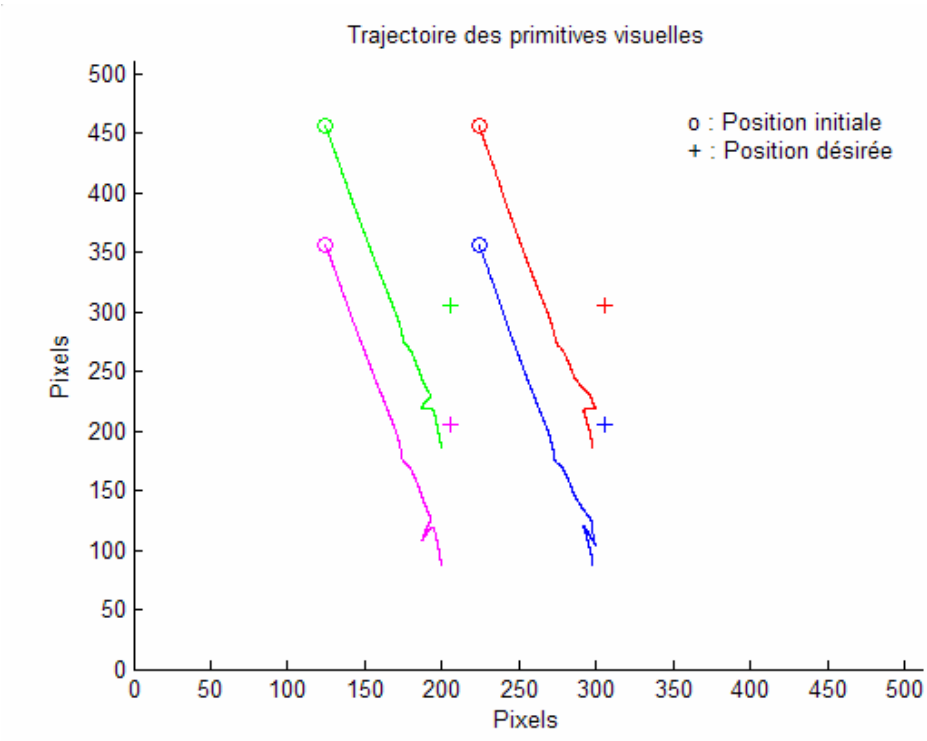
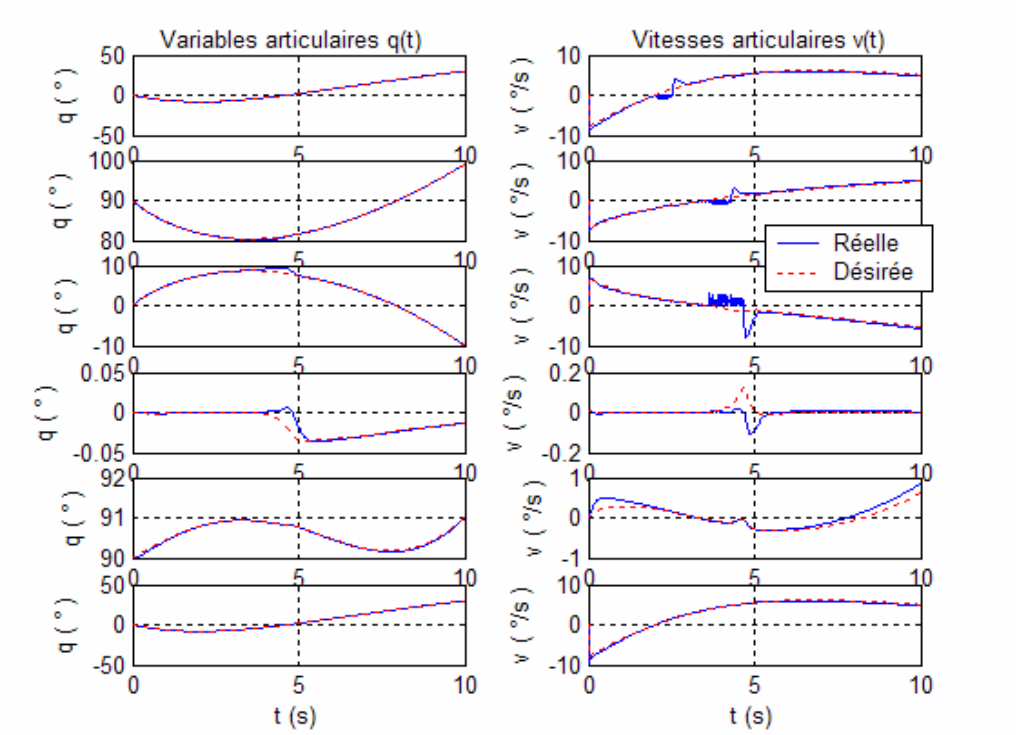


Figure 3.13.a. *Trajectoire dans l'image des primitives visuelle pour une tâche de poursuite, avec g=0.5 (s^{-1}).*



3.13.b. *Variables et vitesses articulaires pour une tâche de poursuite, avec g=0.5 (s^{-1}).*

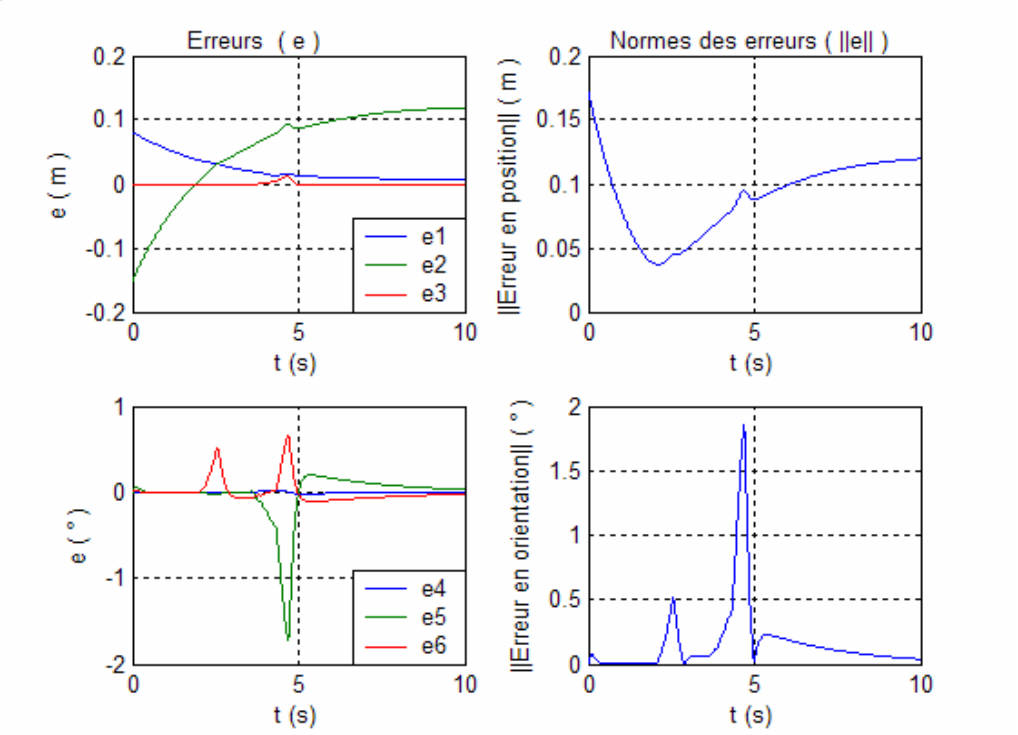


Figure 3.13.c. *Erreurs pour une tâche de poursuite, avec g=0.5 (s^{-1}).*

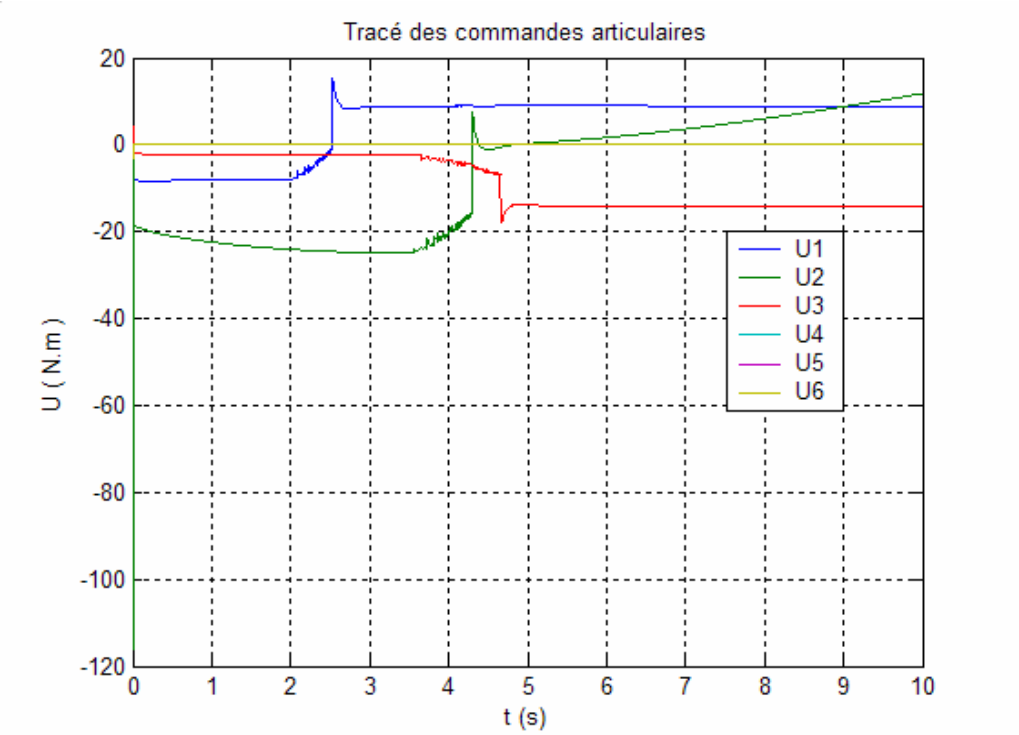


Figure 3.13.d. *Couples articulaires pour une tâche de poursuite, avec g=0.5 (s^{-1}).*

Il est clair que la tâche de poursuite n'est pas réalisée correctement. En rallongeant le temps de simulation, viens un moment où l'objet quitte l'espace de travail du robot, rendant la poursuite irréalisable. Cependant, il est possible d'améliorer cette poursuite en augmentant le gain *g* ; malheureusement, comme dans le cas de positionnement, augmenter la valeur de ce gain entraine l'augmentation de la commande, jusqu'à des seuils non admissibles.

Seul le positionnement reste possible dans ce cas. Cependant, il n'est pas intéressant d'effectuer une tâche positionnement avec des commandes admissibles, en 8 secondes. Nous devons donc trouver à présent des lois de commande visuelles ne saturant pas les commandes articulaires, et convergeant rapidement. C'est ce que nous allons tenter d'élaborer dans le prochain paragraphe.

3.5. Elaboration d'autres méthodes d'asservissement visuel 2D :

Nous tentons à présent d'élaborer d'autres stratégies de commande pour la boucle de vision, mais toujours dans le cadre de l'asservissement visuel 2D : superposer l'image désirée à l'image actuelle.

Dans le paragraphe précédent, nous nous sommes contentés d'imposer une dynamique exponentiellement décroissante à l'erreur, à travers la constante scalaire *g*. Nous testerons à présent l'étude du cas où *g* est une *matrice* constante. Puis, toujours en agissant sur *g*, nous le considérerons comme un scalaire variable *g(t)*. En tentant d'appliquer les outils de l'automatique avancée, nous tenterons de linéariser le comportement du système en y

appliquant une commande visuelle linéarisante. Nous tenterons enfin d'élaborer une commande visuelle par mode de glissement, puis par logique floue.

3.5.1. Cas où *g* est une matrice :

En considérant *g* comme une matrice, il est possible de donnée à chaque erreur une dynamique propre. Il est donc possible de faire converger plus rapidement une variable dont l'erreur initiale est forte, et en même temps, faire converger plus lentement une autre variable dont l'erreur initiale est plus faible. Cette matrice est élaborée en tenant compte des erreurs qui influencent chaque articulation. Nous testerons dans ce paragraphe des matrices élaborées par tâtonnement ; nous verrons par la suite comment il est possible de trouver cette matrice plus intelligemment. Remarquons aussi que si *g* n'est pas une matrice diagonale, il y a couplage des erreurs, comme il est prouvé en (3.42) :

$$\dot{e} = \begin{pmatrix} g_{11} & \cdots & g_{16} \\ \vdots & \ddots & \vdots \\ g_{61} & \cdots & g_{66} \end{pmatrix} e = \begin{pmatrix} g_{11}e_1 + & \cdots & + g_{16}e_6 \\ & \vdots & \\ g_{61}e_1 + & \cdots & + g_{66}e_6 \end{pmatrix} \tag{3.42}$$

Cependant, nous ne considérerons que les matrices diagonales, notées comme suit :

$$diag(g_1, \cdots, g_6) = \begin{pmatrix} g_1 & & 0 \\ & \ddots & \\ 0 & & g_6 \end{pmatrix} \tag{3.43}$$

Commençons par tester le cas d'une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°. Notons qu'il n'y a rotation que selon l'axe $\vec{z}$, ce qui permet de mettre à zéro g_4 et g_5 du fait qu'ils représentent la rotation suivant $\vec{x}$ *et* $\vec{y}$ respectivement. D'autre part, selon l'étude précédemment faite pour *g* constant, le couple U6, sollicité pour la rotation en $\vec{z}$ est assez faible, ce qui permet de choisir g_6 asse forte. Quand à g_1, $g2$ et g_3, ils sont choisis par tâtonnement, en fonction de la rapidité de convergence des couples articulaires. Nous choisissons dans ce cas la matrice *g=diag(1 1 0.8 0 0 3)*. Ce qui permet d'avoir les tracés suivants :

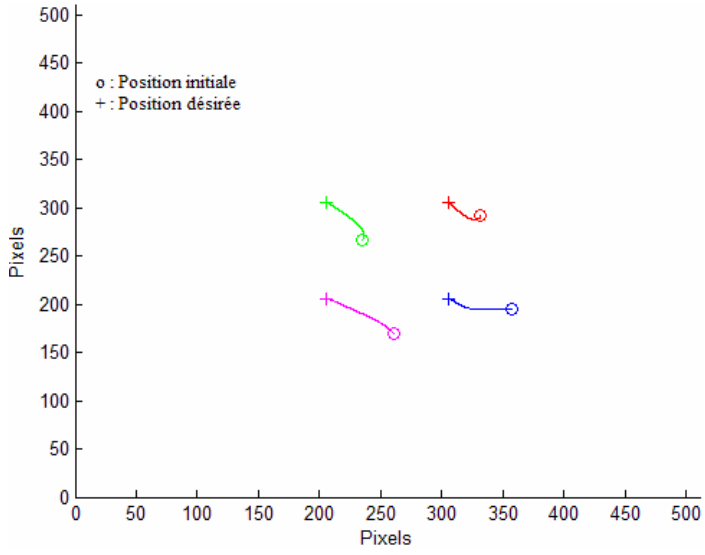
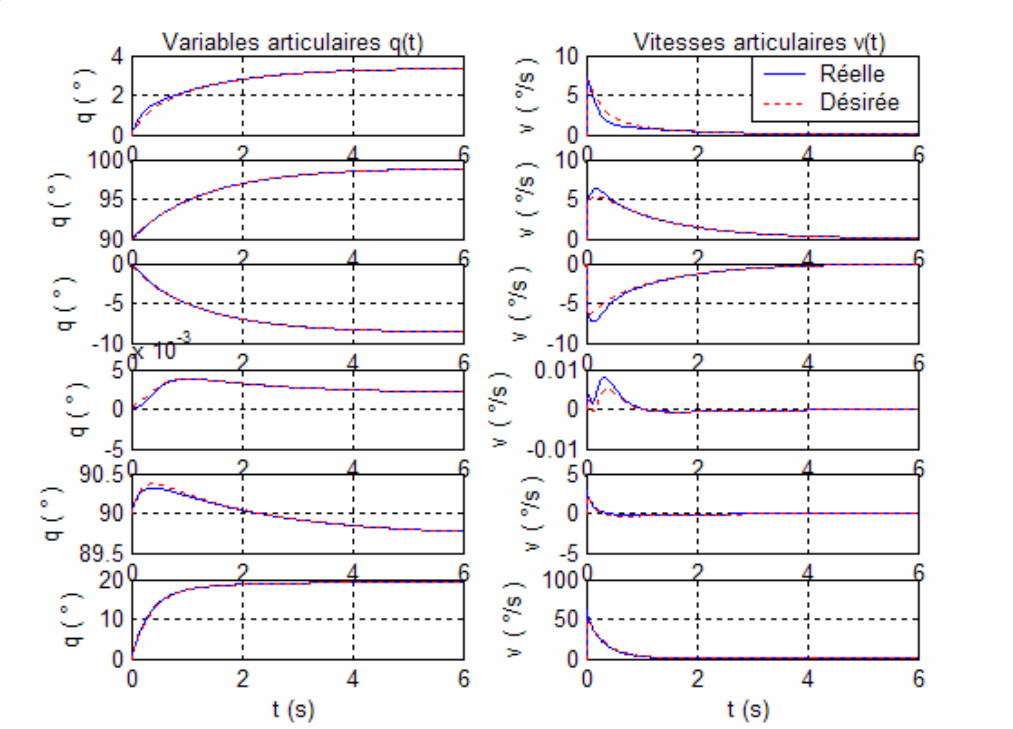


Figure 3.14.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=diag(1 1 0.8 0 0 3).*



3.14.b. *Variables et vitesses articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=diag(1 1 0.8 0 0 3).*

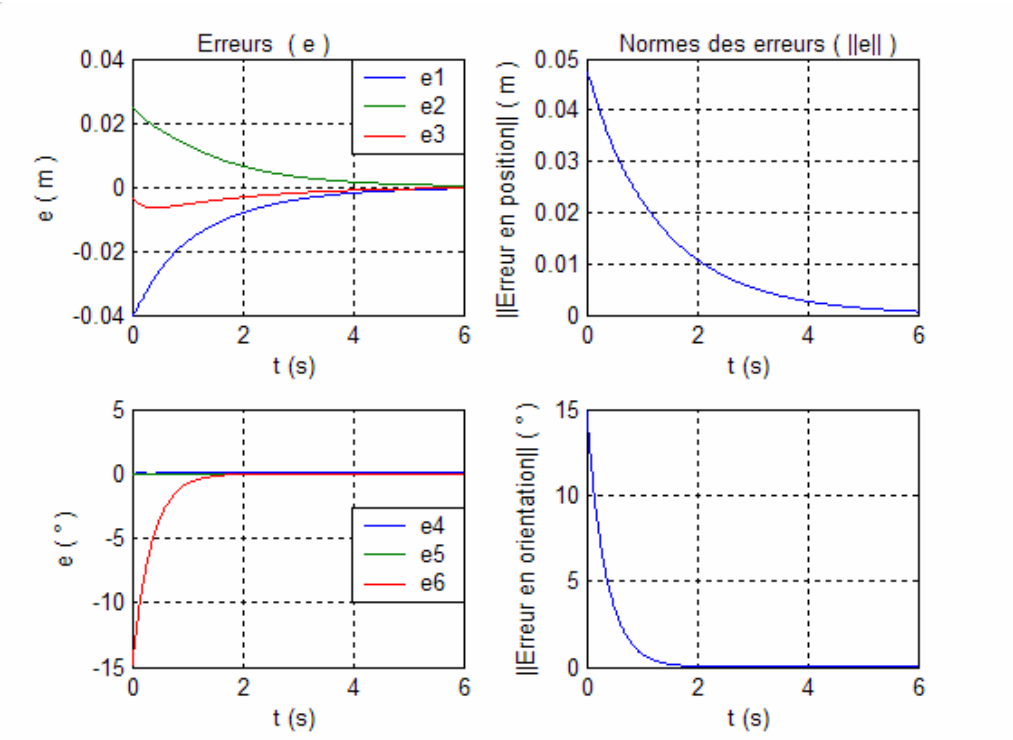


Figure 3.14.c. *Erreurs pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=diag(1 1 0.8 0 0 3).*

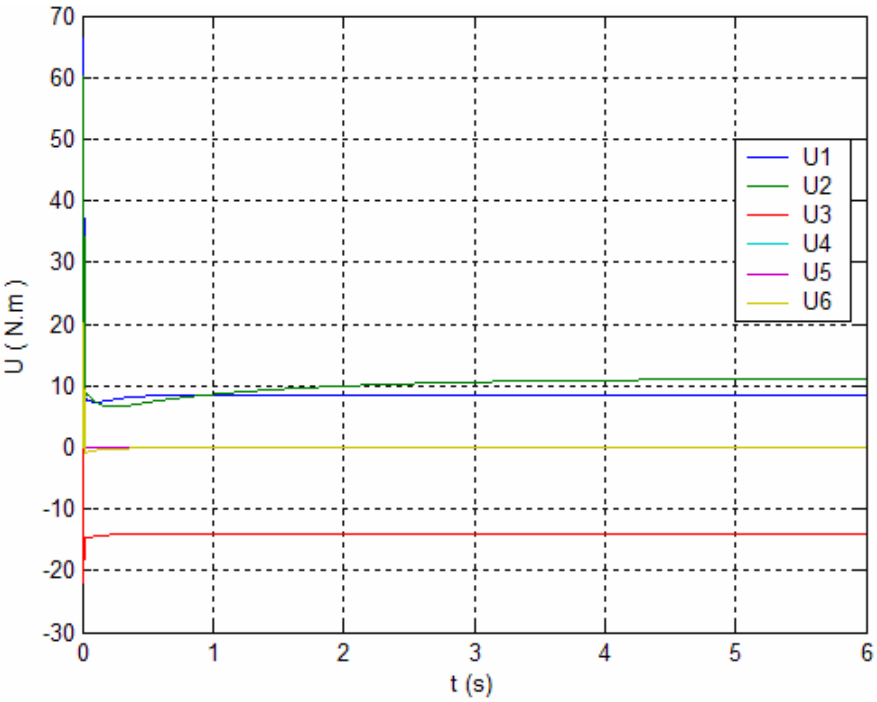


Figure 3.14.d. *Couples articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g=diag(1 1 0.8 0 0 3).*

Remarquons que le positionnement s'est correctement effectué, en un temps moindre que dans le cas précédent, et ceci en jouant sur les paramètres du gain *g* considéré comme une matrice. La trajectoire des primitives visuelle a changé du fait que nous avons accéléré la convergence en rotation plutôt que celle en translation, alors que si *g* était constante, cette accélération aurait été la même pour tous les mouvements.

Etudions à présent le cas de la poursuite de cible. Comme dans le cas de poursuite précédent, la trajectoire ainsi que la vitesse de l'objet sont inconnus. On considère la matrice *g=diag(0.3 0.3 0.2 0 0 1.5)*.

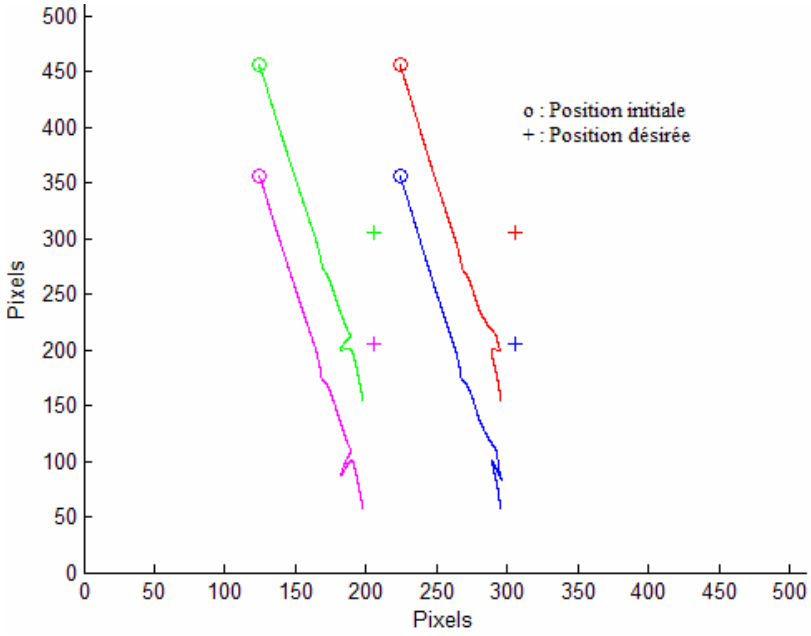


Figure 3.15.a. *Trajectoire dans l'image des primitives visuelle pour tâche de poursuite, avec g=diag(0.3 0.3 0.2 0 0 1.5).*

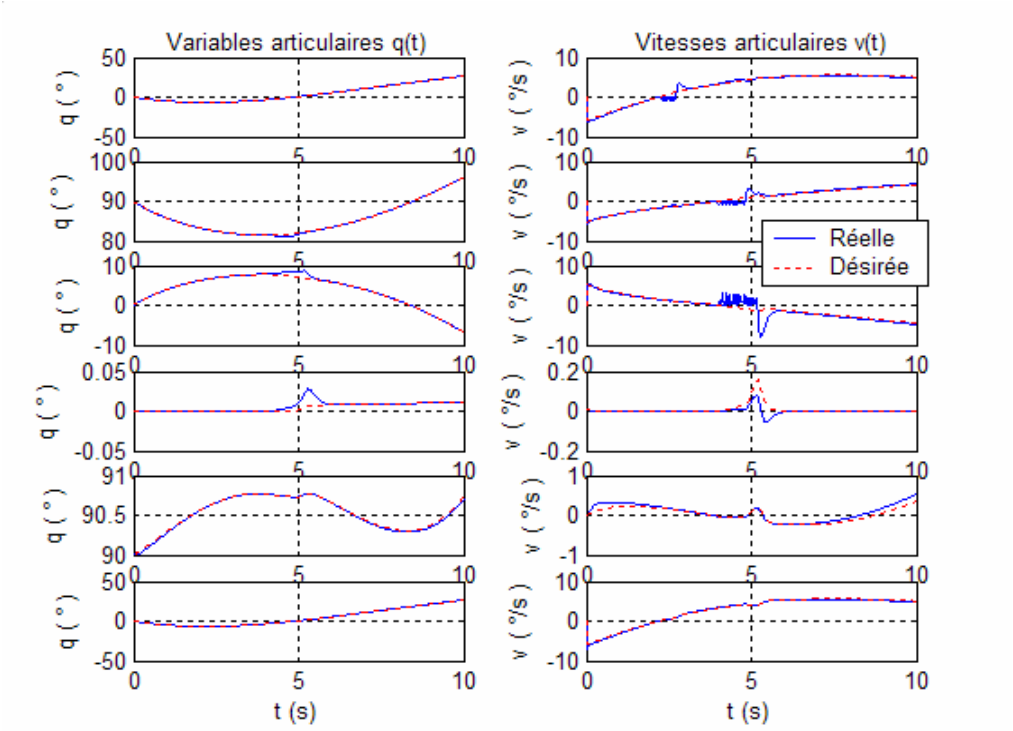


Figure 3.15.b. *Variables et vitesses articulaires pour tâche de poursuite, avec g=diag(0.3 0.3 0.2 0 0 1.5).*

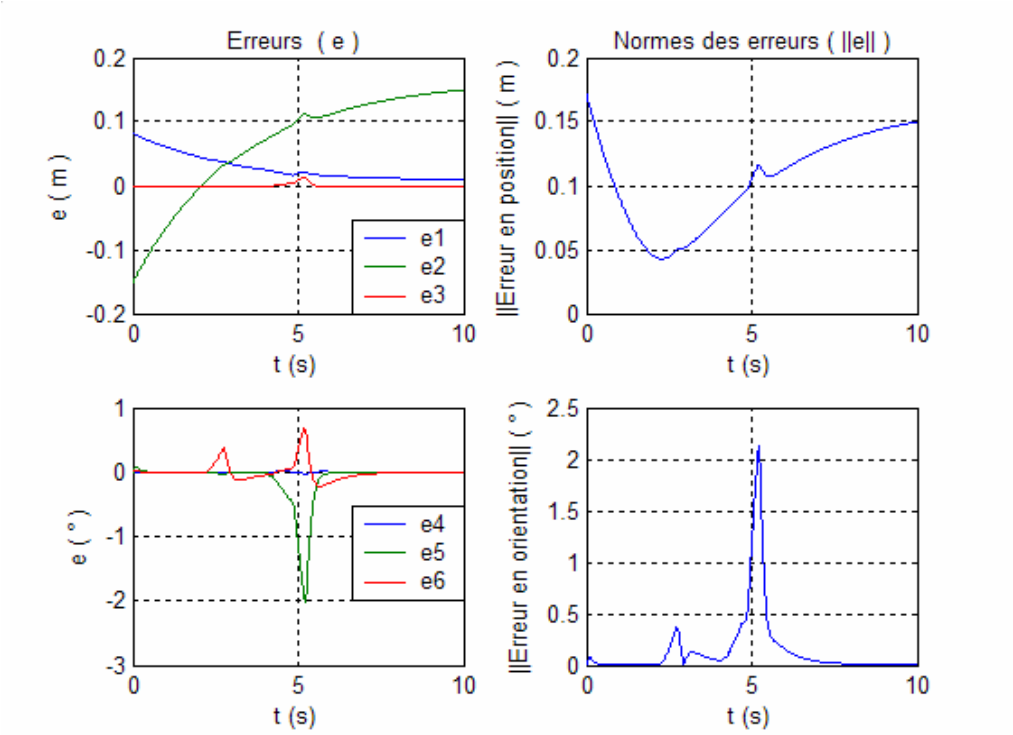


Figure 3.15.c. *Erreurs pour tâche de poursuite, avec g=diag(0.3 0.3 0.2 0 0 1.5).*

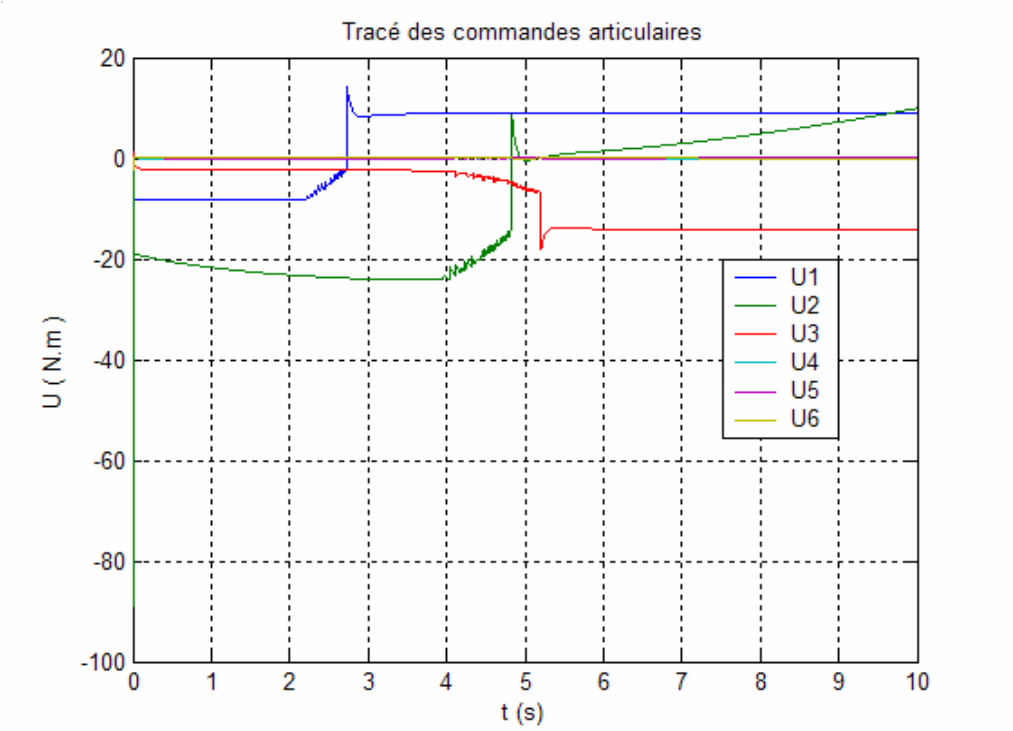


Figure 3.15.d. *Couples pour tâche de poursuite, avec g=diag(0.3 0.3 0.2 0 0 1.5).*

Comme dans le cas où g était scalaire, la tâche de poursuite n'est pas totalement réalisé : subsiste toujours une erreur de traînage. Seul un gain assez fort peut réduire cette erreur, mais dans ce cas, la commande risque fort d'être admissible.

3.5.2. Cas où *g* est variable :

Nous sommes déjà parvenus à la conclusion que plus g est élevée, plus la convergence est rapide et vice versa. Or, une rapide convergence sollicite une forte commande articulaire, notamment à l'instant initial, selon l'équation (3.41). Il serait donc intéressant d'employer un gain g variable dans le temps, de sorte que ce dernier soit asse faible initialement pour ne pas saturer la commande, puis assez fort pour converger rapidement. Nous considérons dans ce cas un gain g nul à l'instant initial, atteignant un gain max g_{max} à l'infini, et positif à chaque instant :

$$\begin{cases} g(0) = 0 \\ g(t) > t,\ t > 0 \end{cases} \tag{3.44}$$

Une fonction candidate vérifiant les conditions précédentes est :

$$g(t) = g_{\max}(1 - e^{-t/a}), \quad g, a > 0 \tag{3.45}$$

g_{max} est le gain maximal à atteindre, et a détermine la rapidité avec laquelle nous atteindrons la gain max. C'est la positivité de cette fonction qui assure la convergence de l'erreur.

Testons à présent cette méthode. Prenons comme premier cas une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°. Prenons aussi $g_{max}=1\ (s^{-1})$, et $a=1(s)$:

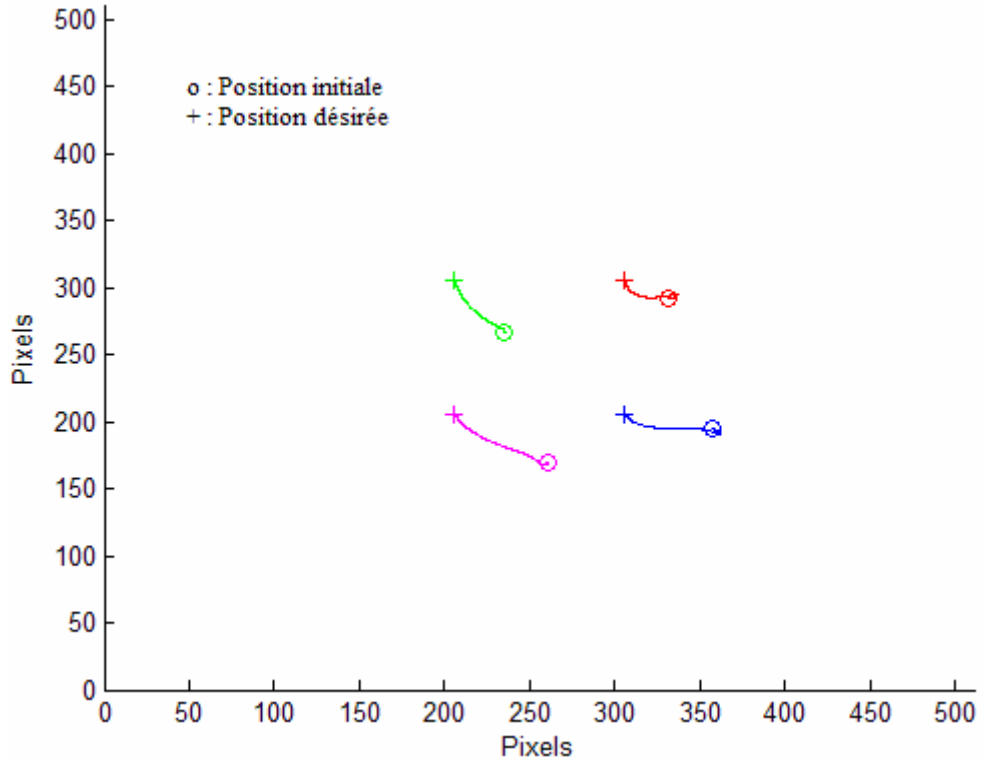


Figure 3.16.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g variable : $g_{max}=1\ (s^{-1})$,* et *$a=1(s)$.*

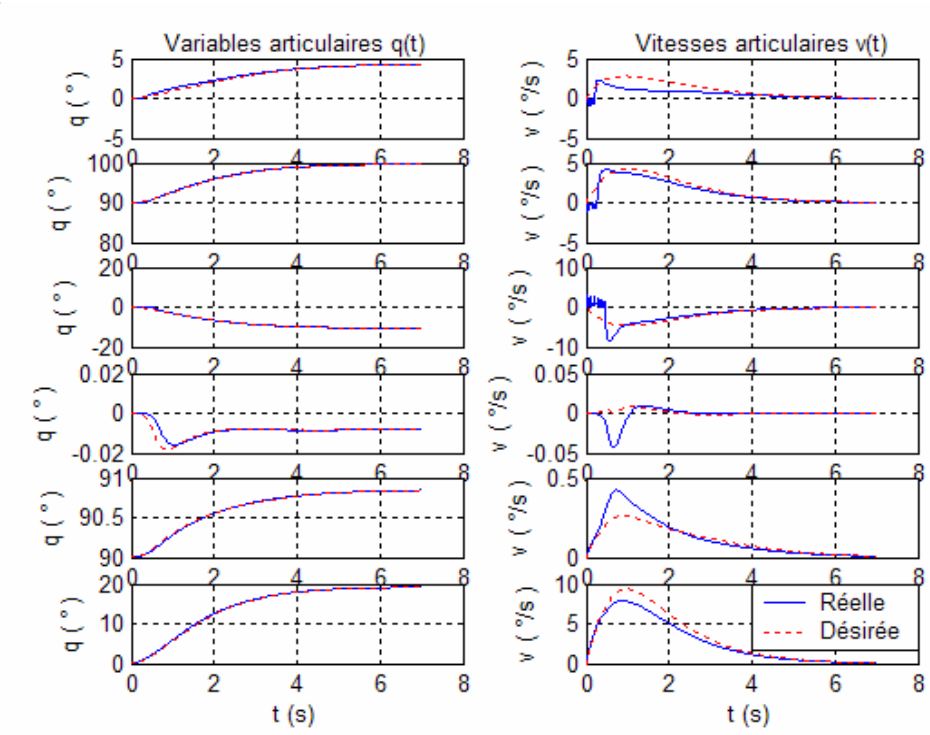


Figure 3.16.b. *Variables et vitesses articulaires visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g variable : $g_{max}=1\ (s^{-1})$,* et *$a=1(s)$.*

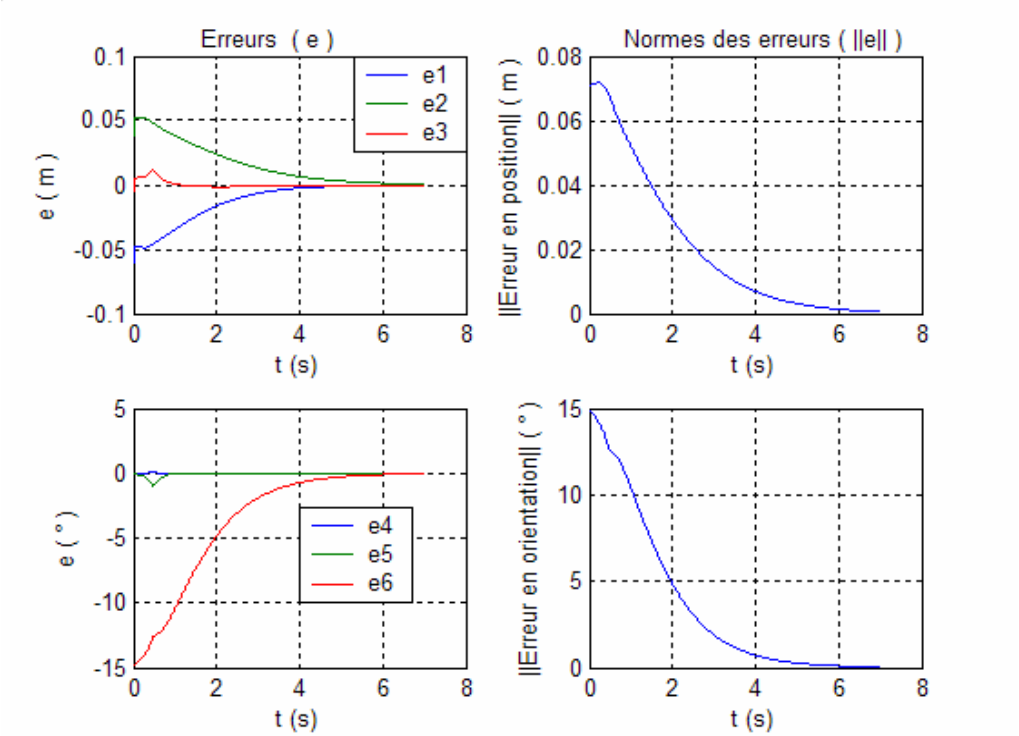
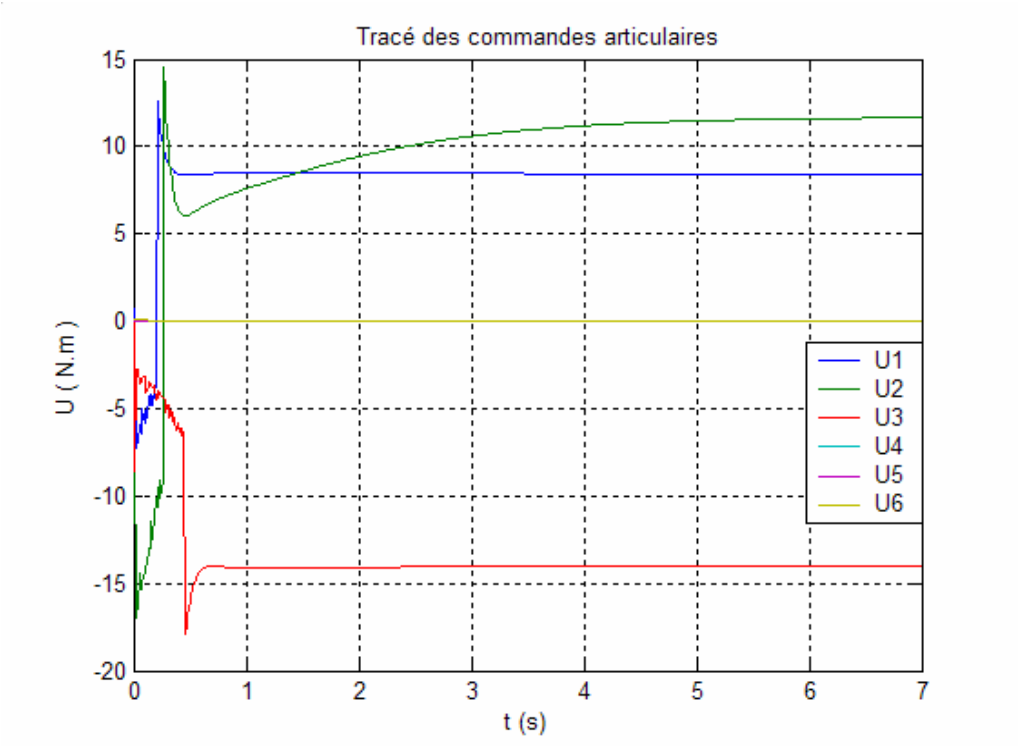


Figure 3.16.c. *visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g variable : g_{max}=1 (s^{-1}),* et *a=1(s).*



3.16.d. *Couples articulaires visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g variable : g_{max}=1 (s^{-1}),* et *a=1(s).*

Comme on peut le constater, le temps de convergence a nettement diminué (de l'ordre de 6s), il y a donc rapidité de convergence. D'autre part, la commande n'excède pas 20 N.m.

Nous avons donc résolu avec cette méthode le dilemme *énergie-temps de convergence*. Pour valider cette méthode, considérons une autre simulation. Pour un mouvement dans l'image de -50 pixels sur *u* et 100 pixels sur *v*, toujours avec $g=1\ (s^{-1})$ et $a=1\ (s)$:

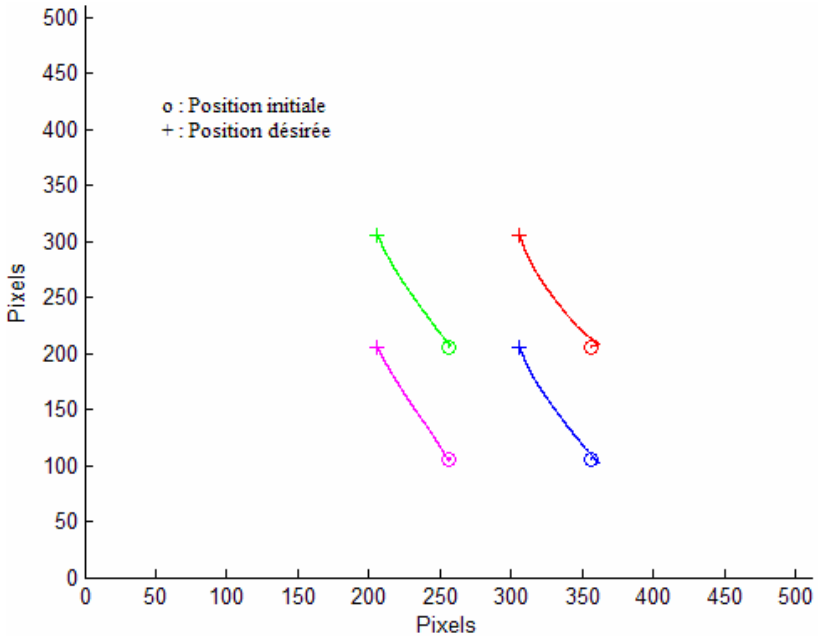


Figure 3.17.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -50 pixels en u et 100 pixels en v, avec g variable : $g_{max}=1(s^{-1})$ et a=1(s).*

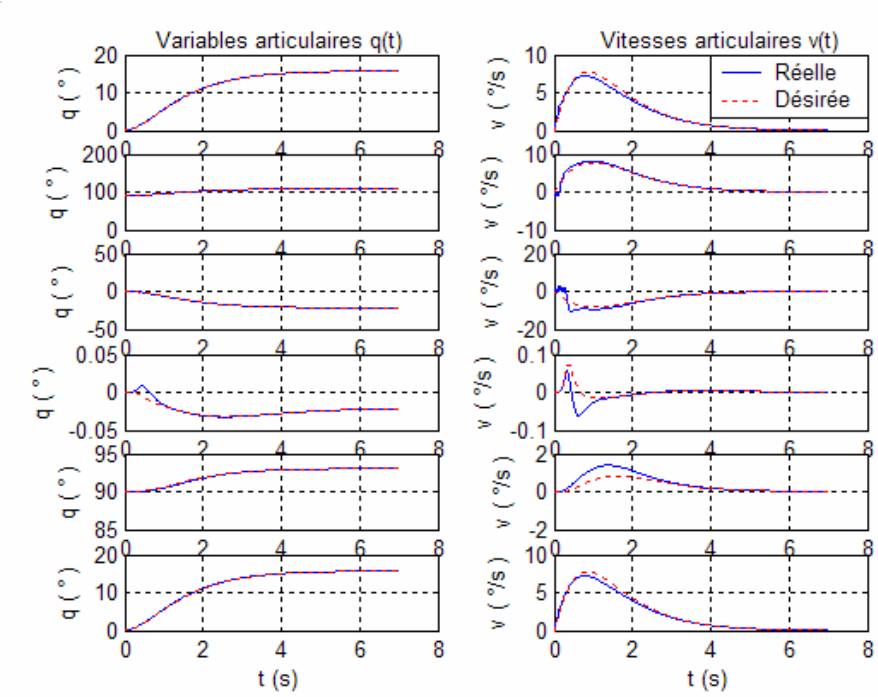


Figure 3.17.b. *Variables et vitesses articulaires pour une translation de -50 pixels en u et 100 pixels en v, avec g variable : $g_{max}=1\ (s^{-1})$ et a=1(s).*

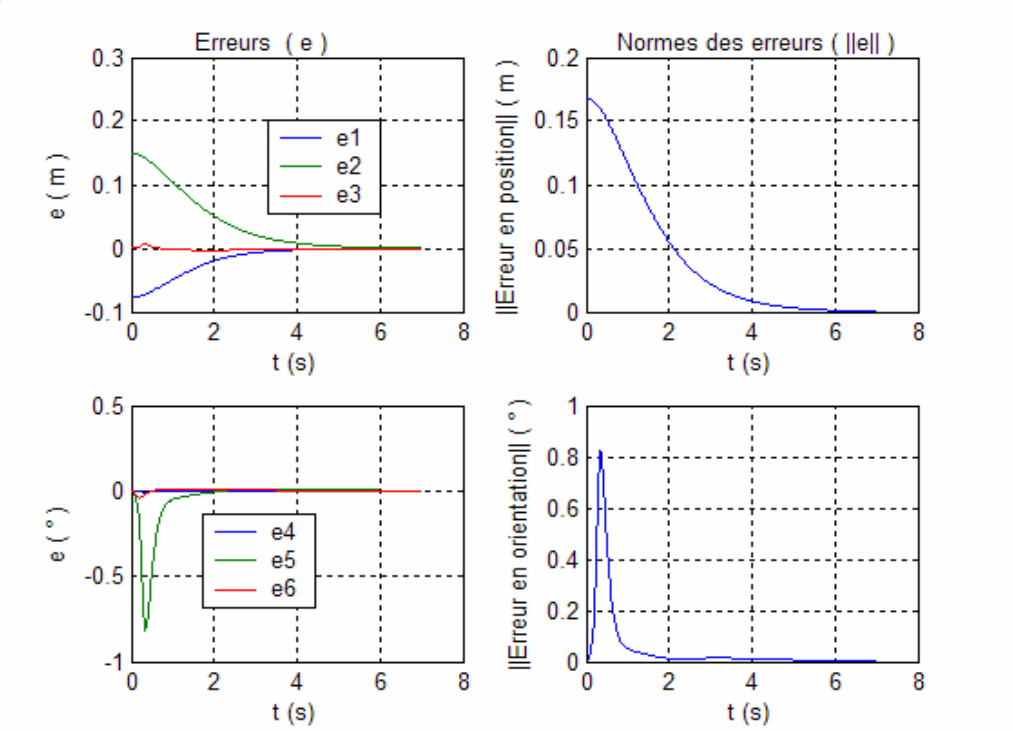
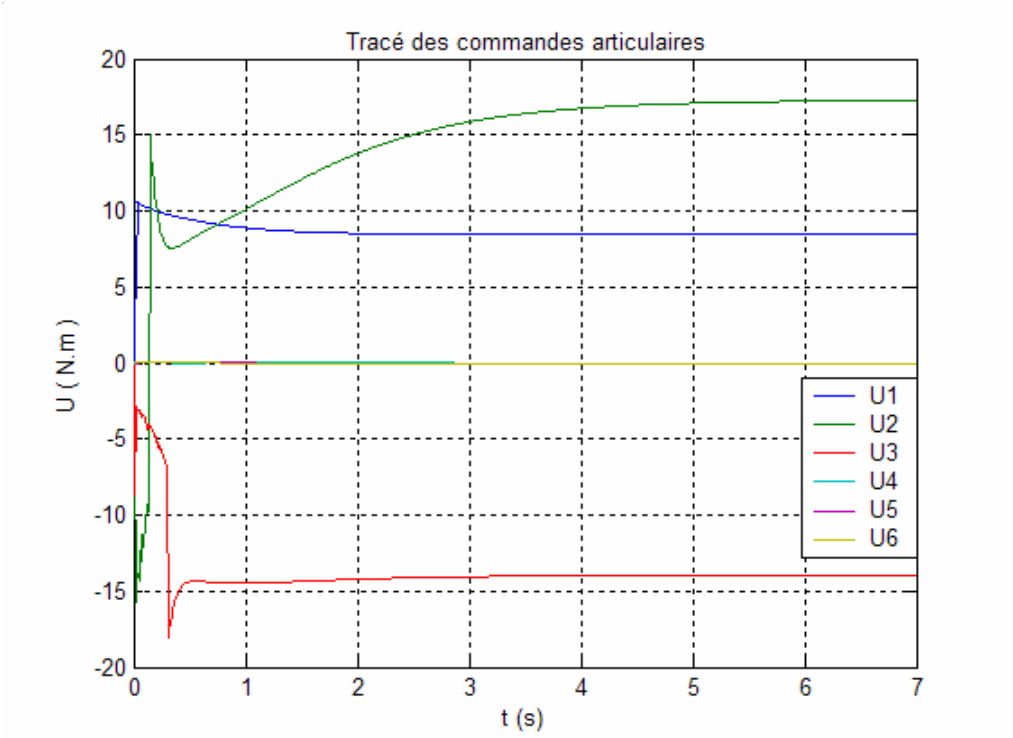


Figure 3.17.c. *Erreurs pour une translation de -50 pixels en u et 100 pixels en v, avec g variable : $g_{max}=1(s^{-1})$ et a=1(s).*



3.17.d. *Couples articulaires pour une translation de -50 pixels en u et 100 pixels en v, avec g variable : $g_{max}=1(s^{-1})$ et a=1(s).*

Comme dans le cas précédent, il y a convergence en 5 secondes avec une commande n'excédant pas 80 N.m. Remarquons cependant que la norme des erreurs en rotation étant nulle au départ (s'agissant d'un mouvement de translation pure) augmente légèrement avant

de tendre vers zéro. Ceci montre bien qu'en asservissement visuel 2D, la commande agit en fonction de l'image et non pas de la configuration du robot, car dans ce cas, il n'y aurait pas eu de mouvement de rotation du robot.

