RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Ecole Nationale Polytechnique

Département d'Automatique

Projet de Fin d'Etude

Mémoire en vue de l'obtention du diplôme d'Ingénieur d'état en Automatique

Thème :

Commande référencée vision du robot Scara à 3dll

Réalisé par :

Mlle Salima IMOUDACHE
Mlle Lyna KHERBOUCHE

Encadré par :

Mr. O. STIHI

Juin 2013

E.N.P- 10, Avenue Hassen Badi, 16200 El Harrach, Alger

ملخص : التحكم البصري لذراع آلي من نوع SCARA

العمل المقدم من خلال هذه المذكرة يتمثل في دراسة التحكم البصري ثنائي الأبعاد لذراع آلي من نوع SCARA مجهز بواجهة PMAC. المهام التي تم القيام بها هي التموقع بالنسبة لأربعة نقاط وتتبع خط . محاكاة هذه المهام تمت وفق ثلاث انظمة تحكم مختلفة وهي: التحكم الكلاسيكي ، التحكم بمنظم PID والتحكم بالنظام الإنزلاقي و من إجراء تطبيق بصري على الآلي ،قمنا بدراسة خوارزميات معالجة الصور : خوارزمية, SIFT, ORB,SURF لاستخراج وموافقة نقاط خاصة في الصور و خوارزمية HOUGH للكشف عن الخطوط.

الكلمات المفتاحية : PMAC ، ذراع آلي SCARA، تحكم بصري ثنائي البعد ، بمنظم PID، التحكم الإنزلاقي، معالجة الصور ، HOUGH ، SURF, SIFT, ORB

Résumé : Commande référencée vision d'un bras manipulateur SCARA DDL Le problème traité dans ce travail est l'étude d'un asservissement visuel 2D d'un bras manipulateur de type SCARA DDL commandé par une interface PMAC de Delta Tau. Deux taches sont réalisées a savoir, le positionnement par rapport a quatre points et le suivi de ligne. Des simulations de ces taches sont données pour trois lois de commande : La commande classique, le régulateur PID et le mode glissant. A fin de permettre une application de vision sur le robot, des algorithmes de traitement d'images ont étaient étudiés : les algorithmes SURF, SIFT et ORB pour l'extraction et l'appariement de primitives et la transformée de Hough pour la détection de droites.

Mots clés : PMAC, robot SCARA, Asservissement 2D, régulateur PID, commande par mode glissant, SURF, SIFT, ORB, Hough.

abstract : Visual-based control applied to 3DOF SCARA robotic arm This wok mainly addresses the problem of studying a 2D visual servoing applied on a 3DOF SCARA robot arm controlled by a Delta tau PMAC interface. The two missions made ere image based positioning and line tracking. Simulations of these missions are given for three control laws : the classical control law, the PID controller and the sliding mode law. To allow a vision application on the robot, image processing algorithms were studied : the SURF, SIFT and ORB algorithms for feature extraction and matching and the hough transform for lines detection. Medskip

Key words : PMAC, SCARA robot, 2D visual servoing, PID controller, Sliding mode, SURF, SIFT, ORB, Hough.

Remerciements

Nous tenons a exprimer nos remerciements a Mr Omar Stihi pour nous avoir encadré, pour sa disponibilité et pour nous avoir facilité l'accès au LCP.

Nous tenons aussi a exprimer notre gratitude a Mr Noureddine Ouadah du CDTA, pour sa précieuse aide, sa disponibilité et sa gentillesse. Nous lui adressons nos vifs remerciements pour nous avoir fait l'honneur de faire partie de notre jury.

Nos remerciements vont aux membres de jury qui nous font l'honneur d'évaluer notre travail.

Nous remercions également vivement Mme Latifa Hamami enseignante a l'ENP pour avoir recu nos questions, pour nous avoir consacré de son temps et avoir fait de son mieux pour nous aider. Sans oublier Mr Rachid Illoul enseignant a l'ENP, pour sa perpétuelle bonne humeur et ses précieux conseils.

Nous tenons a exprimer notre reconnaissance a toute personne qui a contribué par son aide a ce travail, particulièrement, Mr Boussif Abderraouf ingénieur automatique de l'ENP, qui a toujours était serviable et qui n'a jamais manqué de répondre a nos questions.

Nous ne pouvons terminer ces remerciements sans exprimer notre gratitude a deux personnes qui nous ont marqué par leur gentillesse, leur grandeur d'ame et leur sens du service ; Ami Saleh de l'ENP sans qui, l'école polytechnique, et en particulier l'ancienne bibliothèque ne seraient pas les memes, ainsi que Mme Fahssil du service formation du CDTA.