Tentons de simuler à présent une tâche de poursuite. L'objet est en mouvement de translation sur une bande convoyeuse. Sa vitesse et sa trajectoire sont inconnues pour le robot. Voici la simulation d'un asservissement visuel 2D réalisant la tâche de poursuite, avec *g* variable ($g_{max}=10(s^{-1})$ *et* $a=2/3(s)$) :

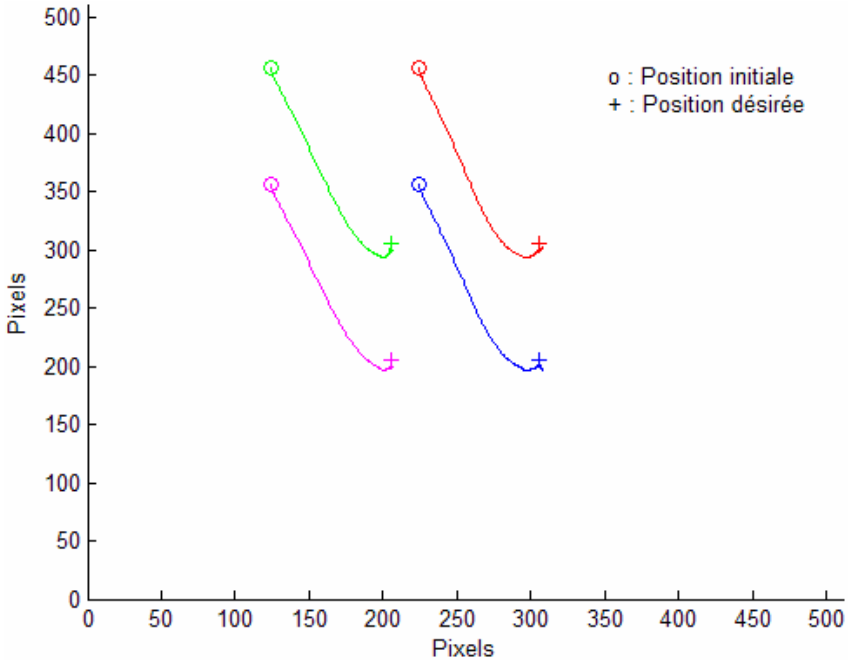


Figure 3.18.a. *Trajectoire dans l'image des primitives visuelle pour une tâche de poursuite, avec g variable : $g_{max}=10(s^{-1})$ et $a=2/3(s)$.*

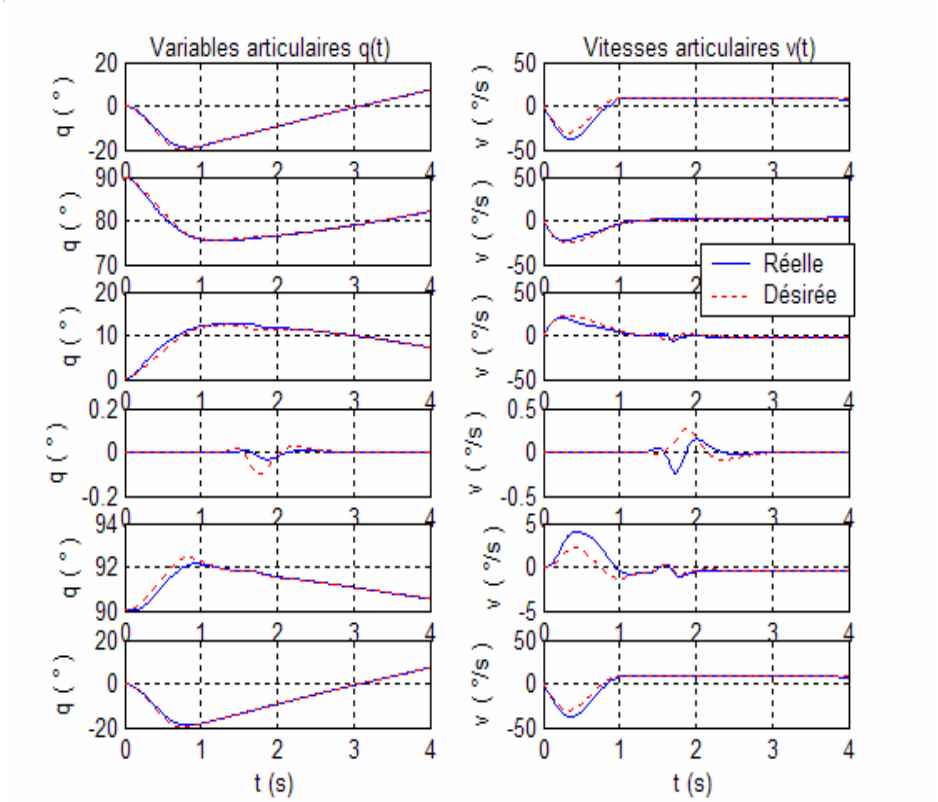


Figure 3.18.b. *Variables et vitesses articulaires pour une tâche de poursuite, avec g variable : $g_{max}=10(s^{-1})$ et $a=2/3(s)$.*

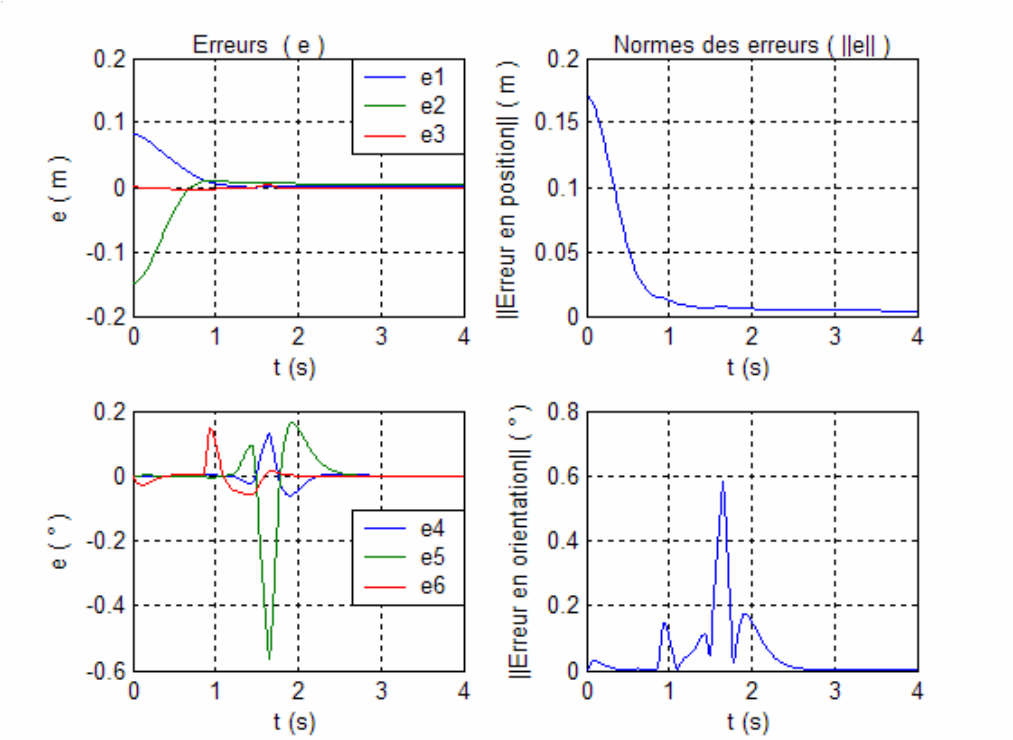
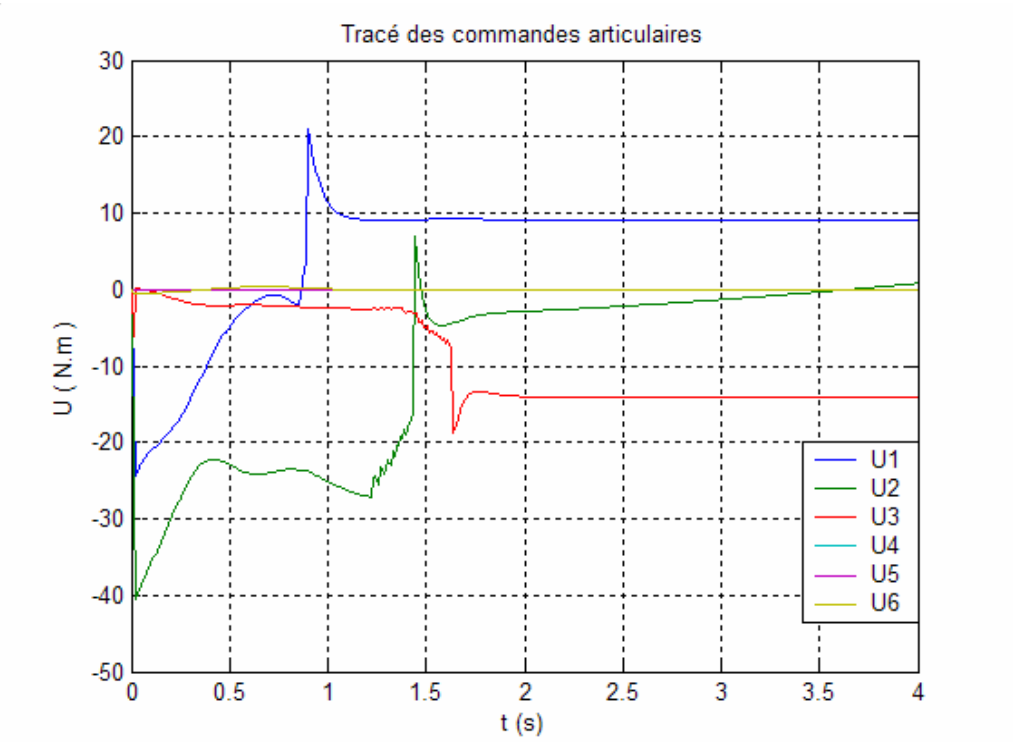


Figure 3.18.c. *Erreurs pour une tâche de poursuite, avec g variable : $g_{max}=10(s^{-1})$ et a=2/3(s).*



3.18.d. *Couples articulaires pour une tâche de poursuite, avec g variable : $g_{max}=10(s^{-1})$ et a=2/3(s).*

Nous constatons donc que la tâche de poursuite est correctement effectuée en employant cette méthode. Notons qu'au départ le robot n'avait aucune information sur la position de l'objet, sa trajectoire et sa vitesse. D'autre part, remarquons que le fait d'effectuer une tâche de poursuite nous a contraint d'employer un gain maximum plus élevé. Les valeurs de g_{max} et a ont été trouvées par tâtonnement, de manière à effectuer correctement la tâche de poursuite, en tentant d'éliminer l'erreur de traînage, et de réduire la valeur du couple articulaire initiale. Remarquons aussi que ce couple est plus élevé que dans le cas de positionnement du robot.

3.5.3. Commande visuelle linéarisante :

L'élaboration de la loi de commande visuelle (3.38) résulte du mélange de trois équations qui sont : la relation fondamentale de l'asservissement visuel traduisant les variations de l'image en fonction des mouvements du robot, la définition de la fonction de tâche et le fait d'imposer une décroissance exponentielle à la fonction de tâche. Tentons à partir de maintenant d'élaborer des lois de commandes visuelles en n'employant que la relation traduisant les variations de l'image en fonction des mouvements du robot, à savoir :

$$\dot{s} = L_s \cdot T \tag{3.46}$$

L'objectif étant de faire tendre l'image actuelle (représentée par s) vers l'image désirée (représentée par s^*). Définissons pour cela la sortie y du système tel que :

$$y = X = s \tag{3.47}$$

Avec X la variable d'état. De ce fait, la dérivée de cette variable par rapport au temps donne :

$$\dot{X} = \dot{s} = L_s T = \mathcal{V} \tag{3.48}$$

Ce qui permet d'aboutir au système *linéaire* suivant :

$$\begin{cases} \dot{X} = \mathcal{V} \\ y = X \end{cases} \tag{3.49}$$

Par superposition à la représentation classique des systèmes linéaires $\dot{X} = AX + B\mathcal{V}$ et $y = CX + D\mathcal{V}$, nous avons $A = D = \mathbb{O}_{2n}$ et $B = C = \mathbb{I}_{2n}$. Ainsi, le système non-linéaire (3.46) s'écrit sous la forme linéaire (3.49) en appliquant la commande linéarisante :

$$T = L_s^+ \mathcal{V} \tag{3.50}$$

$\mathcal{V}$ étant la commande du système linéarisé. Il faut toutefois souligner qu'avec cette méthode, ce n'est pas une linéarisation du système autour d'un point de fonctionnement qui est faite, mais l'application d'une commande dite *linéarisante* qui linéarise le comportement du système quelque soit l'espace de travail. $\mathcal{V}$ peut être calculée par retour d'état sur le système linéaire, et ceci à travers une matrice K tel que :

$$\mathcal{V} = -K\,X + N\,r \tag{3.51}$$

Avec *K* le gain du retour d'état, *N* la matrice de pré compensation et *r* la référence à suivre, qui dans notre cas n'est que l'image désirée : $r = s^*$. D'autre part, la matrice de pré compensation ayant pour but d'annuler l'erreur statique, est donnée par :

$$N = D + (C\,(B\,K - A)^{-1}\,B)^{-1} = K \quad ,dans\ notre\ cas \tag{3.52}$$

Ce qui permet de réécrire l'équation (3.51) comme suit :

$$\mathcal{V} = -K\,(s - s^*) \tag{3.53}$$

Et de ce fait, la loi de commande linéarisante est donnée par :

$$T = -L_s^+ K\,(s - s^*) \tag{3.54}$$

Avec *K* le gain du retour d'état linéaire. Il serait donc intéressant d'avoir la dynamique du système commandé comme tel. Pour cela, injectons l'équation (3.53) dans (3.48), ceci permet d'avoir la dynamique des informations visuelles :

$$\dot{s} = -K\,(s - s^*) \tag{3.55}$$

En multipliant des deux cotés par la matrice *C* définie pour la fonction de tâche (3.25), nous aboutissons à l'équation modifiée :

$$\dot{e} = -(C\,K\,C^+)e = -g\,e \tag{3.56}$$

Nous aboutissons donc à la relation entre cette méthode et la méthode classique : En fait, appliquer la commande linéarisante revient à appliquer un gain *g* matriciel. Or, la grande différence avec la méthode précédente décrite dans le paragraphe 3.5.1. est que cette fois-çi, *g* n'est plus recherchée par tâtonnement, mais calculée directement par retour d'état, à travers la matrice *K*. Il est donc possible à travers cette méthode de trouver le gain *g* permettant d'effectuer au mieux une tâche précise.

Il convient maintenant de calculer le gain du retour d'état *K*. Ce dernier peut être calculé par simple placement de pôle. Mais nous pouvons améliorer cette méthode en calculant *K* par placement de pôle optimal, et ceci en résolvant le critère linéaire quadratique associé au système linéaire :

$$J = \min_{\mathcal{V}} \int_0^{+\infty} \{X^T Q\,X + \mathcal{V}^T R\,\mathcal{V}\}\,dt \tag{3.57}$$

Avec $Q \geq 0$, $R > 0$ deux matrices constantes.

La résolution de ce critère permet d'aboutir à la commande suivante :

$$\mathcal{V} = -R^{-1}\,B^T\,P\,X + N\,r = -(R^{-1}P)\,X + N\,r \tag{3.58}$$

Avec $K = R^{-1}\,P$ et *P* la résolution de l'équation de *RICCATI* :

$$A^T P + P\,A + Q - P\,B\,R^{-1}B^T P = \mathbb{O}_{2n}$$

➔ $Q - P\,R^{-1}P = \mathbb{O}_{2n}$, *dans notre cas* (3.59)

Avec *P* une matrice définie positive. Il ne reste donc plus qu'à choisir les matrices *R* et *Q*. Elles sont élaborées par tâtonnement, mais en suivant une piste bien précise ; en effet, *Q* détermine l'influence de *X* dans le critère, et donc le poids de la variable visuelle *s*, quand à *R*, elle détermine le poids de $\mathcal{V}$ dans le critère, et donc la variation de la variable visuelle s, ie $\dot{s}$.

Ainsi, choisir les constantes *Q* et *R* revient à donner du poids à la variable visuelle ou à sa variation. Remarquons donc qu'on est contraint de faire un compromis entre la convergence et la vitesse de convergence. Quand au choix des valeurs de *Q* (resp. *R*), ils se font sur le poids que l'on désire attribuer à la variable visuelle (resp. sa vitesse), par rapport au reste.

Une fois ces matrices fixées, il nous faut résoudre l'équation (3.59), et injecter le résultat dans (3.58) pour élaborer la commande linéarisante. Tentons de valider donc cette méthode par diverses simulations. Prenons un premier cas où nous avons à effectuer une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°. Dans ce cas, nous prendrons les matrices *R* et *Q* tel que : $Q=0.5\ I_8$, $R=2\ I_8$. Ceci permet après la résolution de (3.59) d'avoir comme gain du retour d'état $K=0.5\ I_8$.

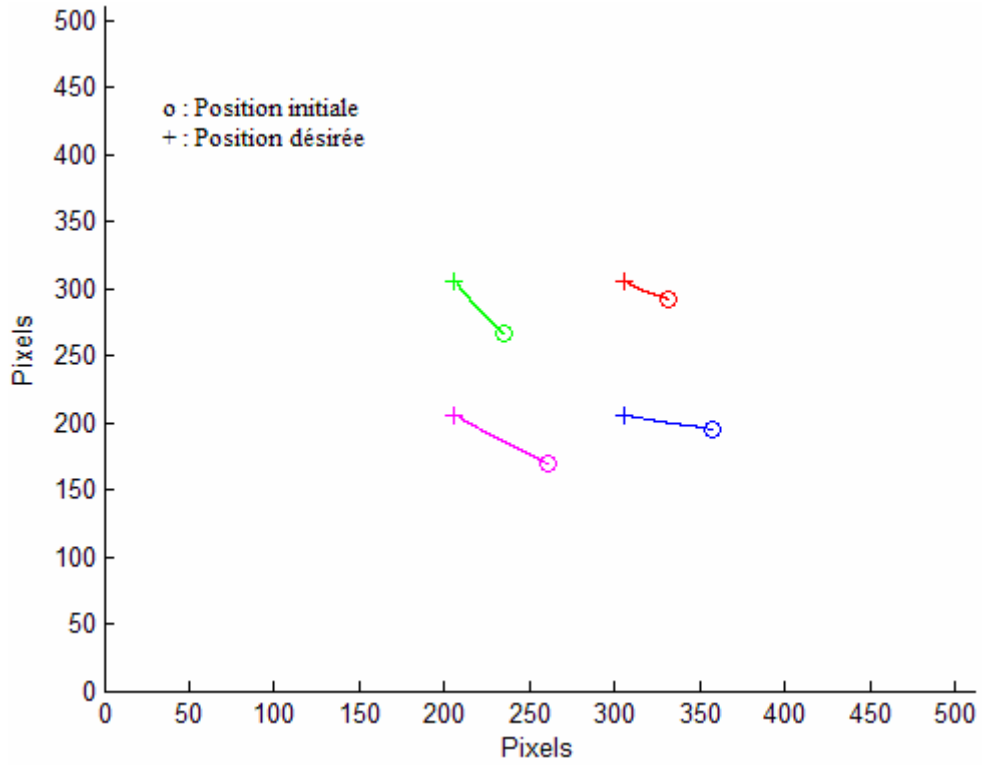
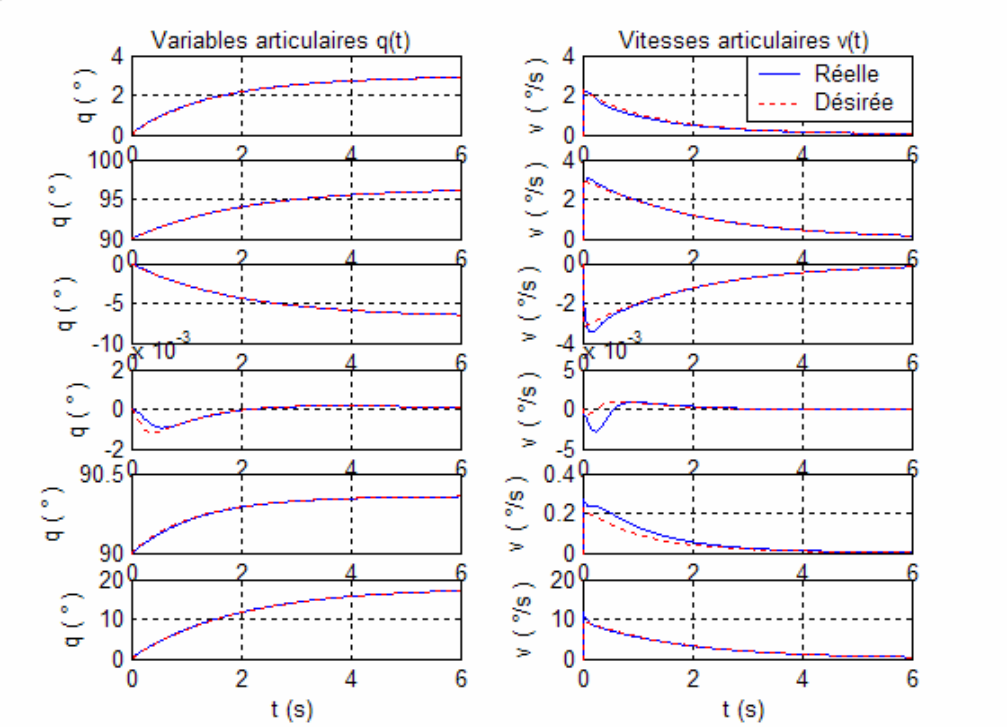


Figure 3.19.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec une commande linéarisante : $Q=0.5I_8$, $R=2\ I_8$.*



3.19.b. *Variables et vitesses pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec une commande linéarisante : Q=0.5 I_8, R=2 I_8.*

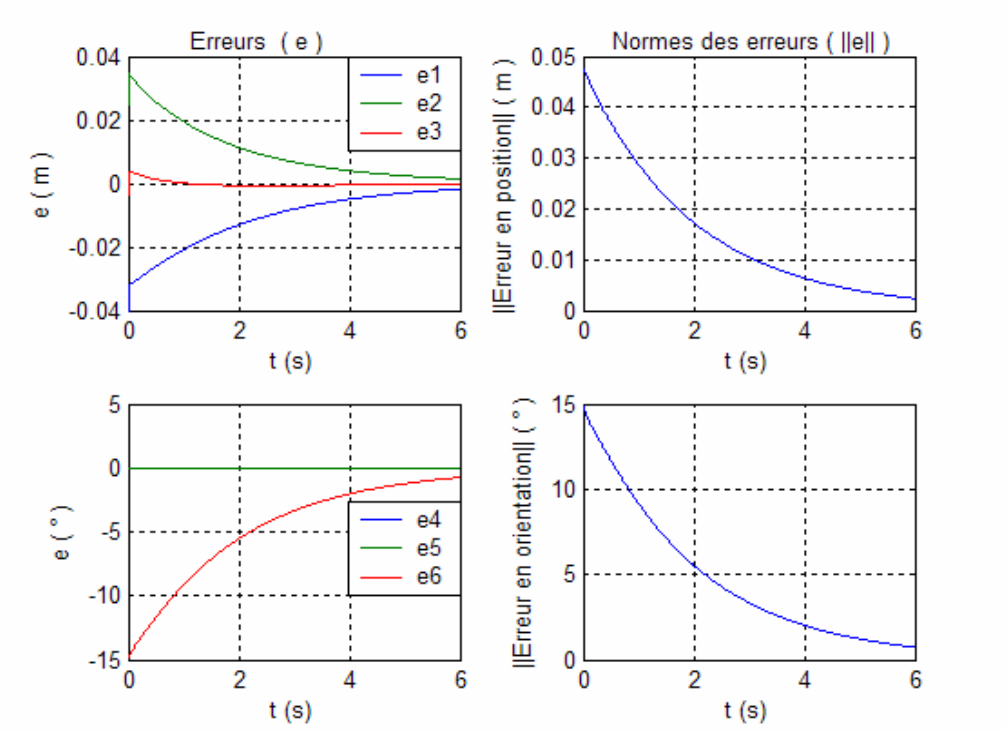


Figure 3.19.c. *Erreurs pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec une commande linéarisante : Q=0.5 I_8, R=2 I_8.*

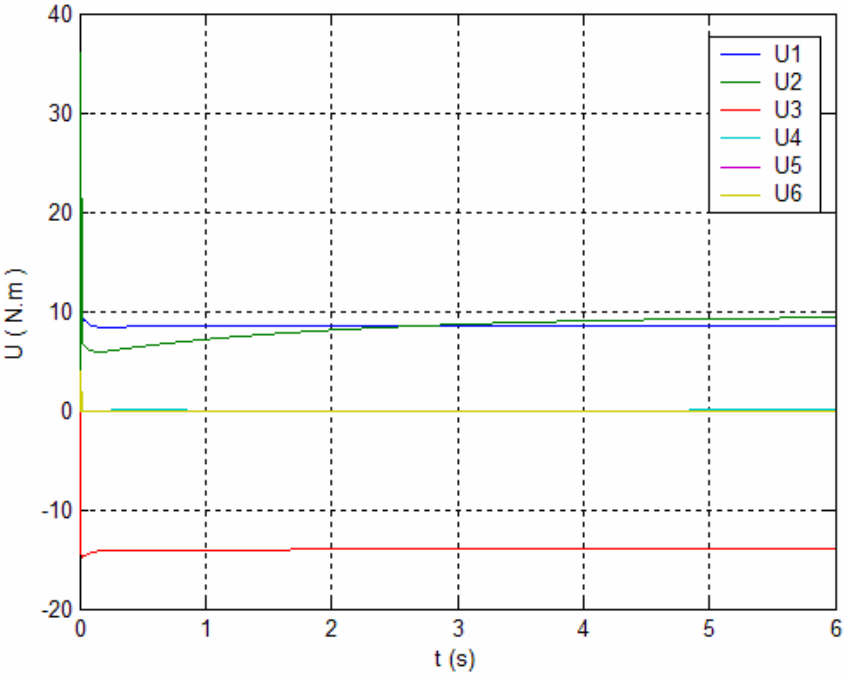


Figure 3.19.d. *Couples articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec une commande linéarisante : $Q=0.5\ I_8$, $R=2\ I_8$.*

Nous remarquons donc que cette méthode permet de réaliser correctement la tâche de positionnement, et avec des commandes tout à fait admissibles. Nous nous sommes donc ramenés à la recherche du gain g matriciel, mais de manière précise. Cette même tâche de positionnement a été réalisée dans le cas où g était matricielle, mais élaborée par tâtonnement : nous avions effectivement convergence, mais avec des commandes plus élevée à l'instant initial. Testons à présent cette méthode dans le cas d'une translation plus grande, de l'ordre de -50 pixels suivant u et 100 pixels suivant v. Nous prendrons pour cela $Q=0.7\ I_8$, $R=2\ I_8$. Ceci permet après la résolution de (3.59) d'avoir comme gain du retour d'état $K \approx 0.6 I_8$, et les tracés suivants :

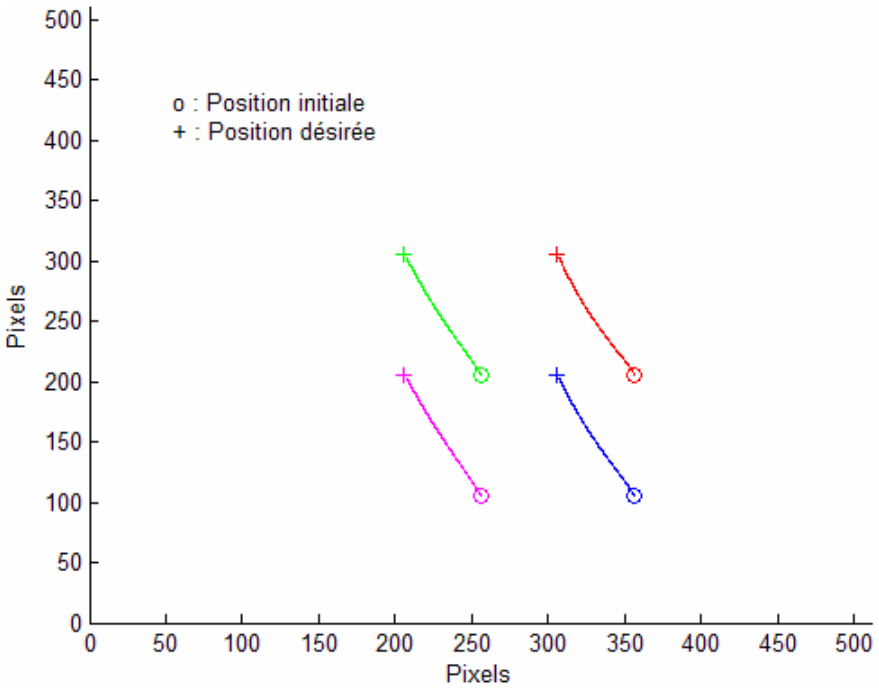
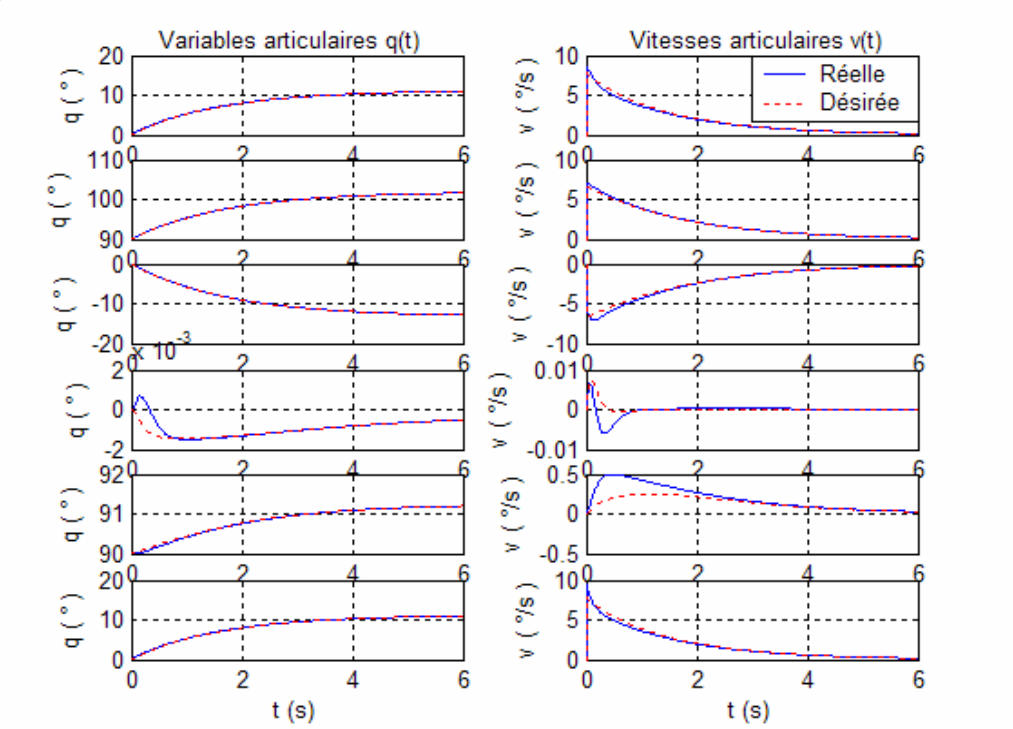


Figure 3.20.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -50 pixels en u et 100 pixels en v, avec une commande linéarisante : $Q=0.7\ I_8$, $R=2\ I_8$.*



3.20.b. *Variables et vitesses articulaires pour une translation de -50 pixels en u et 100 pixels en v, avec une commande linéarisante : $Q=0.7\ I_8$, $R=2\ I_8$.*

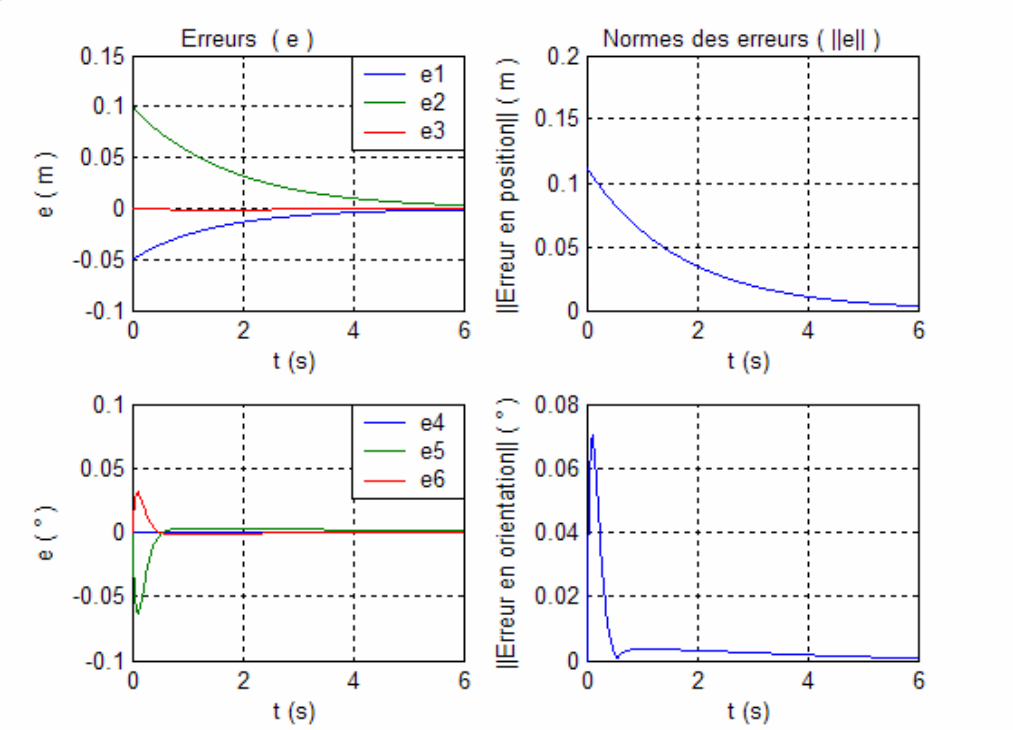


Figure 3.20.c. *Erreurs pour une translation de -50 pixels en u et 100 pixels en v, avec une commande linéarisante : $Q=0.7\ I_8$, $R=2\ I_8$.*

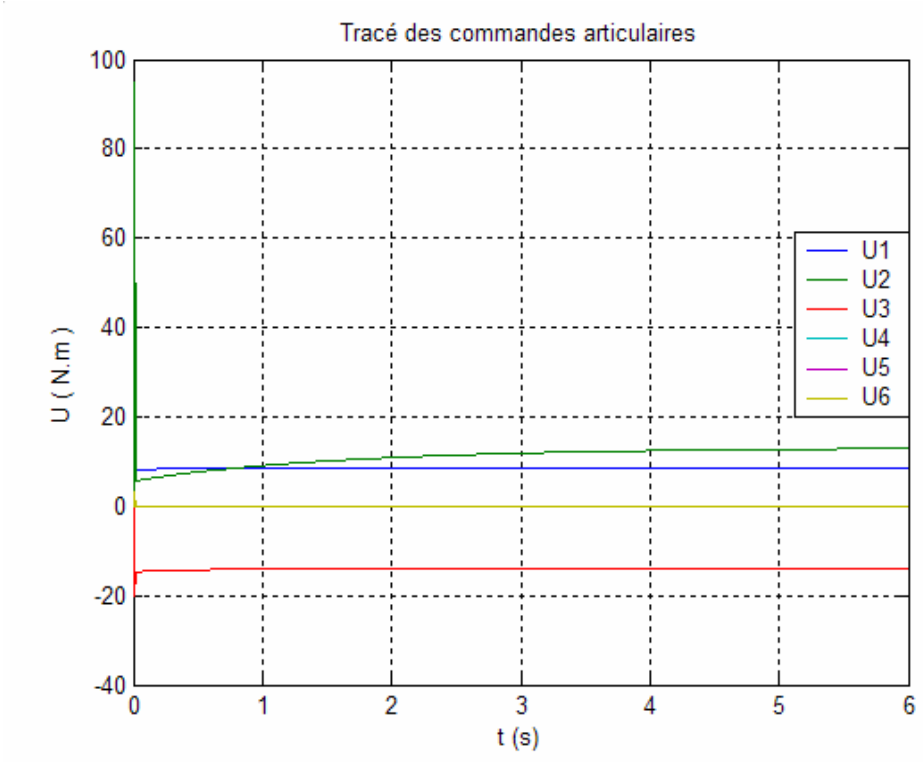


Figure 3.20.d. *Couples articulaires pour une translation de -50 pixels en u et 100 pixels en v, avec une commande linéarisante : $Q=0.7\ I_8$, $R=2\ I_8$.*

Là encore, la tâche de positionnement a bien été réalisée. Cette même tâche a été testée dans le cas où *g* est variable (Figures 3.17) ; nous constatons donc que le couple mis en œuvre dans cette méthode est plus fort que celui mis en œuvre pour *g* variable. Par contre, mis à part cet instant initial, la commande est moins sollicitée que pour *g* variable.

Testons à présent le cas d'une tâche de poursuite. Pour cela, considérons les matrices $Q= I_8$, $R=15\ I_8$. Ceci permet après la résolution de (3.59) d'avoir comme gain du retour d'état $K \approx 0.26\, I_8$, et les tracés suivants :

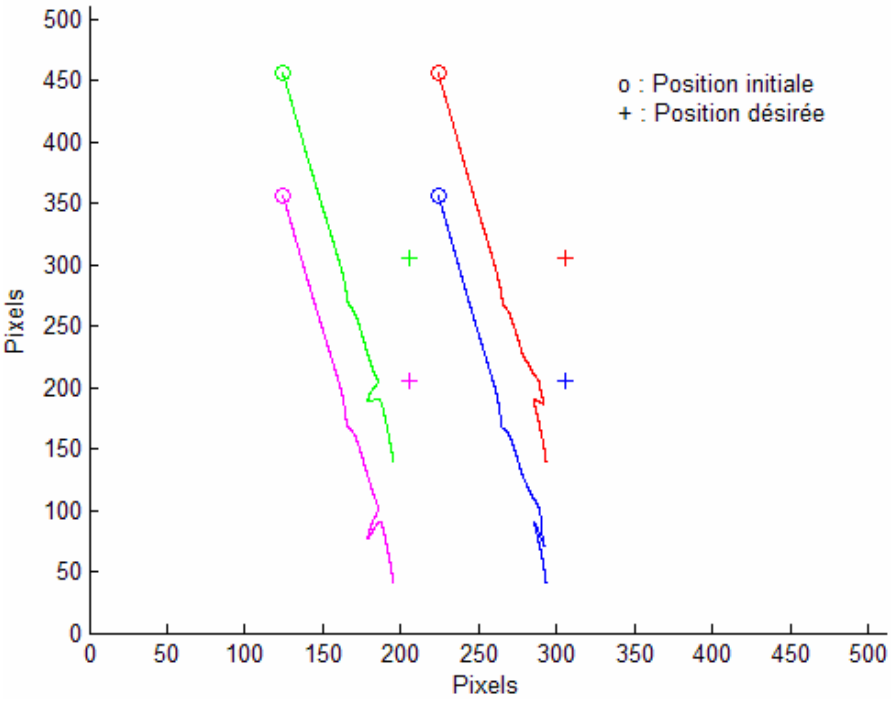


Figure 3.21.a. *Trajectoire dans l'image des primitives visuelle pour une tâche de poursuite, avec une commande linéarisante : $Q= I_8$, $R=15\ I_8$.*

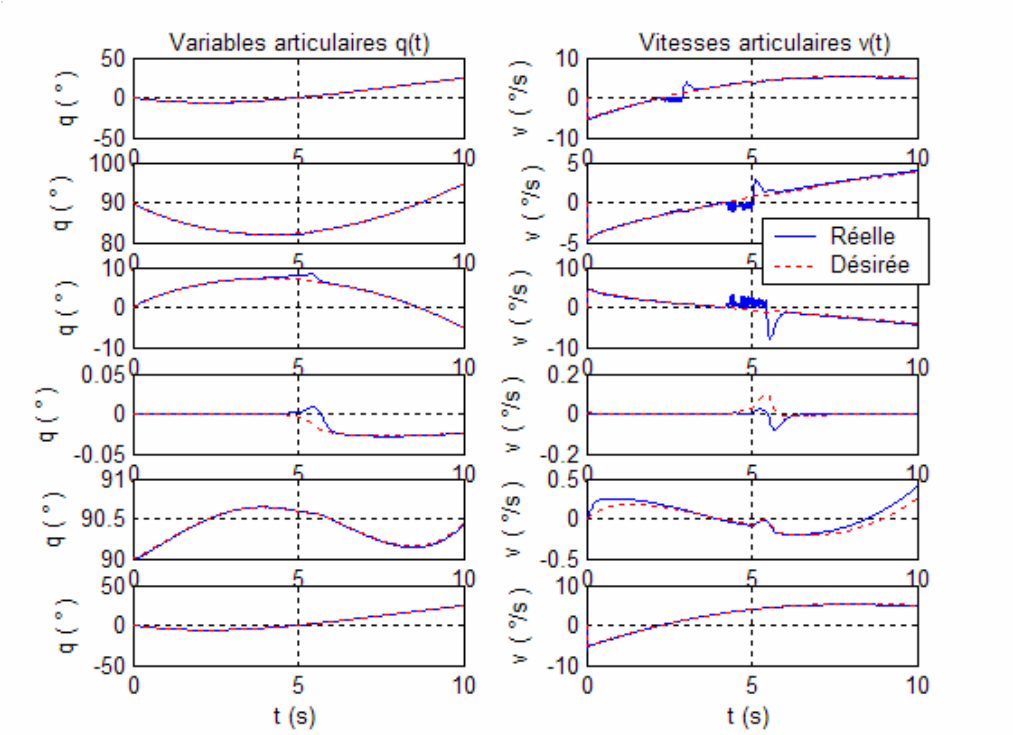


Figure 3.21.b. *Variables et vitesses articulaires pour une tâche de poursuite, avec une commande linéarisante : $Q= I_8$, $R=15\ I_8$.*

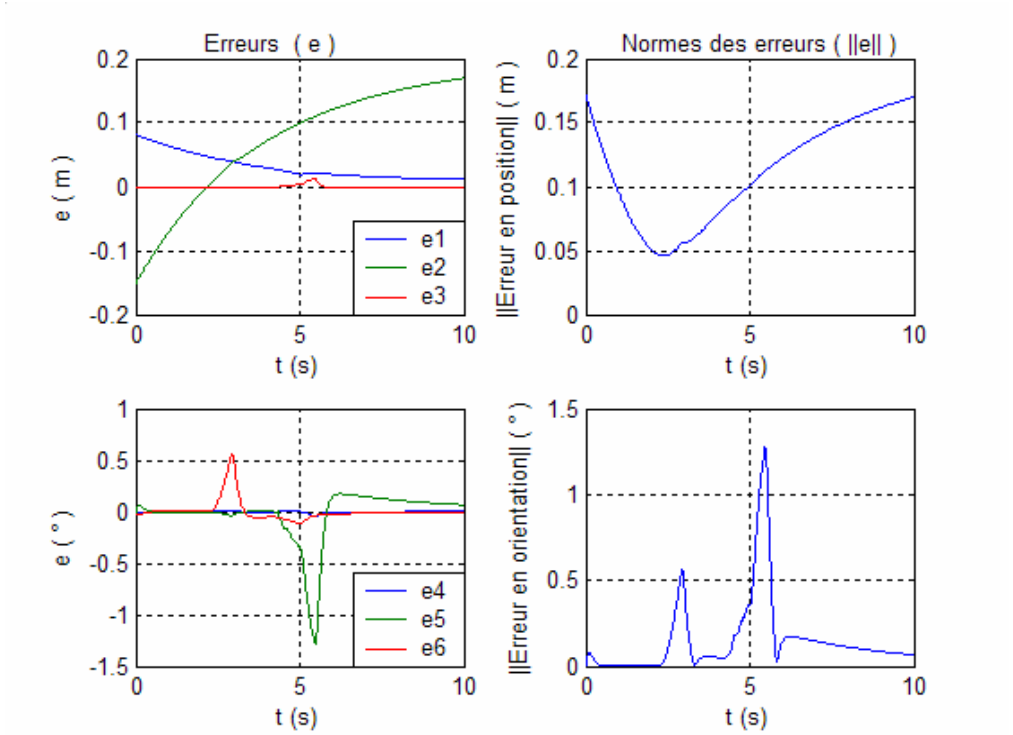


Figure 3.21.c. *Erreurs pour une tâche de poursuite, avec une commande linéarisante : $Q= I_8$, $R=15\ I_8$.*

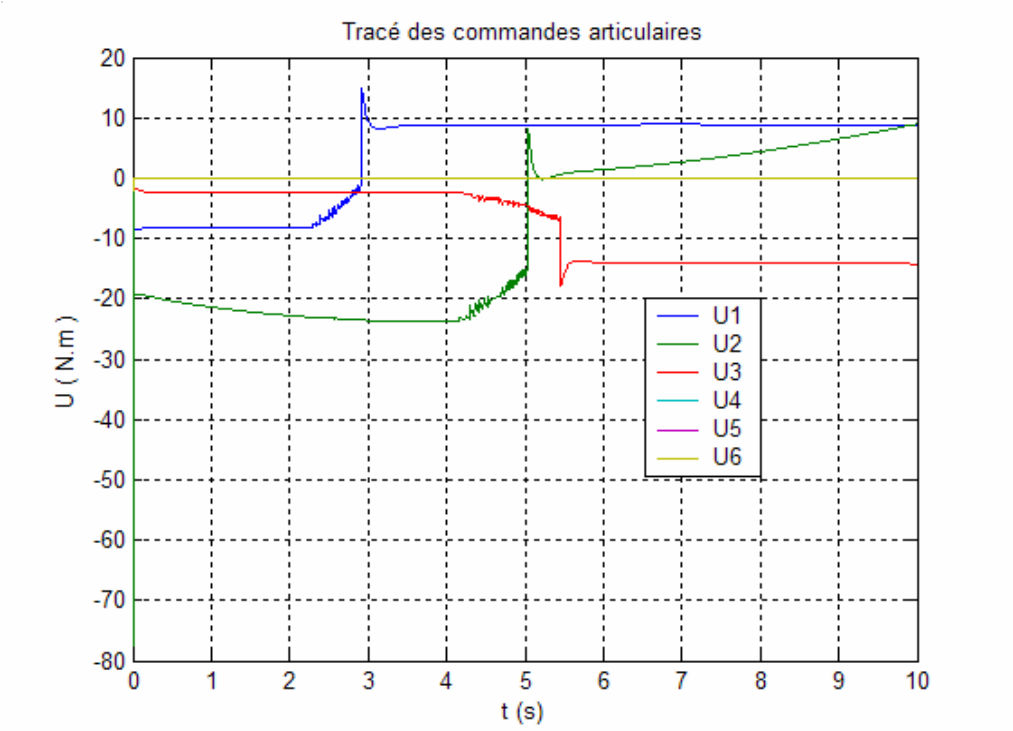


Figure 3.21.d. *Couples articulaires pour une tâche de poursuite, avec une commande linéarisante : $Q= I_8$, $R=15\ I_8$.*

Encore une fois, la poursuite n'est pas réalisée, comme dans le cas précédent lorsque g était calculée par tâtonnement. Ceci confirme la remarque faite précédemment : Pour avoir une bonne poursuite, g doit être élevée, mais pas jusqu'à saturer la commande. Dans notre cas, d'après (3.56), $g \approx 0.26\ \mathbb{I}_8$, ce qui n'a pas permis d'établir la poursuite, alors que pour g variable, le gain g tendait vers *10 !* Et c'est ce qui a permis dans ce cas d'avoir une bonne poursuite de trajectoire.