Enfin ; nous rendons a travers ce mémoire un hommage aux personnes a qui revient tout le mérite ; nos parents qui nous ont soutenu et qui ont tant sacrifié pour nous.

Kherbouche-Imoudache

Dédicaces

*Aux personnes qui ont fait de moi ce que je suis aujourd'hui : mes anges gardiens ;Jeddi Bouhou et Yemma Touma ; et mes chers parents, Vava et Yemma ;A ma chère soeur, Ilhem ;A mes merveilleux frères, Merzouk et Mustafa ;A mon cher oncle, Idir et sa femme,A mes chères tantes et leurs maris,en particulier Hakima et Azdine qui etaient mon deuxième foyer tout au long de ces 5 années.A ma merveilleuse tribu de cousins : Yanis, Hakim, Iles, Aksil, Milissa, Lydia, Nassim, Yacine, Salima, Massi, Katia et Salim,ainsi qu'a Dania et Sonia.A ceux de ma famille qui ne sont plus de ce monde,A la terre qui porte leurs tombes,A Akbou, la ville qui m'a vu grandir ;A mes deux meilleures amies, Dyhia et Nesrine, sans qui ma vie d'étudiante ne serait pas la meme,A ma binome, Salima avec qui j'ai passé tant de moments,A tous mes amis de polytech, surtout Amina, Lamia et Nejla.A tous ceux qui ont peuplé mes 5 années d'études ger ;*Aux souvenirs de nos meilleures années...

Lyna

Dédicaces

A ceux, qui par leurs amour, leurs sacrifices et leurs soutient ont fait de moi la personne que je suis aujourd'hui ; A ce que je possède de plus cher au monde ; mes parents ;

A mes chers grands-parents, Jeddi Seddik , Jidda et Ayyi.

A la mémoire de mon ancle, Khali Mouhammed ;

A mes chers frères, mes chers cousins et cousines, particulièrement Karima ;

A mon amie et binome, Lyna qui m'a tant supporté ;

A tous mes amis pour tous les bons moments ;

A 3ami Saleh pour sa gentillesse et sa grandeur d'ame ;

A Mr Tidjini ;

Salima

Table des matières

Table des figures

Liste des tableaux

Abréviations et Symboles

Abréviations

K	Énergie cinétique
U	Énergie potentielle
L	Lagrangien du système mécanique
X	Vecteur des coordonnées opérationnelles
q	Vecteur des coordonnées généralisées
J	Matrice jacobienne
ω	Vitesse de rotation(rad/s)
$\dot{q}$	Vecteur commande du robot
R_α	Repère cartésien orthonormé associé à α
$[a]_x$	Matrice symétrique de pré-produit vectoriel associée au vecteur a
V_c	Torseur cinématique du robot
α_{V_b}	Matrice de changement de repère d'un torseur cinématique
T_i^j	Transformation homogène entre le repère i et le repère j
s	Vecteur des indices visuels
u	Coordonnée horizontale dans l'image
v	Coordonnée verticale dans l'image
L_s	Matrice d'interaction
L_s^+	Pseudo inverse de la matrice d'interaction
C	Matrice de combinaison
e	Fonction de tâche
f	Distance focale de la camera (mm)
U_0, V_0, α_u, α_v	Caractéristiques intrinsèques de la camera
k_u	Facteur d'échelle horizontal
k_v	Facteur d'échelle vertical
σ	Surface de glissement
λ	Gains de l'asservissement visuel
P	Action proportionnelle
I	Action intégral

Introduction générale

Quand on parle de la robotique, plusieurs idées viennent à l'esprit de chacun de nous. Historiquement, nous pourrions nous référer aux premiers concepts et automates de l'antiquité ou aux premiers robots comme à des personnages de la mythologie. Le mot *Robot* a d'ailleurs était introduit par *Karel capek*(1890–1938), un écrivain de science fiction, dans sa pièce *Rossum's Universal Robots* en 1921. Depuis, le mot *Robot* ne cesse de se rapprocher de sa désignation d'origine, car en robotique on cherche parfois à (faire réalité la fiction). Un exemple de l'influence de cette fiction nous est donné par les lois de la robotique.

Le développement de la robotique a était poussé par les besoins industriels d'après guerre, période qui a vu l'avènement des premiers bras manipulateurs dans l'industrie. Depuis, ce domaine ne cesse de se développer et d'englober d'autres domaines que celui de l'industrie.

De cette ouverture a surgit un nouveau besoin que l'industrie, jusqu'à lors, ne connaissait pas, l'interaction du bras manipulateur avec son environnement. La solution à ce besoin a était amenée par l'intégration de la vision aux robots. C'est la naissance d'une autre branche de la robotique, la commande référencée vision : celle-ci consiste non plus à utiliser les informations fournies par un capteur de vision dans des procédures haut niveau, mais à les intégrer dans des lois de commande en boucle fermée(sur ces informations.)

L'utilisation de la vision, à laquelle nous nous sommes rattaché, est particulièrement intéressante en raison du grand nombre d'informations qu'une camera peut fournir et en raison de la grande variété des tâches qu'elle permet de réaliser.

Cette grande richesse nécessite cependant de disposer d'algorithmes de traitement d'images particulièrement performant afin d'extraire, à une cadence proche de celle de la vidéo, les informations qui seront utilisées en commande.

L'intégration de la vision a ouvert le champ aux applications les plus révolutionnaires de la robotique. Cette capacité a définitivement fait des bras manipulateurs une partie indissociable de plusieurs domaines autres que celui de l'industrie, notamment la médecine

et l'aérospatial. Dans le domaine de la médecine, la vision a permis l'apparition d'une nouvelle discipline, laquelle s'intéresse aux Gestes Médicaux et Chirurgicaux Assistés par Ordinateur (GMCAO), où la tâche est réalisée selon un schéma de commande en boucle fermée utilisant directement l'information visuelle fournie par une modalité d'imagerie médicale telles l'endoscopie, la fibroscopie, l'échographie, la tomodensitométrie (scanner X),la résonance magnétique (IRM) ou autres. Ce qui permet d'assister le praticien de façon quantitative et fiable. L'utilisation de systèmes robotiques permet d'augmenter la précision du geste et d'effectuer des opérations chirurgicales peu invasives pour le patient voir même des interventions ne pouvant pas être réalisées auparavant par une technique manuelle.

Dans le domaine de l'aérospatiale, c'est pour la robotique mobile que la vision a était le plus bénéfique, en effet, l'avènement de la vision marque le début de l'ère des robots explorateurs en mission sur Mars. Mais malgré la dominance des robots mobiles, les bras manipulateurs trouvent largement leur applications dans ce domaine, en effet des travaux sur la manipulation d'objets par asservissement visuel se font en vue d'applications sur la station spatiale internationale.

En fin, dans le domaine de l'industrie, l'avènement de la vision a permis aux bras manipulateurs d'être plus autonomes ce qui leur a permis de se substituer à l'homme lors de tâches en milieu hostile et dangereux comme le nucléaire, ou lors de tâches répétitives et précise comme la détection de défauts, notamment dans l'industrie automobile.

Organisation de la thèse

Cet ouvrage est structuré en quatre chapitres qui portent respectivement sur les points suivants :

1. Après avoir présenté quelques généralités sur les bras manipulateurs, notamment expliquer les étapes de réalisation d'une tâche en robotique et la modélisation des bras manipulateurs, nous nous présentons le système sue lequel s'est porté notre travail, à savoir le robot SCARA et son interface de commande.
2. Le chapitre II, sera quand à lui, consacré à un état de l'art sur l'asservissement visuel. Nous commençons par quelques notions de base sur le traitement d'image puis, nous abordons les différentes transformations de repères. Après la modélisation des informations visuelles et l'exposé des différentes techniques d'asservissement visuel, nous détaillons le, principe d'un asservissement 2D, technique retenue pour notre travail.
3. Le chapitre 3 traite des résultats de simulations d'un asservissement 2D appliqué à notre système. Après avoir expliqué la configuration du vison sur notre système, et la procédure de calibration de la camera, nous donnons les informations visuelles

qui sont utilisées pour chaque tâche traitée dans ce mémoire (positionnement par rapport à quatre points et suivi de ligne) puis, nous synthétisons les lois de commande pour réaliser chaque tâche, dont nous donnons les résultats de simulations à la fin de ce chapitre. .

Chapitre 1

Généralités sur les bras manipulateurs et Présentation de la structure

1.1 Introduction

Depuis que *General Motors* a introduit l'Unimate dans son processus de fabrication en 1961, les bras manipulateurs font partie intégrante du paysage industriel. Ce domaine a d'ailleurs connu un véritable 'Boom' durant les années 80 du fait que les exigences industrielles ,de plus en plus, poussées ont amorcé l'explosion des thèmes de recherche. Ce qui a eu pour conséquences l'enrichissement de la théorie des bras manipulateurs et la variété des applications que ce soit dans le domaine industriel ou même dans la robotique de service.

Dans ce chapitre, on traitera en premier lieu la théorie des bras manipulateurs. Après avoir abordé les différentes définitions, on détaillera les différentes étapes de réalisation d'une tâche en robotique avant d'aborder la modélisation dynamique des bras manipulateurs.La deuxième partie de ce chapitre sera, quant à elle, consacrée à la description du système sur lequel va se porter notre travail ainsi que son interface de commande.