3.5.4. Commande visuelle par mode de glissement :

Considérons pour cela une surface de glissement tel que :

$$\mathcal{S} = s - s^* \tag{3.60}$$

Remarquons que sur cette surface de glissement, les objectifs de commandes sont réalisés, car si $\mathcal{S} = \underline{0}$, alors $s = s^*$. Dérivons à présent cette surface par rapport au temps :

$$\dot{\mathcal{S}} = \dot{s} = L_s\, T \tag{3.61}$$

La commande par mode de glissement repose sur deux conditions qui doivent être vérifiées pour assurer la convergence :

1. *La condition d'attractivité* qui doit être vérifiée à chaque instant, et pour chaque élément de la surface $\mathcal{S}$, tel que : $\dot{\mathcal{S}}_i\, \mathcal{S}_i < 0$;
2. *La condition d'invariance* qui traduit le maintient sur la surface de glissement une fois celle-ci atteinte, pour chaque élément de la surface : $\dot{\mathcal{S}}_i = 0\ si\ \mathcal{S}_i = 0$.

Le choix de la commande T dans l'équation (3.61) doit vérifier les deux conditions citées précédemment. Un choix possible de cette fonction est :

$$T = -L_s^+ \, K \, sat(\mathcal{S}), \qquad \text{Avec } K>0 \tag{3.62}$$

La fonction *Sat* (Saturation) est définie comme suit :

$$sat(x) \overset{def}{\equiv} \begin{cases} x, & |x| \leq \varepsilon \\ \varepsilon \cdot sign(x), & |x| \geq \varepsilon \end{cases} \qquad \text{Avec } \varepsilon > 0 \text{ constante} \tag{3.63}$$

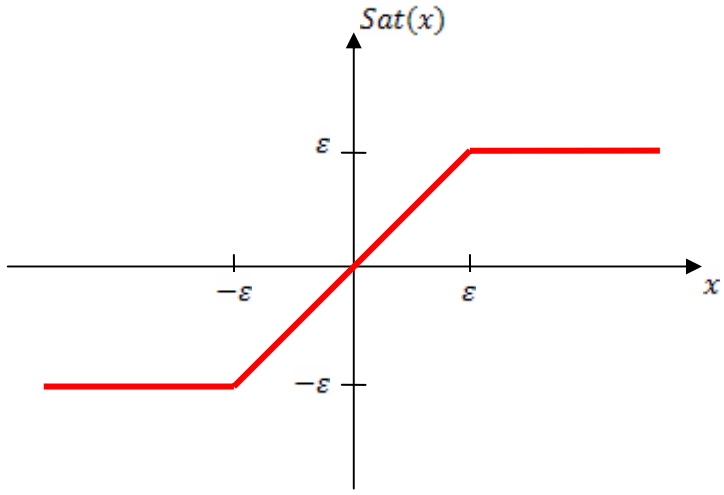


Figure 3.22. *Fonction saturation (sat).*

L'emploi de la fonction *Sat* au lieu de la fonction *Sign* permet de ne pas trop solliciter la commande en lui imposant le passage d'une valeur forte positive, en une valeur forte négative. En remplaçant l'équation (3.62) dans (3.61), nous avons :

$$\dot{\mathcal{S}} = -K \, sat(\mathcal{S}), \qquad \text{Avec } K>0 \tag{3.64}$$

Ce qui vérifie les deux conditions de convergence citées précédemment. En effet, pour tout élément i de la surface de glissement, nous avons :

1. *Condition d'attractivité :*

$$\dot{\mathcal{S}}_i \mathcal{S}_i = -K \, \mathcal{S}_i \, sat(\mathcal{S}_i) = \begin{cases} -K \, \mathcal{S}_i^2 < 0, & |\mathcal{S}_i| \leq \varepsilon \\ -K \, \varepsilon \, |\mathcal{S}_i| < 0, & |\mathcal{S}_i| \geq \varepsilon \end{cases} \qquad \text{Avec } K, \varepsilon > 0 \tag{3.65}$$

2. *Condition d'invariance :*

$$\mathcal{S}_i = 0 \qquad \Rightarrow \qquad \dot{\mathcal{S}}_i = -K \, sat(0) = 0 \tag{3.66}$$

Ainsi, les performances de cette commande dépendent des paramètres K et ε. Il ne nous reste donc plus qu'à valider cette méthode par des simulations. Considérons comme cas de positionnement, une translation dans l'image de -10 pixels sur u et 39 pixels sur v et une rotation de 15°. Les paramètres sont fixés à $K=0.5(s^{-1})$ et $\varepsilon = 50$ *pixels* :

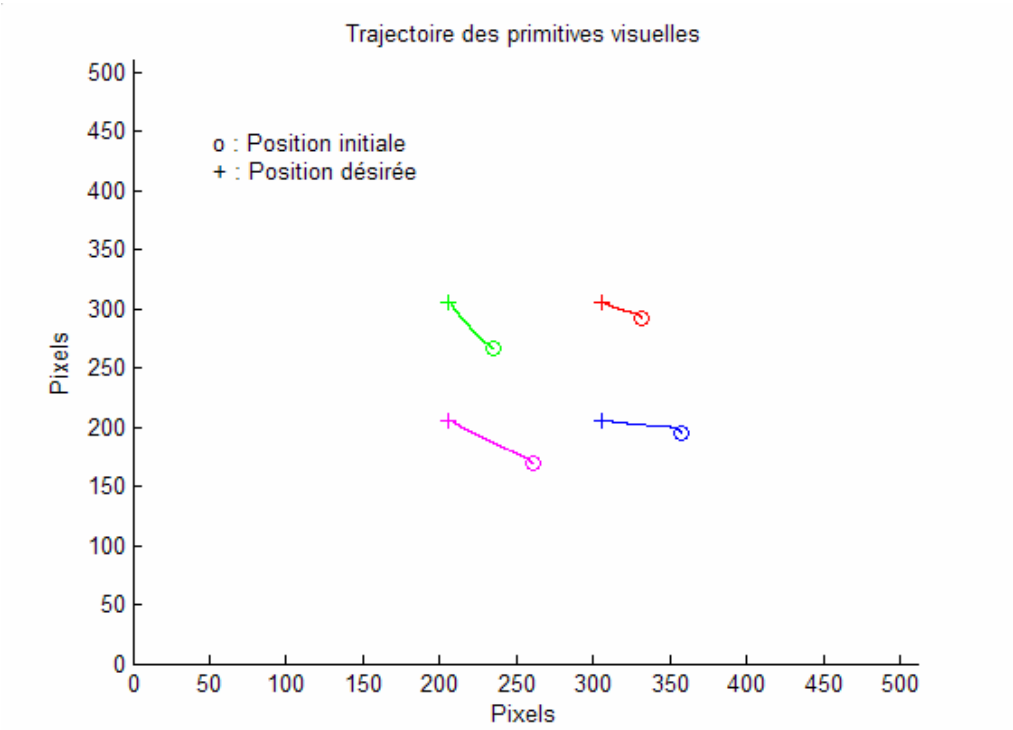


Figure 3.23.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par mode de glissement : K=0.5(s^{-1}) et ε =50 pixels .*

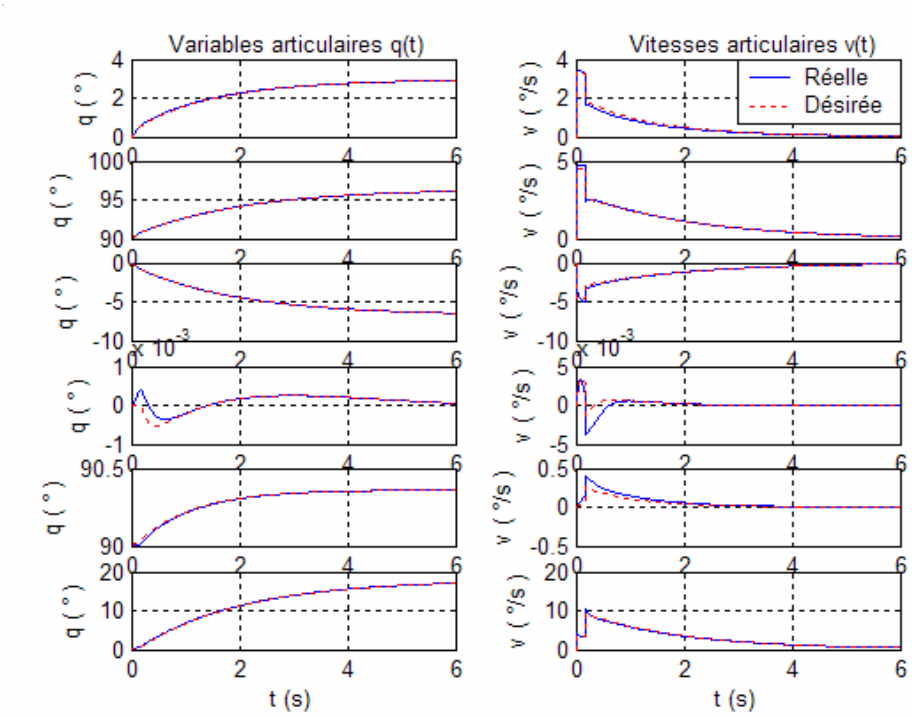


Figure 3.23.b. *Variables et vitesses articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par mode de glissement : K=0.5(s^{-1}) et ε =50 pixels.*

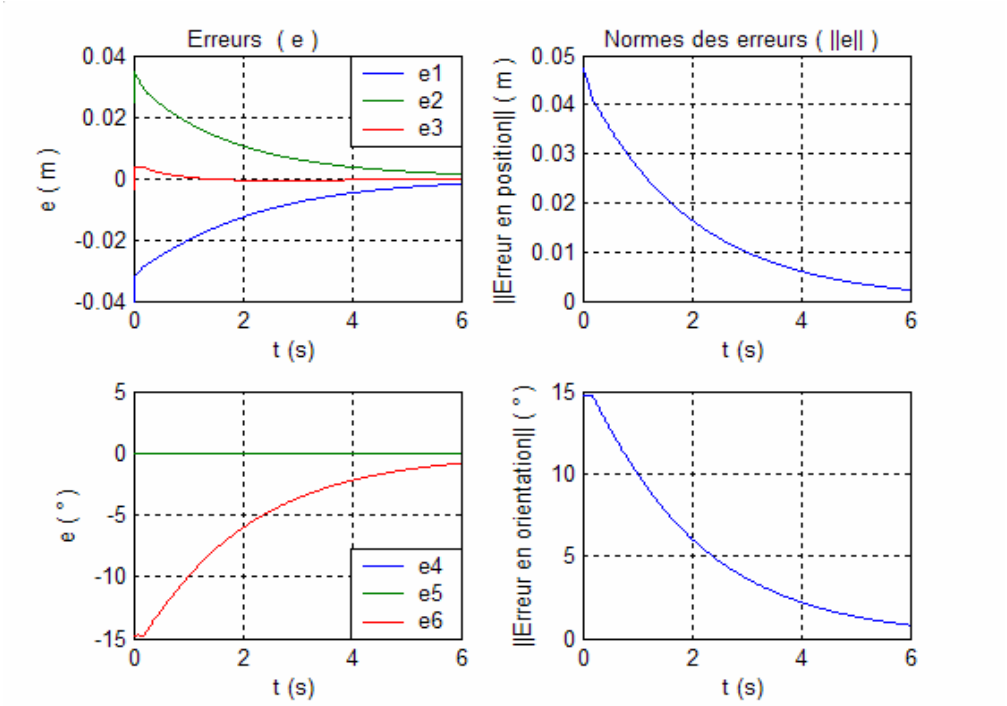


Figure 3.23.c. *Erreurs pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par mode de glissement : K=0.5(s^{-1}) et ε =50 pixels.*

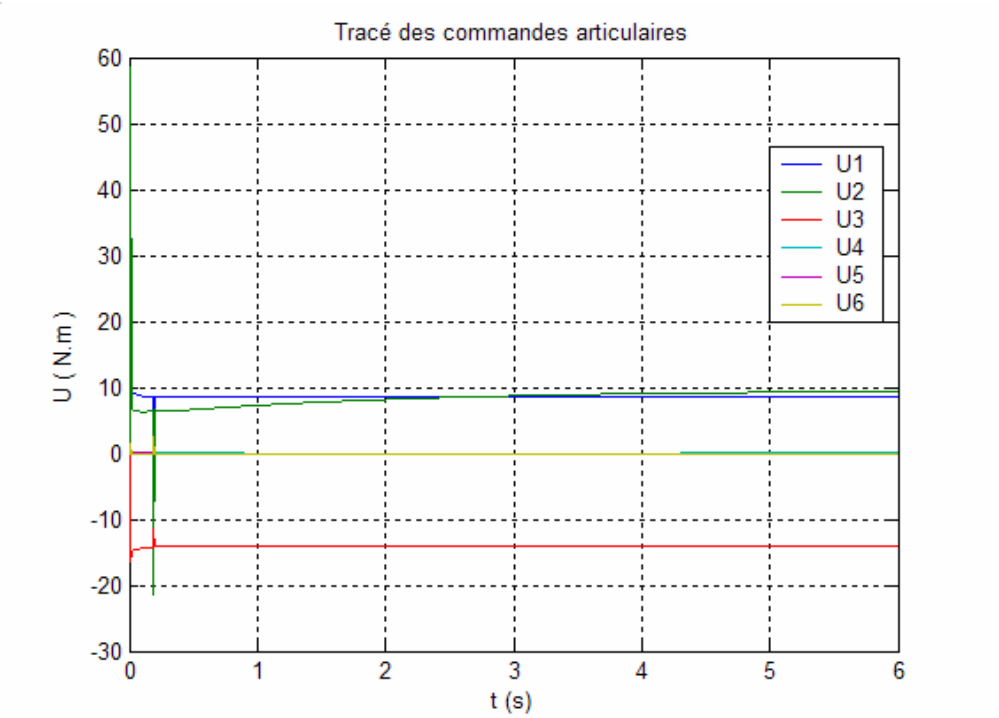


Figure 3.23.d. *Couples articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par mode de glissement : K=0.5(s^{-1}) et ε =50 pixels.*

Il y a effectivement convergence en moins de 6 secondes, avec des commandes admissibles. Etudions à présent le cas de la poursuite. Nous prendrons dans ce cas les mêmes valeurs de K et ε, à savoir $K=0.5(s^{-1})$ et $\varepsilon = 50$ *pixels* :

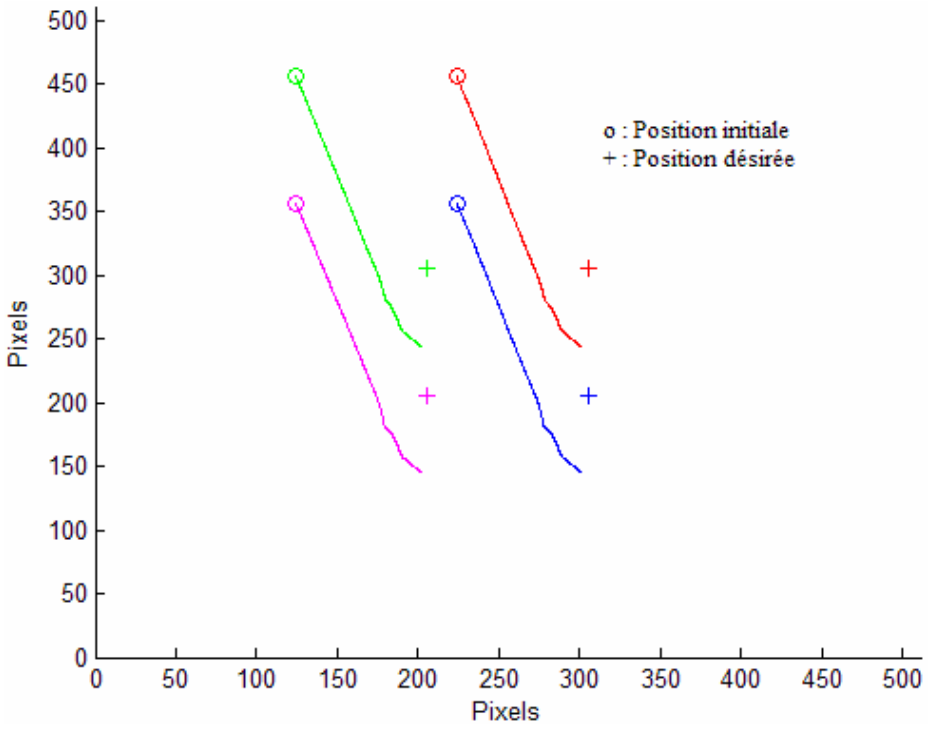


Figure 3.24.a. *Trajectoire dans l'image des primitives visuelle pour une tâche de poursuite, avec la commande par mode de glissement : $K=0.5(s^{-1})$ et $\varepsilon = 50$ pixels.*

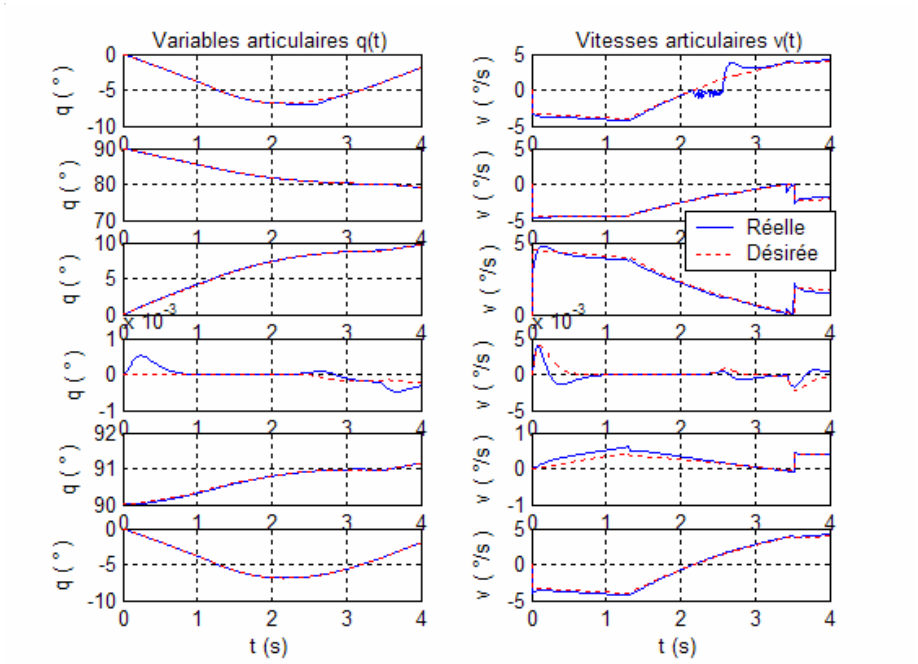


Figure 3.24.b. Variables et vitesses articulaires pour une tâche de poursuite, avec la commande par mode de glissement : $K=0.5(s^{-1})$ et $\varepsilon = 50$ *pixels*.

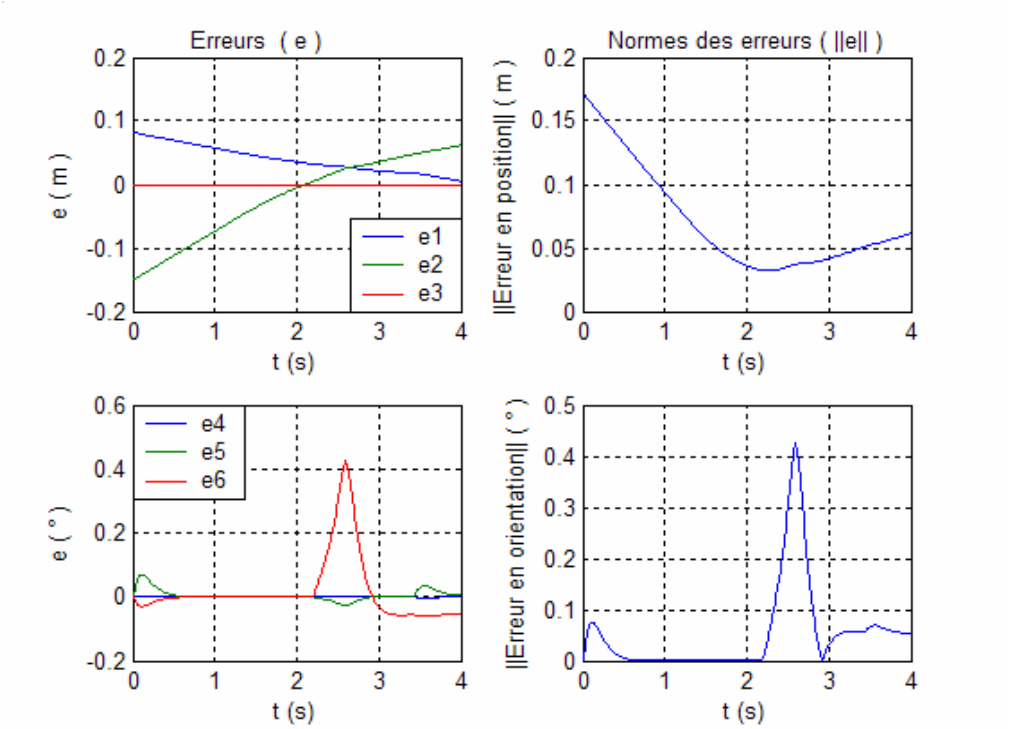


Figure 3.24.c. Erreurs pour une tâche de poursuite, avec la commande par mode de glissement : K=0.5(s^{-1}) et ε =*50 pixels*.

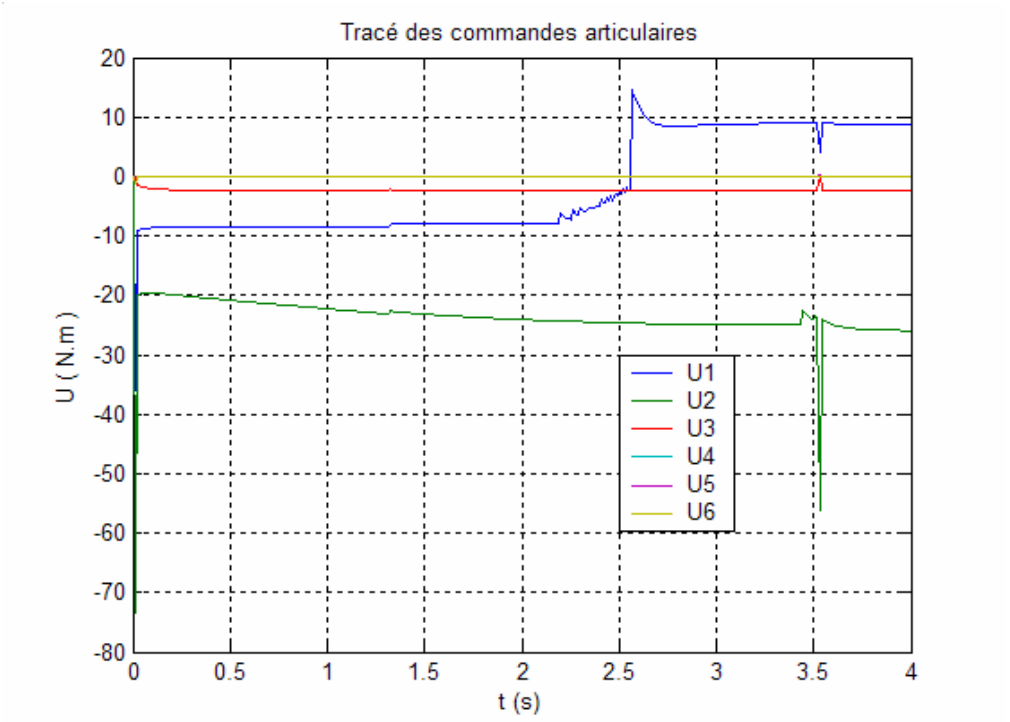


Figure 3.24.d. *Couples articulaires pour une tâche de poursuite, avec la commande par mode de glissement : K=0.5(s^{-1}) et ε =50 pixels.*

Remarquons que dans ce cas, nous n'avons pas une bonne poursuite : il subsiste toujours une erreur statique, qui ne s'annule que pour les fortes valeurs de gain *K*. L'objet à

saisir continu toujours sa course jusqu'au moment où ce dernier quitte l'espace de travail du robot, rendant ainsi sa capture impossible.

Les avantages de la commande par mode de glissement (dans un cas général) sont la *rapidité* de la convergence et la *robustesse vis-à-vis des erreurs de modélisation*. Cependant, la commande appliquée est essez *énergétique* et l'on distingue l'apparition de l'effet de *broutement* au niveau de la commande. D'autre part, cette méthode est équivalente à la méthode de la commande linéarisante. En effet, en partant de l'équation (3.62), et en considérant que l'on soit proche de la convergence ($|\mathcal{S}_i| \leq \varepsilon$), on retrouve l'équation (3.54). Sauf que dans la commande par glissement, le gain K est scalaire et est déterminé par tâtonnement, alors que pour la commande linéarisante, il est matriciel et élaboré à partir du critère linéaire quadratique. Cette méthode trouve donc son avantage par rapport à la précédente, dans la rapidité de la convergence aux instants initiaux ($|\mathcal{S}_i| \geq \varepsilon$).

3.5.5. Commande visuelle par logique floue :

Nous tenterons dans cette partie de concevoir une commande visuelle à base de logique floue, du fait des avantages de celle-ci : à savoir la robustesse vis-à-vis des perturbations et des erreurs de modélisation.

Pour cela, considérons l'erreur entre l'image actuelle et l'image désirée, notée $\varepsilon = s - s^*$, pour éviter toute confusion avec la fonction de tâche e. Les variations de cette erreur sont donnée par :

$$\dot{\varepsilon} = \dot{s} = L_s T = \mathcal{V} \tag{3.67}$$

Ainsi, on a linéarisé le comportement de la boucle de vision, sauf que cette fois-çi, le vecteur d'état n'est plus la variable visuelle s mais l'erreur entre celle-ci et sa valeur désirée. La variation de cette erreur, notée $\mathcal{V}$, est directement liée à la commande T recherchée :

$$T = L_s^+ \mathrm{V} \tag{3.68}$$

La difficulté réside dans le calcul de la commande $\mathcal{V}$. Dans le paragraphe 3.5.3., nous l'avions élaboré à travers un retour d'état. Dans ce paragraphe, nous élaborerons $\mathcal{V}$ à travers un régulateur flou. Nous expliquons à présent les différents constituants de ce régulateur, puis nous le synthétiserons pour notre cas. Nous reportons le lecteur à l'ouvrage [AHM 04] dont nous avons exploité les informations, pour de plus amples détails.

3.5.5.a. Constituants du régulateur flou :

La figure 3.25 schématise les différents modules du régulateur flou, qui sont au nombre de quatre :

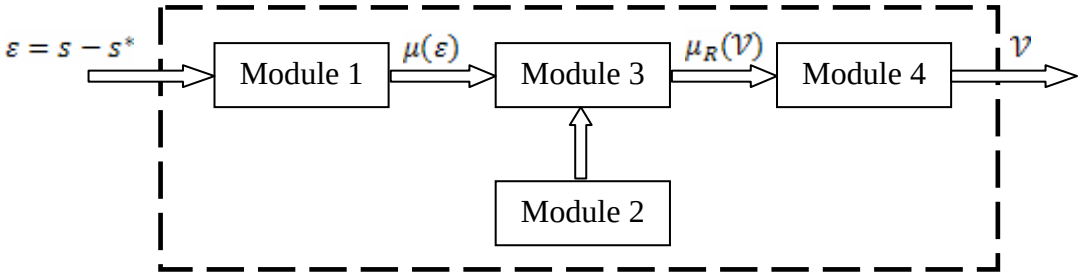


Figure 3.25. *Constituants d'un régulateur flou.*

- Le module 1 est appelé *« interface de fuzzification »*. Son rôle est de transformer les entrées en des ensembles flous ;
- Le module 2 est appelé *« base de connaissance »*. Il a pour rôle de donner le comportement dynamique du système de commande par des règles linguistiques, basées sur la connaissance d'un expert. Les règles sont de la forme : *« si (ensemble de condition) alors (ensemble de conséquences) »* ;
- Le module 3 est appelé *« moteur d'inférence flou »*. Il permet, à partir d'une entrée floue et en exploitant les informations inscrites dans la base de règles, de déduire les conséquences floues ;
- Le module 4 est appelé *« interface de défuzzification »*. Il consiste en la transformation des conséquences floues résultantes du moteur d'inférence en des grandeurs numériques.

Il existe différents outils possibles pour l'interface de fuzzification et de défuzzification. Néanmoins, il a été montré que ces outils sont équivalents en conséquence. La partie la plus importante est l'élaboration des règles floues. Ce sont ces règles qui détermineront la qualité du régulateur flou.

3.5.5.b. Elaboration des règles floues :

L'élaboration des règles floues se fait après une bonne étude du comportement du système. Ils consistent en des règles *linguistiques* imposant au régulateur un certain comportement en fonction des sorties du système à réguler. Il nous est possible, en fonction de l'équation (3.67) de déduire ces règles.

Définissons tout d'abord les différents domaines d'appartenance : $\mu_N(x), \mu_0(x), \mu_P(x)$, correspondant aux domaines : *Négatif, Nul, Positif* respectivement. Nous bornons le signal d'entrée *x* par l'intervalle [x_{min}, x_{max}], ainsi que les signaux de sortie entre *0* et *1*. Comme schématisé dans la figure suivante :

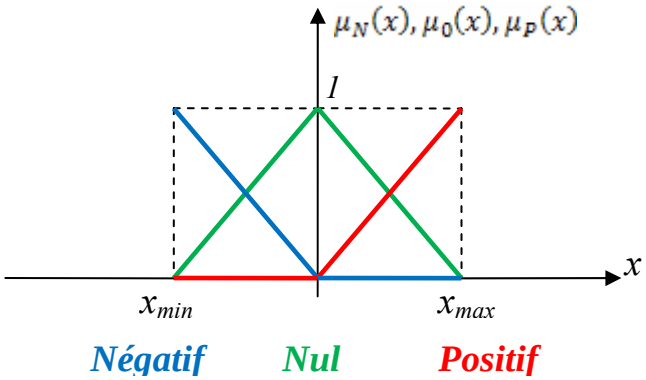


Figure 3.26. *Les différents ensembles flous employés et leur fonction d'appartenance.*

A partir de la définition des différents domaines d'appartenance, et en considérant l'entrée du régulateur comme l'erreur entre la variable visuelle actuelle et désirée, et la sortie $\mathcal{V}$ étant sa vitesse, il nous est possible de définir les règles floues suivantes :

- Si (ε est *Négatif*) Alors ($\mathcal{V}$ est *Positif*) ;
- Si (ε est *Nul*) Alors ($\mathcal{V}$ est *Nul*) ;
- Si (ε est *Positif*) Alors ($\mathcal{V}$ est *Négatif*).

Ces règles permettent d'assurer la stabilité du système de vision. Dans notre cas, nous bornerons l'espace des discours par l'intervalle [$-E_{max}$, E_{max}], et celui de la commande par [$-V_{max}$, V_{max}]. Le schéma de la figure 3.27 donne la méthodologie générale employée dans notre cas [AHM 04], et où $\mathcal{V}_i$ est extraite par le calcul du barycentre :

$$\mathcal{V}_i = \frac{\int_{-V_{max}}^{V_{max}} \mathcal{V}\, \mu_R(\mathcal{V})\, d\mathcal{V}}{\int_{-V_{max}}^{V_{max}} \mu_R(\mathcal{V})\, d\mathcal{V}} \qquad (3.69)$$

Il existe cependant d'autres méthodes de *défuzzification* moins lourdes en temps de calcul [AHM 04]. Mais le résultat est globalement le même.

Ces opérations seront réalisées sur chaque composante de l'erreur ε. La régulation de l'erreur se fait donc de manière découplée, dans le sens où chaque erreur sera traitée séparément. Une fois le vecteur de commande $\mathcal{V}$ calculé, il ne reste plus qu'à l'injecter dans (3.68) pour réaliser la commande linéarisante. Remarquons aussi que les performances de cette méthode dépendent des paramètres E_{max} et V_{max}.

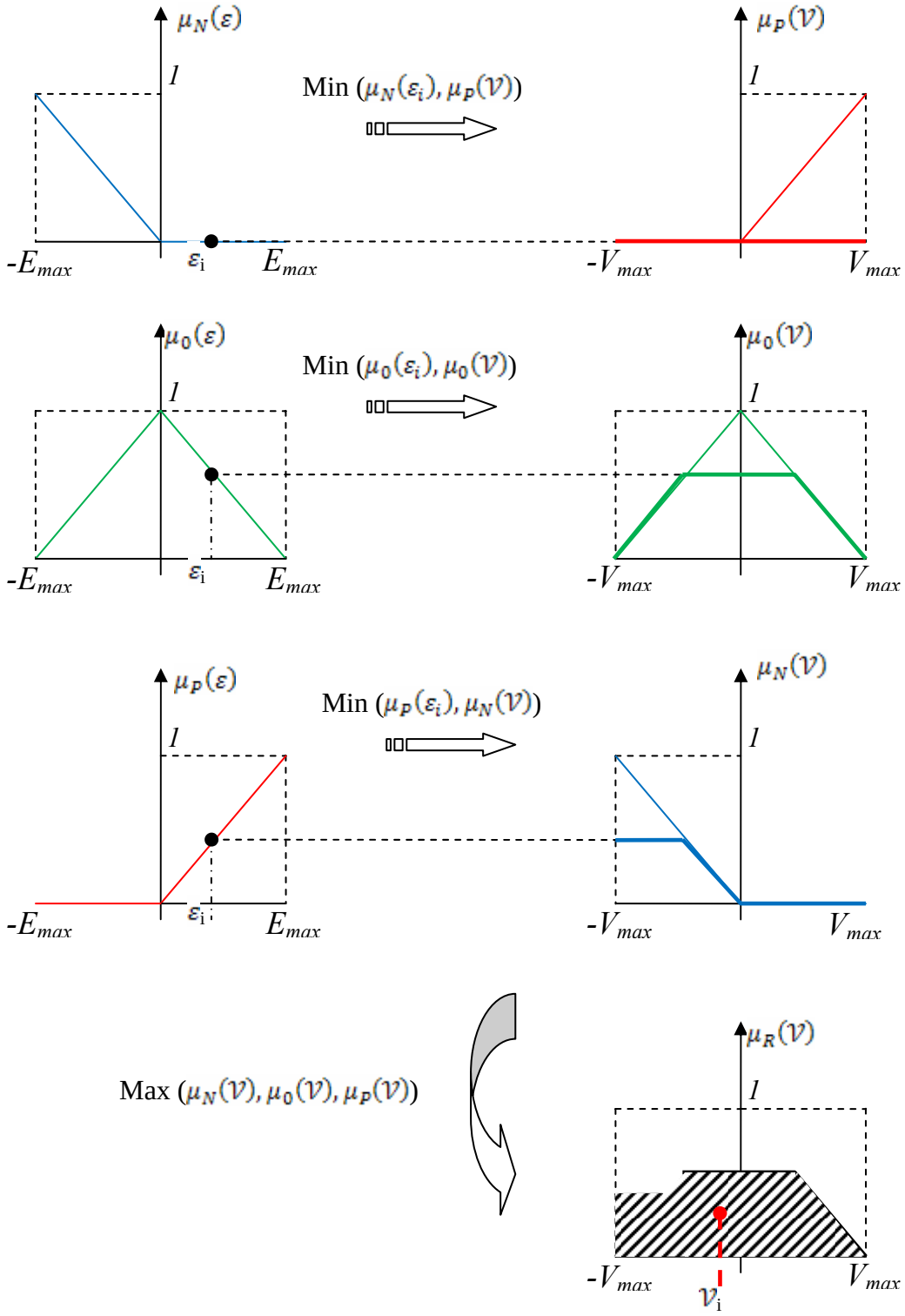


Figure 3.27. *Principe du réglage flou.*

3.5.5.c. Validation de la méthode dans un cas de positionnement :

Nous tentons à présent de réaliser une tâche de positionnement. Tentons pour cela d'effectuer une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°. D'autre part nous prendrons E_{max}=*70 pixels*, et V_{max}=*400 pixels/s* :

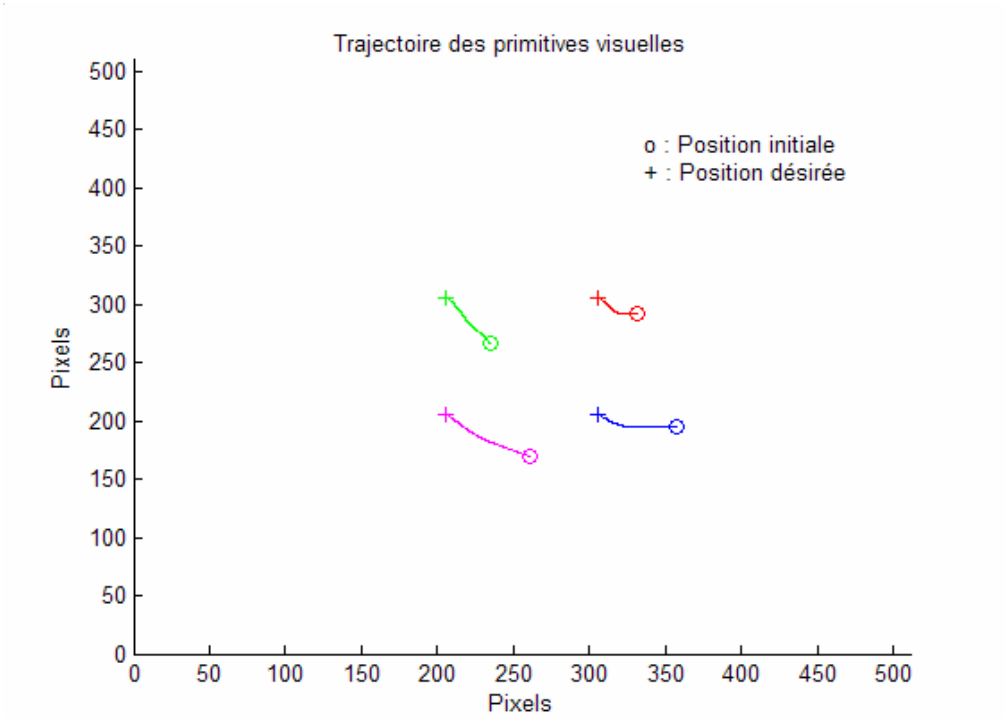


Figure 3.28.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par logique floue : E_{max}=70 pixels*, et *V_{max}=400 pixels/s.*

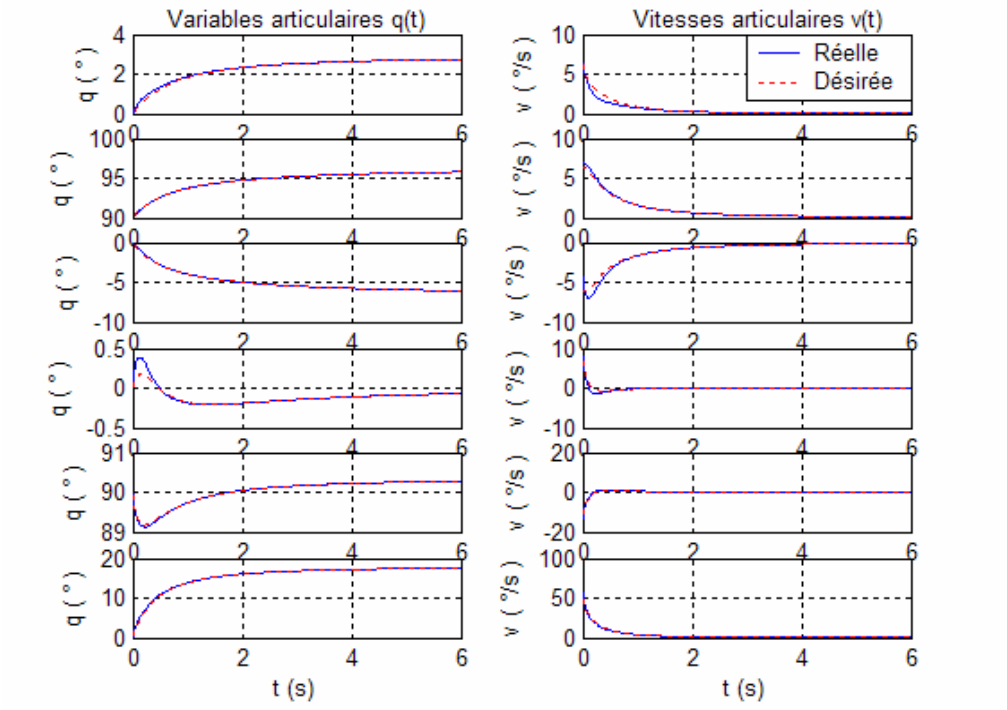


Figure 3.28.b. *Variables et vitesses articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par logique floue : E_{max}=70 pixels*, et *V_{max}=400 pixels/s.*

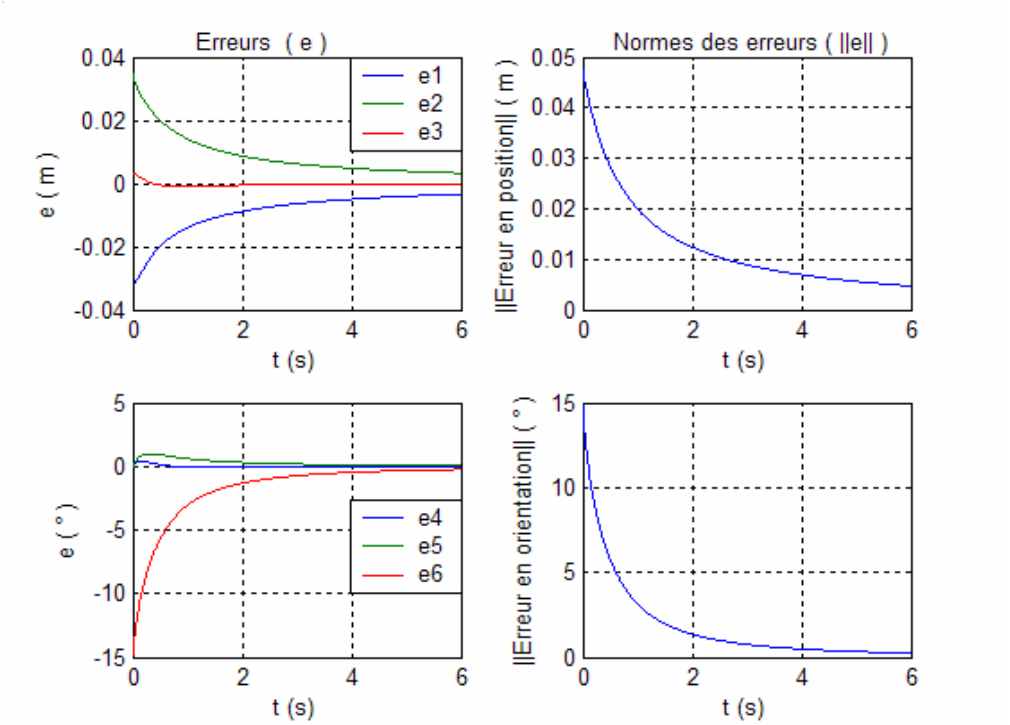


Figure 3.28.c. *Erreurs pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par logique floue : E_{max}=70 pixels*, et *V_{max}=400 pixels/s.*

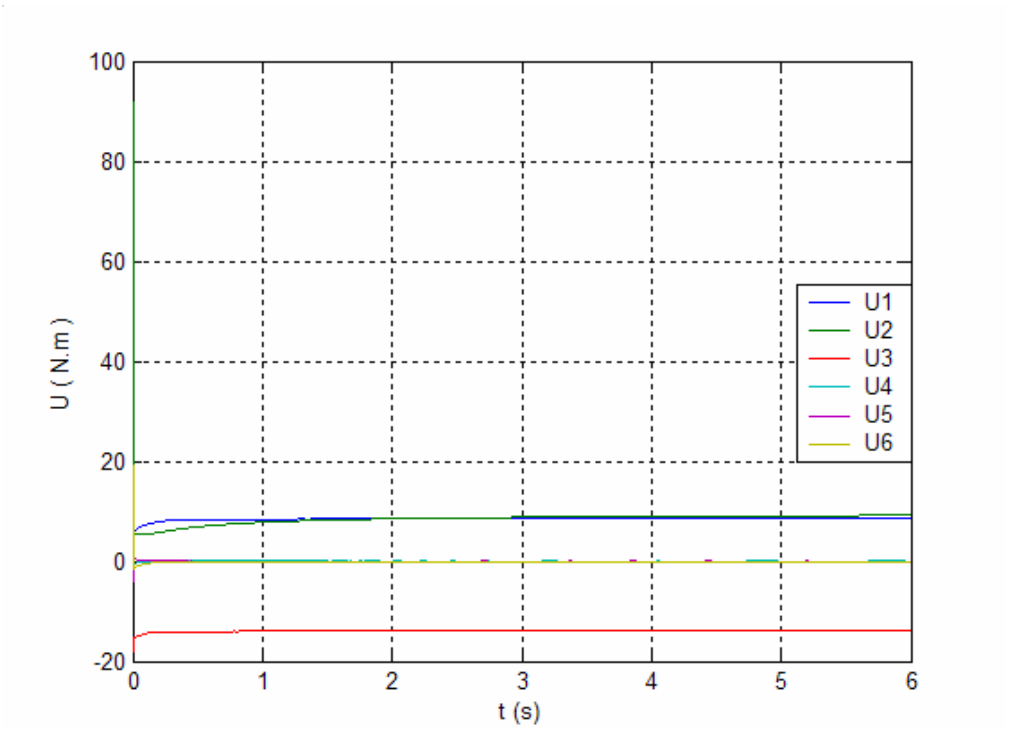


Figure 3.28.d. *Couples articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par logique floue : E_{max}=70 pixels*, et *V_{max}=400 pixels/s.*

La tâche de positionnement est correctement effectuée, en un temps admissible. Remarquons cependant la présence du fort couple articulaire initiale. Encore une fois, nous sommes confrontés au problème du fort couple initiale. Un tâtonnement sur les valeurs de

E_{max} et V_{max} permet de réduire ce fort couple, mais il rallonge le temps de convergence. En fait, si l'on fixe E_{max} proche de l'erreur entre la variable visuelle initiale et désirée ($\varepsilon_0 \approx E_{\max}$), la commande $\mathcal{V}$ sera proche de la valeur max V_{max} ce qui a pour conséquence la convergence rapide, mais aussi un couple articulaire assez important. Ainsi, si l'on cherche à faire varier V_{max} tout en gardant E_{max} constante (proche de l'erreur entre la variable visuelle initiale et désirée), l'idéal serait d'avoir V_{max} faible initialement, puis fort. Un tel cas a été étudié précédemment lorsqu'on cherchait à faire varier le gain g. De même, on peut essayer de faire varier la borne max de la variation des informations visuelles.