1.2 Présentation des bras manipulateurs

Plusieurs définitions des bras manipulateurs existent dans la littérature, nous avons retenu pour notre travail deux définitions qui nous paraissent les plus complètes. *L'AFNOR* définit un manipulateur comme étant « un mécanisme multifonctionnel à plusieurs degrés de liberté commandé directement par un opérateur humain ou par un opérateur logique » ; quant au *Robotics Institute of America* il le définit comme étant « un mécanisme automatique asservi en position, reprogrammable, polyvalent et capable de positionner des matériaux, pièces, outils ou dispositifs spécialisés ».

Un bras de robot est une structure à caractère anthropomorphe, il est constitué d'une série de segments ou membrures, chacun est relié à son prédécesseur et à son successeur par des articulations à un degré de liberté, ces articulations peuvent être de type rotoïde ou prismatique. En mettant en mouvement ces articulations par différents actionneurs, on effectue le placement de l'outil terminal (pince, main anthropomorphe...etc.)[10].

Un robot est donc une structure articulée capable de positionner et d'orienter son outil terminal dans un certain espace. L'espace dans lequel se déplace l'outil est appelé espace opérationnel ou espace de tâche, et l'espace représentatif des configurations des articulations est appelé espace articulaire. L'ensemble des points et directions accessibles constitue ce que nous appellerons l'espace accessible du robot ou espace de travail[11].

Deux mécanismes basiques de bras existent, et à partir desquels dérivent tous les autres types de bras de robot : le PUMA (Vertical multi-joint type) et le SCARA (Horizontal multi-joint type). Chacun de ces deux types est constitué d'au moins 3 degrés de libertés, le nombre minimum pour le placement de l'outil terminal dans l'espace tri dimensionnel[12].

FIGURE 1.1: Robot de type SCARA.

FIGURE 1.2: Robot de type PUMA.

1.3 Stratégie de réalisation d'une tâche

La stratégie en robotique traduit la démarche que va suivre le bras manipulateur ou plutôt son organe terminal pour atteindre un objectif, c'est-à-dire la procédure que suivra le robot lors de son évolution dans son environnement pour réaliser une tâche, une tâche qui peut être la saisie d'un objet, suivi de cible ou navigation.

La stratégie de réalisation d'une tâche en robotique peut être résumée par trois activités qui sont : la perception, la génération de mouvement et la commande des actionneurs[4].

1.3.1 La perception

La notion de perception en robotique consiste en l'obtention et l'interprétation d'informations venant des différents capteurs.

En effet, pour assurer son interaction avec son environnement et afin d'avoir un retour d'information sur l'état de son organe terminal, le robot doit disposer de capteurs qui transformeront les stimuli extérieurs en signaux électriques[13].Selon leurs usages,ces capteurs peuvent être classés en deux catégories :les capteurs proprioceptifs et les capteurs extéroceptifs.

a. Les capteurs proprioceptifs :

Ils donnent des informations propres au comportement interne du robot, l'information fournit par ce type de capteurs est utilisée comme mesure de retour à deux niveaux différents : au plus bas niveau pour l'asservissement des articulations, et au deuxième niveau qui correspond à l'asservissement de la trajectoire en position. Pour ce type de capteurs, on peut citer comme exemples : les odomètres, les accéléromètres, les gyroscopes et les GPSs [4][13].

FIGURE 1.3: Télémètre laser.

FIGURE 1.4: Capteur infrarouge.

b. Les capteurs extéroceptifs :

Ils informent le robot sur son environnement. L'information fournie par ces capteurs est utilisée aux deux niveaux :comme mesure de retour dans le premier cas pour corriger la position et/ou la vitesse dans les boucles d'asservissement de trajectoire, et sous la forme de positions de consigne pour la génération de trajectoire dans le second. C'est dans cette catégorie qu'on retrouve les capteurs de vision .Comme exemples de capteurs extéroceptifs on peut citer : les capteurs infrarouge, les télémètres à ultrasons ou à laser et enfin les capteurs de vision[4][13].

Comme notre travail se porte sur la commande référencée vision, on a jugé nécessaire de détailler ce dernier type de capteurs extéroceptifs à savoir les capteurs de vision.

- Les capteurs de vision :

L'utilisation d'une camera pour percevoir l'environnement est une méthode attractive car elle semble proche des méthodes utilisées par les êtres humains, cependant elle présente

un inconvénient qui est le fait qu'elle est un capteur trop riche pour être utilisée dans des applications en temps réel.[6]

De cette richesse provient la difficulté majeure de l'asservissement visuel, à savoir, parmi l'ensemble des informations potentielles, la manière de sélectionner celles qui fourniront un comportement satisfaisant au système (stabilité, robustesse aux erreurs de modélisation et de mesure, absence de singularités et de minimum local et une trajectoire satisfaisante du robot).