3.5.5.d. Commande visuelle par logique floue adaptative :

Considérons pour cela une borne V_{max} non plus fixe, mais variable à travers le temps :

$$V_{\max}(t) = \overline{V}_{\max}\left(1 - e^{-t/a}\right), \qquad \text{Avec } \overline{V}_{\max} > 0 \tag{3.70}$$

Ainsi, en appliquant cette méthode au cas précédemment étudié, avec *a=1(s)*, E_{max} *= 70 pixels,* et $\overline{V}_{\max}$ *=600 pixels/s*, nous obtenons les figures suivantes :

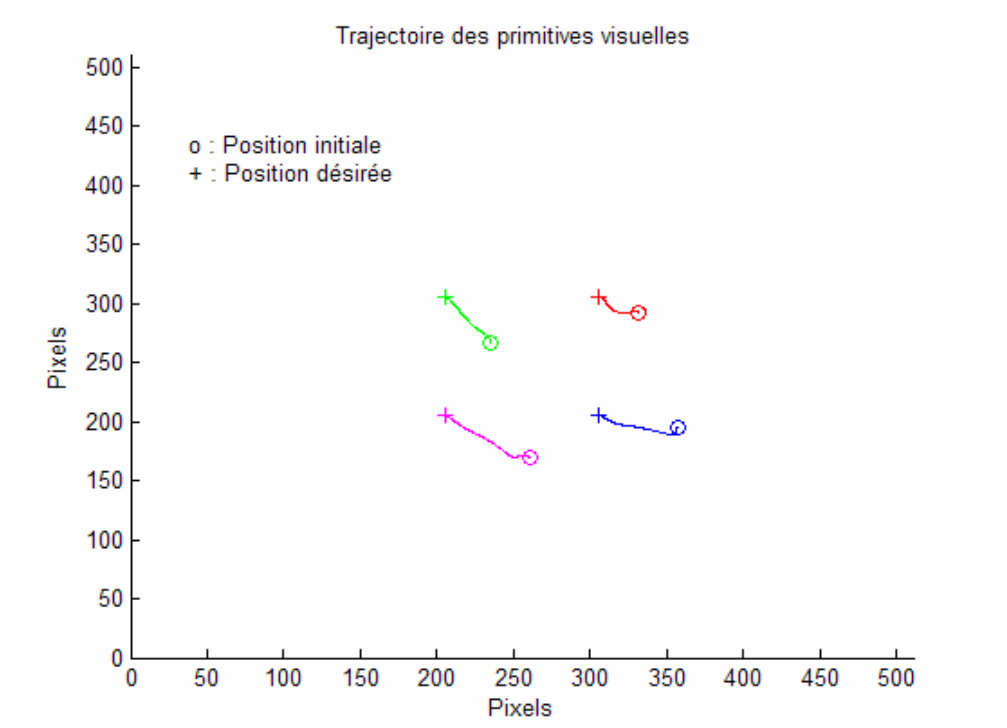


Figure 3.29.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par logique floue adaptative : a=1(s),* E_{max} *= 70 pixels,* et $\overline{V}_{\max}$ *= 600 pixels/s.*

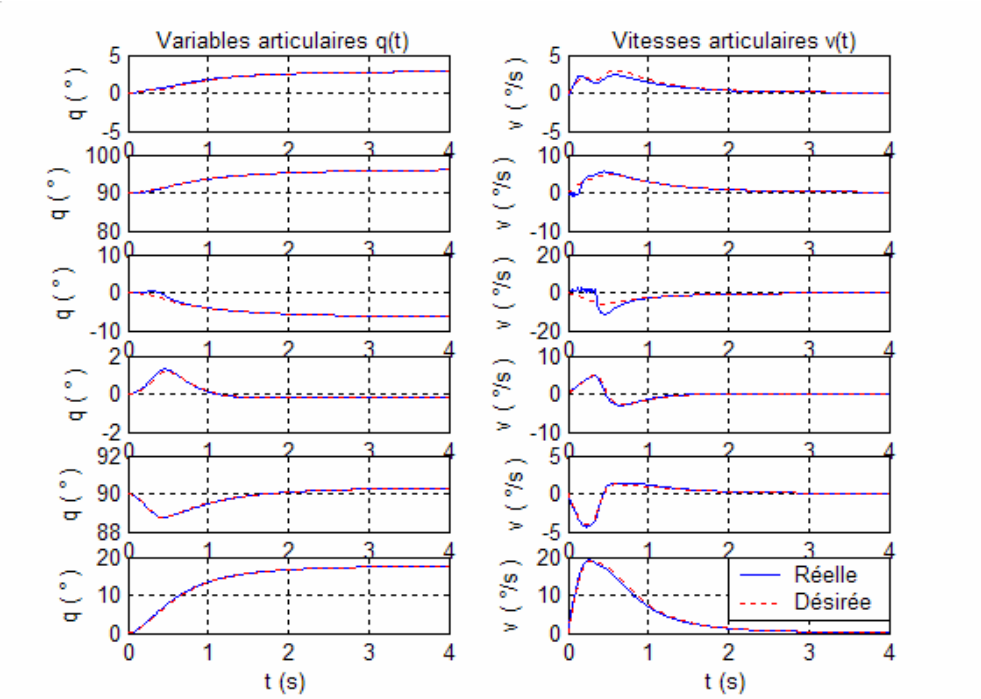


Figure 3.29.b. *Variables et vitesses articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par logique floue adaptative : a=1(s), E_{max} = 70 pixels,* et $\overline{V}_{max}$ *=600 pixels/s.*

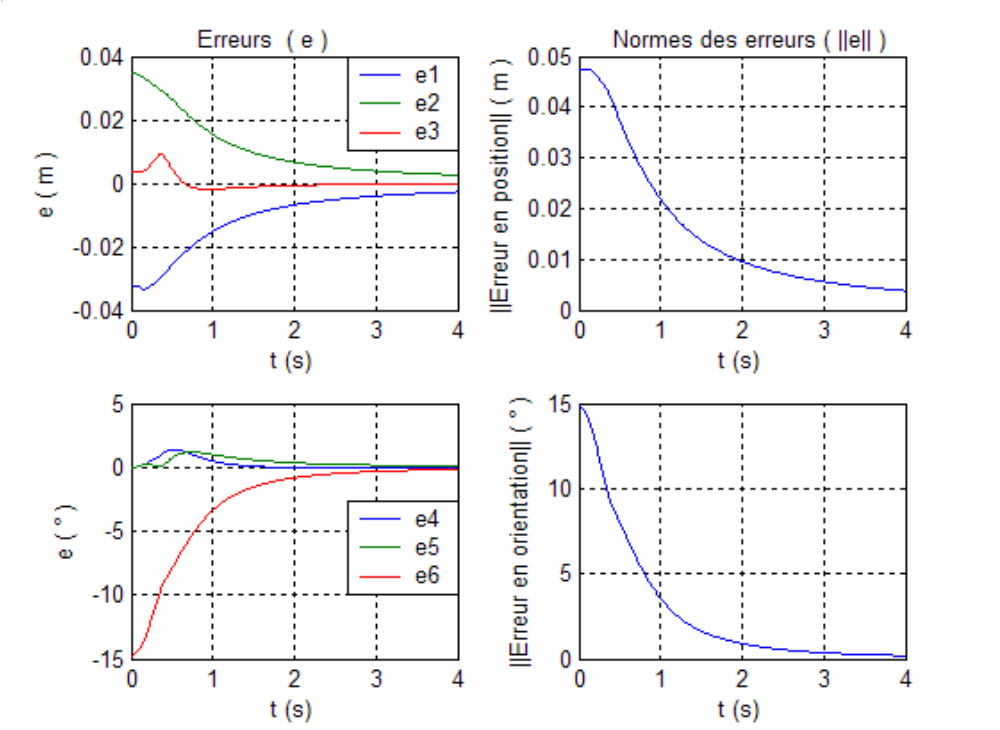


Figure 3.29.c. *Erreurs pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par logique floue adaptative : a=1(s), E_{max} = 70 pixels,* et $\overline{V}_{max}$ *=600 pixels/s.*

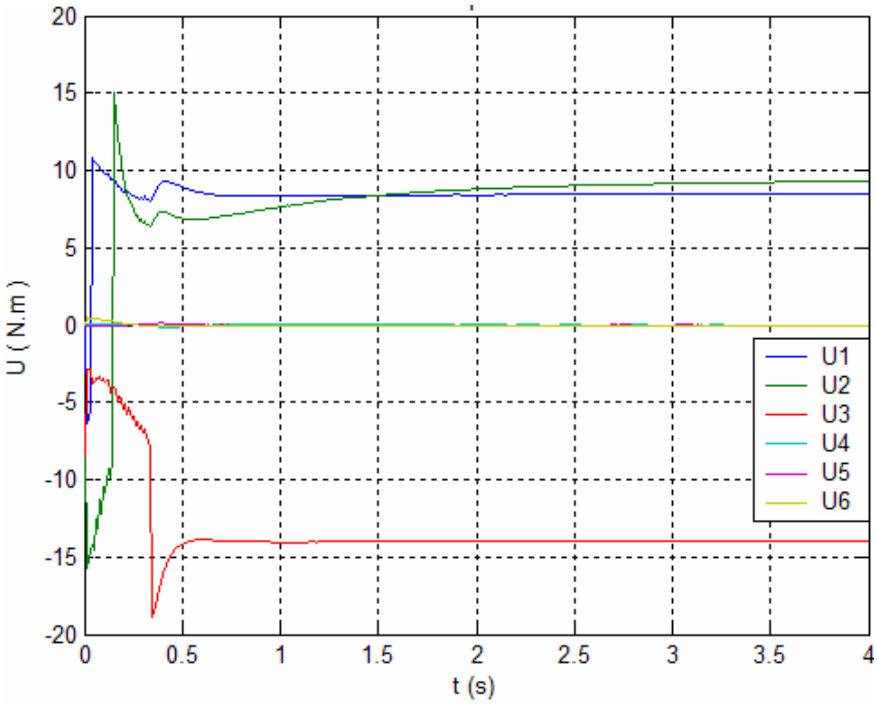


Figure 3.29.d. *Couples articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec la commande par logique floue adaptative : a=1(s),* E_{max} *= 70 pixels,* et $\overline{V}_{max}$ *=600 pixels/s.*

Nous obtenons avec cette méthode une excellente convergence visuelle, et des couples articulaires tout à fait acceptables puisque les pics des couples initiaux ne sont pas trop forts, et qu'aucun couple n'atteint la saturation. D'autre part, la convergence est plus rapide que dans le cas précédent. Il ne nous reste plus qu'à valider cette méthode dans le cas d'une tâche de poursuite. Pour ce faire, considérons *a=0.5(s),* E_{max} *= 200 pixels,* et $\overline{V}_{max}$ *=2000 pixels/s*, nous obtenons les figures suivantes :

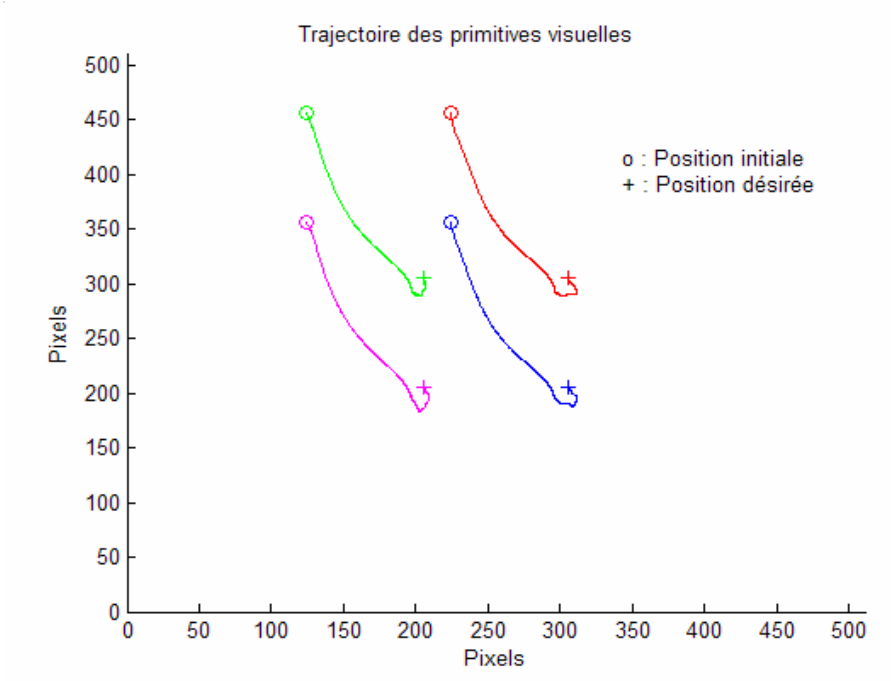


Figure 3.30.a. *Trajectoire dans l'image des primitives visuelle pour tâche de poursuite, avec la commande par logique floue adaptative : a=0.5(s),* E_{max} *= 200 pixels,* et $\overline{V}_{max}$ *=2000 pixels/s.*

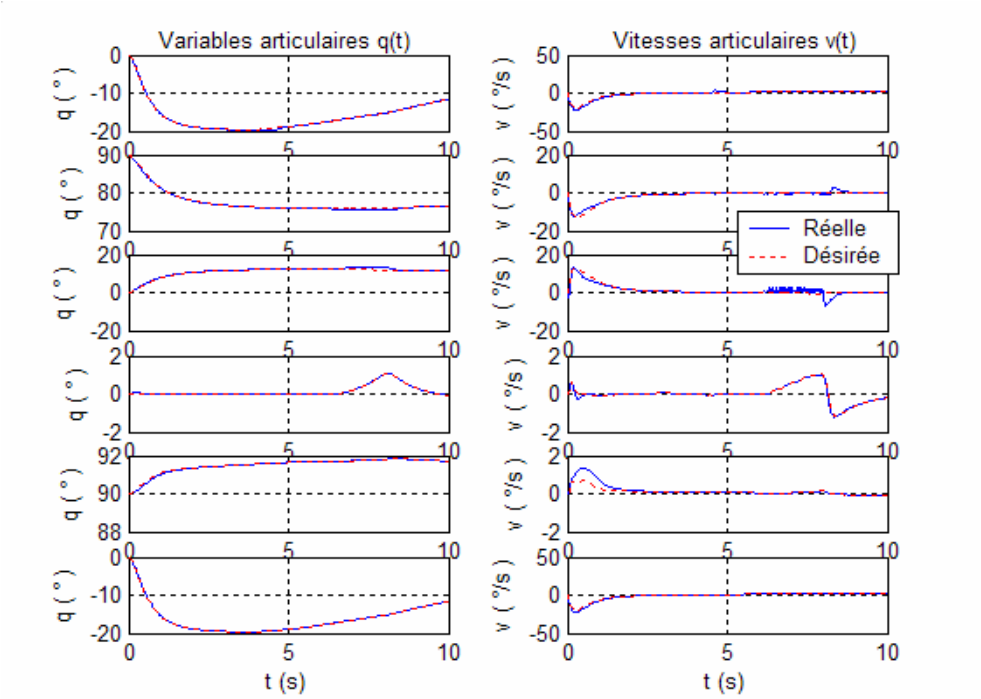


Figure 3.30.b. *Variables et vitesses articulaires pour tâche de poursuite, avec la commande par logique floue adaptative : a=0.5(s), E_{max} = 200 pixels,* et $\overline{V}_{max}$ *=2000 pixels/s.*

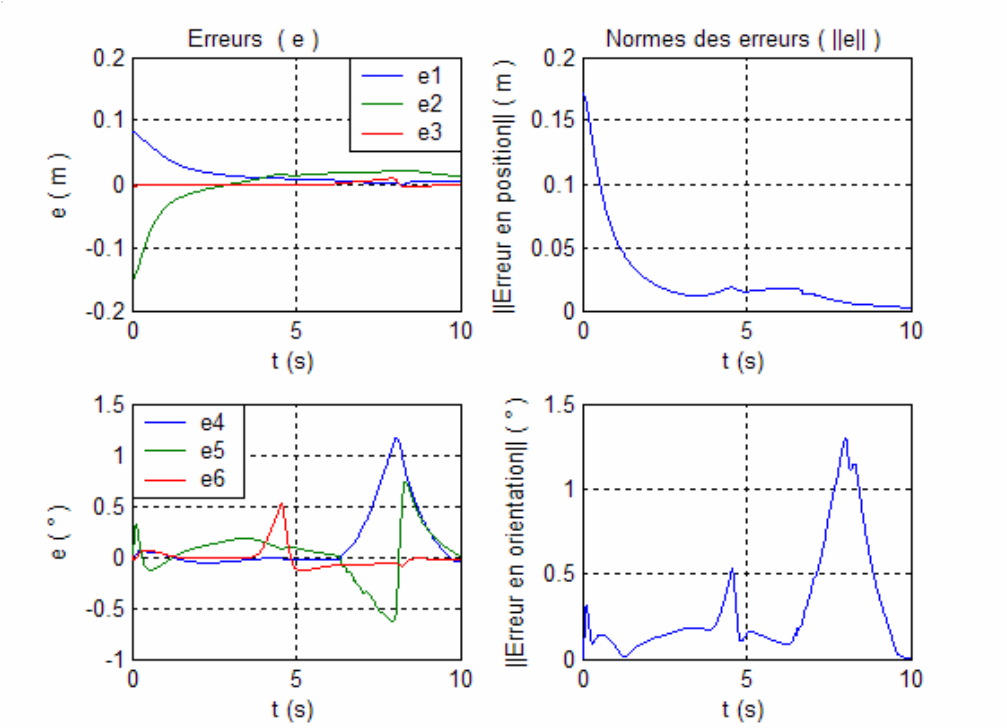


Figure 3.30.c. *Erreurs pour tâche de poursuite, avec la commande par logique floue adaptative : a=0.5(s), E_{max} = 200 pixels,* et $\overline{V}_{max}$ *=2000 pixels/s.*

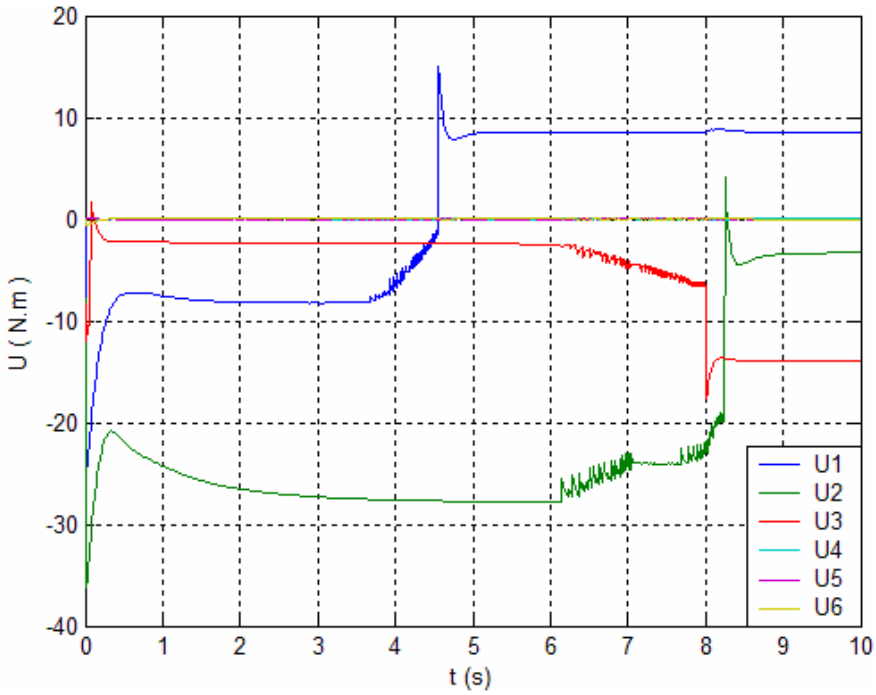


Figure 3.30.d. *Couples articulaires pour tâche de poursuite, avec la commande par logique floue adaptative : a=0.5(s), E_{max} = 200 pixels,* et $\overline{V}_{max}$ *=2000 pixels/s.*

Là aussi, la tâche de poursuite est très bien réalisée. Nous remarquons aussi que les couples articulaires sont tout à fait admissibles. Le mouvement de l'objet ayant été considéré comme perturbation, a très bien été rejeté. Ce qui démontre bien le bon rejet des perturbations de la commande à base de logique floue.

3.6. Etude de la robustesse vis-à-vis des perturbations :

Dans ce paragraphe, on se propose d'étudier la robustesse de l'asservissement visuel vis-à-vis des perturbations. Pour cela, considérons le cas où les perturbations agissent sur les couples articulaires. L'objectif que nous nous fixons est d'assurer la convergence du robot vers la configuration souhaitée, décrite par l'image désirée. Bien que les perturbations agissent sur les couples articulaires, et donc la boucle de commande du robot, nous emploierons un régulateur robuste au niveau de la boucle de vision. L'objectif de la simulation qui va suivre est de montrer la robustesse de l'asservissement visuel vis-à-vis des diverses perturbations présentes sur le système.

Considérons donc des perturbations au niveau de la commande articulaire de l'ordre de 30% de la commande utile. Plaçons nous dans la situation où la tâche à réaliser est une tâche de positionnement, caractérisé par une translation dans l'image de -10 pixels en *u* et 39 pixels en *v*, avec une rotation de 15° autour de l'axe optique *z*. Nous ne testerons que le cas où *g* est variable, avec $g_{max} = 1\ (s^{-1})$ et $a = 1\ (s)$:

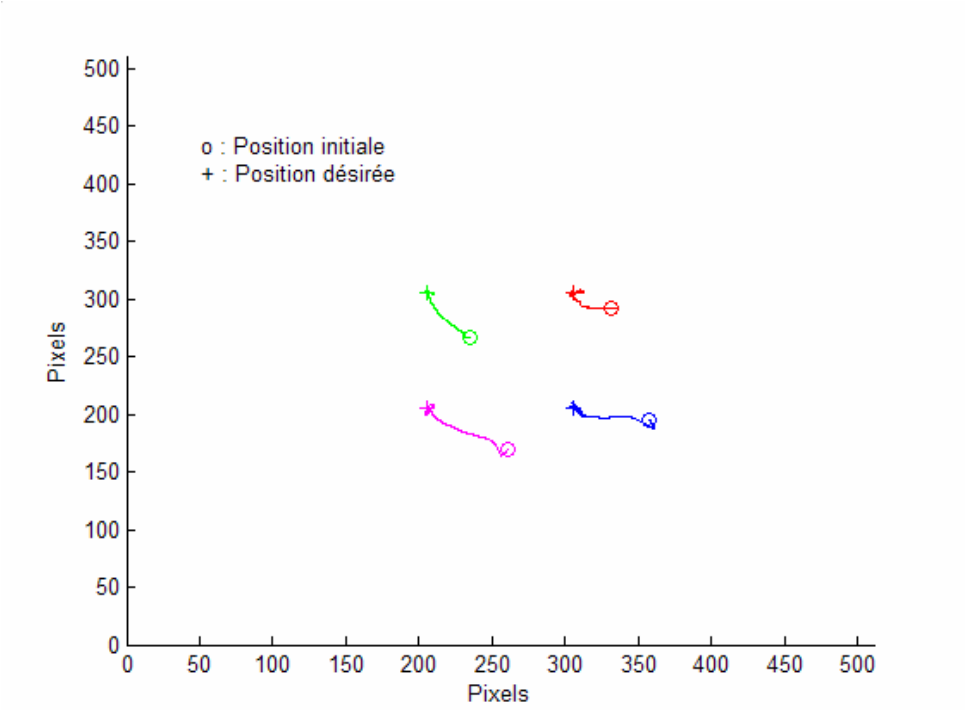


Figure 3.31.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g variable : $g_{max}=1(s^{-1})$ et a=1(s), et des perturbations sur les couples de l'ordre de 30%.*

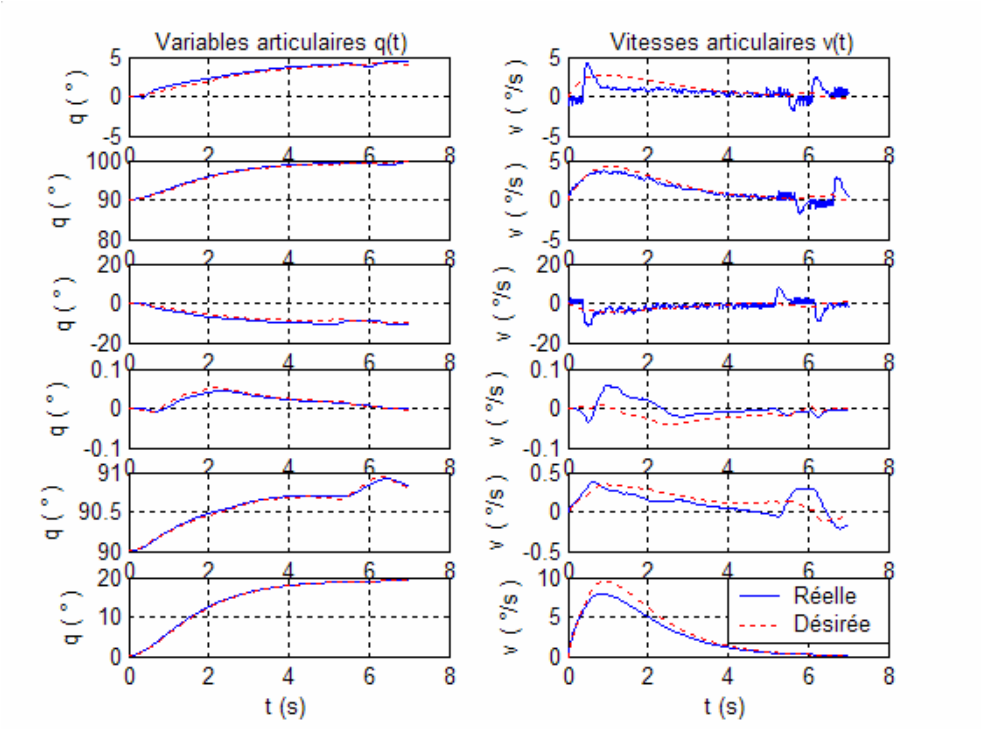


Figure 3.31.b. *Variables et vitesses articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g variable : $g_{max}=1(s^{-1})$ et a=1(s), et des perturbations sur les couples de l'ordre de 30%.*

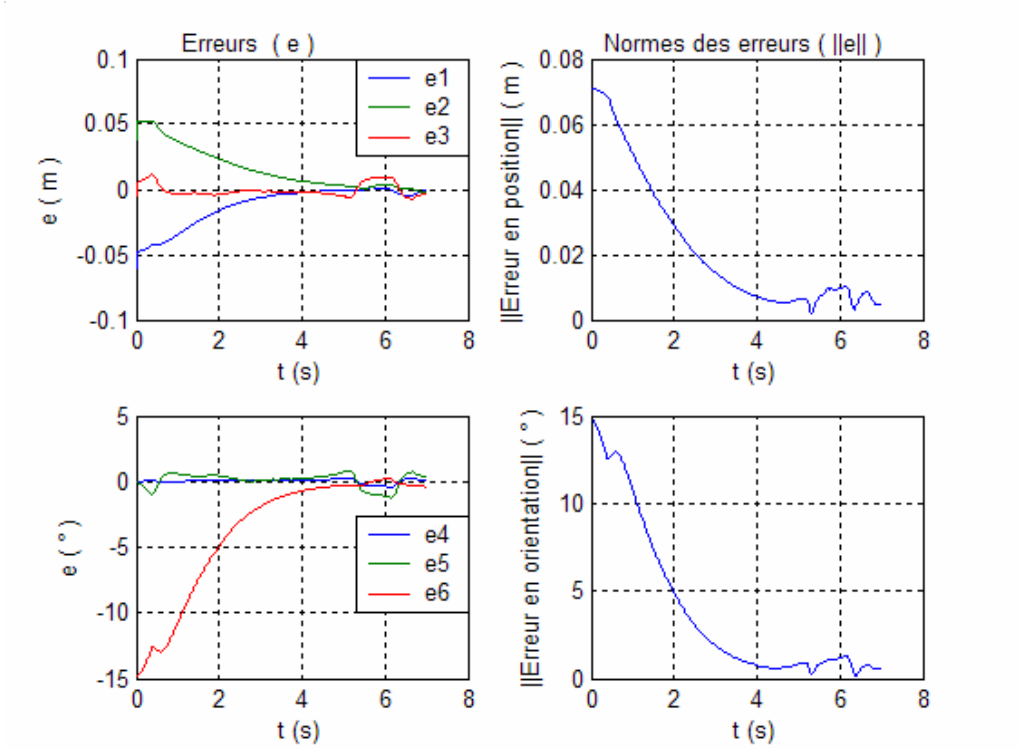


Figure 3.31.c. *Erreurs pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g variable : $g_{max}=1(s^{-1})$ et a=1(s), et des perturbations sur les couples de l'ordre de 30%.*

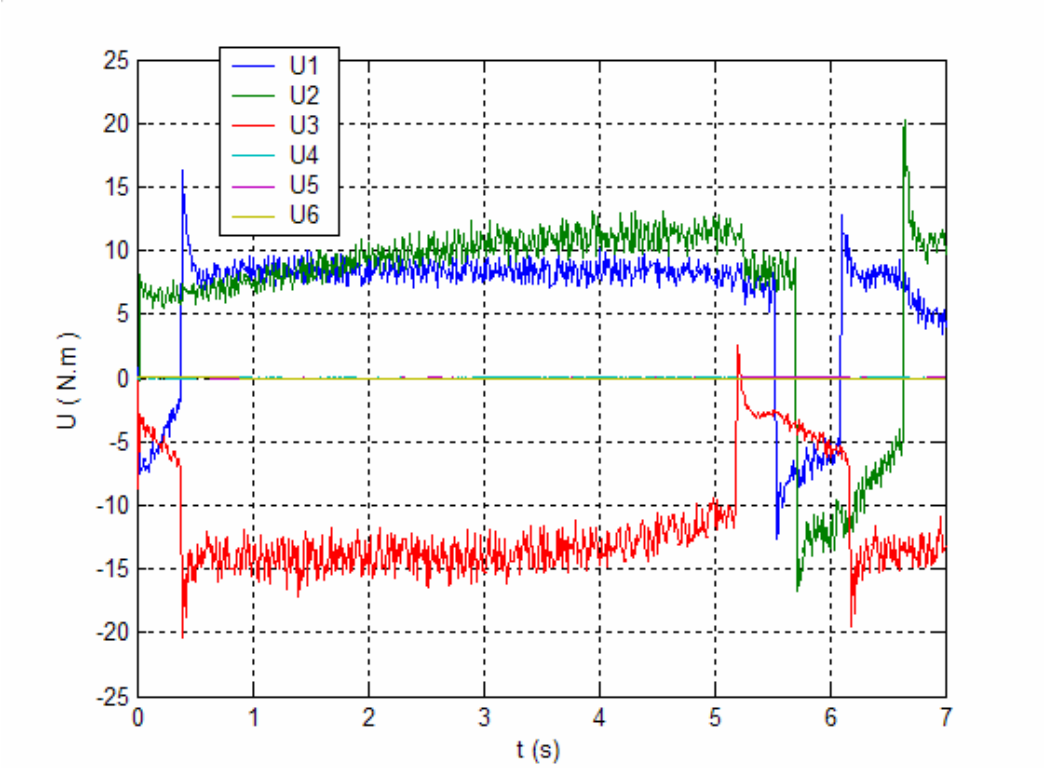


Figure 3.31.d. *Couples articulaires pour une translation de -10 pixels en u et 39 pixels en v et une rotation de 15°, avec g variable : $g_{max}=1(s^{-1})$ et a=1(s), et des perturbations sur les couples de l'ordre de 30%.*

Ce qui démontre bien la robustesse de l'asservissement visuel vis-à-vis des perturbations sur les couples articulaires, du fait de la convergence des informations visuelles.

En effet, l'un des principaux atouts de l'asservissement visuel est la robustesse vis-à-vis des perturbations, comme il a été montré dans la simulation précédente. Les mêmes performances sont constatées pour d'autres méthodes de commande visuelle (commande par mode de glissement, commande par logique flou adaptative, … etc). Il en est de même si la tâche souhaitée est une tâche de poursuite.

Une autre particularité de l'asservissement visuel est la robustesse vis-à-vis des erreurs de modélisations, telle que les erreurs introduites dans la matrice d'interaction ou la matrice de calibrage. Il en est de même si les paramètres du modèle dynamiques du robot sont inconnus ou mal estimés. L'asservissement visuel permet pallier les incertitudes sur l'estimation de certains paramètres, constituant ainsi une commande robuste.

Cette robustesse est due au fait qu'à chaque instant, la commande visuelle agit de manière à faire tendre l'image actuelle (perturbée) vers l'image désirée.

On conçoit donc ainsi une loi de commande robuste indépendante de la loi de commande du robot, qui ne fait qu'assurer la stabilité et la bonne poursuite de ce dernier.

3.7. Conclusion :

Dans ce chapitre, nous avons appliqué la commande visuelle dite 2D par diverses méthodes et simulations. Certaines de ces méthodes permettent d'aboutir à de bons résultats, et d'autres à de moins bons résultats. Or, il ne faut pas oublier que le choix de la méthode se fait non seulement en fonction de ses performances, mais aussi en fonction du temps d'exécution de celle-ci. En effet, dans le cas où il y a nécessite de rapidité des calculs, il est peu probable que la méthode à base de logique floue (par exemple) soit adéquate, du fait des nombreux calculs mis en œuvre. Et dans ce cas, la méthode considérant un gain g scalaire constant peut se révéler comme étant le meilleur choix.

L’asservissement visuel 2D a comme principale avantage d’être rapide en exécution. En effet, seul le temps d’extraction des primitives visuelles et le temps d’exécution de la commande sont à prendre en considération. Or, la commande visuelle ne tient compte que de l’image actuelle et de l’image désirée, mais sans se soucier de la situation du robot, ce qui peut entraîner des mouvements indésirables, comme lors de certaines simulations qui ne nécessitaient que des mouvements de translation, mais où il y a apparition de mouvement de rotation. Il serait intéressant donc de développer des commandes visuelles qui prendraient en considération la pose entre la caméra et l’objet : il s’agit de l’asservissement visuel 3D.

Chapitre

4

Asservissement Visuel 3D

Préambule :

La classification des techniques d'asservissement visuel peut être faite sur la base de la fonction de tâche qui sera choisie de telle manière qu'elle soit nulle si le but de l'asservissement visuel est atteint.

Le principe de l'asservissement visuel 3D est de contrôler le déplacement du robot dans l'espace cartésien. Comme cela a déjà été évoqué, il est possible de choisir des informations visuelles exprimées non plus directement dans l'image mais issues d'une phase de reconstruction ou de localisation 3D.

L'obtention de ces informations 3D peut être effectuée par un simple calcul de pose qui est généralement le plus couramment utilisé en asservissement visuel 3D.

4.1. Introduction

Dans un asservissement visuel 3D, la référence est exprimée sous la forme d'une pose à atteindre. Cette pose définit la position et l'orientation d'un repère lié à l'objet par rapport à celui lié à la caméra. Le calcul des informations 3D à utiliser dans la boucle de commande (pose ou primitives 3D) nécessite la connaissance du modèle 3D de l'objet.

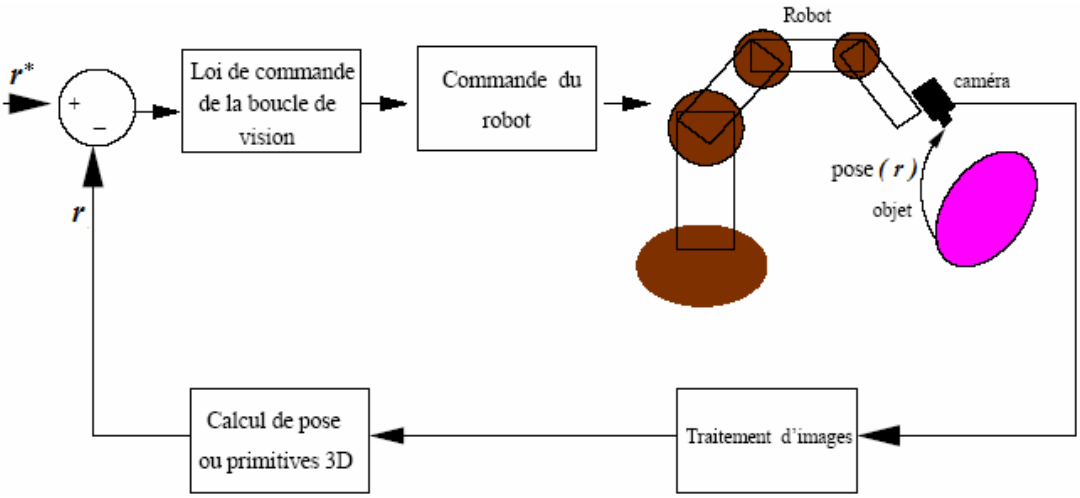


Figure 4.1. *Principe d'un asservissement visuel 3D*

4.2. Modélisation de l'asservissement visuel 3D [MAL 98] :

Le principe de l'asservissement visuel 3D est de contrôler la caméra dans l'espace cartésien en construisant la fonction de tâche à partir de l'estimation du déplacement de la caméra entre deux images. Soient :

R_0 : le repère associé à l'objet observé ;
$R_d=R^*$: le repère associé à la caméra dans sa position désirée ;
$R_c=R$: le repère associé à la caméra dans sa position courante.

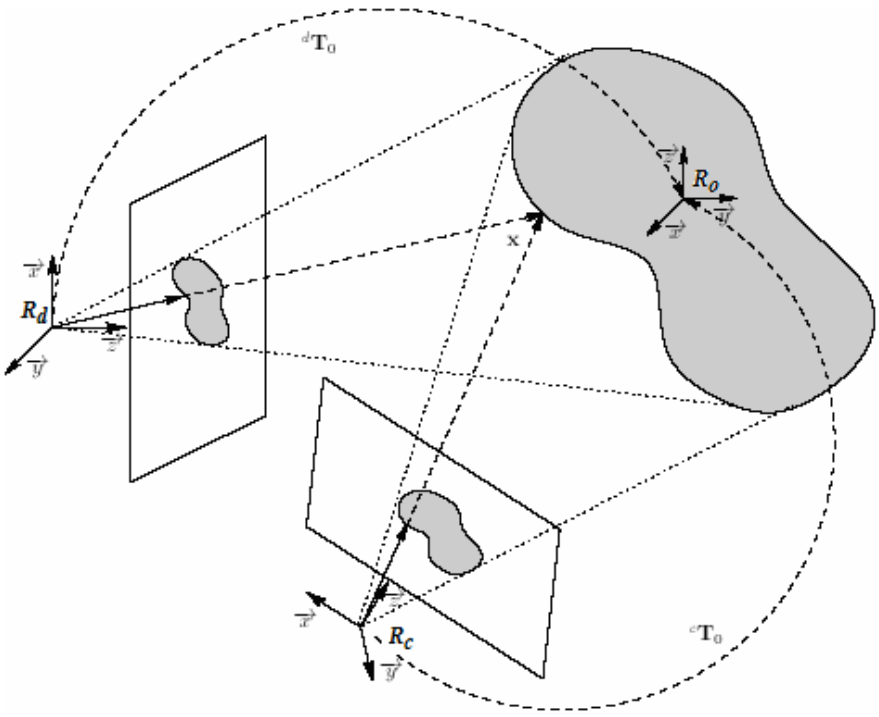


Figure 4.2. *Modélisation de l'asservissement visuel 3D*

En connaissant les coordonnées, exprimées dans R_0, d'au moins quatre points de l'objet (c'est-à-dire que le modèle 3D de l'objet, représenté par un vecteur de paramètres géométriques est supposé être connu), il est possible, à partir de leur projection dans l'image, de calculer la position et orientation désirées et courantes de la caméra (paragraphe 2.3).

A partir de localisation 3D, on dispose non seulement du déplacement de la caméra mais aussi des coordonnées courantes x et désirées x^* des points de la cible exprimées respectivement dans le repère courant R_c et désiré R_d de la caméra. Afin de contrôler l'effecteur du robot, plusieurs façons de procéder ont été proposées. Toutes sont basées sur le contrôle direct de la rotation car, comme cela a déjà été évoqué dans le paragraphe 2.4, à partir de la matrice de rotation ${}^cR_d$, on peut calculer aisément une représentation minimale [KHA 99] : $e_\omega(u,\theta) = u\theta$ où u et θ sont respectivement l'axe et l'angle de rotation. En ce qui concerne la translation, on utilise directement ${}^cP_d$.

Si la caméra n'est pas parfaitement calibrée, ou s'il existe des erreurs sur le modèle 3D de la cible, les positions courantes et désirées de la caméra ne seront pas correctement estimées et présenteront un biais par rapport à leur valeur réelle. Ces erreurs d'estimation sont compensées par une commande en boucle fermée et il n'y a alors pas d'inconvénients à la convergence. Toutefois, les erreurs de calibrage influencent grandement la manière dont le système arrive à la convergence, surtout quand le déplacement de la caméra est important.

4.3. Régulation de la fonction de tâche [MAL 98] :

La notion de fonction de tâche ayant déjà été abordée au premier chapitre, nous passerons donc à sa régulation dans le cas d'une caméra embarquée observant l'objet ; la loi de commande élaborée doit faire tendre cette fonction vers zéro. Rappelons la fonction de tâche dans le cas d'un asservissement visuel 3D :

$$e(q) = r(s(q)) - r(s^*). \tag{4.1}$$

r est l'attitude d'un repère lié à l'objet vu par la caméra par rapport à un repère lié à l'organe terminal du robot, cette attitude est estimée à partir des informations visuelles *s(t).*

Intéressons nous tout d'abord à réaliser simplement une décroissance exponentielle de la fonction de tâche, à savoir :

$$\dot{e} = -g.e \quad , \quad g : scalaire > 0 \tag{4.2}$$

La dérivée de la fonction de tâche peut être obtenue à partir de l'équation :

$$\dot{e} = \frac{\partial e}{\partial r_c}.v_c + \frac{\partial e}{\partial t} \tag{4.3}$$

$$= \frac{\partial e}{\partial r_c}.\frac{\partial r_c}{\partial r_r}.\dot{r}_r + \frac{\partial e}{\partial t} \tag{4.4}$$

$$= \frac{\partial e}{\partial r_c}.\frac{\partial r_c}{\partial r_r}.\frac{\partial r_r}{\partial q}.\dot{q} + \frac{\partial e}{\partial t} \tag{4.5}$$

Où :

- $\frac{\partial e}{\partial t}$: la variation de *e* due au mouvement propre (et généralement inconnu) de l'objet ;

- $\dot{r}_r$ torseur cinématique du robot ;

- $L = \frac{\partial e}{\partial r_c}$: la matrice d'interaction qui lie la variation de la fonction de tâche par rapport à *la vitesse v_c de la caméra*. Comme la fonction de tâche est construite à partir des informations visuelles, cette matrice peut être aussi écrite $L = \frac{\partial e}{\partial s}\frac{\partial s}{\partial r_r}$ mettant ainsi en évidence sa dépendance avec *s*. Cette matrice dépend généralement des paramètres intrinsèques de la caméra et de certains paramètres géométriques de la scène observée.

- $W = \frac{\partial r_c}{\partial r_r}$: la matrice exprimant le passage entre la vitesse de la caméra et la vitesse de l'effecteur du robot :

$$v_c = W\,\dot{r}_r \tag{4.6}$$

Cette matrice est constante si la caméra est rigidement liée à l'organe terminal du robot ; elle contient la rotation ${}^cR_e$ et la translation ${}^cP_e$ entre le repère caméra R_c et le repère de l'effecteur R_e, tel que :

$$W = \begin{bmatrix} I & [{}^cP_e]_x \\ 0 & I \end{bmatrix} \begin{bmatrix} {}^cR_e & 0 \\ 0 & {}^cR_e \end{bmatrix} = \begin{bmatrix} {}^cR_e & [{}^cP_e]_x\,{}^cR_e \\ 0 & {}^cR_e \end{bmatrix} \tag{4.7}$$

- $J(q) = \frac{\partial r_r}{\partial q}$: la matrice jacobienne du robot exprimant le passage entre la vitesse de l'effecteur et la vitesse articulaire du robot :

$$\dot{r}_r = J(q).\dot{q} \tag{4.8}$$

Nous supposerons par la suite que le jacobien du robot est parfaitement connu et inversible. Si l'objet est immobile c'est-à-dire $\frac{\partial e}{\partial t} = 0$, $\dot{r}_r$ aurait donc la forme suivante :

$$\dot{r}_r = -g.W.^{-1}L.^{-1}.e = -g.(L.W)^{-1}.e \tag{4.9}$$

Avec :

$$\dot{e} = L.W.\dot{r}_r \tag{4.10}$$

Si l'on souhaite utiliser les variations articulaires comme signaux de commande, en considérant l'attitude désirée fixe et l'objet immobile, la loi de commande s'écrit :

$$\dot{q}_d = -g.{}^{n}J^{-1}{}_{n}(q).(LW)^{-1}.e \tag{4.11}$$

Où ${}^{n}J_{n}(q)$ est le *jacobien du robot* exprimé dans le repère R_n de son organe terminal [KHA 99].

4.3.1. Estimation du paramètre $\frac{\partial e}{\partial t}$:

Comme cela a déjà été évoqué au paragraphe (3.4.2), le mouvement de l'objet est considéré comme une perturbation à éliminer le plus rapidement possible. La commande de la boucle de vision doit donc d'être robuste vis-à-vis des perturbations. Ce sont ces méthodes que nous testerons dans la suite de ce mémoire. En effet, en remarquant que toutes les perturbations agissant sur le système (couples résistant sur les actionneurs, erreurs de calibrage, erreurs de modélisation et même le mouvement propre de l'objet) ont tous des conséquences dans l'image, il serait plus judicieux de ne concevoir qu'un régulateur robuste vis-à-vis des perturbations, dans la boucle de vision.

Il s'agira donc de commander le torseur cinématique, noté *T*, entre la caméra et son environnement. Elle est donnée selon (3.9) :

$$T = v_c = \begin{pmatrix} V \\ \omega \end{pmatrix} = W.\dot{r}_r = W \cdot {}^{n}J_n(q) \cdot \dot{q} \tag{4.12}$$

Selon (4.11) et en considérant le mouvement de l'objet comme une perturbation au système, la loi de commande de la boucle de vision s'écrit :

$$T_d = -g.\hat{L}^{-1} \cdot e \tag{4.13}$$

4.3.2. Matrice d'interaction d'information 3D [KHA 02] :

Si l'on choisit de réguler la fonction de tâche suivante :

$$e = \begin{bmatrix} {}^{c}P_o - {}^{d}P_o \\ {}^{d}\theta u_c \end{bmatrix} \tag{4.14}$$

Avec :

$${}^{c}T_o = \begin{bmatrix} ({}^{c}R_o)_{3\times3} & ({}^{c}P_o)_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}_{R_c} \tag{4.15}$$

- ${}^{c}T_{o}$ est la matrice homogène de transformation entre le repère courant de la caméra R_c et le repère objet R_o, exprimée dans le repère caméra R_c ;

- ${}^{d}T_{o}$ est la matrice homogène de transformation entre le repère désiré de la caméra R_d et le repère objet R_o, exprimée dans le repère caméra R_c :

$$ {}^{d}T_{o} = \begin{bmatrix} ({}^{d}R_{o})_{3\times3} & ({}^{d}P_{o})_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}_{R_c} \tag{4.16} $$

- ${}^{d}\theta u_{c}$ est la représentation minimale (2.28) de la matrice ${}^{d}T_{c}$ qui représente la transformation entre le repère désiré de la caméra R_d et le repère courant de cette dernière R_c, exprimée dans le repère caméra R_c, avec :

$$ {}^{d}T_{c} = {}^{d}T_{o} \cdot {}^{c}T_{o}^{-1} \tag{4.17} $$

Les matrices de transformation ${}^{d}T_{o}$ et ${}^{c}T_{o}$ sont obtenues en utilisant le calcul de pose issu des informations visuelles $s(t)$ et s_d (paragraphe 2.3).

La matrice d'interaction associée à la fonction de tâche choisie est [KHA 02] :

$$ L = \begin{bmatrix} -I_3 & [{}^{c}P_{o}]_{X} \\ 0_3 & J_{c\omega} \end{bmatrix} \tag{4.18} $$

Avec :

$$ J_{c\omega} = I_3 - \frac{\theta}{2}[u]_x + (1 - \frac{\sin c\theta}{\sin c^2 \frac{\theta}{2}})[u]_x^{\,2} \tag{4.19} $$

Où : θ et u sont les éléments de la représentation minimale de la rotation ${}^{c}R_{o}$ de la matrice de transformation homogène ${}^{c}T_{o}$.

La condition de stabilité de la matrice d'interaction suffisante pour assurer la décroissance de $\|e\|$ dans tout l'espace de travail est (3.23) :

$$ L\hat{L}^{-1} = LL^{-1} = I_6 > 0 \tag{4.20} $$

4.3.3. Schéma de simulation d'une boucle d'asservissement visuel 3D :

Les relations (4.12), (4.13) et (4.10) nous permettent de tracer le schéma de simulation de la boucle de commande 3D :

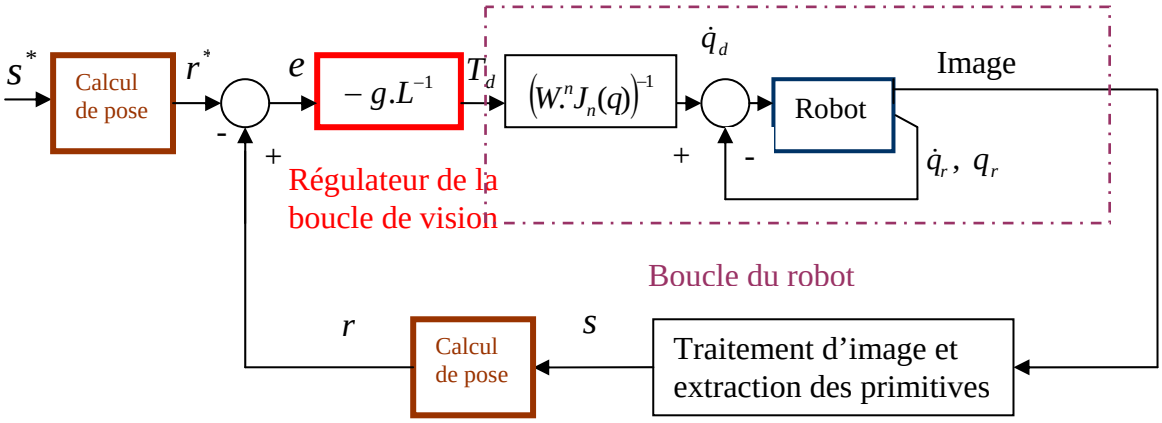


Figure 4.3. *Schéma bloc d'un asservissement visuel 3D*

Le bloc associé au robot, ainsi que quelques commandes sont décrits dans le paragraphe (3.4.3), nous décrirons à présent d'autres commandes qu'on va utiliser dans ce type d'asservissement visuel.

4.3.4. Commande du robot :

Comme il a été mentionné dans le paragraphe 3.3, la loi de commande choisie pour le robot doit assurer la stabilité de ce dernier, ainsi que ses performances dans le cas d'une consigne variable. Différentes méthodes de régulation existent, nous choisirons la *commande par Backsteeping*, déjà développée dans l'annexe B :

$$\Gamma = H(q,\dot{q}) + A(q)\left[\ddot{q}_d + (\dot{q} - \dot{q}_d)(\alpha + \beta) - (q - q_d)(1 + \alpha\beta)\right] \qquad (4.21)$$

Ainsi que la *commande linéarisante* pour commander les *vitesses articulaires* avec utilisation du retour d'état pour le système linéarisé dont le développement et aussi décrits dans l'annexe B :

$$\Gamma = H(q,v) + A(q)\ \mathcal{V} \qquad \text{Avec : } \mathcal{V} \text{ la commande linéarisante} \qquad (4.22)$$

La commande linéarisante est aussi utilisée pour commander les *variables articulaires* (annexe B) :

$$\Gamma = H(q,v) + A(q)\ \mathcal{V} \qquad \text{avec } \mathcal{V} \text{ la commande linéarisante} \qquad (4.23)$$

L'utilisation de *la commande par mode de glissement* s'avère aussi intéressante pour commander le robot. Son expression est donnée ci-dessous (annexe B) :

$$\Gamma = H(q,\dot{q}) + A(q)\cdot\left(-\lambda\cdot\dot{q} - K\cdot Sat(S(X))\right) \quad K > 0 \quad \lambda > 0 \qquad (4.24)$$

4.4. Simulations de l'asservissement visuel 3D :

Après avoir décrit la boucle de commande du robot et celle de l'asservissement visuel, passons à la simulation de ces derniers, disposés en cascade comme décrit dans la figure 4.3. Notons que la structure en cascade impose à la boucle de la commande du robot d'être plus *rapide* que celle de l'asservissement visuel. Ce qui impose pour la commande par Backsteeping de prendre les valeurs des paramètres α, β assez grands, mais les commandes résultantes doivent être admissibles. Quant à la commande linéarisante, il faut veiller à bien choisir les valeurs propres du retour d'état. Ainsi que le bon choix des valeurs de λ et K pour la commande par mode de glissement.

L'environnement de simulation a déjà été décrit dans le paragraphe (3.4.4) ainsi que toutes les lois de commandes utilisées pour la boucle de vision.

Vu que la boucle de commande du robot *PUMA* 560 et celle de l'asservissement visuel sont en cascade, (voir figure 4.3), il sera préférable de garder la même loi de commande pour une boucle et de varier l'autre et vice versa afin d'étudier l'influence de chaque loi de commande sur les réponses. Les simulations seront donc effectuées de la manière suivante :

1er cas : Loi de la boucle de robot invariante : Backsteeping

$$\text{Boucle de vision : } \begin{cases} g \text{ constent} \\ g \text{ adaptatif} \\ \text{glissement} \end{cases}$$

2^{em} cas : Loi de la boucle de vision invariante : g adaptatif

$$\text{Boucle du robot : } \begin{cases} \text{le glissement} \\ \text{linéarisation en vitesses articulaires} \\ \text{linéarisation en variables articulaires} \end{cases}$$

Nous aborderons ensuite l'effet de présence de perturbations. Finalement, on aura à étudier le problème de suivi de cible.

4.4.1. Variation dans les commandes visuelles :

Nous choisirons le Backsteeping comme loi de commande de la boucle du robot *PUMA* 560, cette dernière a déjà été donnée dans le paragraphe (4.3.5), les constantes $\alpha = \beta = -15$.

4.4.1.1. Utilisation de la commande visuelle classique avec g constant :

Simulons le comportement du robot pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec g =0.3 s^{-1} :

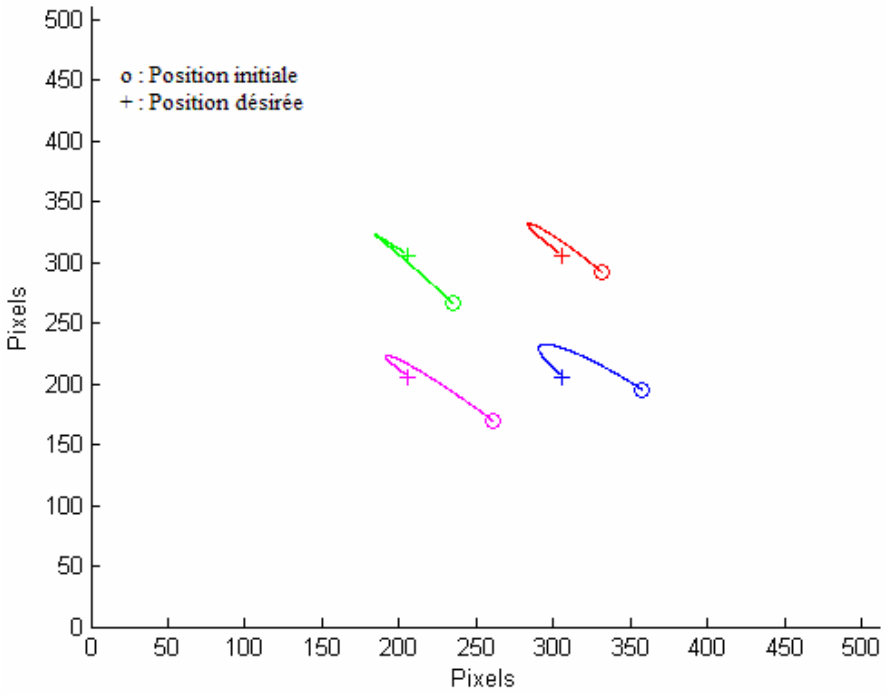


Figure 4.4.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15°, avec g=0.3 (s^{-1}).*

Nous avons une bonne convergence vers l'image désirée avec des trajectoires courbées dues aux limitations physiques du système et le type d'asservissement visuel utilisé. Nous donnons à présent le tracé des variables articulaires du robot, puis celui des erreurs en position et orientation ainsi que leurs normes.

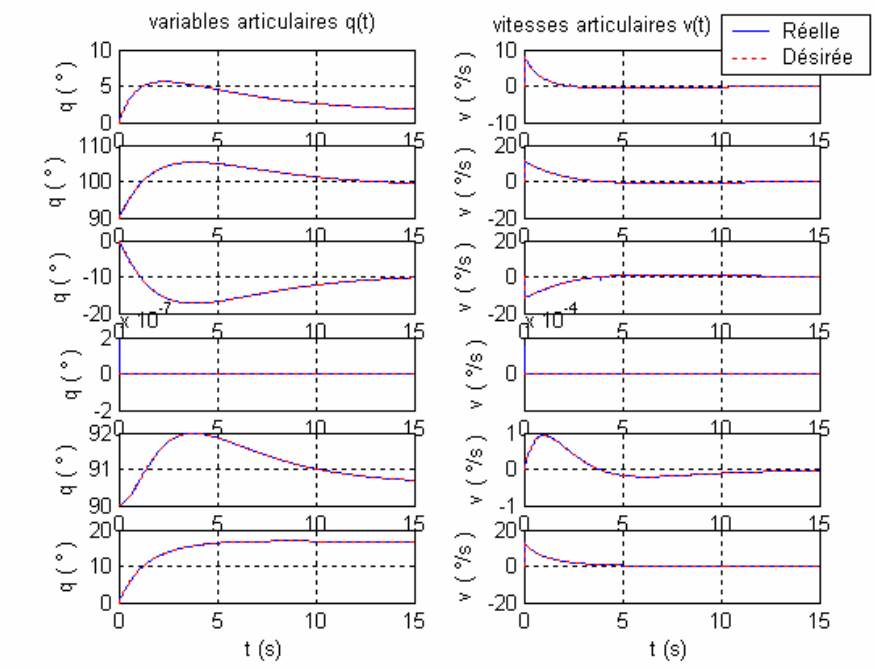


Figure 4.4.b. *Variables et vitesses articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15°, avec g=0.3 (s^{-1}).*

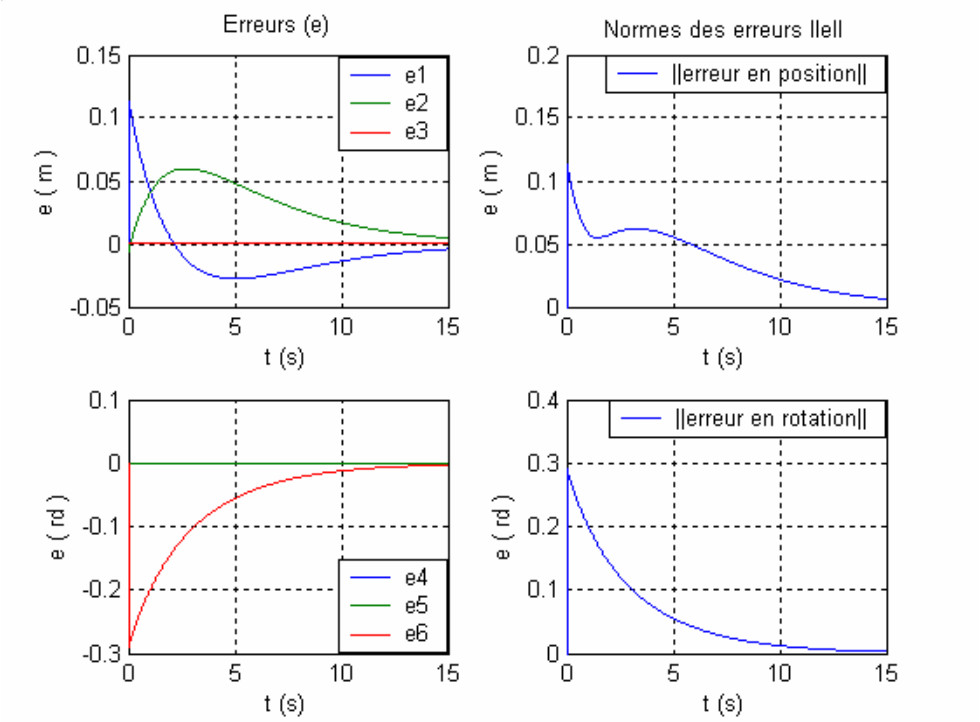


Figure 4.4.c. *Erreurs pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15° avec g=0.3(s^{-1}).*

La convergence s'effectue en 15s ! Ce qui est trop lent pour une tâche de positionnement.

Le tracé des différents couples appliqués à chaque articulation est le suivant :

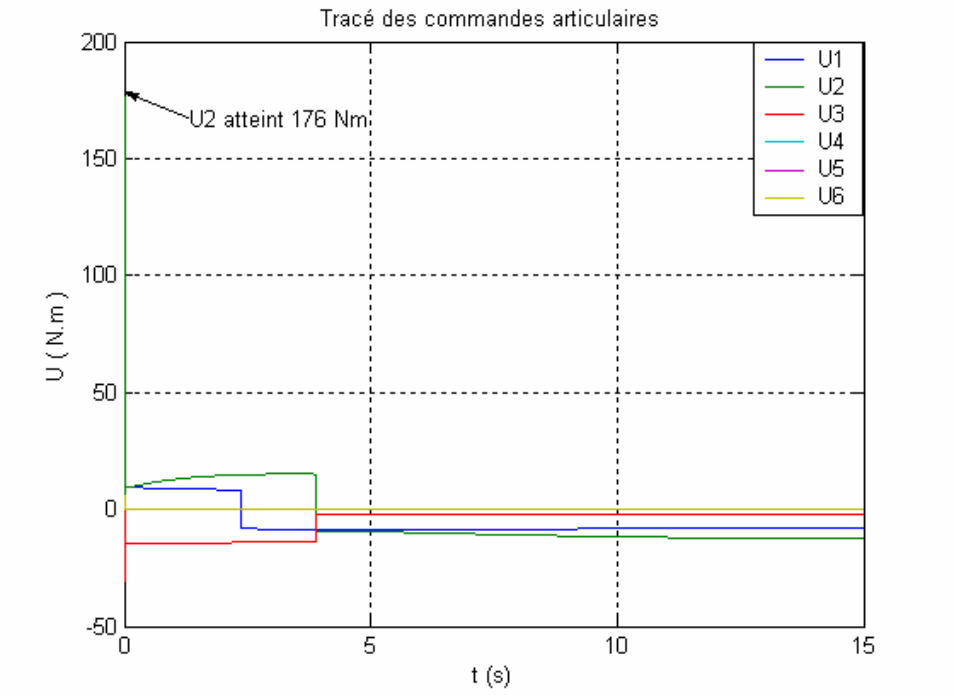


Figure 4.4.d. *Couples articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15° avec g=0.3(s^{-1}).*

Ce tracé révèle une forte commande sur la seconde articulation à l'instant initial ; celle-ci atteint 176 N.m, ce qui est proche de sa limite supérieure : 186.4 N.m, selon le tableau (3.1). La cause de ce pic à l'instant initial est expliquée dans les relations (3.40) et (3.14).

On déduit d'après tous ces résultats que l'application de *g* constant en asservissement visuel 3D, n'est pas très intéressante si l'on souhaite que l'erreur converge rapidement avec des commandes admissibles.

4.4.1.2. Utilisation de la commande visuelle classique avec g adaptatif :

Comme il a déjà été expliqué au paragraphe (3.5.2), l'utilisation de *g* adaptatif permet d'éviter la forte impulsion à l'instant initial, et de réduire le temps de convergence du système.

Testons à présent cette méthode pour différents cas de figures. Prenons comme premier cas une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec $g_{max} = 4\ s^{-1}$ *et* $a = 1(s)$:

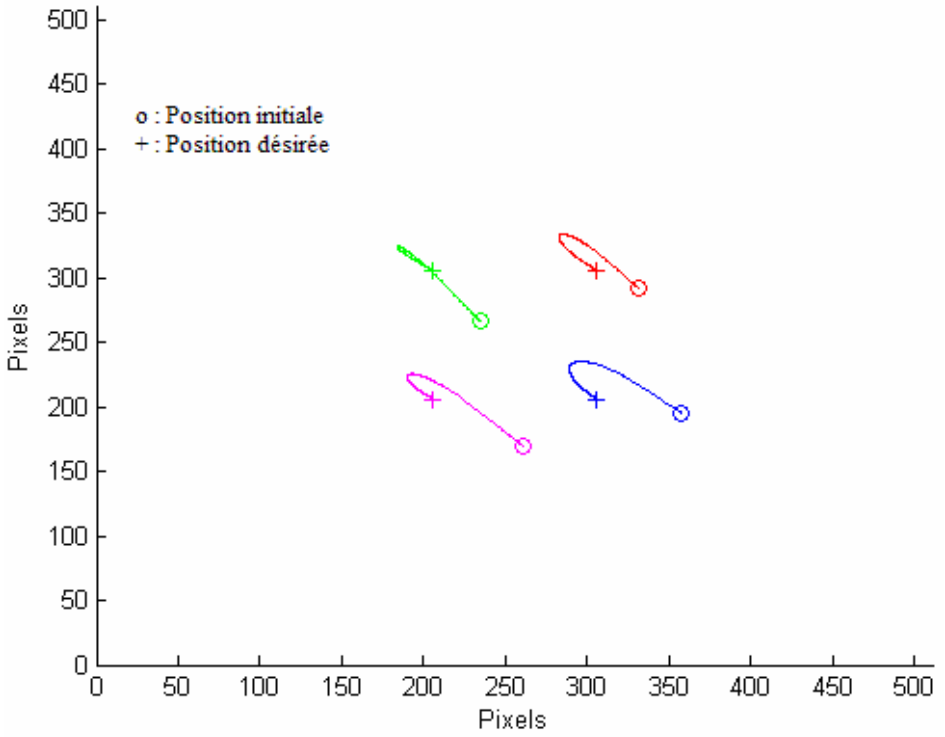


Figure 4.5.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15°, avec* $g_{max}=4\ (s^{-1})$ *et* $a=1(s)$.

On voit bien que la variation de *g* n'a pas influencé sur les trajectoires de la convergence vers l'image désirée.

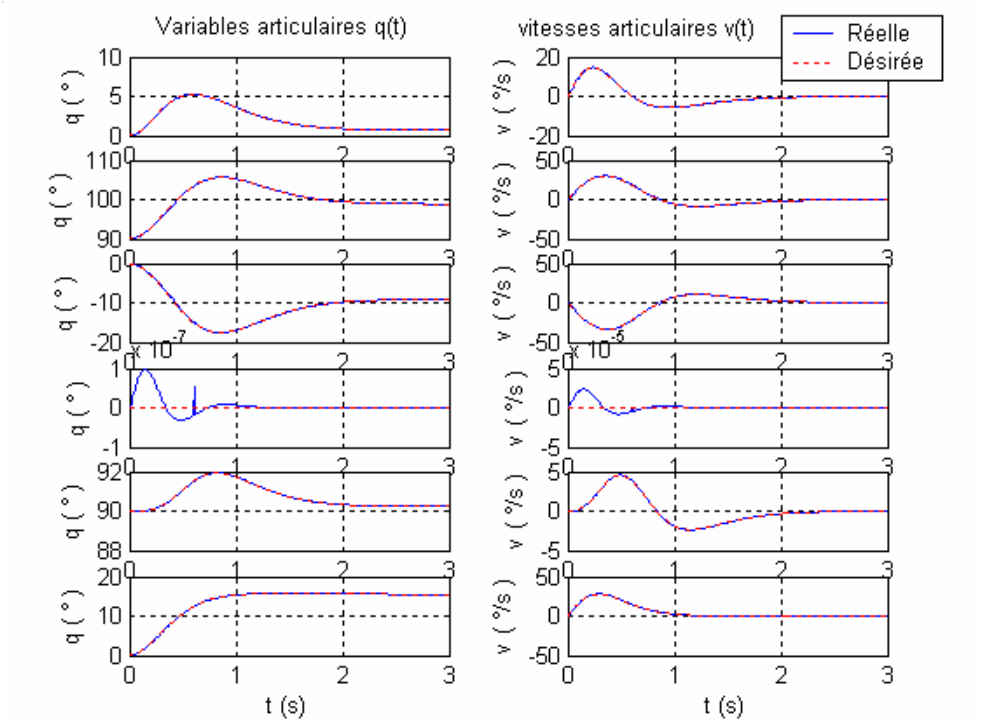


Figure 4.5.b. *Variables et vitesses articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15°, avec* $g_{max}=4\ (s^{-1})$ *et a=1(s).*

Les variables articulaires ainsi que leurs vitesses sont sensibles aux variations du paramètre *g*. Cette sensibilité se voit sur le temps de convergence et sur leurs amplitudes.

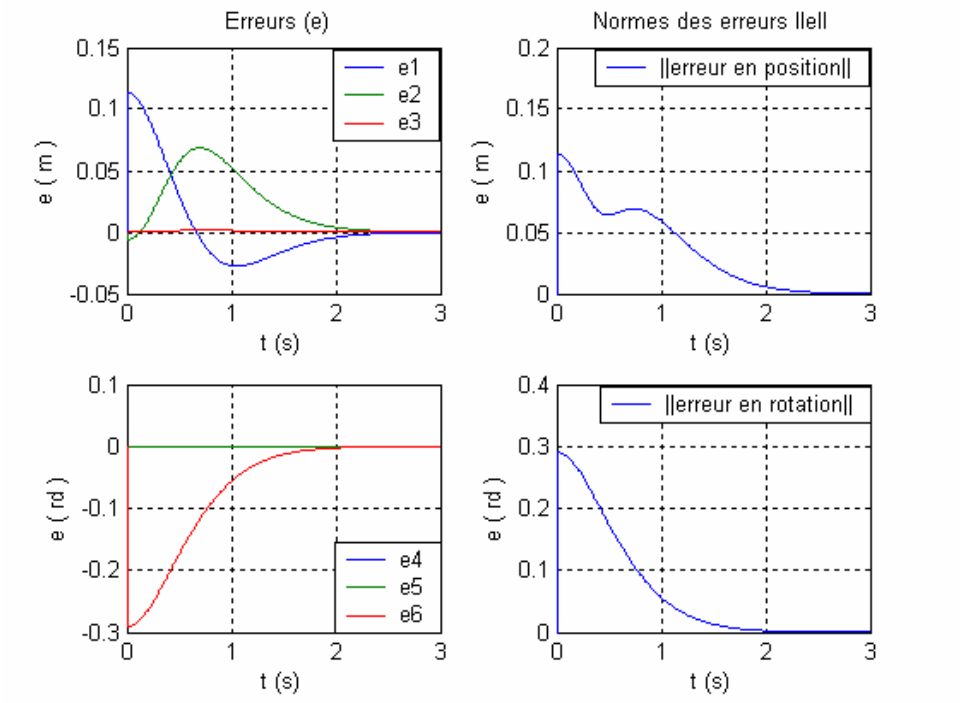


Figure 4.5.c. *Erreurs pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15°, avec* $g_{max}=4\ (s^{-1})$ *et a=1(s).*

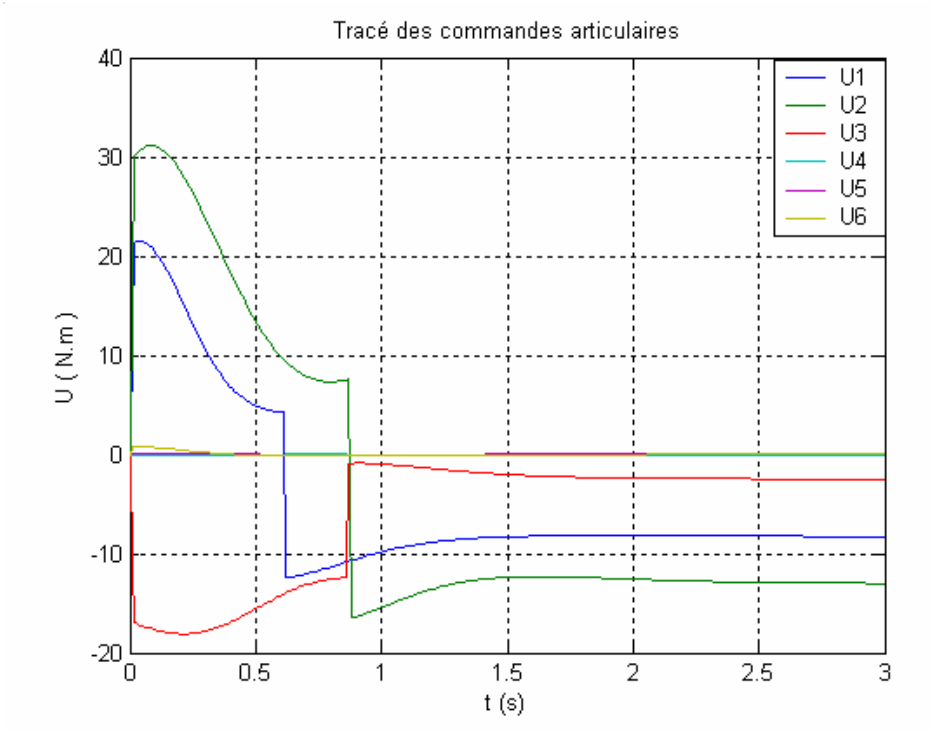


Figure 4.5.d. *Couples articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15°, avec g_{max}=4 (s^{-1}) et a=1(s).*

Contrairement au cas précédent, toutes les commandes sont admissibles par le système, malgré la présence d'un pic initial, le temps de convergence du robot est nettement diminué à 3s.

Prenons comme deuxième cas une translation de 50 pixels selon *u* et -100 pixels selon *v* avec g_{max} =4 s^{-1} *et a=3(s)* :

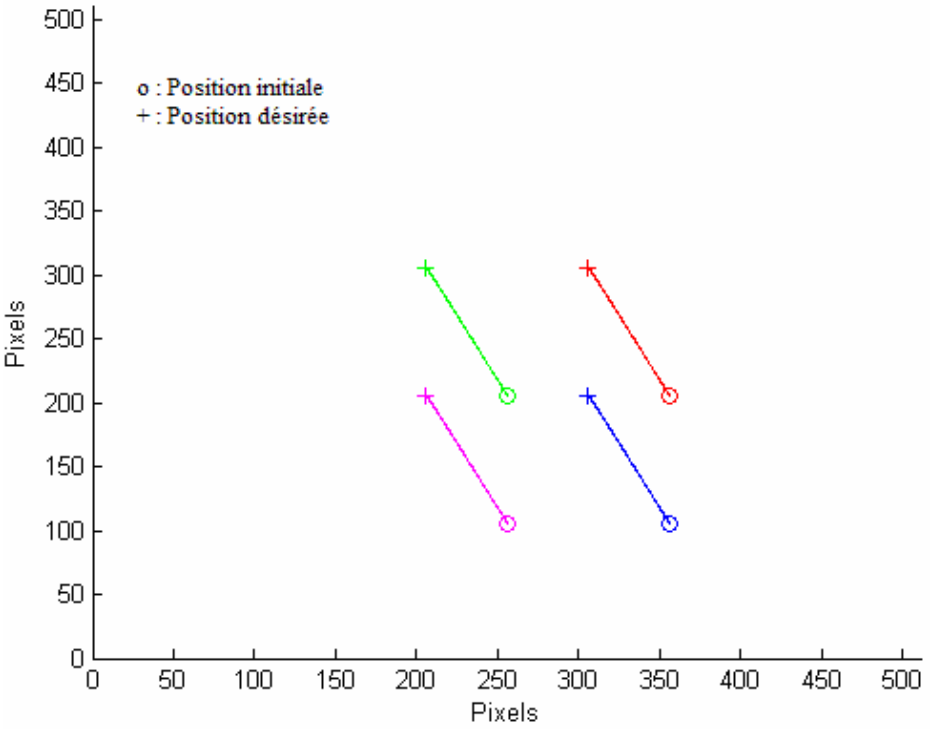


Figure 4.6.a. *Trajectoire dans l'image des primitives visuelle pour une translation de 50 pixels sur u et -100 pixels sur v, avec g_{max}=4 (s^{-1}) et a=1(s).*

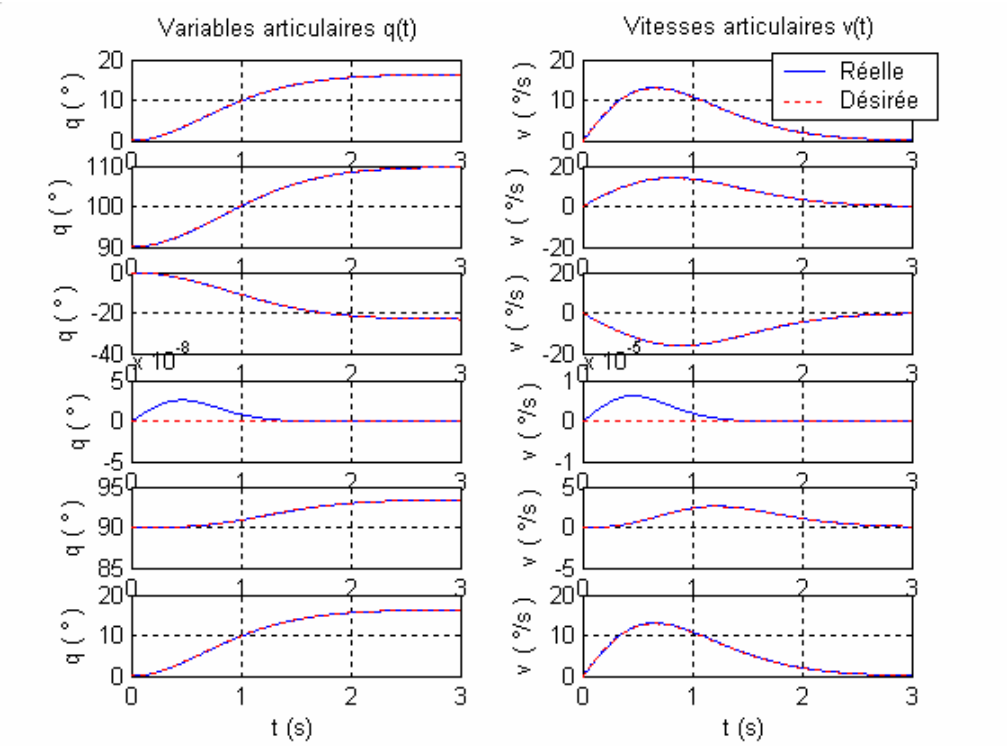


Figure 4.6.b. *Variables et vitesses articulaires pour une translation de 50 pixels sur u et -100 pixels sur v, avec g_{max}=4 (s^{-1}) et a=1(s).*

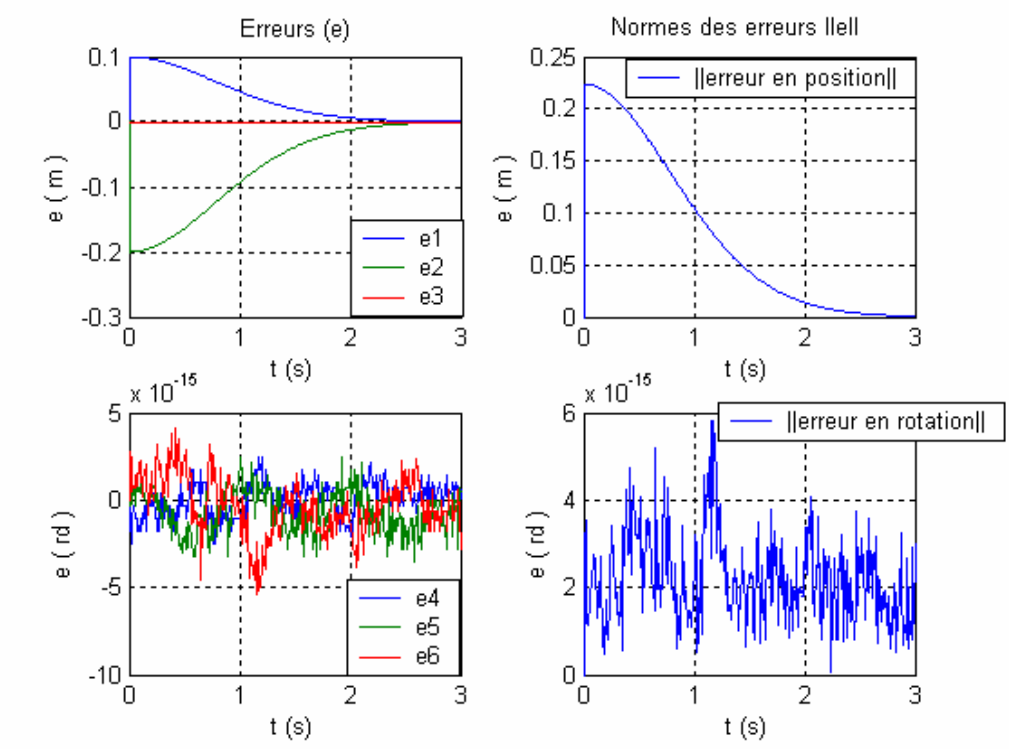


Figure 4.6.c. *Erreurs pour une translation de 50 pixels sur u et -100 pixels sur u, avec g_{max}=4 (s^{-1}) et a=1(s).*

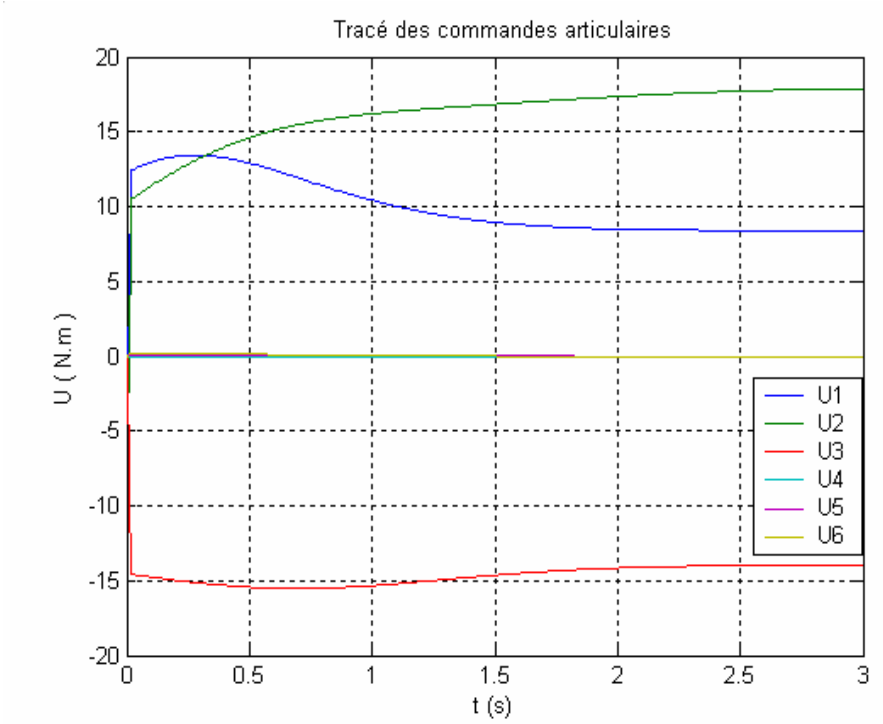


Figure 4.6.d. *Couples articulaires pour une translation de 50 pixels sur u et -100 pixels sur v, avec g_{max}=4 (s^{-1}) et a=1(s).*

Remarquons que le positionnement s'est correctement effectué en 3s comme dans le cas précédent. Les erreurs en rotation du robot sont pratiquement nulles de l'ordre de 10^{-15}rd car on n'a pas eu une rotation dans l'image mais des translations sur *u* et *v*. Les commandes sont admissibles et ne dépassant pas 20 N.m.

4.4.1.3. Utilisation de la commande visuelle par mode de glissement avec g adaptatif :

L'élaboration de cette loi de commande a été faite dans le chapitre précédent dans le paragraphe 3.5.4. où on propose d'annuler la fonction de tâche *e* au lieu de lui imposer une décroissance exponentielle en gardant *g* adaptatif. Testons cette méthode en prenant une translation de 50 pixels sur *u* et 100 pixels sur *v* avec g_{max} *=1.1* s^{-1} *et a=3(s)* :

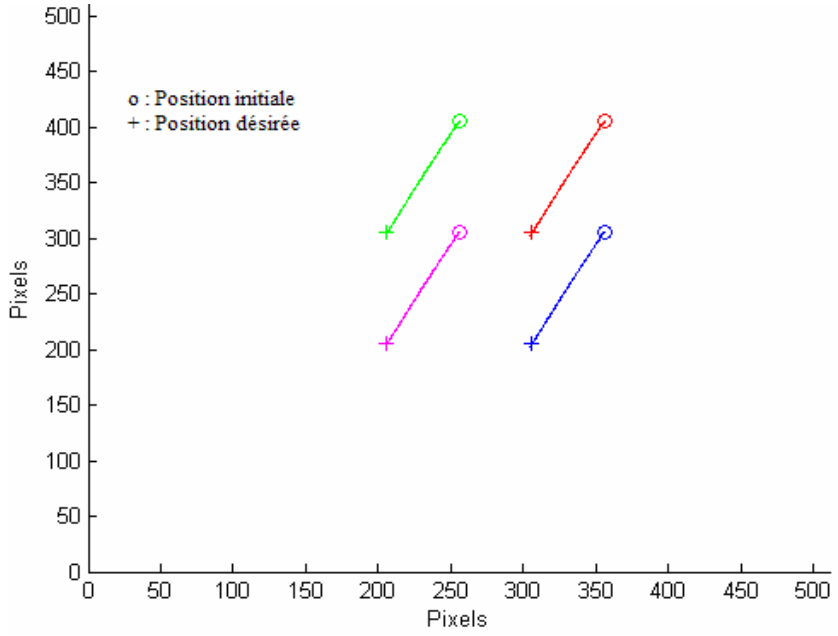


Figure 4.7.a. *Trajectoire dans l'image des primitives visuelle pour une translation de 50 pixels sur u et 100 pixels sur v, avec g_{max}=1.1(s^{-1}) et a=3(s).*

On remarque qu'il y a eu une bonne convergence vers l'image désirée avec des trajectoires rectilignes car on a que des translations dans l'image.

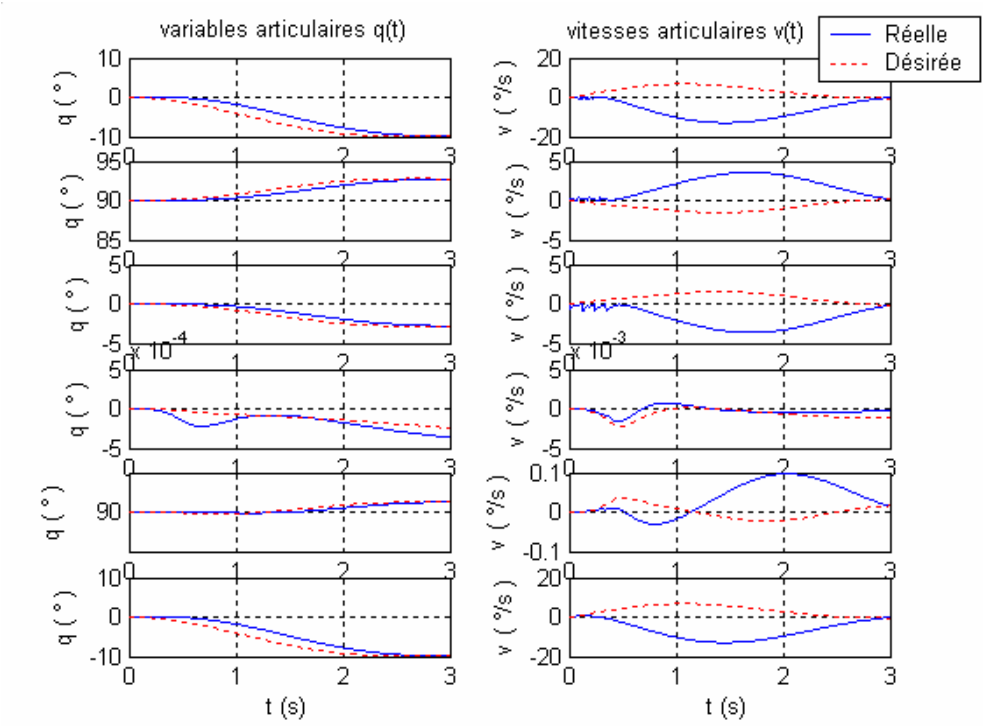


Figure 4.7.b. *Variables et vitesses articulaires pour une translation de 50 pixels sur u et 100 pixels sur v, avec $g_{max}=1.1(s^{-1})$ et a=3(s).*

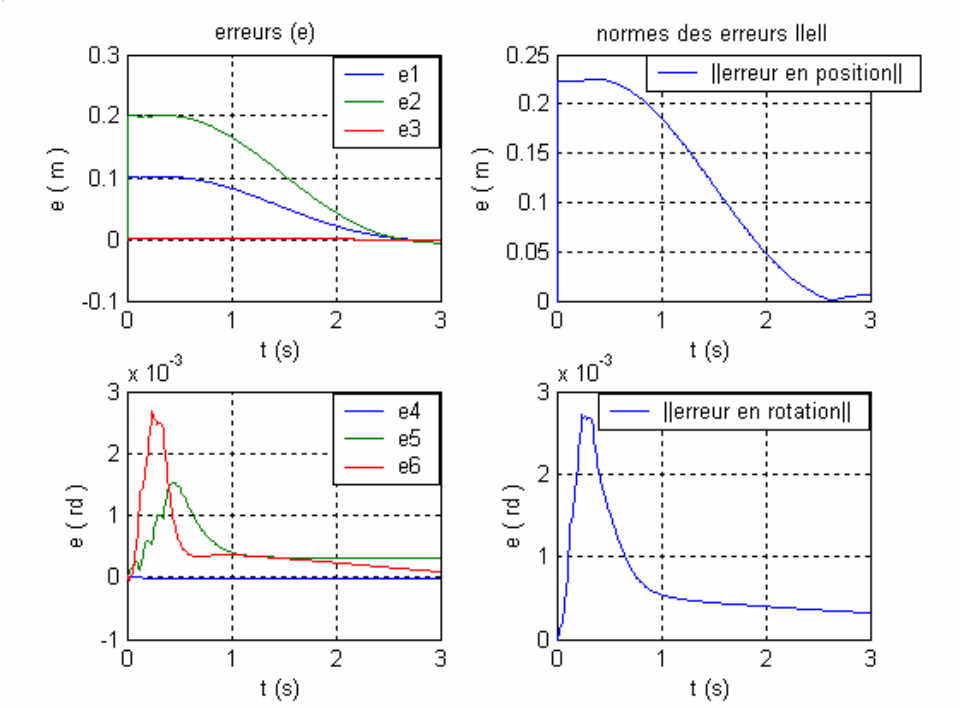


Figure 4.7.c. *Erreurs pour une translation de 50 pixels sur u et 100 pixels sur v, avec $g_{max}=1.1(s^{-1})$ et a=3(s).*

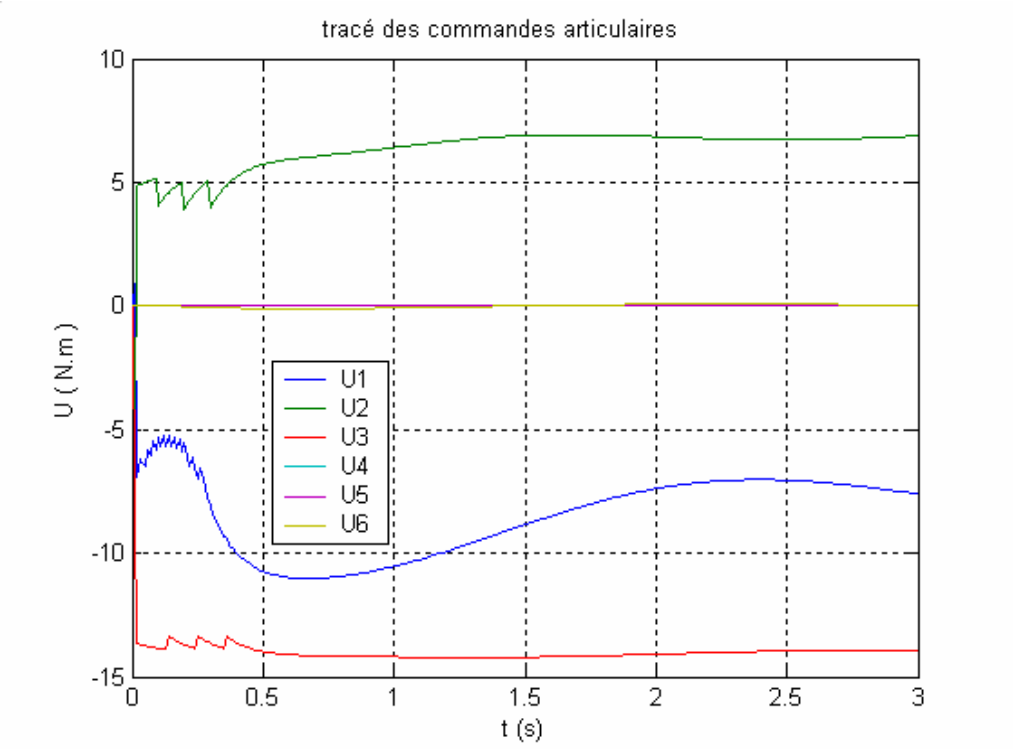


Figure 4.7.d. *Couples articulaires pour une translation de 50 pixels sur u et 100 pixels sur v, avec* $g_{max}=1.1(s^{-1})$ *et a=3(s).*

Comme dans la simulation précédente, cette loi de commande ne permet pas une bonne poursuite en vitesses articulaires ; les erreurs en rotations sont de l'ordre de (10^{-3})rd car on a que des translations dans l'image et le temps de convergence est toujours de l'ordre de 3s. Quant aux commandes, elles sont admissibles et sans pics ne dépassant pas 15 N.m ; ces commandes présentent quelques broutements aux instants initiaux à cause de la commande par mode de glissement utilisée pour la boucle de vision.

D'après les résultats obtenus par toutes ces simulations, on déduit que *la commande classique avec un g variable* est la plus convenable pour ce type d'asservissement visuel, car la convergence vers l'image désirée s'effectue d'une façon rapide avec des poursuites en articulations ainsi que leurs vitesses, les commandes dans tous les cas sont admissibles et ne présentant pas de très fortes impulsions.

4.4.2. Variation dans la commande de la boucle du robot:

Nous choisirons la loi classique avec un *g* adaptatif pour la boucle visuelle et on essayera de varier la commande de la boucle du robot PUMA 560.

4.4.2.1. Utilisation de la commande par mode de glissement :

La commande par mode de glissement utilisée a déjà été décrite dans le paragraphe (4.3.5). Testons cette méthode en prenons une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec $g_{max}=3.2\ s^{-1}$ *et* $a=2(s)$ pour la commande visuelle et $K=10(s^{-1})$ *et* $\lambda=1(s^{-1})$ pour la commande glissante du robot:

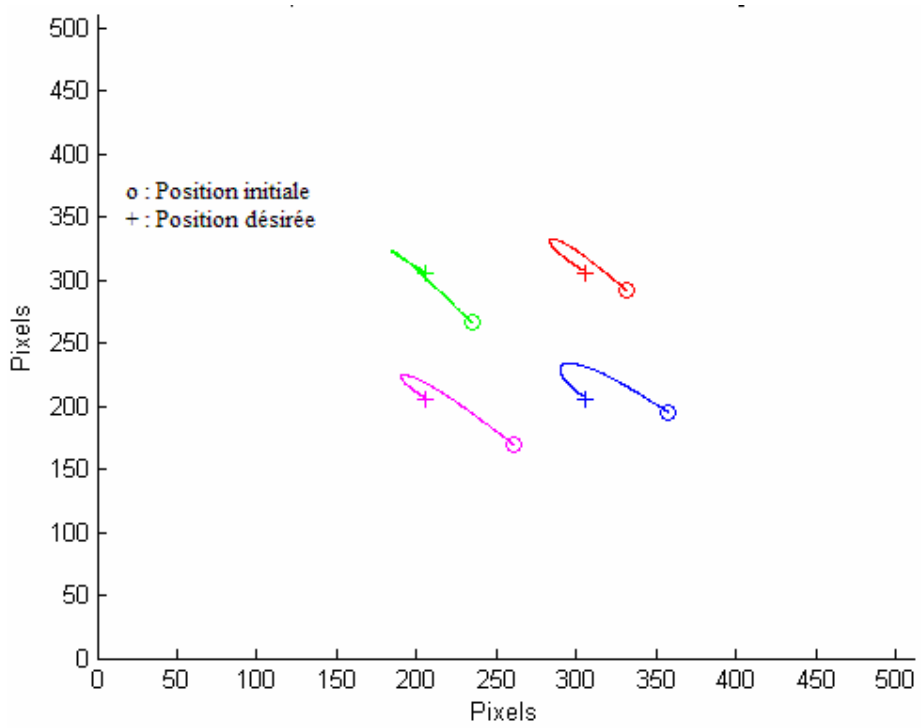


Figure 4.8.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec $g_{max}=3.2(s^{-1})$ et a=2(s).*

On a bel et bien convergence vers l'image désirée avec les mêmes trajectoires courbée trouvées dans le paragraphe précèdent dans le cas d'utilisation de la commande visuelle classique.