La sélection des informations visuel se fait par extraction de l'image numérique, de mesures 2D, tel que les coordonnées de points caractéristiques dans l'image par exemple. L'extraction des mesures 2D se fait par les différentes techniques de traitement d'images qui seront exposées plus loin dans le chapitre II [1].

1.3.2 La génération de mouvement

La génération de mouvement désigne la fonction de calcul des consignes articulaires du robot destinées à réaliser une tâche interprétée sous forme de positions successives de l'outil du robot et de contraintes cinématiques ou dynamiques. Cette fonction est centrale au sein du contrôleur du robot, dans le sens où elle fait la transition entre le niveau informatique d'interprétation-exécution du programme de la tâche et le niveau de commande proprement dit[1].

Les méthodes de génération de trajectoires peuvent être classées de différentes façons : selon l'espace utilisé, articulaire ou cartésien ; selon que le calcul est effectué en ligne ou hors ligne ; selon le type de données d'entrée et les contraintes imposées : trajectoire rectiligne, suivi de chemin ou point à point[4].

Dans les paragraphes suivants, on va exposer de manière succincte quelques généralités sur la génération de mouvement sans détailler ces méthodes qui sont largement traitées dans la littérature notamment dans [1].

- Génération de mouvements dans l'espace articulaire et dans l'espace opérationnel :

La génération de mouvements dans l'espace articulaire applique à chaque articulation du robot une loi de mouvement dont les contraintes sont définies dans l'espace articulaire. Dans le cas le plus général où la durée de mouvement n'est pas imposée, chaque articulation a une durée propre de mouvement déduite de la satisfaction des contraintes cinématiques ou dynamiques. Il est alors nécessaire, dans une seconde étape de synchroniser l'ensemble des articulations sur la plus lente, dite articulation maître ; à l'issue de leur synchronisation, les mouvements recalculés sont directement les consignes de commande, comme l'illustre la figure (1.5) [1].

La manière dont nous avons défini la trajectoire dans l'espace articulaire peut conduire à des trajectoires non désirées dans l'espace opérationnel. Cette façon de faire se justifie cependant lorsqu'aucune limitation n'est imposée à la trajectoire dans l'espace opéra-

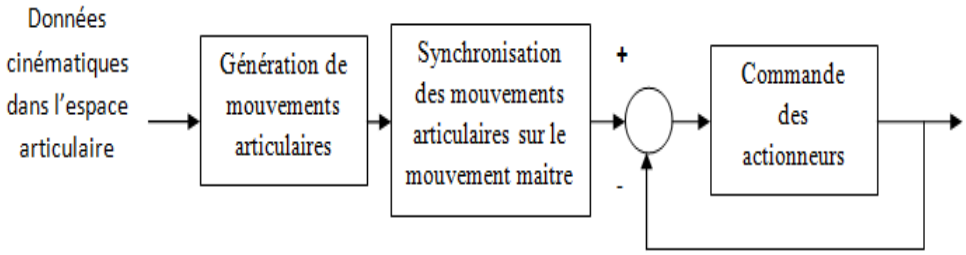

FIGURE 1.5: Organisation fonctionnelle de la génération de mouvement dans l'espace articulaire[1].

tionnel, si ce n'est de passer d'une configuration à une autre, en respectant une certaine continuité (de position, de vitesse ou d'accélération). Si nous fournissons par exemple les conditions extrémales et le temps de mouvement, la génération de mouvement doit impérativement se faire dans l'espace opérationnel, et l'appel au modèle géométrique inverse est alors nécessaire pour transformer le mouvement de consigne cartésien en mouvement de consigne articulaire, comme le montre la figure 1.6 [11].

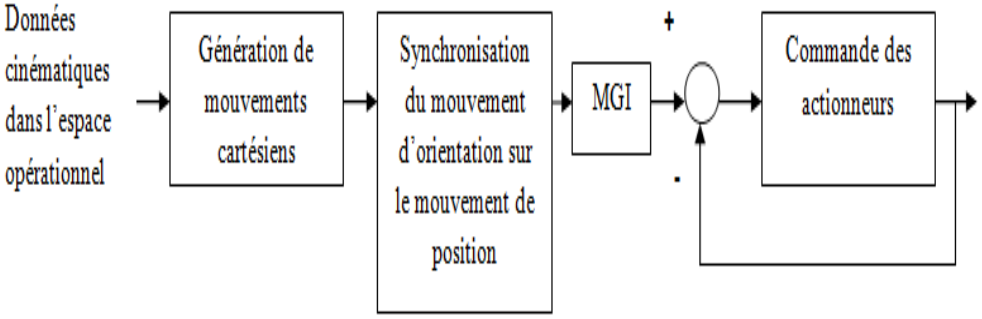

FIGURE 1.6: Organisation fonctionnelle de la génération de mouvements dans l'espace opérationnel [1].

1.3.3 Commande des actionneurs

Le problème de la commande d'un robot manipulateur peut être formulé comme la détermination de l'évolution des forces généralisées (forces ou couples) que les actionneurs doivent exercer pour garantir l'exécution de la tâche tout en satisfaisant certains critères de performance.

La plupart des robots utilisent des servomoteurs électriques comme actionneurs. Dans le cas de servomoteurs ayant un faible rapport de réduction, ce sont les servomoteurs qui doivent compenser les effets des variations des forces d'inertie et de gravité. Dans le cas de servomoteurs avec de forts rapports de réduction, l'inertie vue par le moteur varie beaucoup moins et il est alors possible de modéliser le robot par un système linéaire qui permet de découpler les articulations[4].

Nous ne pouvons pas, dans le cadre de ce mémoire, traiter toutes les méthodes de commande des bras manipulateurs en détail alors, on se limitera dans les paragraphes suivants à une description simple de ces différentes méthodes.

1. **La commande décentralisée :**
   Cette technique est utilisée par des robots manipulateurs qui utilisent des servomoteurs avec de forts rapports de réduction. Lorsque le système présente un comportement linéaire, l'asservissement du mouvement peut être réalisé par des techniques classiques de commande. Nous parlons alors d'une commande décentralisée de type PID[4]. L'utilisation courante, dans le milieu industriel, de ce type de commande est liée à sa simplicité de réalisation et d'utilisation, mais ce type de commande ne pourra pas être utilisé dans le cas d'une tâche complexe où le manipulateur devra par exemple transporter des charges de masses différentes à des vitesses différentes avec l'exigence d'une faible erreur de poursuite[1].
2. **La commande par découplage non linéaire :**
   La commande par découplage non linéaire consiste à transformer par retour d'état un problème de commande d'un système non linéaire en un problème de commande linéaire[14].
   Ce type de technique permet la commande dans l'espace des articulations ou dans l'espace cartésien, avec l'avantage que les articulations sont découplées et peuvent évoluer à grandes vitesses avec de fortes inerties. Cette méthode dépend fortement du modèle du système, elle est très sensible aux imprécisions du modèle qui entraînent un découplage imparfait. Ceci constitue son principal inconvénient[1].
3. **La commande fondée sur une fonction de Lyapunov :**
   Des méthodes basées sur une fonction de Lyapunov ont été utilisées pour la commande des bras manipulateurs de façon satisfaisante pour des tâches de suivi. Particulièrement lorsqu'on cherche à garantir la convergence asymptotique et non à linéariser le système ou à obtenir le découplage [1].
4. **La commande adaptative :**
   Il est parfois nécessaire de modifier les paramètres d'une loi de commande lorsque certains paramètres intrinsèques au robot ou à son environnement ont évolué. En effet, l'usure, la modification des réglages des mécanismes entrainent des modifications du comportement dynamique du robot. Aussi est-il intéressant, dans certains

cas, d'utiliser une loi de commande adaptative[1]. Les avantages de ce type de techniques sont évidents, malheureusement la puissance de calcul demandée au système constitue un inconvénient important[4].

5. **La commande robuste :**
   Les algorithmes de commandes classiques de type (PID) et variantes ont plus de cinquante ans de savoir faire. Cependant, pour les systèmes dynamiques qui présentent des variations paramétriques et des non linéarités pouvant être soit inhérentes au système, soit présentes en cours d'utilisation (hystérésis, couplage,... etc.), ces algorithmes montrent leurs limites lors de leur implémentation [15]. Dans ce cas, le choix de la commande est orienté vers les techniques robustes, notamment le mode glissant appelé aussi commande à structure variable[4].
   On définit alors n surfaces de glissement, correspondant aux n articulations du manipulateur. L'objectif est d'atteindre pour chaque variable commandée le régime glissant. Le système dans sa globalité, aurait alors le comportement de n systèmes du premier ordre linéaires découplés[1].

1.4 Modélisation des bras manipulateurs

1.4.1 Modèle géométrique

- **Modèle géométrique direct**
  Le modèle géométrique direct d'un robot manipulateur est la fonction F qui permet d'exprimer la situation de l'organe terminal du robot manipulateur en fonction de sa configuration [16].
  La situation S de l'organe terminal est définie par m coordonnées que nous appelons opérationnelles et que nous notons :

$$x_1, x_2, x_3, ..., x_m \tag{1.1}$$

  La configuration du robot manipulateur est définie par n coordonnées que nous appelons généralisées et que nous notons :

$$q_1, q_2, q_3, ..., q_n \tag{1.2}$$

  Si X désigne le m-uplet des coordonnées opérationnelles et le n-uplet des coordonnées généralisées,le modèle géométrique du robot manipulateur est :

$$X = F(q) \tag{1.3}$$

  Le détail du calcul de la fonction F sera présenté en annexe A.