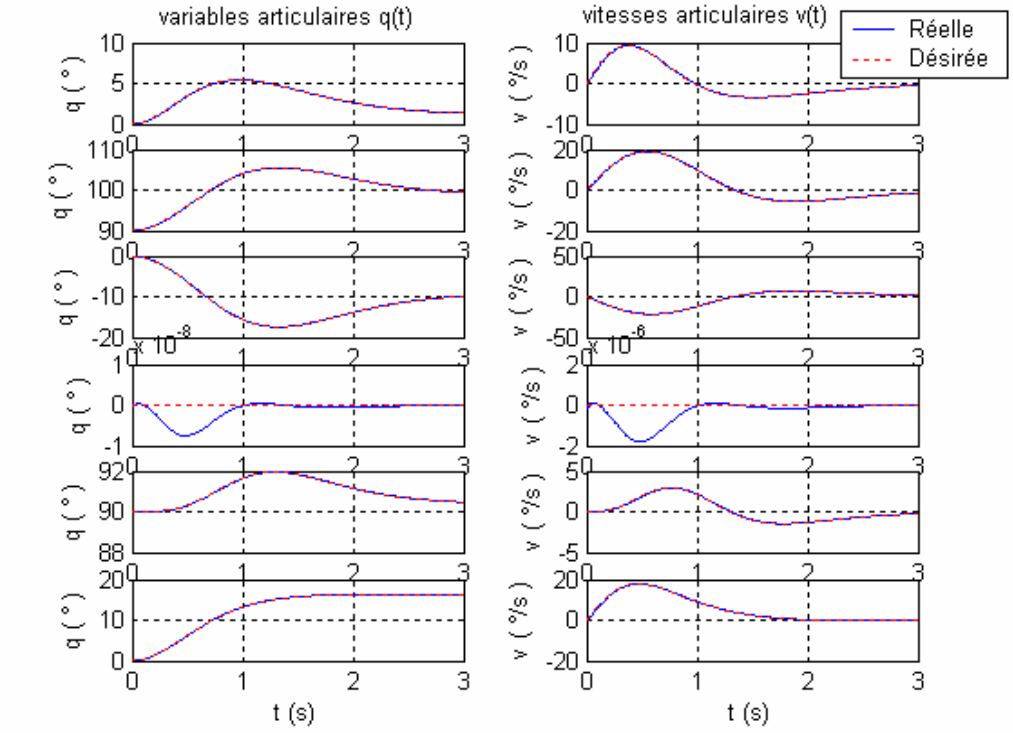


Figure 4.8.b. *Variables et vitesses articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec $g_{max}=3.2\ (s^{-1})$ et a=2(s).*

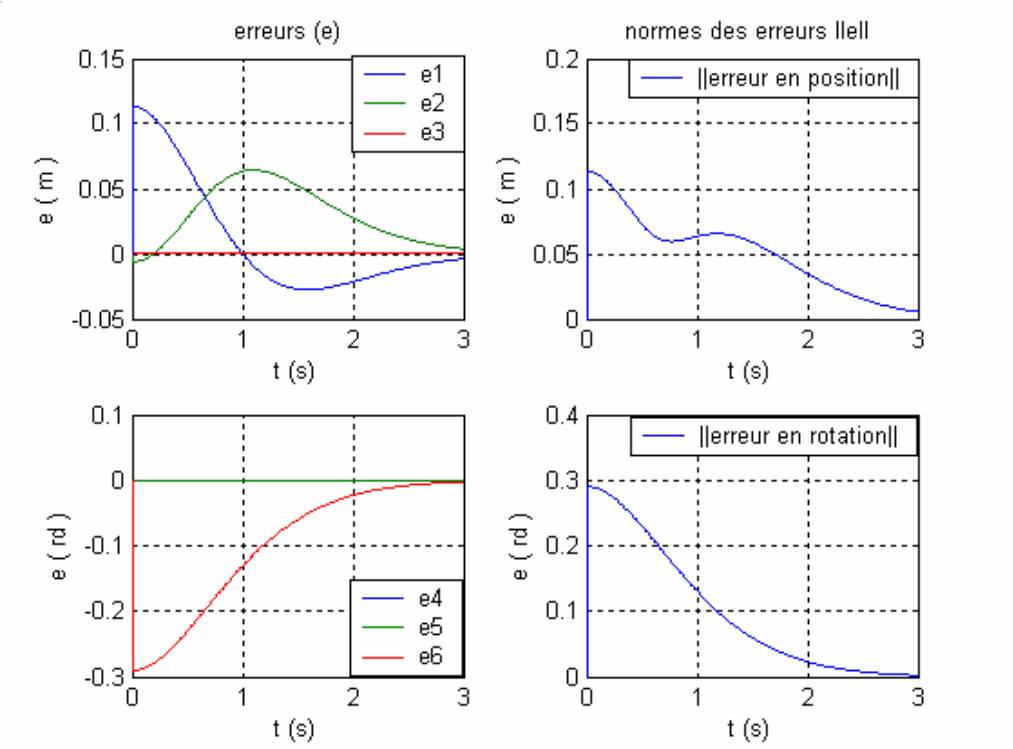


Figure 4.8.c. *Les erreurs pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec g_{max}=3.2 (s^{-1}) et a=2(s).*

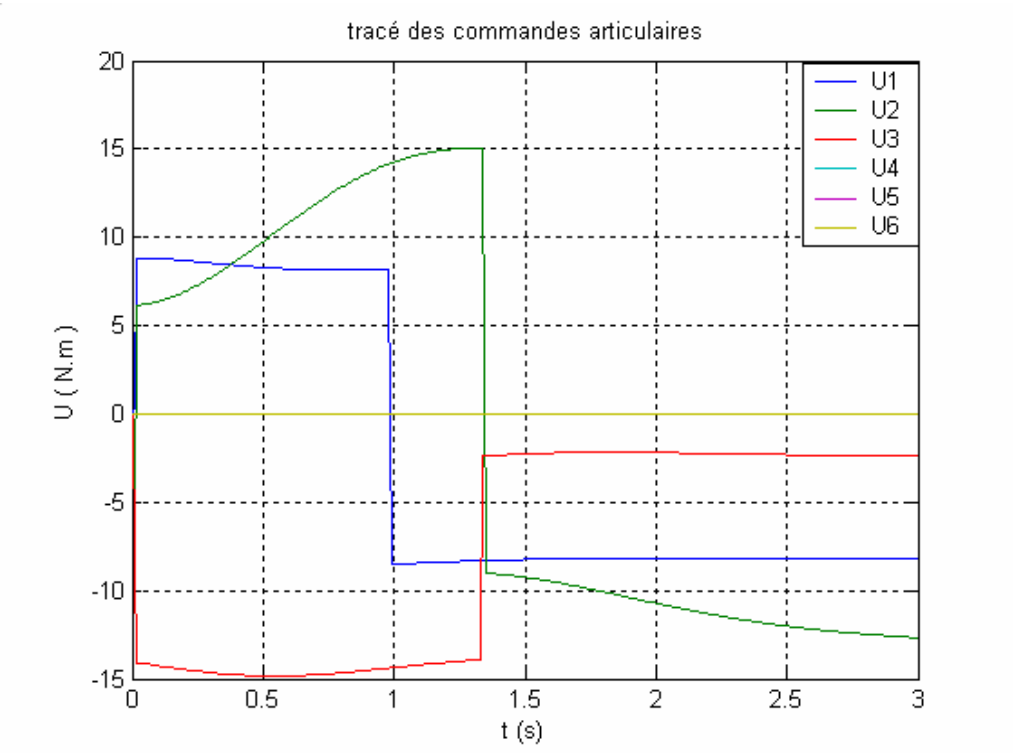


Figure 4.8.d. *Les couples articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec g_{max}=3.2 (s^{-1}) et a=2(s).*

On remarque qu'on a une convergence rapide (3s), et de bonnes poursuites en articulations et leurs vitesses. Les commandes sont admissibles ne dépassent pas les 15 N.m ;

contrairement à celles obtenues pour le même cas de figure en utilisant le Backsteeping où la commande atteint les 30 N.m ; mais dans les deux cas, les commandes sont loin d'atteindre les bornes supérieures du robot.

D'après ces résultats, on déduit que la commande par mode de glissement dans la boucle du robot possède les mêmes effets que celle par Backsteeping, soit de point de vue rapidité, énergie de la commande ou poursuite des articulations ainsi que leurs vitesses.

4.4.2.2. Utilisation de la commande linéarisante en vitesses articulaires, avec l'utilisation du retour d'état pour le système linéarisé :

La loi de commande utilisée a déjà été décrite dans le paragraphe (4.3.5). Testons cette loi de commande pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec g_{max} =3 s^{-1} et a=2s pour la commande visuelle, et les valeurs propres {-1 ;-2 ;-3 ; -4 ;-5 ;-6} pour la commande par retour d'état du système linéarisé en vitesse :

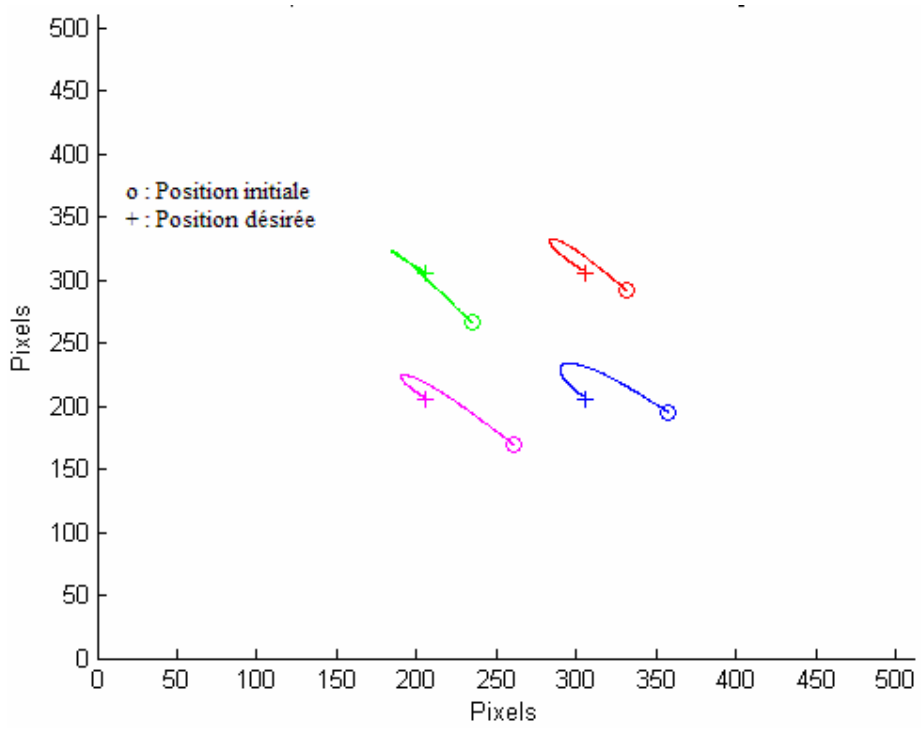


Figure 4.9.a. *Trajectoire dans l'image des primitives visuelles pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec $g_{max}=3(s^{-1})$ et a=2(s).*

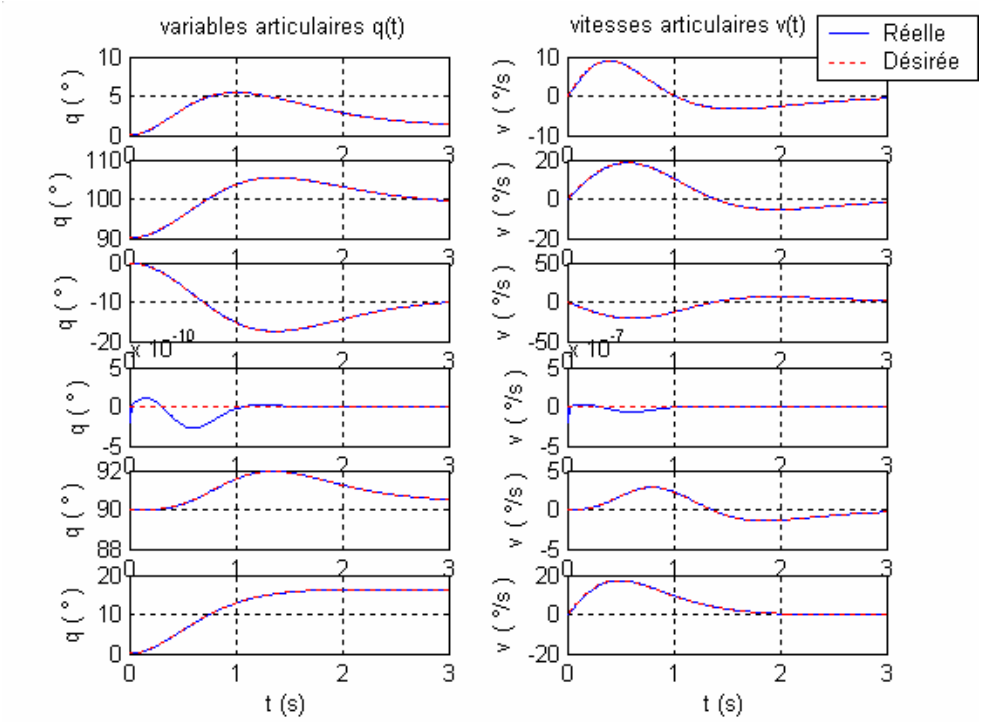


Figure 4.9.b. *Variables et vitesses articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec g_{max}=3 (s^{-1}) et a=2(s).*

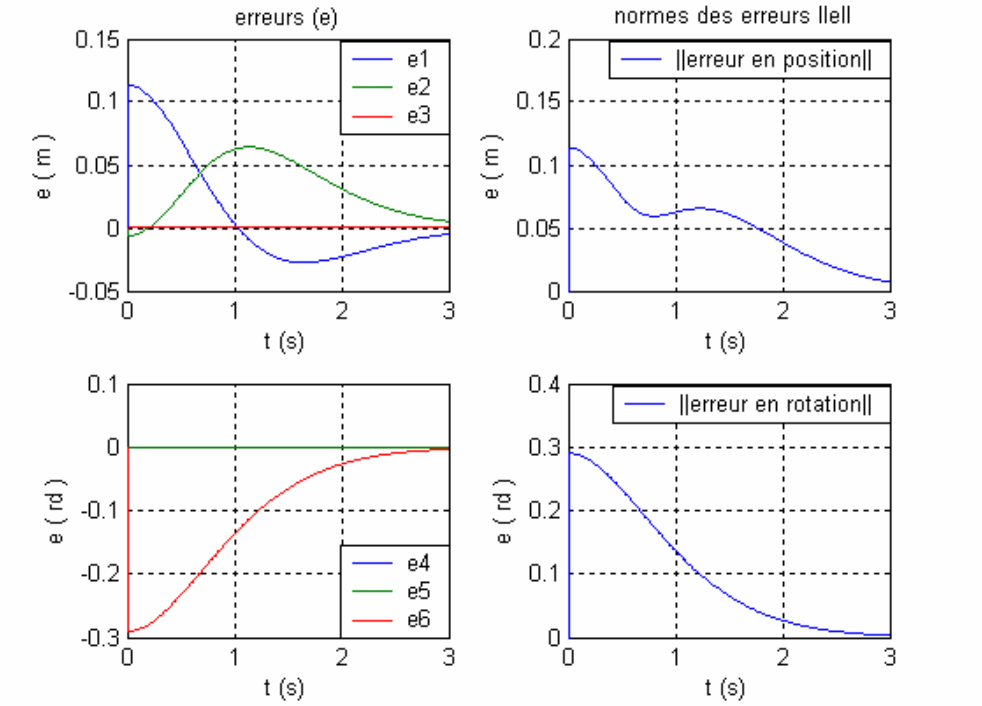


Figure 4.9.c. *Les erreurs pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec g_{max}=3.2 (s^{-1}) et a=2(s).*

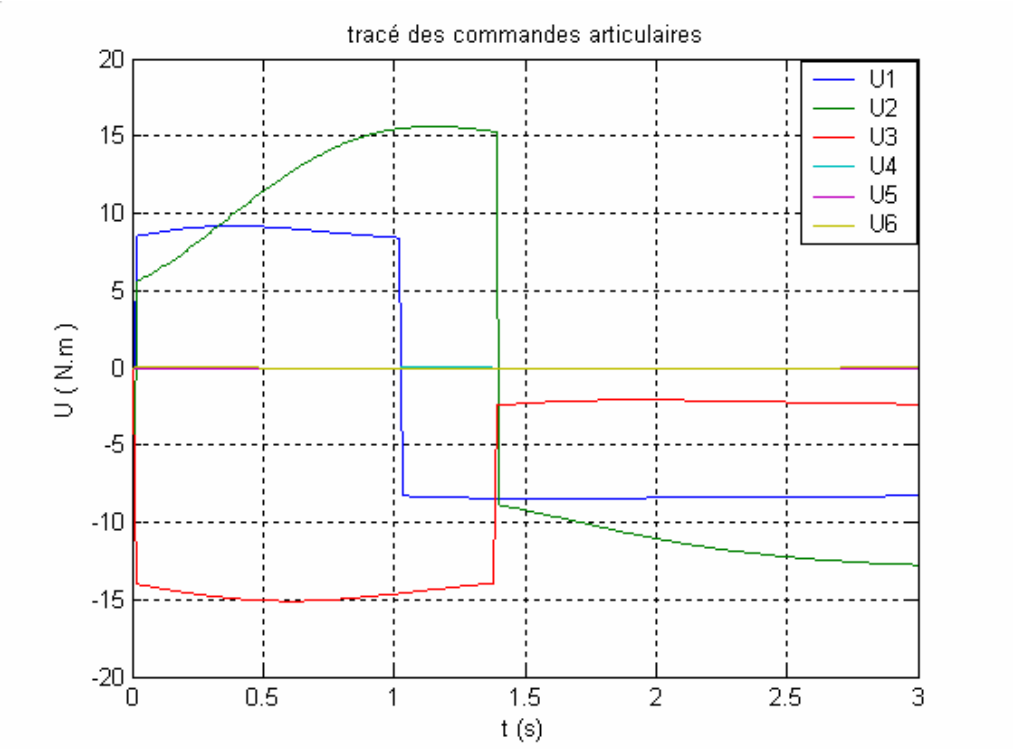


Figure 4.9.d. *les couples articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec g_{max}=3 (s^{-1}) et a=2(s).*

On remarque qu'il y a eu une convergence rapide vers l'image désirée. On remarque aussi qu'on a de très bonnes poursuites en vitesses et même en variables articulaires.

Les commandes sont admissibles et ne dépassent pas 16 N.m avec la présence de quelques variations rapides du sens de rotations des couples articulaires comme le cas de la commande par Backsteeping et le glissement.

D'après ces résultats, on déduit que la commande par linéarisation en vitesses articulaires avec l'emploi du retour d'état pour le système linéarisé dans la boucle du robot possède les mêmes effets que celle par Backsteeping ou le mode glissant, soit du point de vue rapidité, énergie de la commande ou poursuite des articulations ainsi que leurs vitesses.

4.4.2.3. Utilisation de la commande linéarisante en variables articulaires, en utilisant le retour d'état pour le système linéarisé :

La loi de commande utilisé a aussi été décrite dans le paragraphe (4.3.5) ; prenons une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec g_{max} =3 s^{-1} et a=2(s) pour la commande visuelle et les valeurs propres {-1 ;-2 ;-3 ;-4 ;-5 ;-6 ;-7 ;-8 ;-9 ;-10 ;-11 ;-12} pour la commande par retour d'état du système linéarisé en variables articulaires :

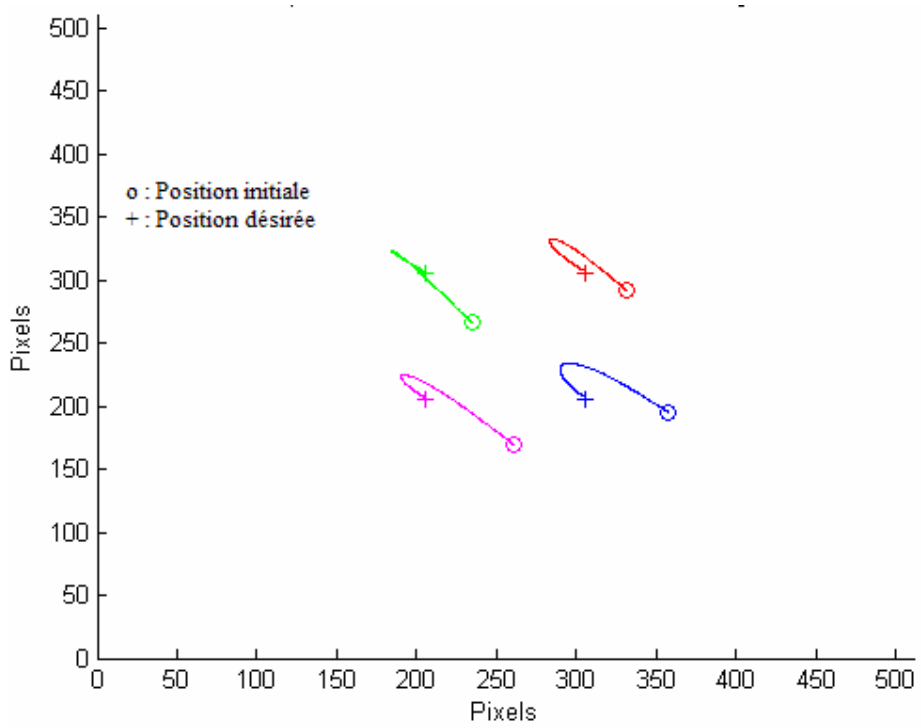


Figure 4.10.a. *Trajectoire dans l'image des primitives visuelles pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15° avec $g_{max}=3(s^{-1})$ et a=2(s).*

On remarque qu'il y a eu une convergence vers l'image désirée, en gardant les mêmes courbes des trajectoires car on a gardé la même loi de commande visuelle.

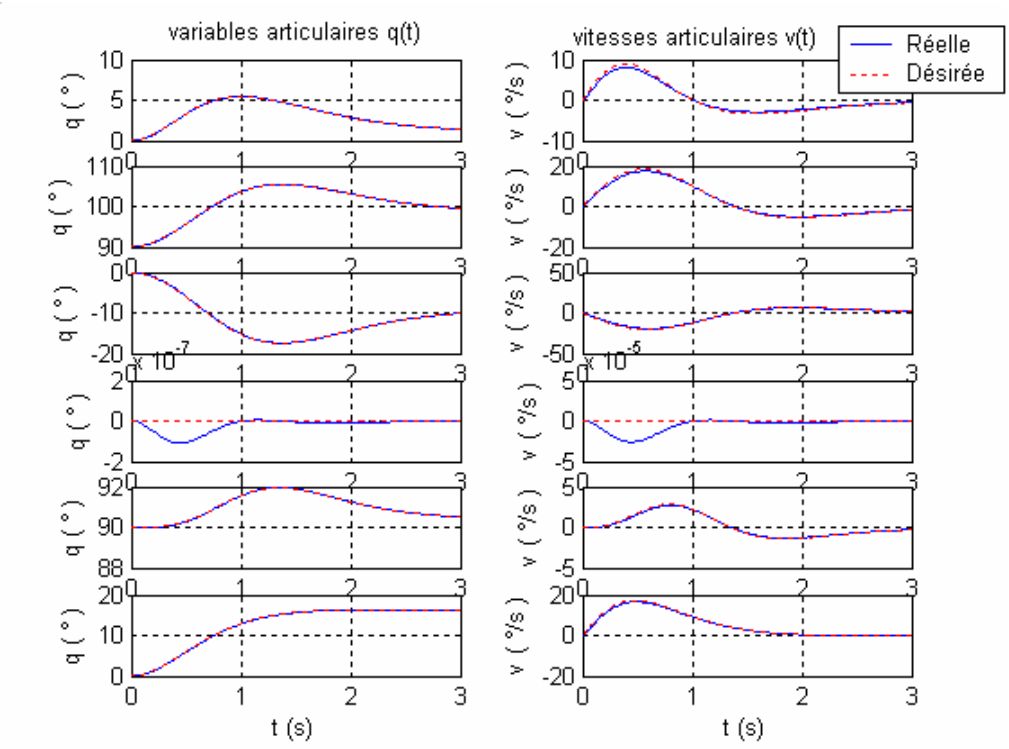


Figure 4.10.b. *Variables et vitesses articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15° avec $g_{max}=3\ (s^{-1})$ et a=2(s).*

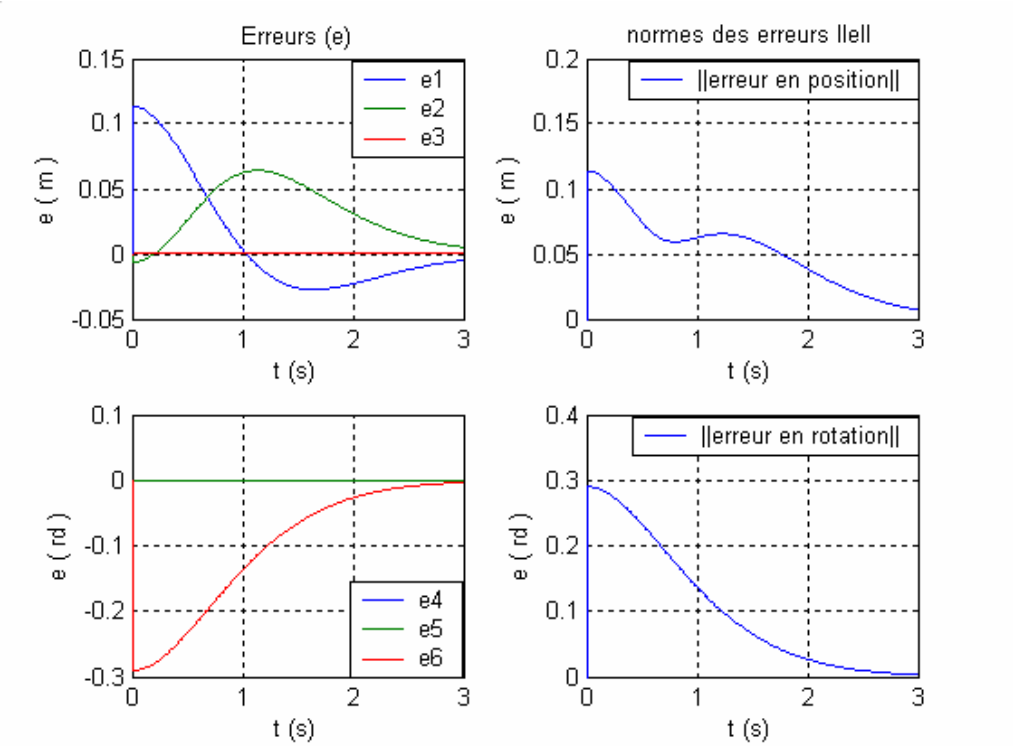


Figure 4.10.c. *Les erreurs pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15° avec gmax=3.2 (s^{-1}) et a=2(s).*

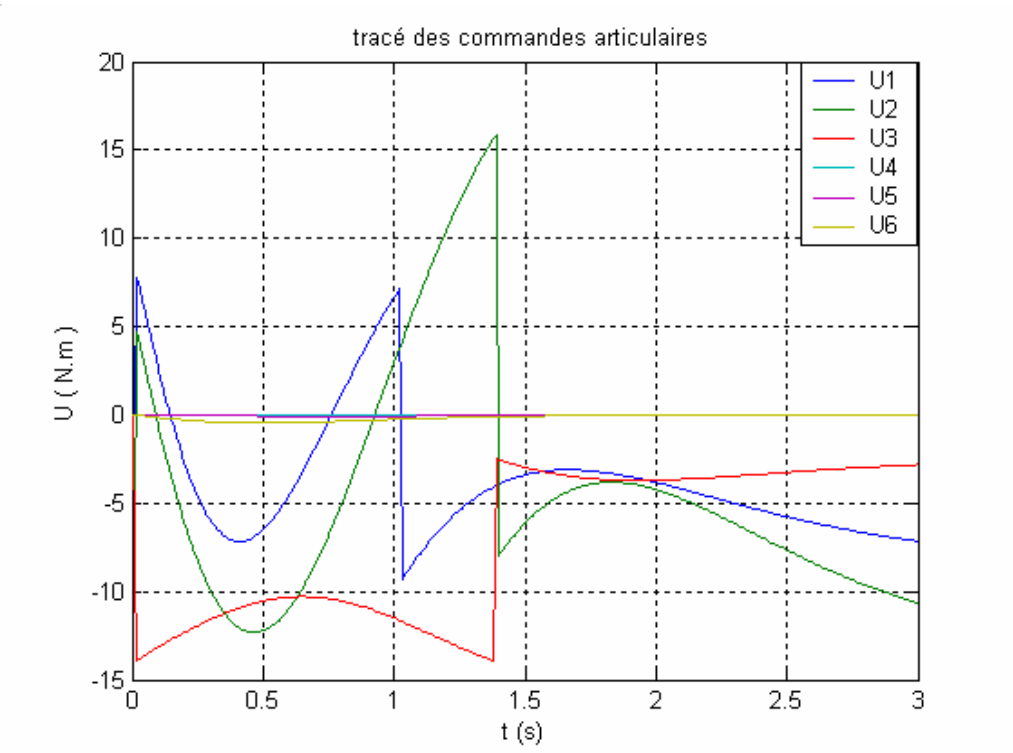


Figure 4.10.d. *les couples articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et une rotation de 15° avec g_{max}=3 (s^{-1}) et a=2(s).*

La linéarisation par rapport aux variables articulation n'influence pas la poursuite en articulations et leurs vitesses ; ainsi que la rapidité de convergence qui est toujours de l'ordre de 3s. Les commandes par contre ont été légèrement influencées ; les bornes supérieures n'ont pas dépassé 20 N.m mais il y avait plus de variations dans les signes des couples articulaires.

En conclusion ; on peut dire que le type de commande utilisée pour la commande du robot ne modifie pas le comportement de ce dernier ; seule la commande visuelle est susceptible de modifier son comportement de point de vu rapidité, énergie et même les trajectoires des primitives visuelles dans l'image, comme il a déjà été montré en variant la loi visuelle entre la loi classique et celle par mode de glissement.

4.4.3. Influence des perturbations sur la commande visuelle :

Comme il a déjà été expliqué dans le paragraphe (3.4.2), la commande de la boucle de vision doit d'être robuste vis-à-vis des perturbations, en remarquant que toutes les perturbations agissant sur le système (couples résistant sur les actionneurs, erreurs de calibrage, erreurs de modélisation et même le mouvement propres de l'objet) ont tous des conséquences dans l'image, il serait plus judicieux de ne concevoir qu'un régulateur robuste vis-à-vis des perturbations dans la boucle de vision.

Tentons alors de simuler le comportement du robot dans le cas de présence des perturbations dans la commande ; on utilise pour cela le Backsteeping avec $\alpha = \beta = -15$ pour la commande du robot et la loi classique avec g adaptatif pour la boucle de vision. Prenons pour notre simulation une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec $g_{max}=4(s^{-1})$ *et* $a=1(s)$:

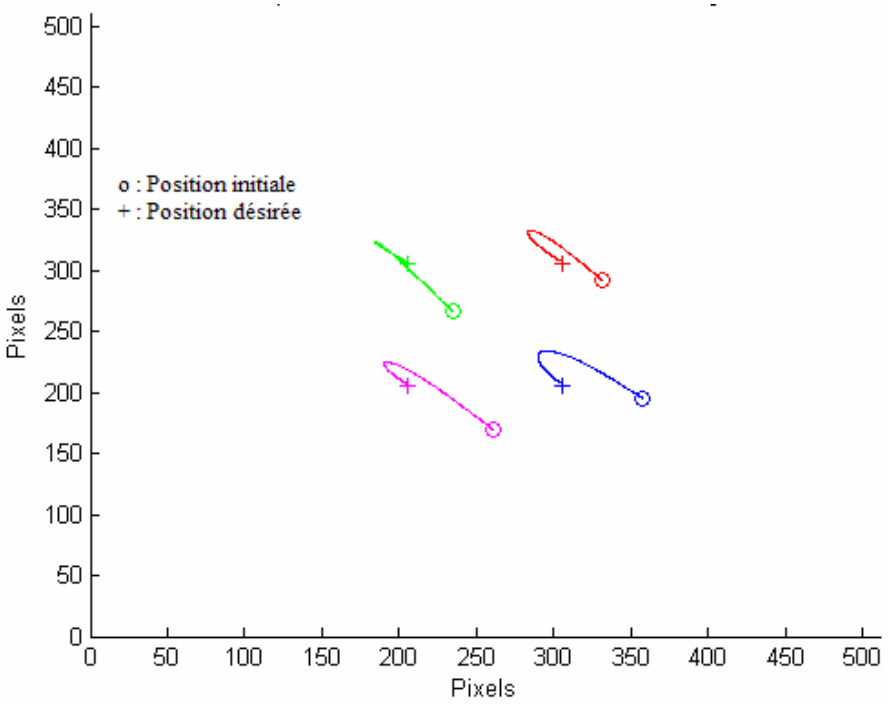


Figure 4.11.a. *Trajectoire dans l'image des primitives visuelles pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° en présence des perturbations avec* $g_{max}=4(s^{-1})$ *et* $a=1(s)$.

On remarque qu'on a bel et bien convergence vers l'image désirée sans présence de perturbation dans leurs trajectoires.

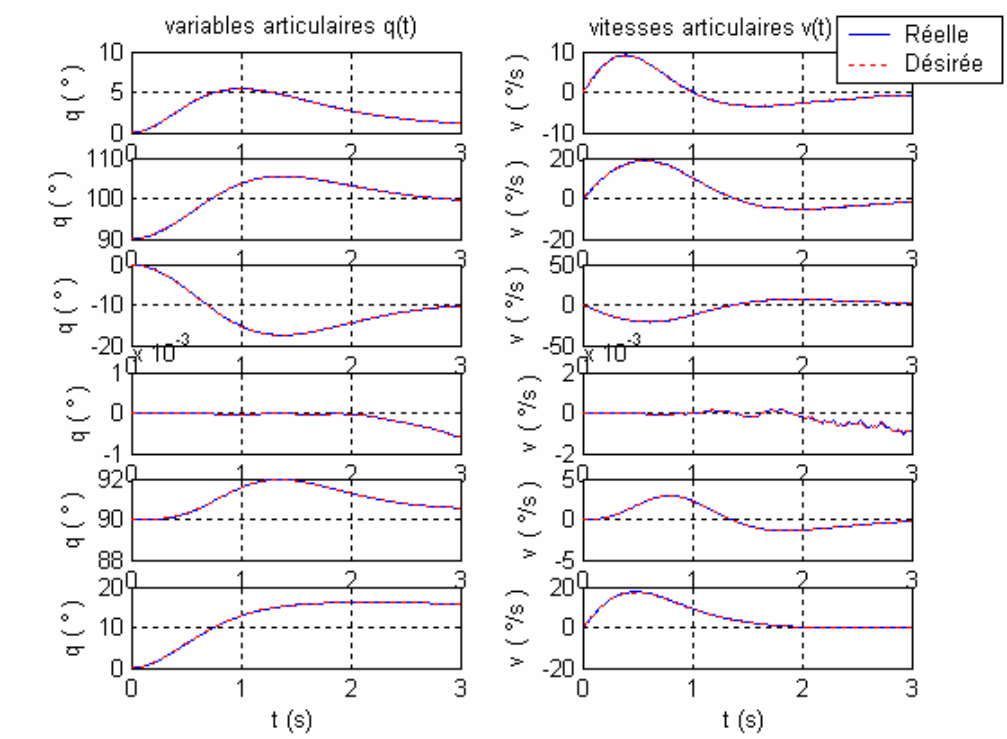


Figure 4.11.b. *Variables et vitesses articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec g_{max}=4 (s^{-1}) et a=1(s).*

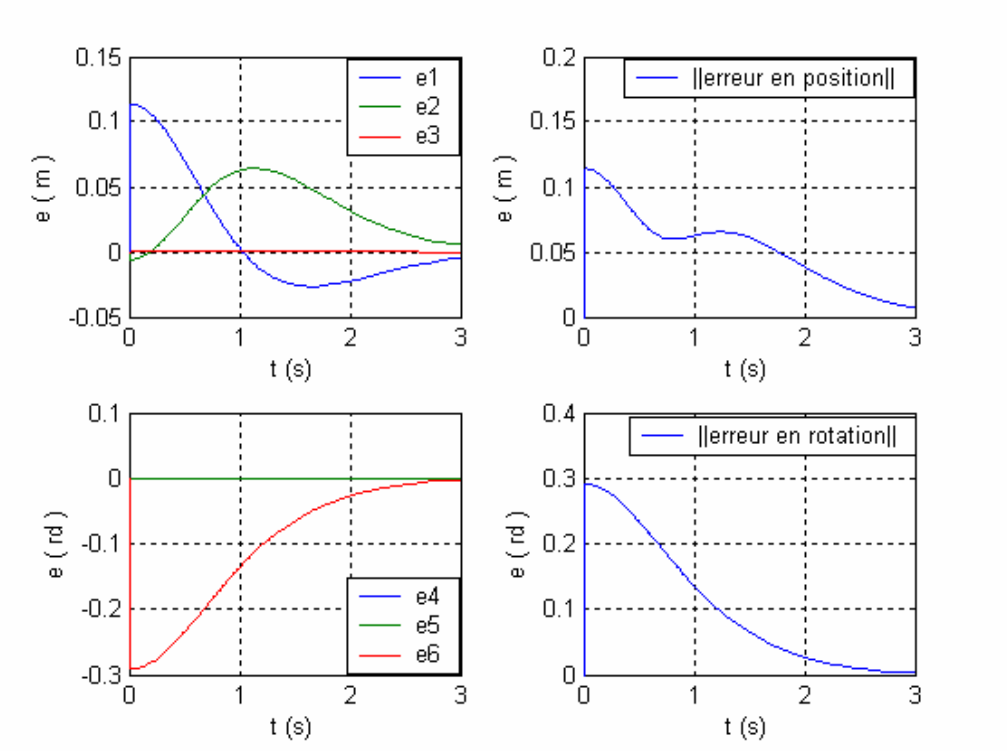


Figure 4.11.c. *Les erreurs pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° avec g_{max}=4 (s^{-1}) et a=1(s).*

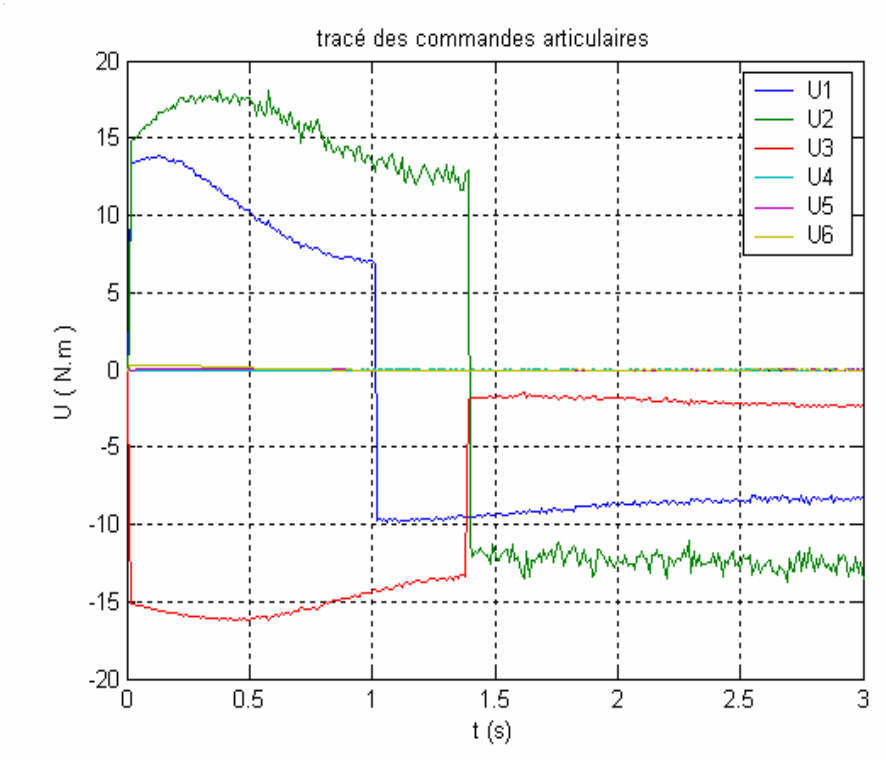


Figure 4.11.a. *Couples articulaires pour une translation de -10 pixels selon u et 39 pixels selon v et rotation de 15° en présence des perturbations avec $g_{max}=4(s^{-1})$ et a=1(s).*

On remarque que la convergence reste toujours rapide, les articulations et leurs vitesses sont faiblement influencées ; les erreurs se sont annulées ; seules les couples articulaires présentent quelques broutements légers mais restent toujours admissibles.

Prenons comme deuxième cas de figure une translation de 50 pixels sur u et 100 pixels sur v avec les mêmes caractéristiques de simulations que précédemment :

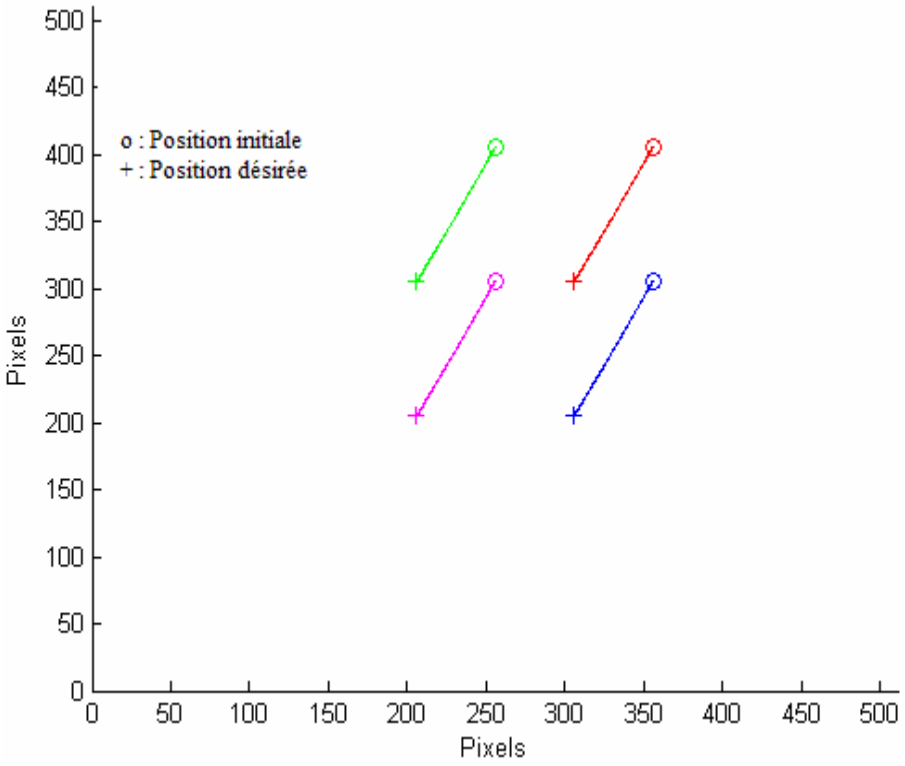


Figure 4.12.a. *Trajectoire dans l'image des primitives visuelles pour une translation de 50 pixels sur u et 100 pixels sur v avec $g_{max}=4(s^{-1})$ et a=1(s).*

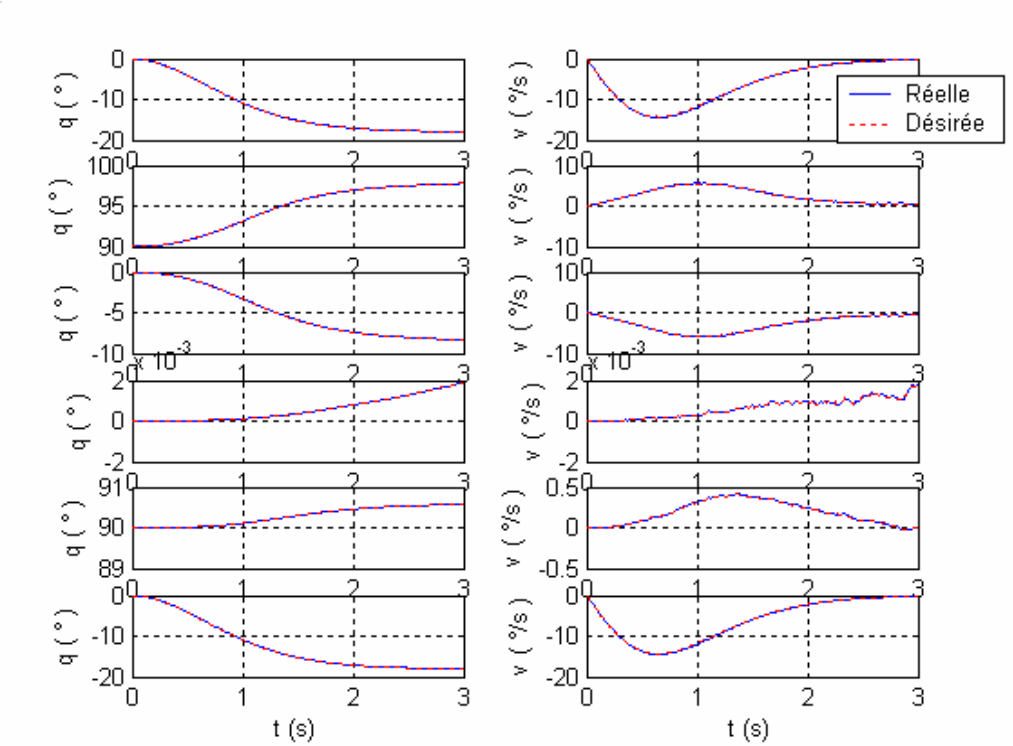


Figure 4.12.b. *Variables et vitesses articulaires pour une translation de 50 pixels sur u et 100 pixels sur v avec g_{max}=4 (s^{-1}) et a=1(s).*

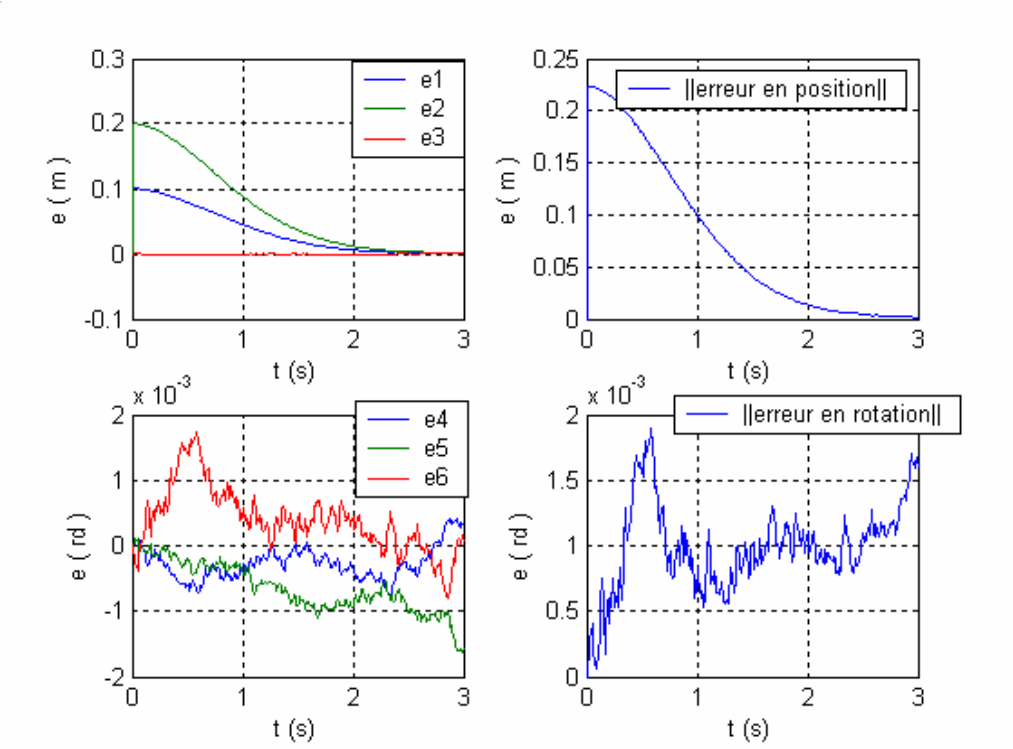


Figure 4.12.c. *Les erreurs pour une translation de 50 pixels sur u et 100 pixels sur v avec g_{max}=4 (s^{-1}) et a=1(s).*

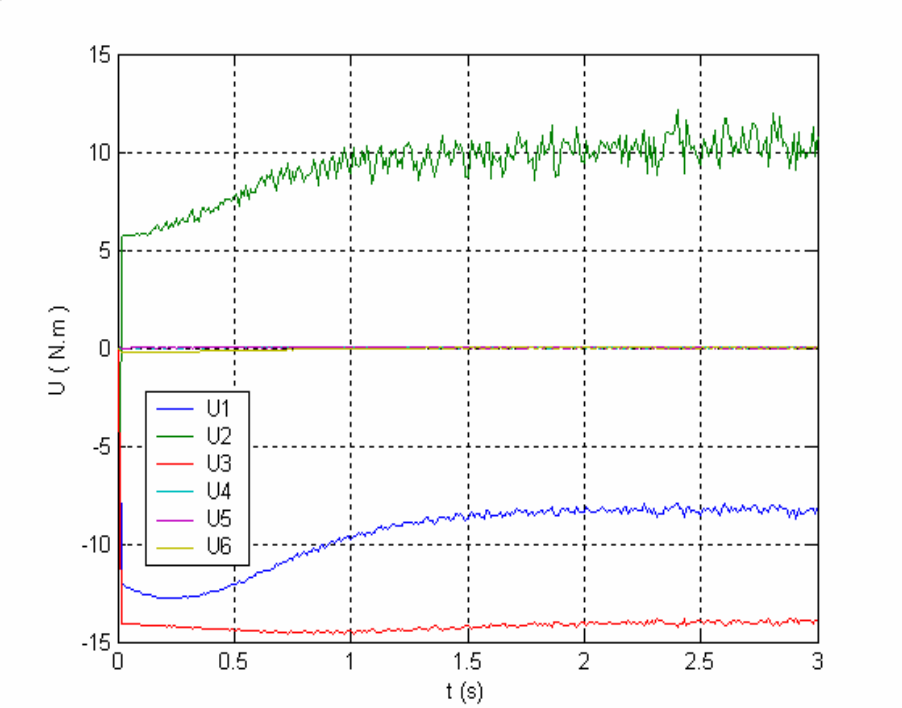


Figure 4.12.d. *Les couples articulaires pour une translation de 50 pixels sur u et 100 pixels sur v avec $g_{max}=4\ (s^{-1})$ et a=1(s).*

Les erreurs se sont annulées rapidement ; l'ordre des erreurs en rotation est de 10^{-3} car on est dans le cas de translation pure.

La commande est admissible mais présentant des broutements à cause des perturbations.

4.4.4. Cas de poursuite de cible :

Tentons de simuler à présent une tâche de poursuite. L'objet est en mouvement de translation sur une bande convoyeur. Sa vitesse et sa trajectoire sont inconnues pour le robot. Voici la simulation d'un asservissement visuel 3D réalisant la tâche de poursuite, avec *g* adaptatif $g_{max}=5(s^{-1})$ *et a=3(s)* (la commande du robot est toujours le Backsteeping) :

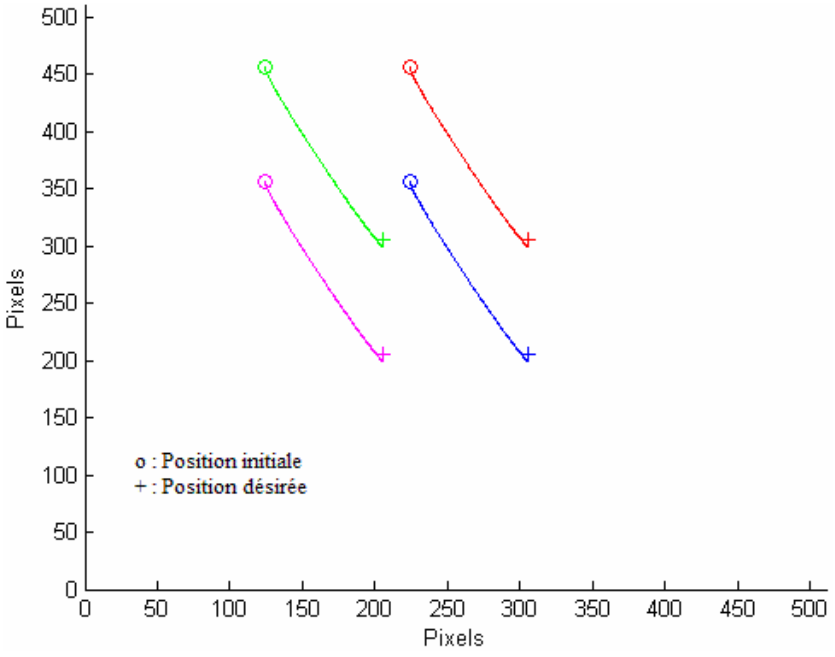


Figure 4.13.a. *Trajectoire dans l'image des primitives visuelle pour une tâche de poursuite, avec g variable : $g_{max}=5(s^{-1})$ et a=3(s).*

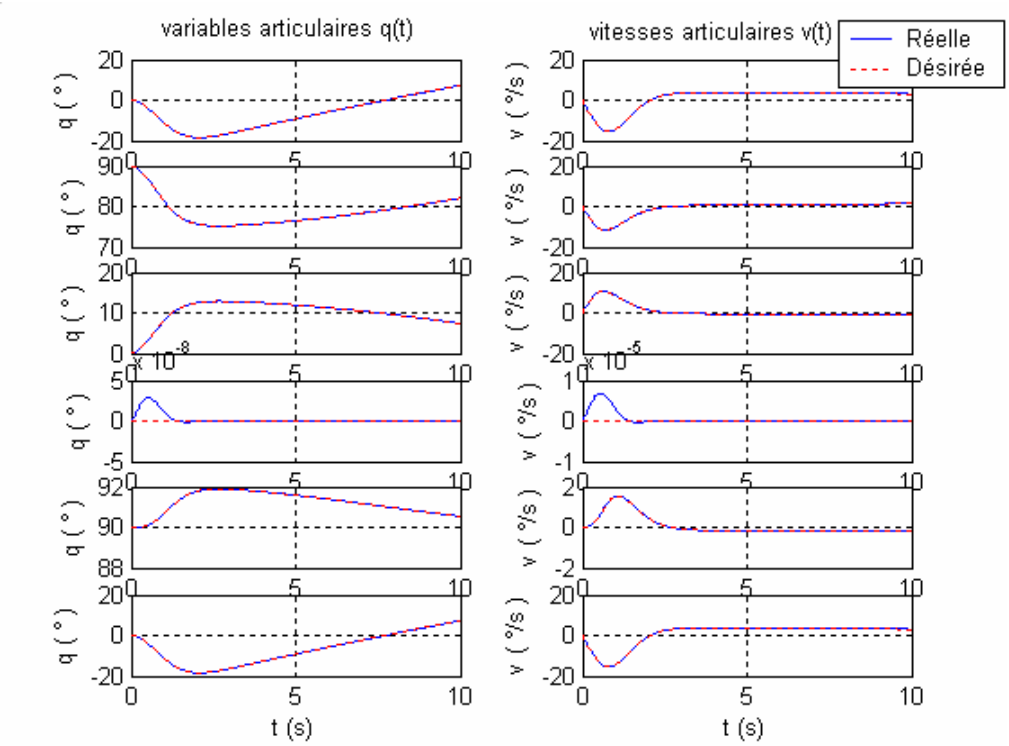


Figure 4.13.b. *Variables et vitesses articulaires pour une tâche de poursuite, avec g variable : $g_{max}=5(s^{-1})$ et a=3(s).*

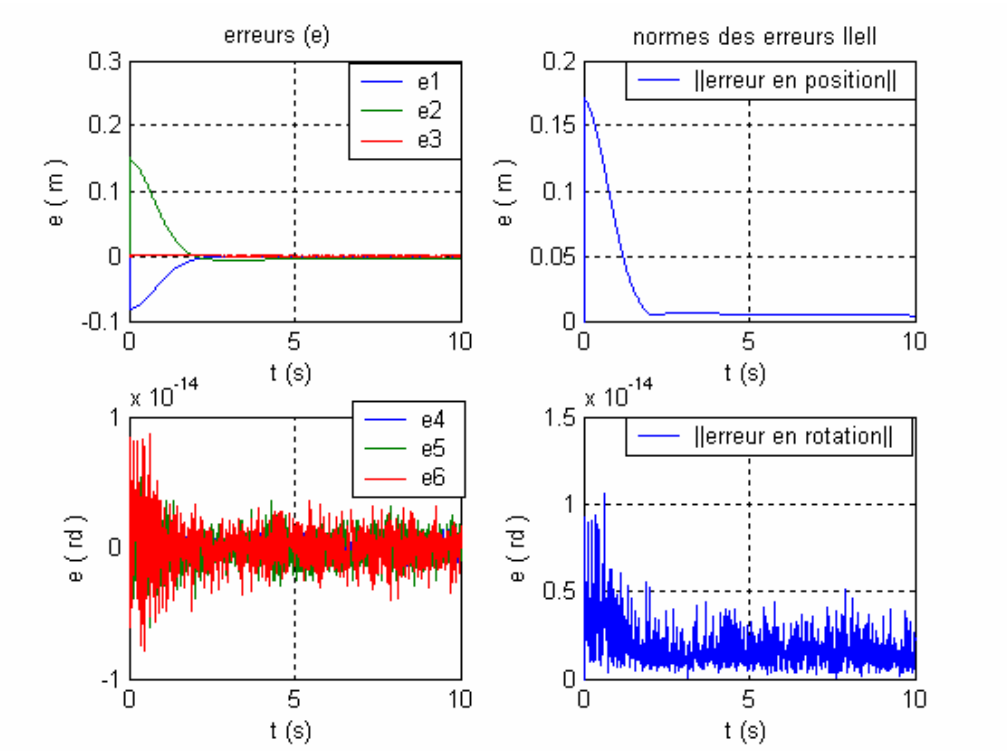


Figure 4.13.c. *Les erreurs pour une tâche de poursuite, avec g variable : $g_{max}=5(s^{-1})$ et a=3(s).*

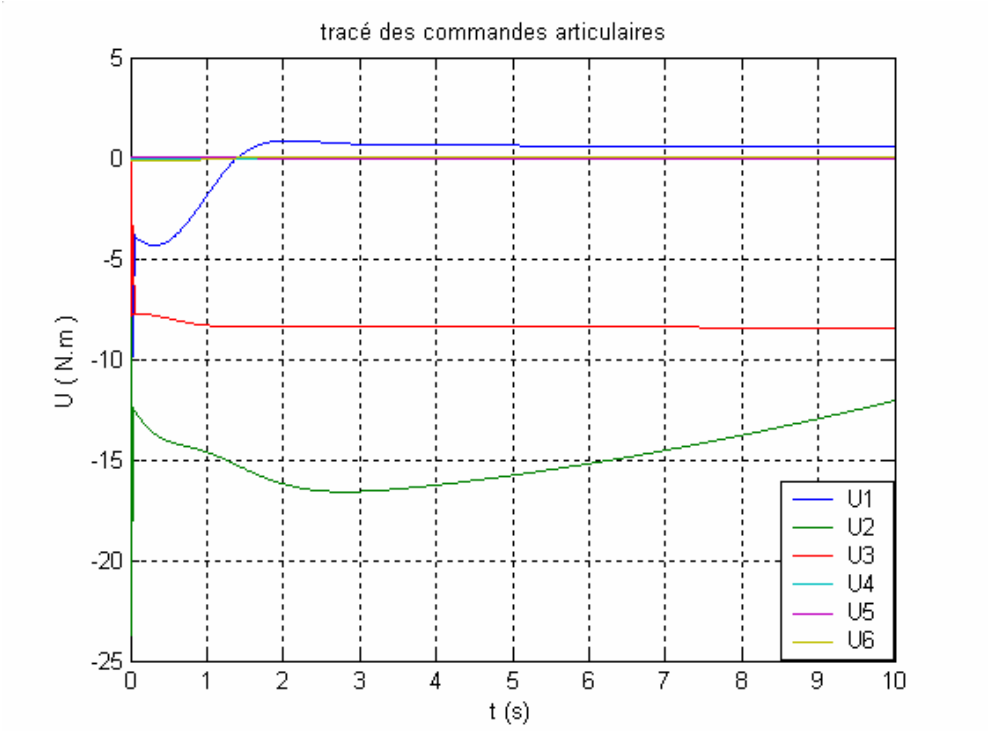


Figure 4.13.d. *Les couples articulaires pour une tâche de poursuite, avec g variable : $g_{max}=5(s^{-1})$ et a=3(s).*

Nous remarquons que la tâche de poursuite est correctement effectuée en employant cette méthode. Notons qu'au départ le robot n'avait aucune information sur la position de l'objet, sa trajectoire et sa vitesse. D'autre part, remarquons que le fait d'effectuer une tâche de poursuite nous a contraint d'employer un gain maximum plus élevé (*gmax*=5). Les valeurs de g_{max} et *a* ont été trouvées par tâtonnement, de manière à effectuer correctement la tâche de poursuite, en tentant d'éliminer l'erreur de traînage, et la valeur du couple articulaire initiale qui n'atteint que 24 N.m et ne dépassant pas après les 17 N.m.

4.5. Conclusion :

Le comportement dans le domaine 3D de ce type d'asservissement en cas de convergence du système est généralement satisfaisant du point de vue de la trajectoire de la caméra.

Ce type d'asservissement visuel présente également l'inconvénient d'être sensible aux perturbations qui se répercutent sur la commande. Le problème de maintien de l'objet dans le champ de vision de la caméra se pose aussi pour de grands déplacements, étant donné que le contrôle se fait dans l'espace cartésien, sans prendre en compte ce qui ce passe dans l'image.

Pour remédier à ce problème ainsi que celui de la sensibilité aux perturbations, un compromis des asservissements 2D et 3D a été réalisé en combinant ces deux types d'asservissement, c'est ce qui fera l'objet du prochain chapitre.

Chapitre

5

Asservissement Visuel 2D½

Préambule :

La prise en considération d'informations visuelles 3D repose sur l'hypothèse que ces données peuvent être mesurées de manière fiable. En pratique, elles sont plus sensibles aux erreurs de mesure que les informations visuelles 2D, puisque obtenues à partir de celles-ci et d'un calcul de pose.

Il est donc intéressant de combiner des informations visuelles 2D et 3D pour gagner en robustesse aux erreurs de mesure tout en conservant de bonnes propriétés de découplage.

L'approche 2D½ s'appuie sur le découplage entre la boucle d'asservissement en translation et la boucle d'asservissement en rotation. Elle permet de combiner un contrôle partiel des trajectoires dans l'espace cartésien et dans l'image.

5.1. Introduction :

La commande 2D1/2 utilise une combinaison d'informations exprimées pour certaines d'entre elles dans l'image, et pour d'autres dans le repère caméra (repère cartésien).

La rotation de la caméra entre deux prises de vue de l'objet (courante et désirée) peut être calculée à chaque itération de la loi de commande. Par conséquent, la boucle d'asservissement en rotation peut être découplée de celle en translation, ce qui permet d'obtenir une forme simple du jacobien de la tâche. Afin de contrôler les degrés de liberté en translation, on introduit les coordonnées images d'un point de référence de la cible. Celles-ci sont obtenues à partir des coordonnées image métriques classiques, et en ajoutant une troisième coordonnée normalisée qui peut être mesurée à partir de la reconstruction partielle. Cette troisième coordonnée n'est rien d'autre qu'un rapport entre deux distances.

5.2. Modélisation de l'asservissement visuel 2D1/2 [MAL 98] :

La commande 2D1/2 utilise deux contrôles, celui de la rotation et celui de la translation.

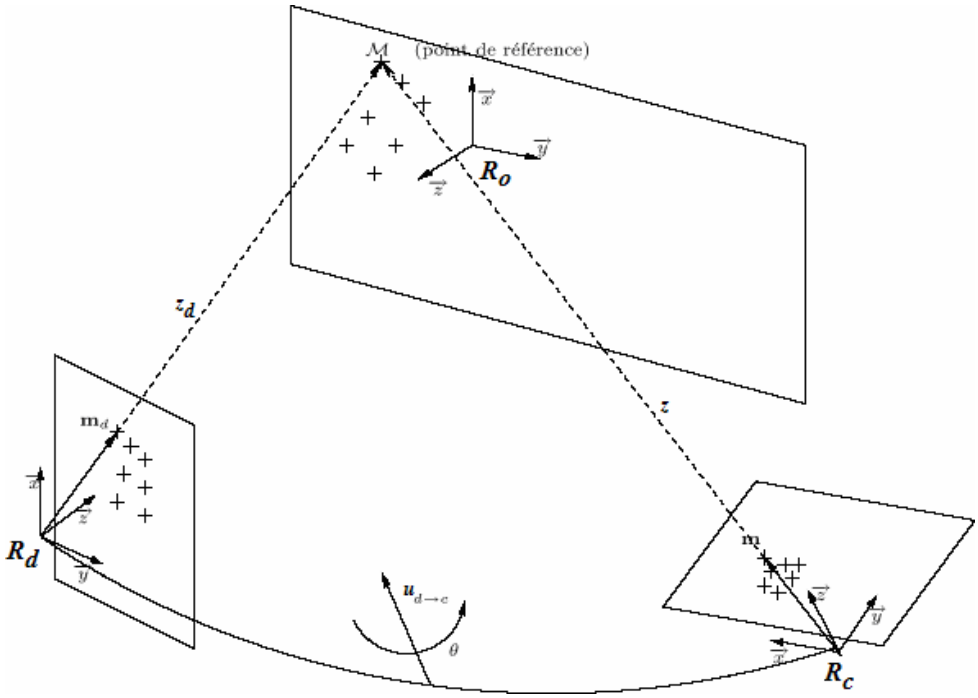


Figure 5.1. *Modélisation d'un asservissement visuel 2D1/2*

Afin de contrôler l'orientation de la caméra, on utilise la rotation ${}^{d}R_{c}$ entre le repère désiré R_d et le repère courant R_c (${}^{d}R_{c}$ doit se ramener à une identité), pour cela, on choisit la représentation vectorielle θu de la rotation ${}^{d}R_{c}$ (2.24).

La position de la caméra peut être contrôlée à partir d'informations dans l'image et dans le repère cartésien. Pour garder la cible dans le champ de vision de la caméra, on utilise

trois informations indépendantes, à savoir les deux coordonnées visuelles d'un point de la cible dans l'image, et le rapport Z/Z_d ; avec Z et Z_d la distance courante et désirée de ce point de la cible par rapport à la caméra.

5.3. Régulation de la fonction de tâche [KHA 02] :

La notion de fonction de tâche ayant déjà été abordé au premier chapitre, nous passerons donc à sa régulation dans le cas d'une caméra embarquée observant l'objet ; la loi de commande élaborée doit faire tendre cette fonction vers zéro.

La fonction de tâche dans le cas d'un asservissement visuel 2D1/2 est définie de la manière suivante :

$$e = \begin{pmatrix} x - x^* & y - y^* & Ln(\frac{Z}{Z_d}) & \theta u^T \end{pmatrix}^T \tag{5.1}$$

Où :

- $(x, y)^T$ et $(x^*, y^*)^T$ sont respectivement les coordonnées métriques courantes et désirées d'un point caractéristique dans l'image ;
- Z / Z_d est le rapport entre les distances courante et désirée de ce point par rapport à la caméra ;
- θu la représentation minimale de la rotation à réaliser (2.24).

Intéressons nous à réaliser simplement une décroissance exponentielle de la fonction de tâche, à savoir :

$$\dot{e} = -g\, e \quad , \quad g \text{ scalaire} > 0 \tag{5.2}$$

Si l'objet est immobile, c'est-à-dire $\frac{\partial e}{\partial t} = 0$, $\dot{r}_r$ (torseur cinématique du robot) aurait la forme suivante (paragraphe 4.3.1) :

$$\dot{r}_r = -g.W.^{-1} L.^{-1}.e = -g.(L.W)^{-1}.e \tag{5.3}$$

Avec :

$$\dot{r}_r = J(q).\dot{q} \tag{5.4}$$

$$W = \begin{bmatrix} I & \left[{}^c P_e\right]_x \\ 0 & I \end{bmatrix} \begin{bmatrix} {}^c R_e & 0 \\ 0 & {}^c R_e \end{bmatrix} = \begin{bmatrix} {}^c R_e & \left[{}^c P_e\right]_x {}^c R_e \\ 0 & {}^c R_e \end{bmatrix} \tag{5.5}$$

$$v_c = W\, \dot{r}_r \tag{5.6}$$

Où v_c le torseur cinématique de la caméra. Si l'on souhaite utiliser les variations articulaires comme signaux de commande, et en considérant l'attitude désirée fixe avec l'objet immobile, la loi de commande sera la suivante :

$$\dot{q}_d = -g.{}^{n}J^{-1}{}_{n}(q).(LW)^{-1}.e \tag{5.7}$$

Où ${}^{n}J_{n}(q)$ est le *Jacobien du robot* exprimé dans le repère R_n de son organe terminal [KHA 99].

L'estimation du paramètre $\frac{\partial e}{\partial t}$ est la même que celle illustrée dans le (4.3.2).

La commande étant le torseur cinématique T entre la caméra et son environnement, elle est donnée par :

$$T = v_c = \begin{pmatrix} V \\ \omega \end{pmatrix} = W.\dot{r}_r = W \cdot {}^{n}J_n(q) \cdot \dot{q} \tag{5.8}$$

Selon (5.7) et en considérant le mouvement de l'objet comme une perturbation au système, la loi de commande de la boucle de vision s'écrit :

$$T_d = -\hat{L}^{-1} \cdot g \cdot e \tag{5.9}$$

Ceci permet aussi de tracer le schéma de la boucle de commande :

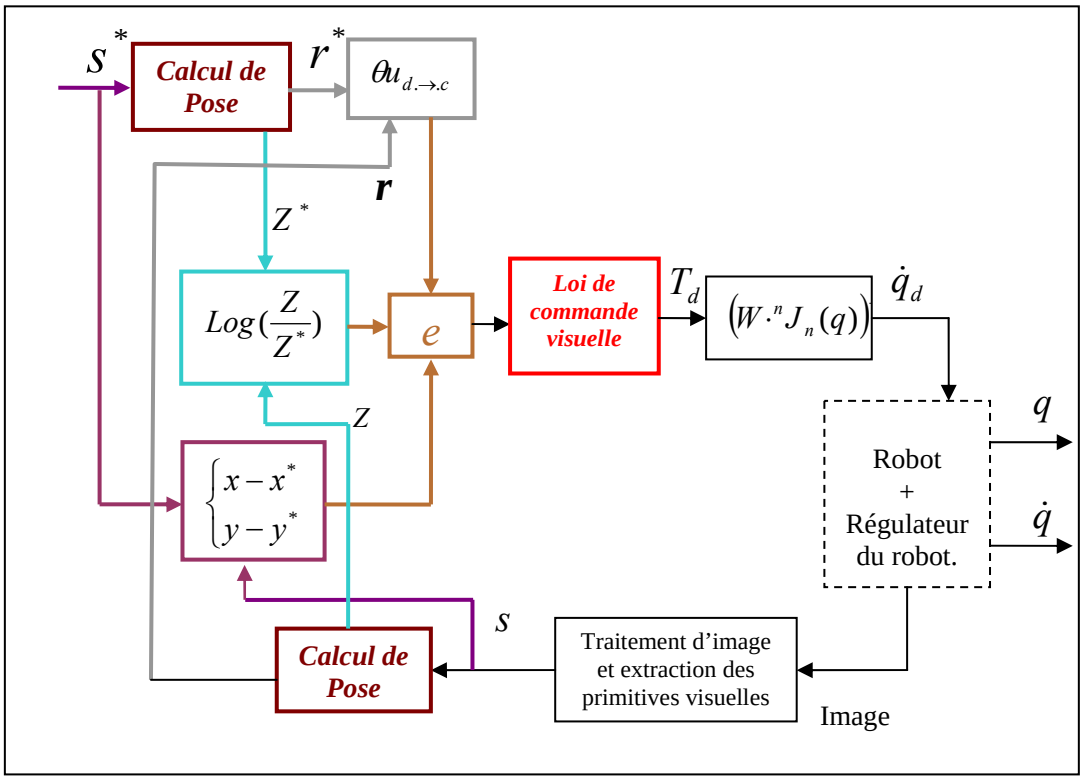


Figure 5.2. *Schéma de commande de type2D1/2*

Suivant la fonction de tâche choisie dans (5.1), la matrice d'interaction d'information 3D est [MAL 98, KHA 02] :

$$L = \begin{pmatrix} \frac{1}{Z}.L_{ev} & L_{ev\omega} \\ 0_3 & I_3 \end{pmatrix} \tag{5.10}$$

Avec :

$$L_{ev} = \begin{pmatrix} -1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1 \end{pmatrix} \tag{5.11}$$

$$L_{ev\omega} = \begin{pmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \\ -y & x & 0 \end{pmatrix} \tag{5.12}$$

La condition de stabilité de la matrice d'interaction suffisante pour assurer la décroissance de $\|e\|$ dans tout l'espace de travail est (3.23) :

$$L\hat{L}^{-1} \approx LL^{-1} = I_6 > [0] \tag{5.13}$$

Le bloc associé au robot, ainsi que toutes les commandes ont été décrits dans le paragraphe (3.4.3).

5.4. Simulations de l'asservissement visuel 2D1/2 :

La boucle d'asservissement 2D1/2 est décrite sur la figure (5.2) . Les conditions de stabilisation de la structure en cascade sont similaires à celles utilisées dans le paragraphe (3.4.4). Nous considérons le cas de la commande par asservissement visuel 2D1/2 du robot PUMA 560, où nous choisirons le Backsteeping comme loi de commande du robot, les constantes $\alpha = \beta = -15$.

Simulons le comportement du robot pour une translation de -10 pixels sur u et 50 pixels sur v et une rotation de 15° en utilisant *g* constant égal à 0.2 :

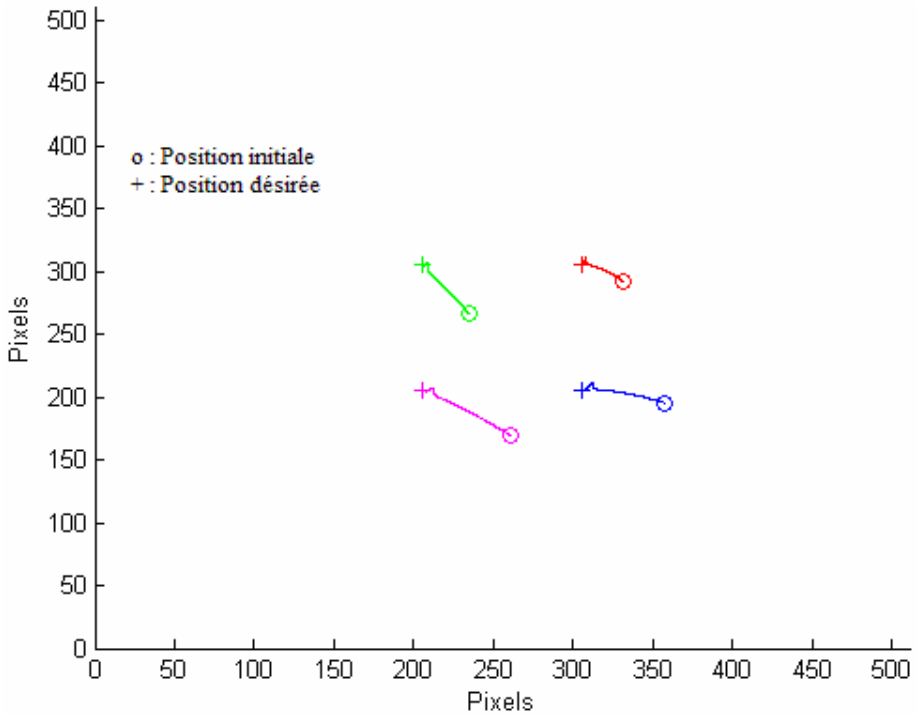


Figure 5.3.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels sur u et 50 pixels sur v et une rotation de 15°, avec g=0.2 (s^{-1}).*

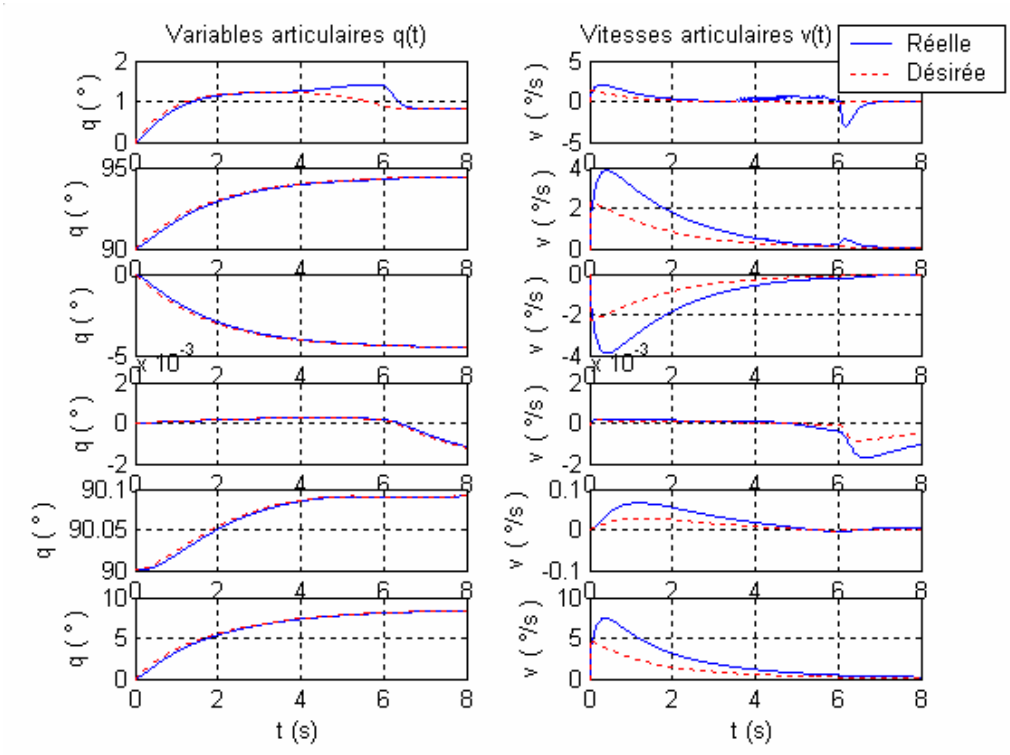


Figure 5.3.b. *Variables et vitesses articulaires pour une translation de -10 pixels sur u et 50 pixels sur v et une rotation de 15°, avec g=0.2 (s^{-1}).*

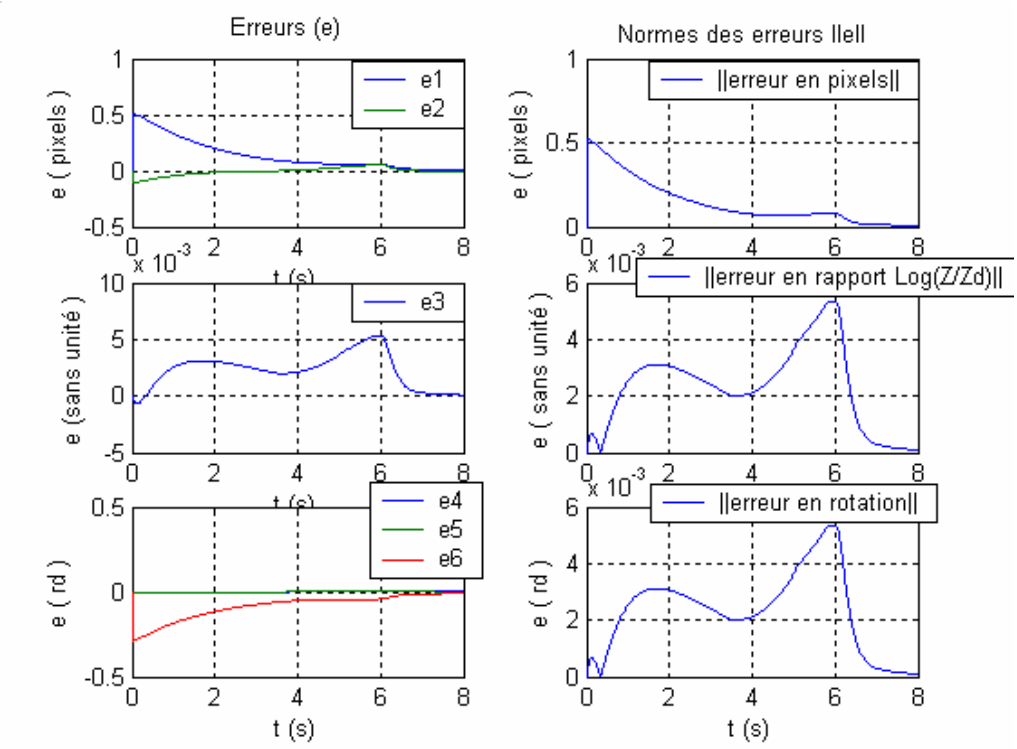


Figure 5.3.c. *Les erreurs pour une translation de -10 pixels sur u et 50 pixels sur v et une rotation de 15°, avec g=0.2 (s^{-1}).*

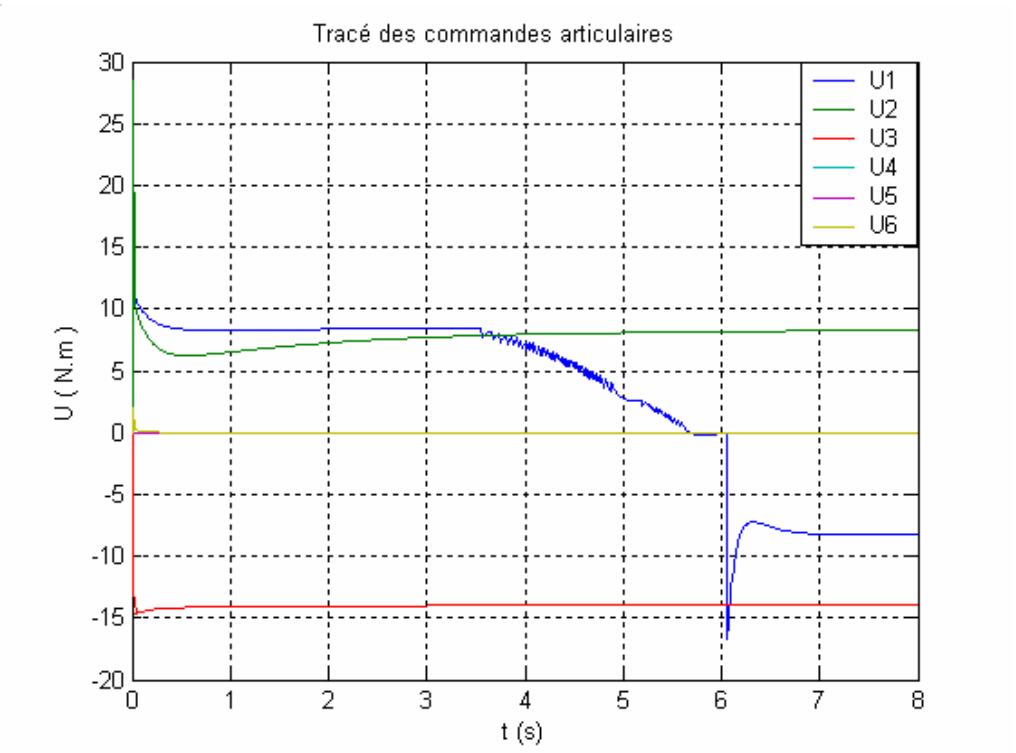


Figure 5.3.d. *Les couples articulaires pour une translation de -10 pixels sur u et 50 pixels sur v et une rotation de 15°, avec g=0.2 (s^{-1}).*

Nous avons une bonne convergence vers l'image désirée avec des trajectoires presque rectilignes bien que les primitives visuelles effectuent une rotation de 15°.

La poursuite en articulation est bonne, mais celle en vitesses articulaires n'est pas vraiment atteinte comparé à l'asservissement visuel 3D ; les erreurs se sont bel et bien annulées au bout de 8s.

Ce type d'asservissement a réduit le temps de convergence de 15s à 8s et les commandes articulaires de 176 N.m à 30N.m comparée aux résultats trouvée dans les simulations (4.4.d), où on avait utilisé aussi un *g* constant. Malgré cela, la convergence en 8s est relativement lente pour une tâche de positionnement.

Nous tenterons à présent d'utiliser un gain *g* adaptatif, (paragraphe 3.5.2), afin de diminuer le temps de convergence et l'amplitude de l'impulsion initiale de commande. Testons cette méthode en prenant comme premier cas une translation de -10 pixels sur u et 50 pixels sur v et rotation de 15° avec $g_{max}=1.1(s^{-1})$ et $a=3(s)$:

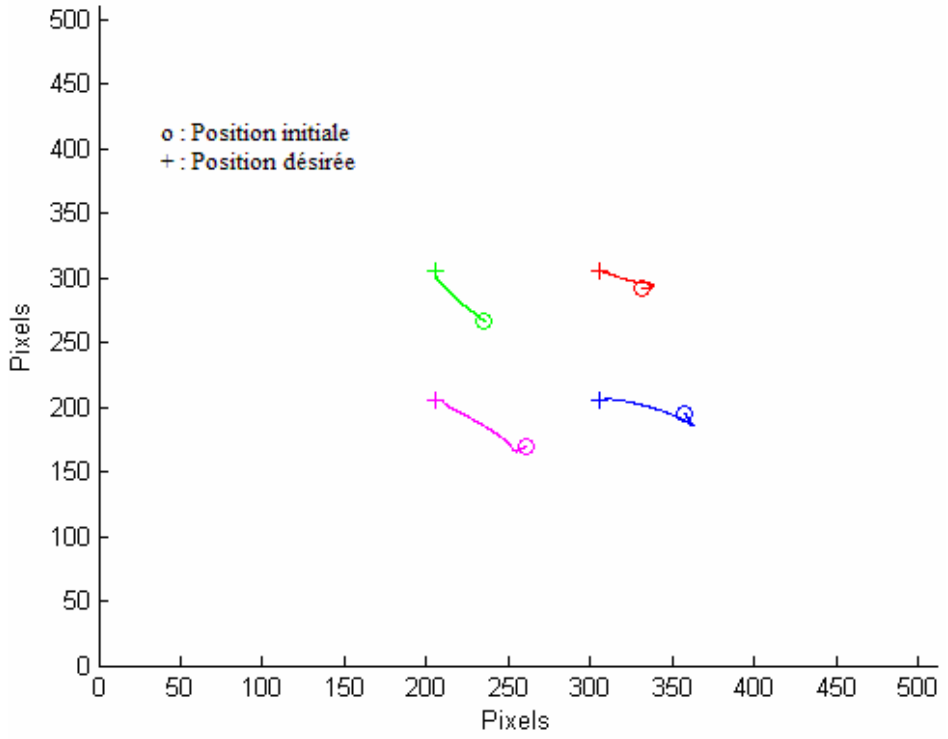


Figure 5.4.a. *Trajectoire dans l'image des primitives visuelle pour une translation de -10 pixels sur u et 50 pixels sur v et une rotation de 15°, avec* $g_{max}=1.1\ (s^{-1})$ *et* $a=3(s)$.

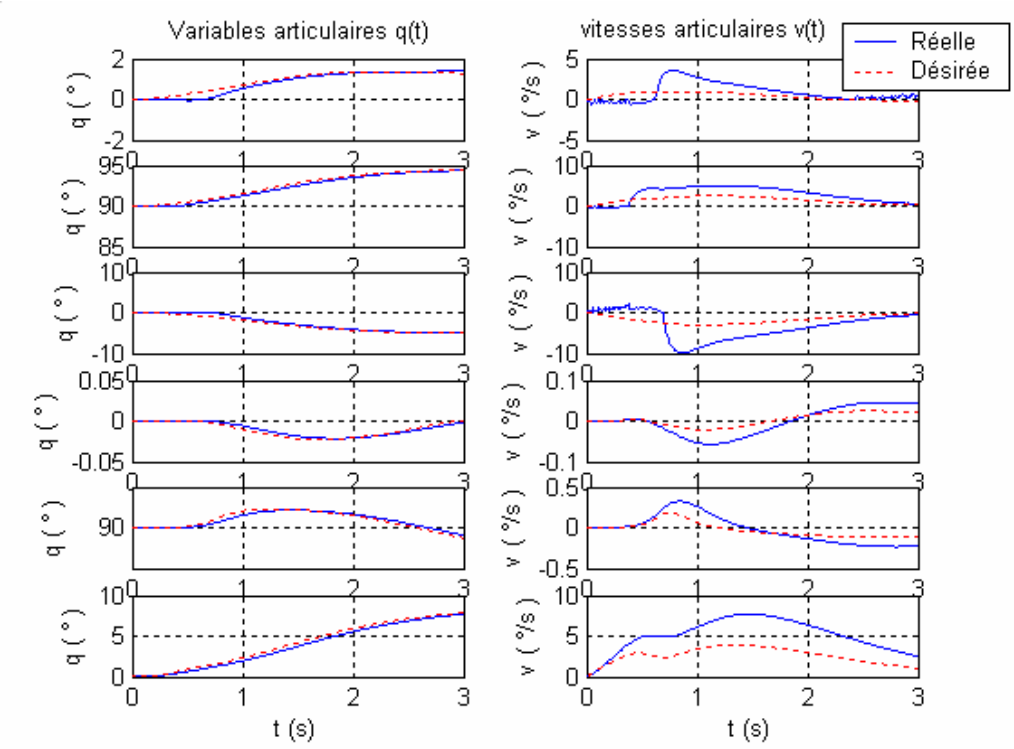


Figure 5.4.b. *Variables et vitesses articulaires pour une translation de -10 pixels sur u et 50 pixels sur v et une rotation de 15°, avec g_{max}=1.1 (s^{-1}) et a=3(s).*

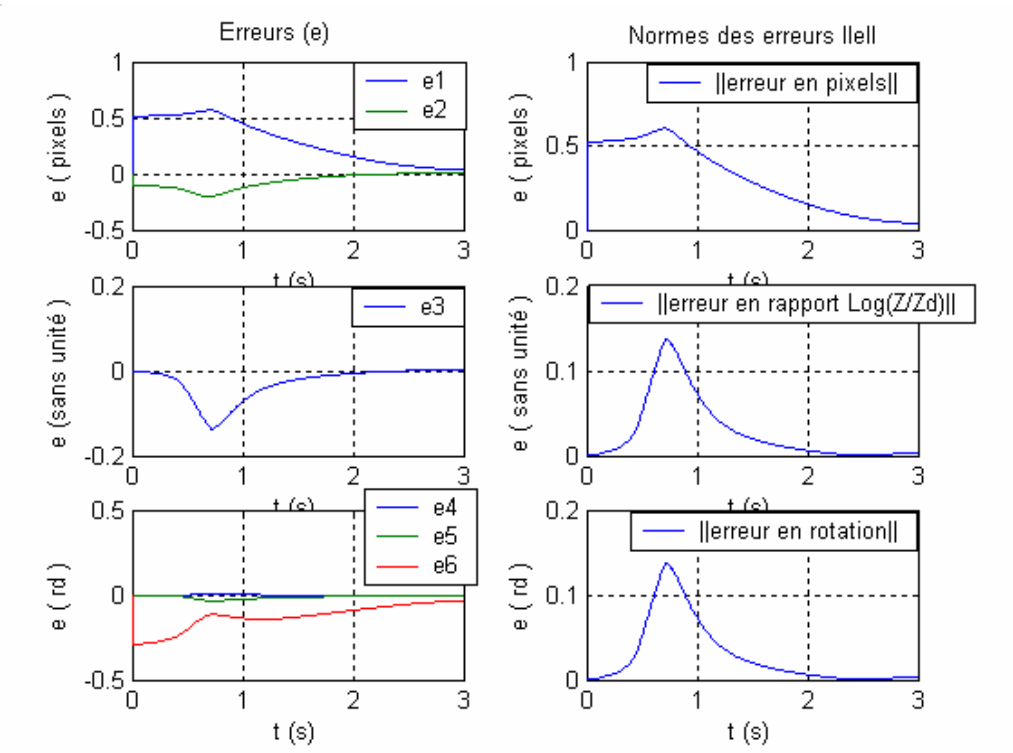


Figure 5.4.c. *Les erreurs pour une translation de -10 pixels sur u et 50 pixels sur v et une rotation de 15°, avec g_{max}=1.1 (s^{-1}) et a=3(s).*

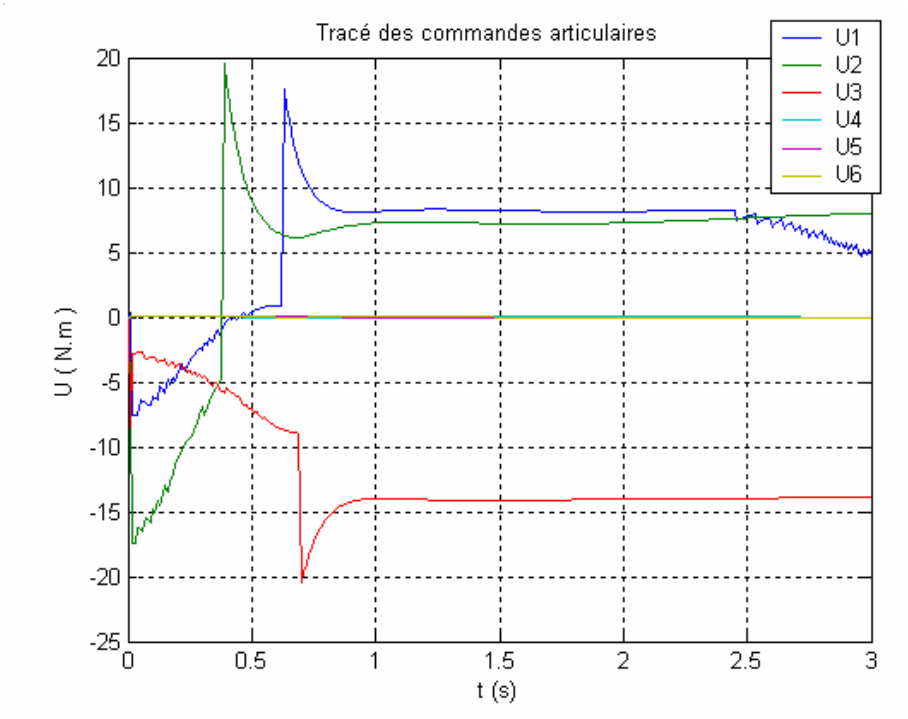


Figure 5.4.d. *Les couples articulaires pour une translation de -10 pixels sur u et 50 pixels sur v et une rotation de 15°, avec g_{max}=1.1 (s^{-1}) et a=3(s).*

Comme on peut le constater, le temps de convergence a nettement diminué (de l'ordre de 3s), il y a donc rapidité de convergence. D'autre part, la commande n'excède pas 20 N.m malgré la présence de brusques changements de signes des commandes. On remarque aussi l'obtention d'une bonne poursuite en articulations mais une poursuite moyenne en vitesses.

Prenant comme deuxième cas une translation de 50 pixels sur *u* et -100 pixels sur *v* avec g_{max}=1.1(s^{-1}) et *a=2(s)* :

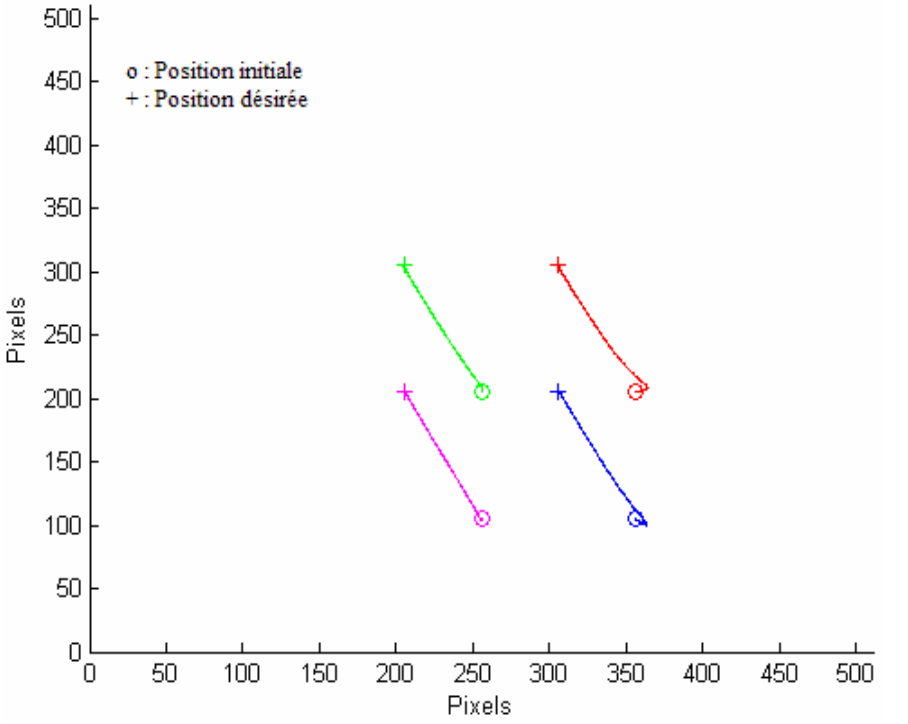


Figure 5.5.a. *Trajectoire des primitives visuelle pour une translation de 50 pixels sur u et -100 pixels sur v, avec g_{max}=1.1 (s^{-1}) et a=2(s).*

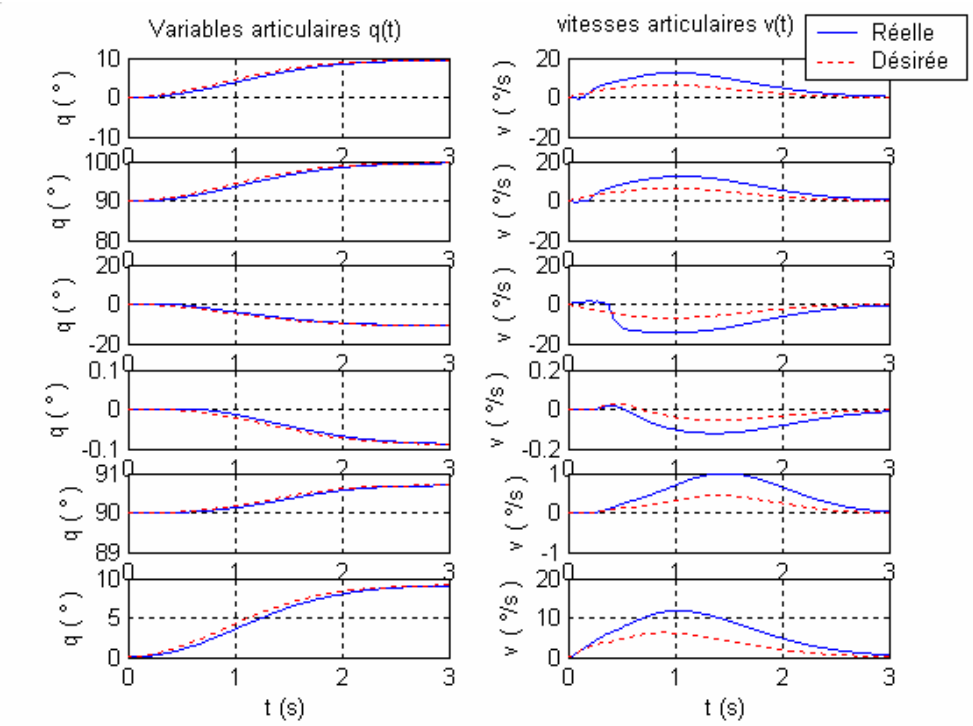


Figure 5.5.b. *Variables et vitesses articulaires pour une translation de 50 pixels sur u et -100 pixels sur v, avec g_{max}=1.1 (s^{-1}) et a=2(s).*

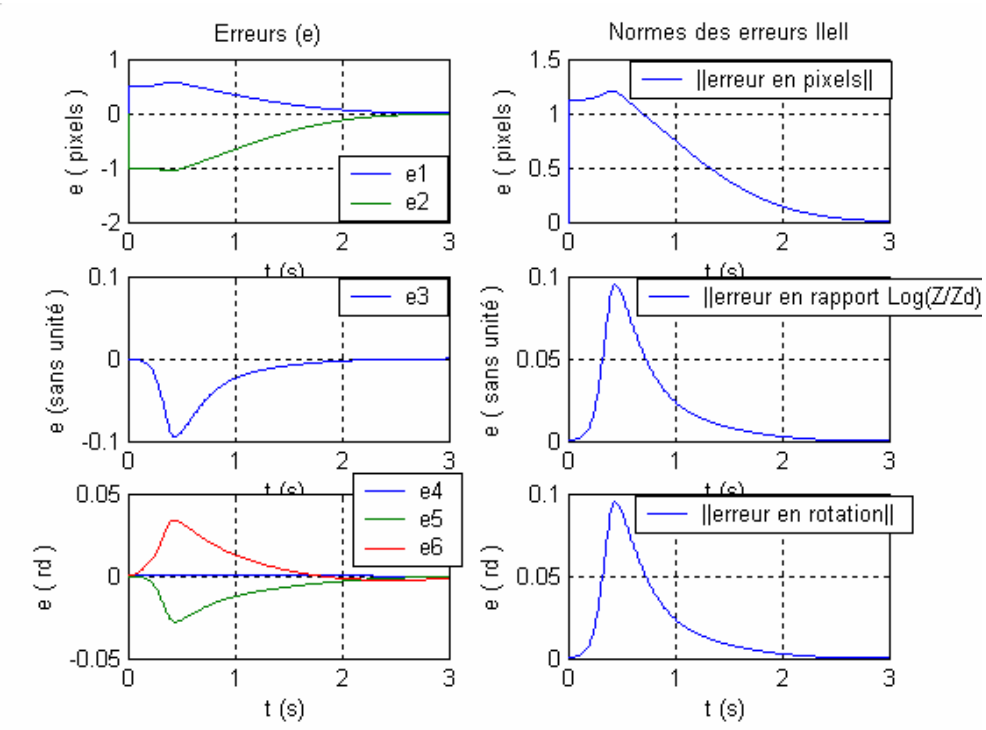


Figure 5.5.c. *Les erreurs pour une translation de 50 pixels sur u et -100 pixels sur v, avec g_{max}=1.1 (s^{-1}) et a=2(s).*