- **Modèle géométrique inverse**
  Le modèle géométrique inverse MGI permet de calculer les coordonnées articulaires correspondant à une situation donnée de l'organe terminal.Plus précisément, c'est

la forme explicite, qui lorsqu'elle existe, donne toutes les solutions possibles dans l'espace articulaire (il y a rarement unicité de solution) à une situation donnée de l'effecteur dans l'espace opérationnel[17].

$$q = F^{-1}(X) \tag{1.4}$$

Plusieurs méthodes de calculs sont disponibles dans la littérature[1][10][18] dans notre travail on considère la méthode de Paul qui facilite énormément les calculs,Les détail seront présentés en annexe A.

1.4.2 Modèle cinématique

Le modèle cinématique exprime les relations entre vitesses et accélérations articulaires de chaque joint et vitesses et accélérations cartésiennes d'un point de la chaine cinématique, généralement l'organe terminal. Ce modèle est donc un modèle par accroissement élémentaire : chaque variation élémentaire de la vitesse/accélération d'une articulation implique la variation de celle de l'organe terminal, et inversement[17].

- **Modèle cinématique direct**
  Le modèle cinématique direct donne la relation entre les vitesses opérationnelles et généralisées en utilisant la matrice jacobienne J :

$$\dot{X} = J(q)\dot{q} \tag{1.5}$$

  Le détail du calcul de la matrice jacobienne J sera présenté en annexe A.

- **Modèle cinématique inverse**
  L'objectif du modèle cinématique inverse est de calculer à partir d'une configuration donnée, les vitesses articulaires qui assurent au repère terminal une vitesse opérationnelle imposée[8].Pour obtenir le modèle cinématique inverse, on inverse le modèle cinématique direct ;si $n = m$; en résolvant un système d'équations linéaires.

$$\dot{q} = J^{-1}\dot{X} \tag{1.6}$$

1.4.3 Modèle dynamique

Un modèle dynamique d'un robot permet de déterminer la position, la vitesse et l'accélération d'un élément quelconque du robot, à tout instant, connaissant les couples et les forces agissant sur ce robot.
Les différents couples et forces qui interviennent dans un tel modèle se repartissent sui-

vant :

-Les couples et forces engendrés par les actionneurs ;
-Les effets inertiels des membrures et charges ;
-Les forces centrifuges et de Coriolis ;
-Les effets de la gravité ;
-Les couples et forces de frottement ;
-Les diverses interactions avec l'environnement.

L'établissement du modèle dynamique peut se faire à partir des différents formalismes ; les plus utilisés sont le formalisme de *Lagrange-Euler*et celui de *Newton-Euler* [11].

Dans notre cas nous avons opté pour le formalisme de Lagrange-Euler pour sa simplicité. Le paragraphe suivant en expliquera le principe.

Formalisme de Lagrange-Euler

- **Définition**

  Suivant le formalisme de Lagrange-Euler, les équations du mouvement sont régies par :

$$\frac{d\left(\frac{\partial L}{\partial \dot{q}}\right)}{dt} - \frac{\partial L}{\partial q} = \Gamma \tag{1.7}$$

  où : $\Gamma \in R^N$:Ensemble des forces et couples conjugués aux coordonnées généralisées $q \in R^N$,

$$L(\dot{q}, q) = K(\dot{q}, q) - U(q) \tag{1.8}$$

  avec ;

  – K : Énergie cinétique,
  – U : Énergie potentielle.

- **Energie cinétique**

$$K = \sum_{0<i<n} K_i \tag{1.9}$$

  L'énergie cinétique totale d'un système articulé est égale à la somme des énergies cinétiques de chacune des membrures :

L'énergie de chacune des membrures est définit par :

$$K_i = \int \left[\dot{P}_i^T P_i\right] dm \tag{1.10}$$

Ce que nous pouvons encore écrire :

$$K_i = \frac{1}{2} tr(\int \left[\dot{P}_i^T P_i\right] dm) \tag{1.11}$$

En exprimant les points P_i de la membrure i dans leurs coordonnées homogènes locales, il vient :

$$K_i = \frac{1}{2} tr(\dot{T}_i \int \left[p_i^i P_i^{iT}\right] dm \dot{T}_{iT}^i) \tag{1.12}$$

T_i : Matrice homogène de transformation.