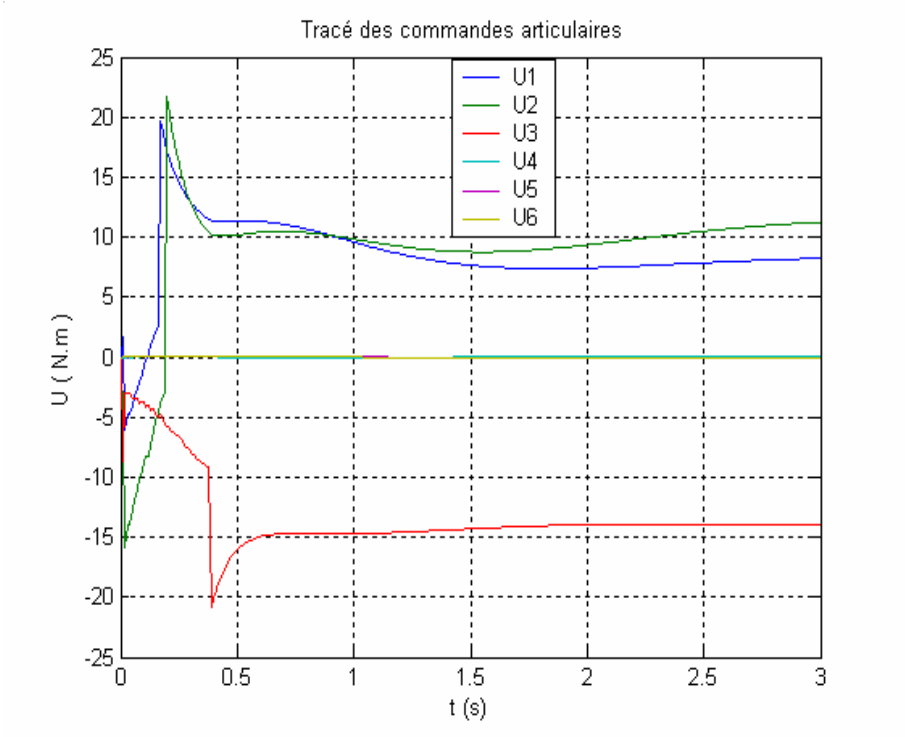


Figure 5.5.d. *Les couples articulaires pour une translation de 50 pixels sur u et -100 pixels sur v, avec $g_{max}=1.1\ (s^{-1})$ et a=2(s)*

Remarquons que le positionnement s'est correctement effectué en 3s comme dans le cas précédent. Les commandes sont admissibles et ne dépassant pas 21 N.m. Tentons à présent de simuler le comportement du robot dans le cas de présence de perturbations au niveau de la commande de l'ordre de 30%; on utilise le Backsteeping avec $\alpha = \beta = -15$ pour la commande du robot et la loi classique avec g adaptatif pour la boucle de vision. Prenons pour notre simulation une translation de 50 pixels sur u et 100 pixels sur v avec $g_{max}=4(s^{-1})$ et $a=1(s)$:

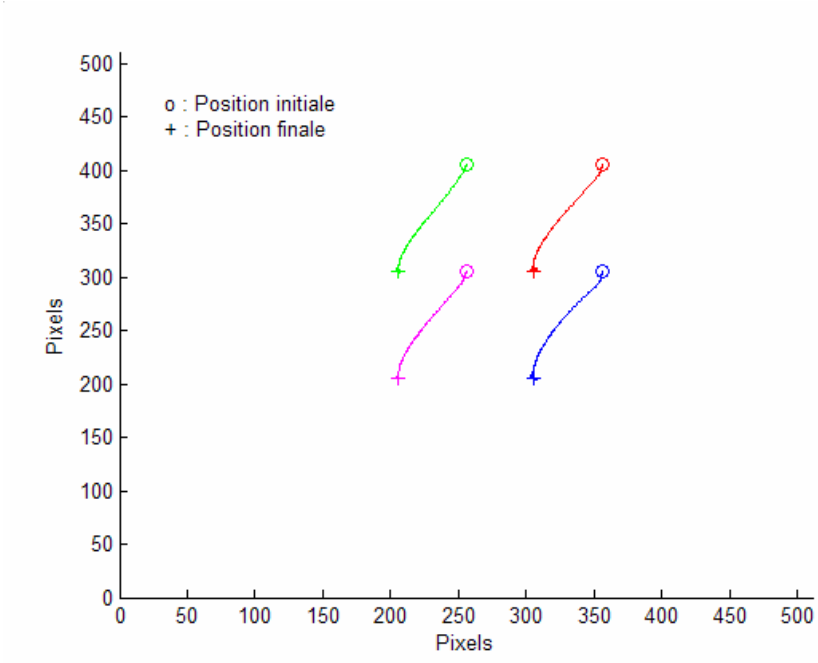


Figure 5.6.a. *Trajectoire des primitives visuelle pour une translation de 50 pixels sur u et -100 pixels sur v en présence des perturbations, avec $g_{max}=1.1\ (s^{-1})$ et a=2(s).*

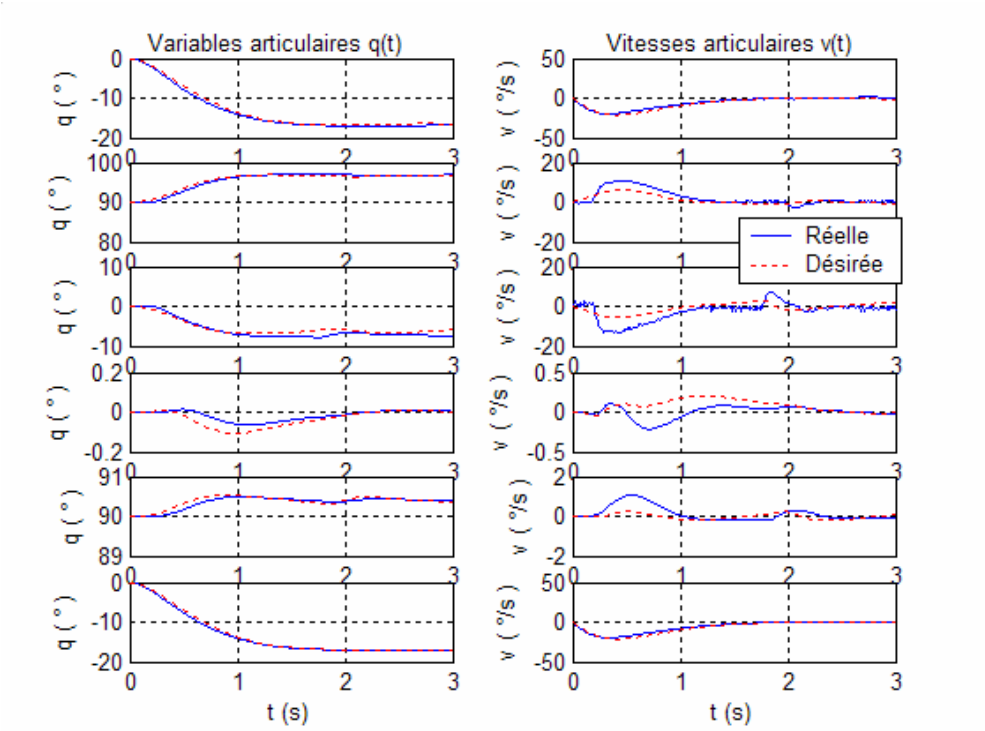


Figure 5.6.b. *Variables et vitesses articulaires pour une translation de 50 pixels sur u et -100 pixels sur v en présence des perturbations, avec g_{max}=1.1 (s^{-1}) et a=2(s).*

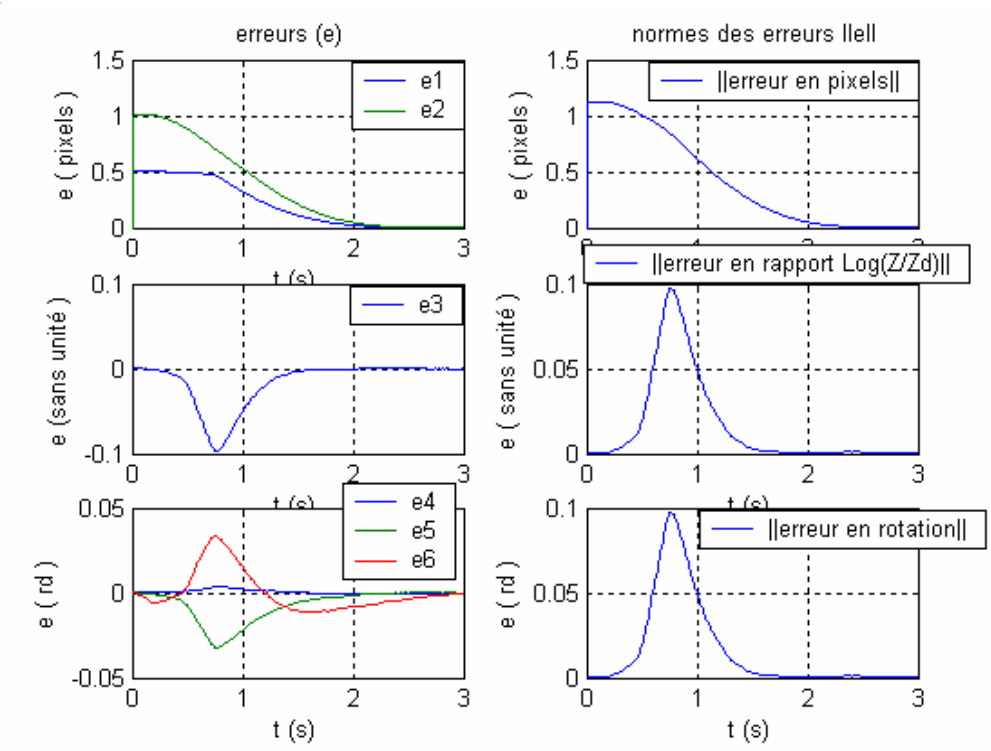


Figure 5.6.c. *Les erreurs pour une translation de 50 pixels sur u et -100 pixels sur v en présence des perturbations, avec g_{max}=1.1 (s^{-1}) et a=2(s).*

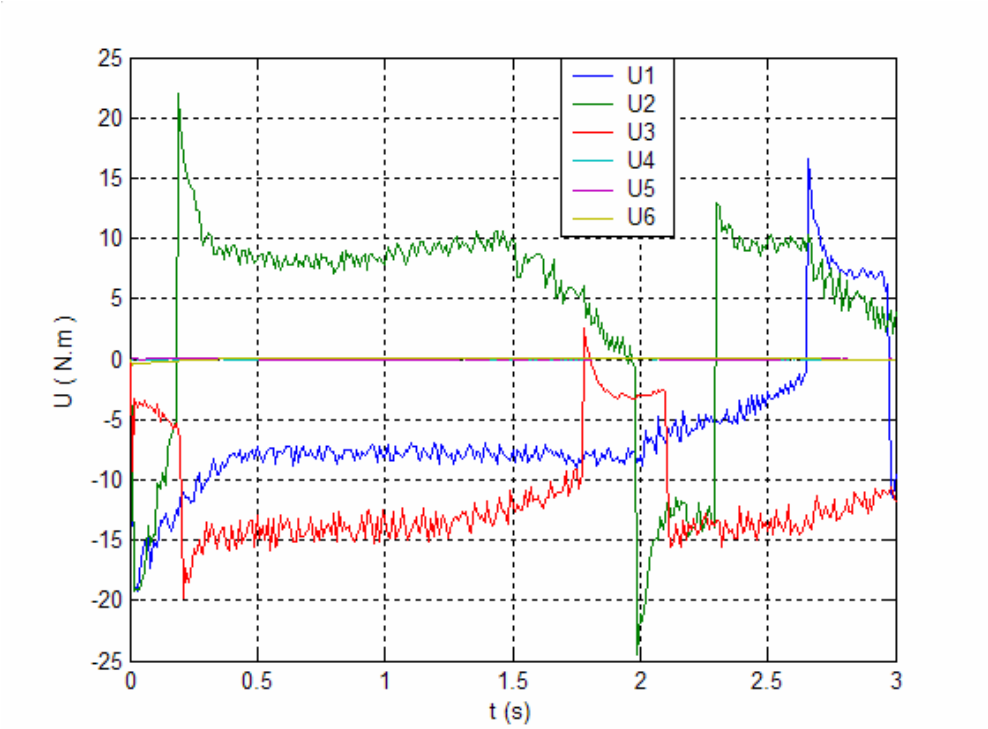


Figure 5.6.d. *Les couples articulaires pour une translation de 50 pixels sur u et -100 pixels sur v en présence des perturbations, avec* $g_{max}=1.1\ (s^{-1})$ *et a=2(s).*

On remarque que la convergence vers l'image désirée reste toujours rapide. Les articulations et leurs vitesses sont faiblement influencées par ces perturbations ; les erreurs se sont complètement annulées.

Contrairement à l'asservissement visuel 3D où les couples articulaires présentaient quelques broutements, l'asservissement 2D1/2 est complètement insensible aux perturbations.

Simulons à présent une tâche de poursuite. L'objet est en mouvement de translation sur une bande convoyeur. Sa vitesse et sa trajectoire sont inconnues pour le robot.

Voici la simulation d'un asservissement visuel 2D1/2 réalisant la tâche de poursuite, avec *g* adaptatif $g_{max}=5(s^{-1})$ *et a=3(s)* (la commande du robot est toujours le Backsteeping) :

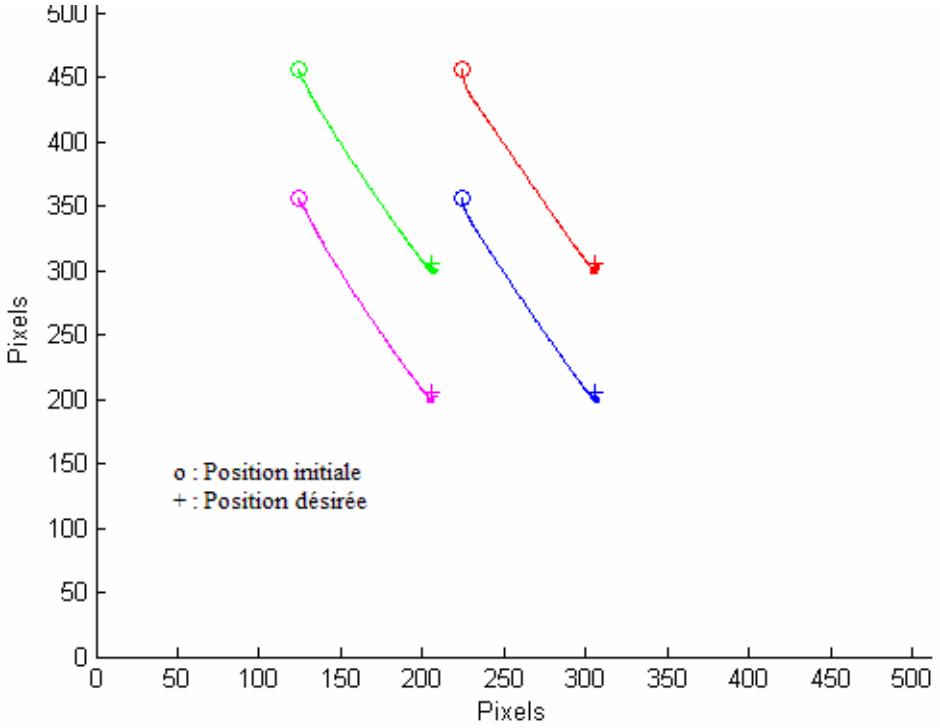


Figure 5.7.a. *Trajectoire dans l'image des primitives visuelle pour une tâche de poursuite, avec g variable : $g_{max}=5(s^{-1})$ et a=3(s).*

Nous constatons donc que la tâche de poursuite est correctement effectuée. Notons qu'au départ le robot n'avait aucune information sur la position de l'objet, sa trajectoire et sa vitesse.

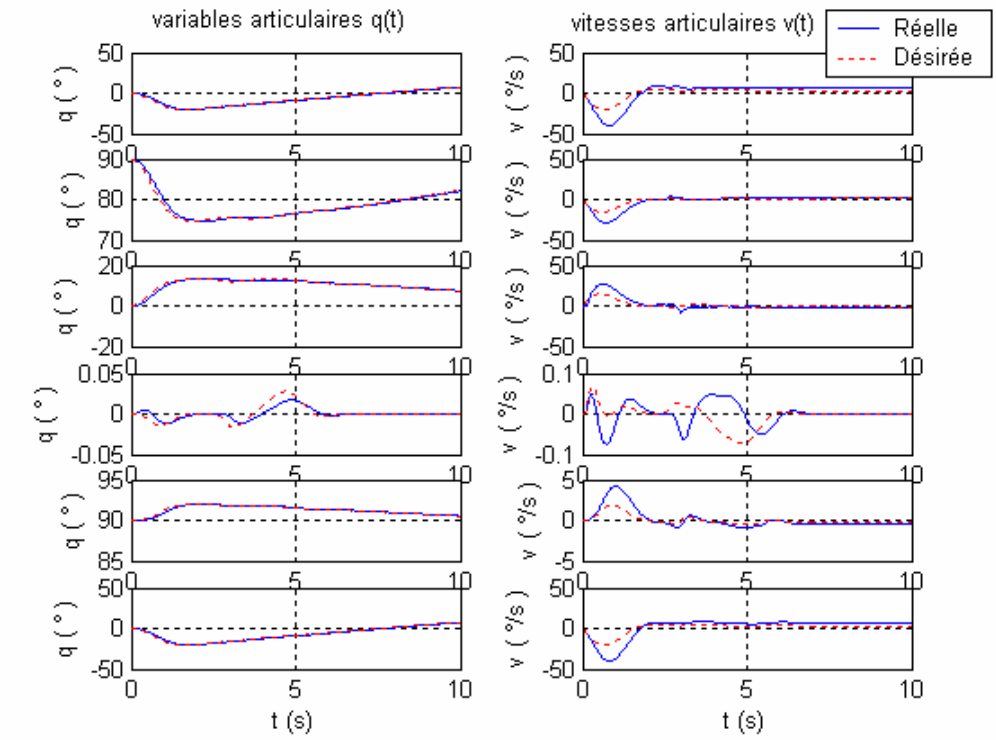


Figure 5.7.b. *Variables et vitesses articulaires pour une tâche de poursuite, avec g variable : $g_{max}=5(s^{-1})$ et a=3(s).*

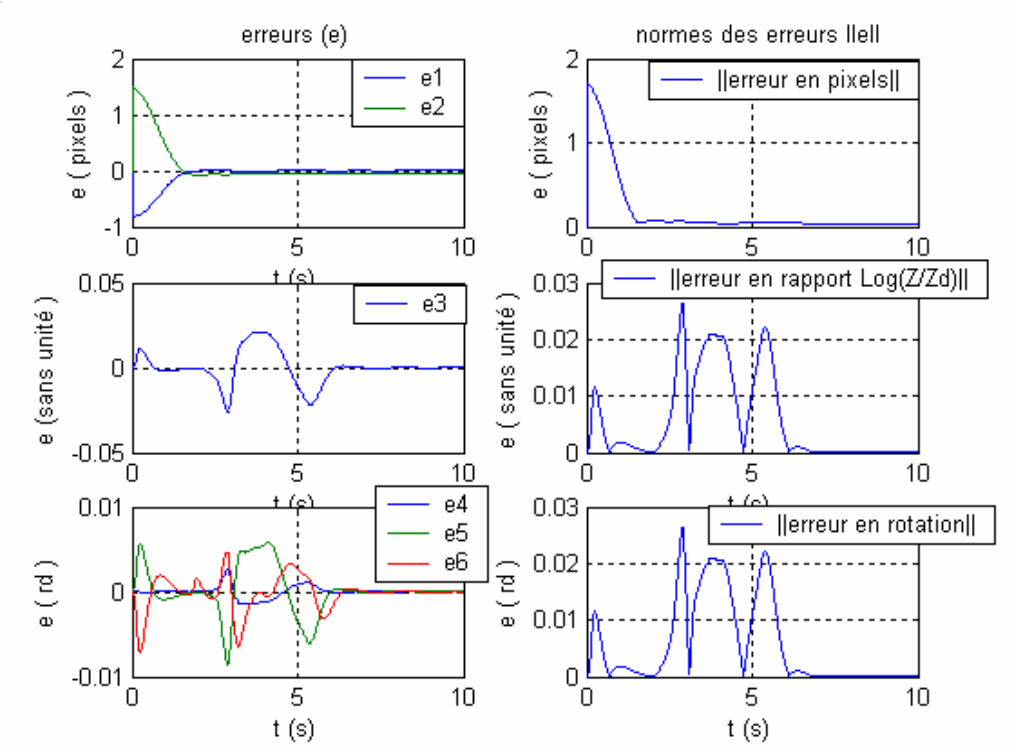


Figure 5.7.c. *Les erreurs pour une tâche de poursuite, avec g variable : $g_{max}=5(s^{-1})$ et a=3(s).*

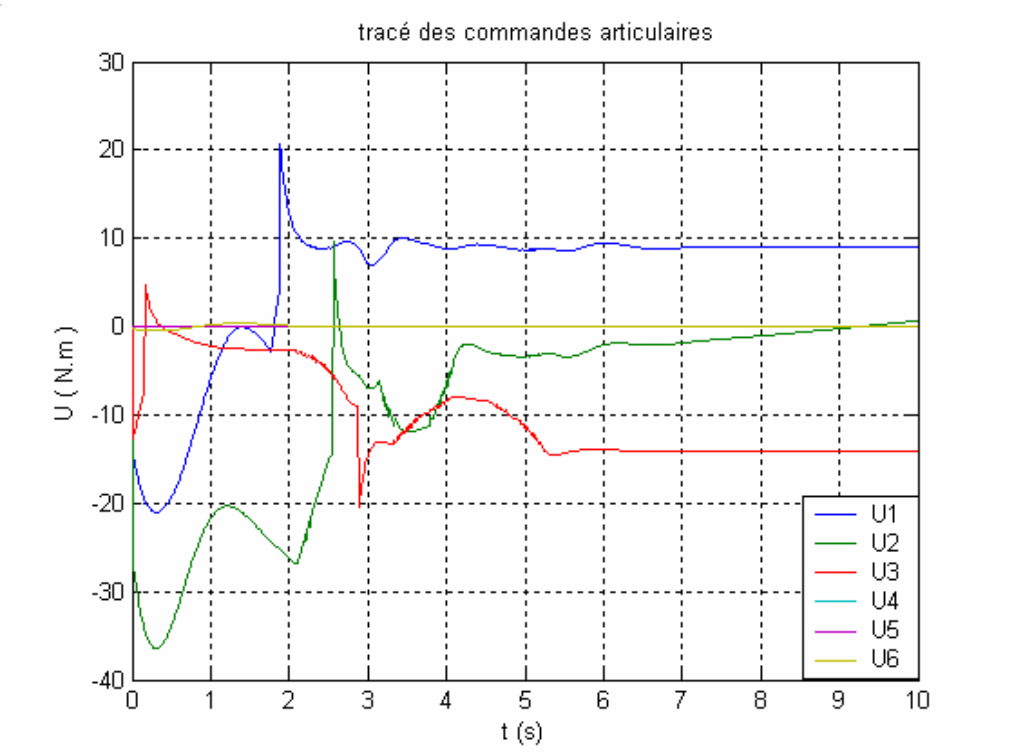


Figure 5.7.d. *Les couples articulaires pour une tâche de poursuite, avec g variable : $g_{max}=5(s^{-1})$ et a=3(s).*

On remarque que le fait d'effectuer une tâche de poursuite nous a contraint d'employer un gain maximum plus élevé ($g_{max}=5(s^{-1})$) afin d'effectuer correctement la tâche de poursuite, en tentant d'éliminer l'erreur de traînage, et de réduire la valeur du couple articulaire initial, qui n'atteint que 35 N.m dans notre cas.

5.5. Conclusion :

Dans l'asservissement visuel 2D1/2, les trajectoires dans l'image étaient moins courbées par rapport à l'asservissement 3D car une partie du contrôle s'effectue dans l'image.

D'autre part, le point de référence utilisé dans la fonction de tâche garantit qu'au moins un point de la cible restera dans le champ de vision de la caméra.

La poursuite en vitesses articulaires est effectuée d'une façon moins robuste dans ce type d'asservissement, comparé à l'asservissement 3D, car le contrôle est effectué en partie dans l'espace cartésien et en partie dans l'image ; contrairement à l'asservissement 3D où le contrôle est totalement effectué dans l'espace cartésien, d'où la possibilité d'appliquer la commande linéarisante et le glissement dans l'asservissement 3D et l'impossibilité de le faire pour les asservissements 2D et 2D1/2.

Conclusion générale

Ce mémoire porte sur l'étude de l'asservissement visuel. Après avoir décrit les différents types d'asservissement visuel et les outils nécessaires à l'introduction du capteur caméra comme *seul capteur extéroceptif*, nous abordons la commande liée à chaque type d'asservissement visuel, à savoir le 2D, 3D et 2D½. Diverses simulations sur le cas de positionnement et de poursuite sont réalisées afin de valider les résultats théoriques.

Notre contribution s'est portée sur l'élaboration de lois de commande robustes au niveau de la commande visuelle, tel que l'emploi d'un gain variable, commande linéarisante, commande par mode de glissement, commande par logique floue et commande par logique floue adaptative. Il n'est donc plus nécessaire de vouloir estimer certains paramètres tel que le mouvement propre de l'objet ; il suffit de considérer ce dernier comme une perturbation et d'appliquer les lois de commandes robustes, vis-à-vis des perturbations, pour assurer la convergence de la tâche robotique. Ces nouvelles commandes sont appliquées sur les trois types d'asservissement visuel étudié dans ce mémoire.

La comparaison entre les différents résultats obtenus permet de tirer les avantages et les inconvénients de chaque commande. Il est donc impossible d'affirmer à l'avance qu'une commande permet d'avoir de meilleurs résultats qu'une autre commande. En effet, d'après l'étude menée dans ce mémoire, la bonne réalisation d'une certaine tâche robotique par asservissement visuel dépend :

- Du robot asservis ;
- De la tâche à effectuer (positionnement, poursuite) ;
- Des critères de commandes (temps de convergence, couples max tolérés,…) ;
- Du type d'asservissement visuel utilisé ainsi que de la commande visuelle employée.

Il est donc nécessaire de simuler le comportement du robot afin de veiller à ce que ce dernier respecte le cahier de charge établi. Il est aussi nécessaire de tenir compte des limitations techniques tel que le temps de traitement, qui doit non seulement inclure le temps de calcul de la commande, mais aussi celui du traitement d'image.

Perspectives :

Ce mémoire étudie les différents types d'asservissement visuel d'un bras de robot, en y appliquant des commandes visuelles robustes. Il serait intéressant de traiter par la suite les thèmes suivants :

- Expérimenter les lois de commandes conçues dans ce mémoire afin de mieux les valider ;
- Développer des lois de commandes visuelles afin d'effectuer une tâche d'évitement d'obstacles ;
- Contrôler la trajectoire imposée par la commande visuelle afin d'éviter les configurations singulières du robot. En effet, selon l'image initiale, il est possible que le robot se trouve, à un certain moment, sur une configuration singulière, notamment lors de fort mouvement de rotation ;
- Appliquer les théories de l'asservissement visuel à d'autres systèmes que le robot, tel que l'aéronautique, l'aérospatiale ou la navigation par exemple.

ANNEXE A : Présentation du robot PUMA 560

A.1. présentation générale du robot PUMA 560 :

Dans cette annexe nous présenterons les différentes caractéristiques du robot PUMA 560. On tentera ensuite de modéliser ce dernier afin de le commander. Cependant, nous n'aborderons pas les différents modèles du robot (modèle géométrique et cinématique, direct et indirect) ; seul le modèle dynamique sera abordé dans le cas général.

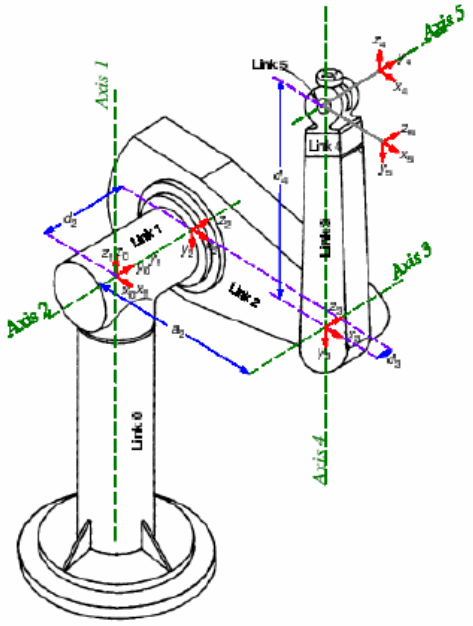


Figure A.1. *Système de coordonnées des liaisons et paramètres des articulations pour le bras de robot PUMA 560 à sa position d'origine.*

Les paramètres D-H du puma 560 sont donnés par le tableau suivant :

Articulation	θ_i	a_i (mm)	b_i (mm)	α_i
1	θ_1	0	0	-90°
2	θ_2	431.81	0	0°
3	θ_3	-20.31	149.10	+90°
4	θ_4	0	433.04	-90°
5	θ_5	0	0	+90°
6	θ_6	0	56.23 + 97.5	0°

Tableau A.1. *Paramètres D-H du PUMA 560.*

A.2. Modélisation dynamique des bras manipulateurs :

Le modèle dynamique est la relation entre les couples (et/ou force) appliqués aux actionneurs et les positions, vitesses et accélérations articulaires. On représente le modèle dynamique par une relation de la forme :

$$\Gamma = F(q, \dot{q}, \ddot{q}, f_e) \tag{A.1}$$

Avec :

- Γ : vecteur des couples/forces des actionneurs, selon que l'articulation est rotoïde ou prismatique. Dans la suite, on écrira tout simplement *couples* ;
- q : vecteur des positions articulaires ;
- $\dot{q}$: vecteur des vitesses articulaires ;
- $\ddot{q}$: vecteur des accélérations articulaires ;
- f_e : vecteur représentant l'effort extérieur (forces ou moments) qu'exerce le robot sur l'environnement.

L'équation dynamique peut aussi se mettre sous la forme générale :

$$\Gamma = A(q)\ddot{q} + C(q, \dot{q})\dot{q} + Q(q) + \Gamma_f + \Gamma_e \tag{A.2}$$

- A (q) est la matrice (n x n) de l'énergie cinétique, appelée aussi *matrice d'inertie* du robot ;
- $C(q, \dot{q})\dot{q}$ Vecteur de dimension (n x 1) représentant les couples/forces de *Coriolis* et des forces *centrifuges* ;
- Q(q) vecteur des forces/couples de gravité ;
- Γ forces/couples à appliquer aux articulations du robot ;
- $\mathbf{\Gamma}_f$ forces/couples de frottement ;
- $\mathbf{\Gamma}_e$ forces/couples que doivent fournir les actionneurs d'un robot pour que son organe terminal puisse exercer un effort statique $\mathbf{f_e}$ sur l'environnement.

Le robot PUMA 560 disposant de 6 degrés de libertés, l'expression du modèle dynamique de ce dernier serai trop longue et très encombrante. Vous pouvez vous référer à [ARM 86].

A.3. Mise sous forme d'équation d'état :

Définissons tout d'abord le vecteur d'état :

$$\underline{X} = \begin{pmatrix} q \\ \dot{q} \end{pmatrix} \tag{A.3}$$

Ce qui permet de réécrire l'équation (I.2) sous la forme suivant :

$$\Gamma = A(q)\ddot{q} + C(q,\dot{q})\dot{q} + Q(q) + \Gamma_f + \Gamma_e \tag{A.4}$$

En considérant les simplifications suivantes :

$$\begin{cases} H(q,\dot{q}) = C(q,\dot{q})\dot{q} + Q(q) \\ \Gamma_f = \underline{0} \quad \Gamma_e = \underline{0} \end{cases} \tag{A.5}$$

Ceci permet d'aboutir à l'équation d'état d'un bras manipulateur :

$$\sum : \begin{cases} \dot{q} \\ \ddot{q} = A^{-1}(q) \cdot (\Gamma - H(q,\dot{q})) \end{cases} \tag{A.6}$$

Remarquons que cette écriture est générale à tous les robots manipulateurs. Si le robot est à ***n*** degrés de libertés, le vecteur d'état ***X*** est de dimension ***2n***. Dans le cas du robot PUMA 560, le nombre de degré de liberté étant ***6***, l'équation vectorielle (I.6) se forme donc de ***12*** équations différentielles.

Toutes les commandes décrites dans ce mémoire tiendront comptes de l'équation (A.6). L'entrée du système étant la commande Γ exprimée en couple appliquée au robot, et de dimension ***6***. Remarquons aussi la forte non linéarité du système, de la forme :

$$\dot{X} = f(X) + g(X) \cdot \Gamma \tag{A.7}$$

Le robot étant un système physique, la commande à appliquer est bornée. Voici un tableau donnant les valeurs maximales en valeur absolue [ALI 05] :

Numéro de l'articulation	1	2	3	4	5	6
$\lvert\Gamma_{Max}\rvert$ (N.m)	97.6	186.4	89.4	24.2	20.1	21.3

Tableau A.2. *Couples maximaux applicables.*

ANNEXE B : Développement des différentes commandes appliquées au robot

B.1. Synthèse de la loi de commande par Backsteeping:

L'équation dynamique du robot PUMA 560 peut s'écrire d'une manière simple comme suit :

$$\Gamma = A(q)\ddot{q} + H(q,\dot{q}) \tag{B.1}$$

On pose le modèle d'état suivant :

$$\begin{cases} x_1 = q \\ x_2 = \dot{q} \end{cases} \tag{B.2}$$

On aura donc :

$$\begin{cases} \dot{x}_1 = \dot{q} = x_2 \\ \dot{x}_2 = \ddot{q} = A^{-1}(q)\,\Gamma - A^{-1}(q)\,H(q,\dot{q}) \end{cases} \tag{B.3}$$

On essayera de trouver une commande Γ pour avoir une poursuite des positions articulaires c'est-à-dire $q \to q_d$ donc une stabilisation de l'erreur $e = q - q_d \to 0$, cherchons cette commande par Backsteeping :

Etape1 : Cherchons x_2 pour que $x_1 \to q_d$ c'est-à-dire $e \to 0$

Prenons la fonction de Lyapunov $V_1 = \frac{1}{2} e^T e$ tel que V_1 est une fonction définie positive sur R^6 ; la dérivée de cette fonction nous donne :

$$\dot{V}_1 = e_1\dot{e}_1 = (x_1 - q_d)(\dot{x}_1 - \dot{q}_d) = (x_1 - q_d)(x_2 - \dot{q}_d)$$

Si on prend : $\varphi_1 = \dot{q}_d + \alpha(x_1 - q_d)$ avec $\alpha < 0$

Alors : $\dot{V}_1 = -\alpha(x_1 - q_d)^2$ fonction définie négatives sur R^6

Etape2 : Cherchons u pour que $x_2 \to \varphi_1$

Prenons la fonction de Lyapunov $V_2 = \frac{1}{2}(x_1 - q_d)^2 + \frac{1}{2}(x_2 - \varphi_1)^2$ tel que V_2 fonction définie positive sur R^{12} ; Sa dérivée nous donne :

$$\dot{V}_2 = (x_1 - q_d)(\dot{x}_1 - \dot{q}_d) + (x_2 - \varphi_1)(\dot{x}_2 - \dot{\varphi}_1)$$

$$\dot{V}_2 = \alpha(x_1 - q_d)^2 + (x_1 - q_d + \dot{x}_2 - \dot{\varphi}_1)(x_2 - \varphi_1)$$

$$\dot{V}_2 = \alpha(x_1 - q_d)^2 + (x_1 - q_d + A^{-1}(q)u - A^{-1}(q)H(q,\dot{q}) - \dot{\varphi}_1)(x_2 - \varphi_1)$$

Si on prend $\Gamma = A(q)\left[A^{-1}(q)H(q,\dot{q}) - x_1 + q_d + \dot{\varphi}_1 + \beta(x_2 - \varphi_1)\right]$ avec : $\beta < 0$

Alors : $\dot{V}_2 = \alpha(x_1 - q_d)^2 + \beta(x_2 - \varphi_1)^2$ Fonction définie négative sur R^{12}

Après l'explicitation des expressions de φ_1 et de $\dot{\varphi}_1$ on aboutira à l'expression finale de la commande stabilisante Γ :

$$\Gamma = H(q,\dot{q}) + A(q)\left[\ddot{q}_d + (\dot{q} - \dot{q}_d)(\alpha + \beta) - (q - q_d)(1 + \alpha\beta)\right] \tag{B.4}$$

B.2. Synthèse de la loi de commande par linéarisation en articulations:

La représentation d'état du robot est donnée par les équations (B.2) et (B.3) :

$$\begin{cases} \dot{x}_1 = \dot{q} = x_2 \\ \dot{x}_2 = \ddot{q} = A^{-1}(q)\,\Gamma - A^{-1}(q)\,H(q,\dot{q}) \end{cases}$$

Pour trouver la forme normale on suit les étapes suivantes :

$y = x_1$

$\dot{y} = \dot{x}_1 = x_2$

$$\ddot{y} = \dot{x}_2 = A^{-1}(x_1)u - A^{-1}(x_1)H(x_1, x_2) \qquad \text{Avec} \begin{cases} L_g L_f h(x) = A^{-1}(x_1) \\ L_{f^2} h(x) = -A^{-1}(x_1)H(x_1, x_2) \end{cases}$$

Alors le degré relatif est r =2.

La commande linéarisante sera calculée comme suit : $\Gamma = \dfrac{1}{L_g L_f h(x)}(-L_{f^2} h(x) + \mathcal{V})$

Alors : $\Gamma = A(x_1)[A^{-1}(x_1)H(x_1, x_2) + \mathcal{V}]$ Finalement on aura :

$$\Gamma = H(x_1, x_2) + A(x_1)\ \mathcal{V} \tag{B.5}$$

La forme normale sera écrite :

$$\begin{cases} z_1 = y = x_1 \\ z_2 = \dot{y} = \dot{x}_1 = x_2 \end{cases} \text{ alors : } \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} 0_{6\times6} & \mathrm{I}_{6\times6} \\ 0_{6\times6} & 0_{6\times6} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} 0_{6\times6} \\ \mathrm{I}_{6\times6} \end{pmatrix} \mathcal{V} \tag{B.6}$$

Pour la commande du système linéarisé on utilise le retour d'état :

$$\mathcal{V} = -KZ + N_r r \quad \text{et} \quad \dot{Z} = AZ + B\,\mathcal{V} \quad \text{donc :} \quad \dot{Z} = (A - BK)Z + BN_r r \tag{B.7}$$

Pour notre cas la sortie $y = x_1 = q$ c'est-à-dire les positions articulaires alors la référence $r = x_{1d} = q_d$ c'est-à-dire les positions articulaires désirées.

B.3. Synthèse de la loi de commande par linéarisation en vitesses articulaires:

D'après l'équation d'état donnée par (B.3), pour trouver la forme normale afin d'avoir une poursuite en vitesse, on suit les étapes suivantes :

$y = x_2$

$$\dot{y} = \dot{x}_2 = A^{-1}(x_1)\Gamma - A^{-1}(x_1)H(x_1, x_2) \qquad \text{Avec :} \begin{cases} L_g h(x) = A^{-1}(x_1) \\ L_f h(x) = -A^{-1}(x_1)H(x_1, x_2) \end{cases}$$

Alors : $\Gamma = A(x_1)[A^{-1}(x_1)H(x_1, x_2) + \mathcal{V}$

Finalement on aura : $\Gamma = H(x_1, x_2) + A(x_1)\,\mathcal{V}$

La forme normale sera écrite :

$$\begin{cases} z_1 = y = x_2 \\ z_2 = \lambda(x) \end{cases} \tag{B.8}$$

Pour calculer $\lambda(x)$ on utilise les deux conditions suivantes :

$$\begin{cases} \left|\dfrac{\partial \phi}{\partial x}\right| \neq 0 \quad \text{(Jacobien)} \\ L_g \lambda_i = 0 \end{cases} \tag{B.9}$$

$$\begin{cases} \begin{vmatrix} 0 & 1 \\ \dfrac{\partial \lambda}{\partial x_1} & \dfrac{\partial \lambda}{\partial x_2} \end{vmatrix} \neq 0 \Rightarrow \dfrac{\partial \lambda}{\partial x_1} \neq 0 \\ L_g \lambda = 0 \Rightarrow \begin{bmatrix} \dfrac{\partial \lambda}{\partial x_1} & \dfrac{\partial \lambda}{\partial x_2} \end{bmatrix} \begin{bmatrix} 0 \\ A^{-1}(x_1) \end{bmatrix} = 0 \Rightarrow A^{-1}(x_1) \dfrac{\partial \lambda}{\partial x_2} = 0 \Rightarrow \dfrac{\partial \lambda}{\partial x_2} = 0 \end{cases} \Rightarrow \lambda(x) = x_1$$

La forme normale sera réécrite comme suit :

$$\begin{cases} z_1 = x_2 \\ z_2 = x_1 \end{cases} \quad \text{Donc :} \quad \begin{cases} \dot{z}_1 = \mathrm{V} \\ \dot{z}_2 = z_1 \end{cases} \tag{B.10}$$

Quand z_1=0 on aura $\dot{z}_2 = 0$

Alors le système sera à minimum de phase lorsque z_2= q= *cste*

Pour ce cas la sortie $y = x_2 = \dot{q}$ c'est-à-dire les vitesses articulaires, alors la référence $r = x_{2d} = \dot{q}_d$ c'est-à-dire les vitesses articulaires désirées.

B.3. Synthèse de la loi commande par mode de glissement:

L'objectif que nous nous fixons est d'élaborer une loi de commande permettant d'amener la trajectoire d'état sur la surface de glissement, notée *S(X)*. Il serait donc intéressant de définir cette surface comme une *expression vérifiant les objectifs de commande*. Ainsi *S(X)* est définie telle que :

Objectifs de commande atteints ⇔ *S(X)* = 0

La commande recherchée *U* doit pouvoir ramener le vecteur d'état, des positions initiales quelconques, vers la surface de glissement et le maintenir sur cette surface.

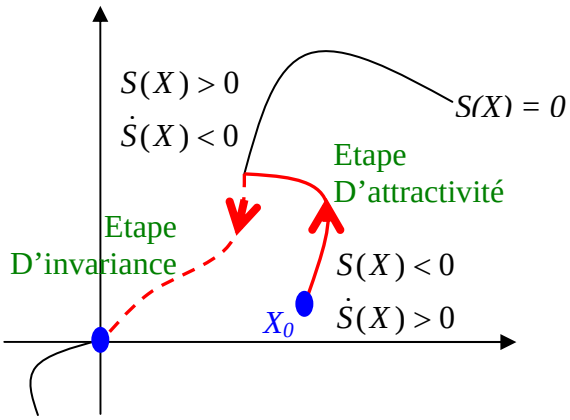


Figure B.1. *Evolution désirée de la trajectoire d'état, dans le cas d'une commande par mode de glissement.*

Ce qui permet d'écrire les deux conditions d'attractivité et d'invariance :

- Condition d'attractivité : $S(X)\cdot\dot{S}(X)<0$;
- Condition d'invariance : $\dot{S}(X)=0$ pour $S(X)=0$.

De ce fait, la loi de commande par mode de glissement (commande à structure variable) , pour un système d'état de la forme : $\dot{X}=f(X)+g(X)\cdot U$,s'écrit :

$$U=\left[\frac{\partial S(X)}{\partial X}\cdot g(X)\right]^{-1}\cdot\left(-\frac{\partial S(X)}{\partial X}\cdot f(X)-K\cdot Sign\big(S(X)\big)\right),\quad K>0 \tag{B.11}$$

Cette loi de commande est bel et bien formée des deux termes : attractivité et invariance. On montre bien que celle-ci vérifie les deux conditions citées précédemment :

- $S(X)\cdot\dot{S}(X)=-K\cdot|S(X)|<0 \qquad \forall S(X)\neq 0$ et
- $\dot{S}(X)=-K\cdot Sign\big(S(X)\big)=0 \qquad si \quad S(X)=0$.

Il est très important de noter que dans cette loi de commande, $\frac{\partial S(X)}{\partial X}\cdot g(X)$ doit être inversible, ce qui limite encore le choix de *S(X)*.

Seul problème, la commande *U* dépendant du signe de *S(X)*, le système sera sujet à de grandes oscillations. Ces dernières peuvent être limités en remplaçant la fonction *Sign* par la fonction *Sat* (Saturation), tel que décrit dans la fonction (B.11) :

$$U=\left[\frac{\partial S(X)}{\partial X}\cdot g(X)\right]^{-1}\cdot\left(-\frac{\partial S(X)}{\partial X}\cdot f(X)-K\cdot Sat\big(S(X)\big)\right),\quad K>0 \tag{B.12}$$

$$\text{Avec : } Sat(A)=\begin{cases} A & si \quad |A|\le\varepsilon \\ \varepsilon\cdot Sign(A) & si \quad |A|\ge\varepsilon \end{cases}.$$

Application de la loi de commande par mode de glissement au robot PUMA 560 :

Sachant que nous traitons ici un problème de poursuite, définissons les erreurs en positions et vitesses articulaires comme suit :

$$\begin{pmatrix} e_q \\ e_{\dot{q}} \end{pmatrix}=\begin{pmatrix} q-q_d \\ \dot{q}-\dot{q}_d \end{pmatrix} \tag{B.13}$$

Nous devons tout d'abord définir la surface de glissement de façon à vérifier les objectifs de commande :

$$S=\dot{e}_q+\lambda\cdot e_q \qquad \text{Avec } \lambda>0 \tag{B.14}$$

Sur cette surface, les objectifs de commande sont vérifiés, car en *S=0* :

$$\dot{e}_q=-\lambda\cdot e_q \quad \Rightarrow \quad e_q\to 0 \quad Et \quad \dot{e}_q=e_{\dot{q}}\to 0$$

Il est donc possible maintenant de calculer la commande par mode de glissement Γ selon (B.12), et en considérant :

$$f(X)=\begin{pmatrix} \dot{q} \\ -A^{-1}(q)\cdot H(q,\dot{q}) \end{pmatrix} \quad et \quad g(X)=\begin{pmatrix} 0 \\ A^{-1}(q) \end{pmatrix}$$

$$\Gamma=H(q,\dot{q})+A(q)\cdot\big(-\lambda\cdot\dot{q}-K\cdot Sat(S(X))\big) \tag{B.15}$$

C'est en agissant sur *K* et λ que nous pourrons modifier les performances de la commande.

B I B L I O G R A P H I E

[AHM 04] Ahmad M. Ibrahim, - *Fuzzy logic for embedded systems applications* – Elsevier Science (USA), 2004.

[ALI 05] M. Ali-Bey., - *Commande en effort des robots manipulateurs* – L'Université Mouloud Mammeri de Tizi-Ouzou, Juin 2005.

[ALL 93] P.K. Allen., B. Yoshimi., A. Timcenko., P. Michelman., - *Automated tracking and grasping of a moving object with a robotic hand-eye system – IEEE Transactions on Robotics and Automation*, vol. 9, n°2, p. 152-165, 1993.

[ARM 86] B. Armstrong., O. Khatib., J. Burdick., - *The explicit dynamic model and inertial parameters of the PUMA 560 arm* - dans *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, p. 510 – 518, 1986.

[CHA 90] F. Chaumette., - *La relation vision-commande : théorie et application à des tâches robotiques.* - Thèse de l'Université de Rennes I, IRISA, Juillet 1990.

[CHA 98] F. Chaumette., – *Potential problems of stability and convergence in image-based and position-based visual servoing* – dans *The confluence of vision and control*, Springer-Verglas, LNCIS 237, p. 66-78, 1998.

[COL 98] E Colle., F Chavand., - *Perception de l'environnement en robotique* - Paris, Hermès, 1998.

[DOR 95] F. Dornaika., - *Contributions à l'intégration Vision/Robotique : Calibrage, Localisation et Asservissement.* – Thèse de l'INP Grenoble, Septembre 1995.

[GAN 98] J. Gangloff., M. De Mathelin., G. Abba., - *6 DOF high speed dynamic visual servoing using GPC controllers* – dans *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'98, Louvain)*, p.1008-1013, 1998.

[GAN 99] J. Gangloff., - *Asservissements visuels rapides d'un robot manipulateur à dix degrés de liberté* – Thèse de doctorat, Université Lois Pasteur de Strasbourg, LSIIT, 1999.

[HAS 93] K. Hashimoto., H. Kimura., – *LQ optimal and non linear approach to visual servoing* – dans Hashimoto K. (dir.), *Visual servoing : Real time control of robot manipulator based on visual sensory feedback*, World Scientific Press, Singapour, World Scientific Series in Robotics and Automated Systems, vol 7, 1993.

[HAS 95] K. Hashimoto., H. Kimura., - *Visual servoing with non linear observer* – dans *Proceeding of the IEEE International Conference on robotics and Automation (ICRA'95, Nagoya)*, p. 484-489, 1995.

[KHA 99] W. Khalil., E. Dombre., – *Modélisation, identification et commande des robots.* – Hermès, Paris, deuxième édition, 1999.

[KHA 02] W. Khalil., – *Commande des robots manipulateurs*. – Hermès Science, 2002.

[MAL 98] E. Malis., - *Contributions à la modélisation et à la commande en asservissement visuel*. – Thèse de l'Université de Rennes I, IRISA, Novembre 1998.

[PAP 93] N. Papanikolopoulos., P.K.Khosla., 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*. Vol. 9, n°1, p. 14-35, 1993.

[RIZ 96] A. Rizzi., D. Koditschek., - *An active visual estimator for dexterious manipulation – IEEE Transactions on Robotics and Automation*. Vol. 12, n°5, p. 697-713, 1996.

[SHE 90] R.J. Shelling., - *Fundamentals of robotics: Analysis and Control* - New Jersey, Prentice Hall, 1990.

[WIL 96] W.J. Wilson., C.C.W. Hulls., G.S. Bell., -*Relative end-effector control using Cartesian position-based visual servoing - IEEE Transactions on Robotics and Automation*. Vol. 12, n°5, p. 684-696, Octobre 1996.

[ZAN 03] P. Zanne., -*Contribution à l'asservissement visuel robuste*.- Thèse de l'université Louis Pasteur, Strasbourg I, Juillet 2003.