Nous introduisons le moment d'inertie en les coordonnées homogènes locales :

$$J_i^i = \int \left[p_i^i P_i^{iT}\right] dm \tag{1.13}$$

De manière que :

$$K_i = \frac{1}{2} tr(T_i \dot{J_i^i} \dot{T_i^T} \tag{1.14}$$

En explicitant P_i suivant :

$$P_i = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{1.15}$$

Nous trouvons :

$$J_i^i = \begin{pmatrix} \int x^2\, dm & \int xy\, dm & \int xz\, dm & \int x\, dm \\ \int yx\, dm & \int y^2\, dm & \int yz\, dm & \int y\, dm \\ \int zx\, dm & \int zy\, dm & \int z^2\, dm & \int z\, dm \\ \int x\, dm & \int y\, dm & \int z\, dm & \int dm \end{pmatrix} \tag{1.16}$$

Or la masse du corps est donnée par :

$$m_i = \int dm \tag{1.17}$$

Son centre de gravité est :

$$\begin{cases} m_i x_G = & \int x \, dm \\ m_i y_G = & \int y \, dm \\ m_i z_G = & \int z \, dm \end{cases} \tag{1.18}$$

Et son moment d'inertie autour de ce centre de gravité est donné par :

$$\begin{cases} I_{xx} = & \int \left(z^2 + y^2\right) dm \\ I_{yy} = & \int \left(x^2 + z^2\right) dm \\ I_{zz} = & \int \left(x^2 + y^2\right) dm \\ I_{xy} = & \int xy \, dm \\ I_{yz} = & \int yz \, dm \\ I_{xz} = & \int xz \, dm \end{cases} \tag{1.19}$$

On peut alors écrire le moment d'inertie J_i^i en coordonnées homogènes locales :

$$J_i^i = \begin{pmatrix} \frac{1}{2}(I_{zz} + I_{yy} - I_{xx}) & I_{xy} & I_{xz} & m_i x_G \\ I_{yx} & I_{xy} & \frac{1}{2}(I_{xx} + I_{zz} - I_{yy}) & m_i y_G \\ I_{zx} & I_{zy} & \frac{1}{2}(I_{xx} + I_{yy} - I_{zz}) & m_i z_G \\ m_i x_G & m_i y_G & m_i z_G & m_i \end{pmatrix} \tag{1.20}$$

Finalement, l'énergie cinétique totale s'écrit :

$$K = \frac{1}{2} tr(\sum_{0<i<n} \dot{T}_i J_i^i \dot{T}_i^T) \tag{1.21}$$

- **Énergie potentielle :**

L'énergie potentielle d'un système articulé est la somme des énergies potentielles de chacune des membrures :

$$U = \sum_{0<i<n} U_i \tag{1.22}$$

Chacune des membrures a pour énergie potentielle :

$$U_i = \int g^T p_i \, dm \tag{1.23}$$

Avec g représente le vecteur homogène de l'accélération de la pesanteur dans les coordonnées absolues :

$$g = \begin{pmatrix} 0 \\ 0 \\ -g \\ 1 \end{pmatrix} \tag{1.24}$$

En notants_i et s_i^i les coordonnées homogènes du centre de gravité de la membrure i, dans les repères respectivement absolu et local, l'énergie potentielle de chaque membrure peut alors s'écrire :

$$U_i = m_i g^T s_i = m_i g^T T_i s_i^i \tag{1.25}$$

Ce qui nous donne une énergie potentielle totale s'exprimant sous la forme

$$U = g^T \sum_{0<i<n} m_i T_i s_i^i \tag{1.26}$$

- **Équation du mouvement**

Le Lagrangien du système est donné par :

$$L = \frac{1}{2} tr(\sum_{0<i<n} \dot{T}_i J_i^i \dot{T}_i^T) - g^T \sum_{0<i<n} m_i T_i s_i^i \tag{1.27}$$

Les équations de mouvement sont régies par l'équation 1.7, nous devons donc évaluer les équations : $\frac{d\left(\frac{\partial K}{\partial \dot{q}}\right)}{dt}$, $\frac{\partial K}{\partial q}$, $\frac{\partial U}{\partial q}$

Il vient :

$$\frac{\partial U}{\partial q_k} = g^T \sum_{k<i<n} m_i \frac{\partial T_i}{\partial q_k} s_i^i \tag{1.28}$$

Notons que nous nous sommes servis du fait que $\frac{\partial T_i}{\partial q_k} = 0$ si $i < k$, pour commencer notre sommation en $i = k$.

Nous avons aussi :

$$\frac{\partial K}{\partial q_k} = tr(\sum_{k<i<n} \frac{\partial \dot{T}_i}{\partial q_k} J_i^i \dot{T}_i^T) \tag{1.29}$$

Et :

$$\frac{\partial K}{\partial \dot{q_k}} = tr(\sum_{k<i<n} \frac{\partial \dot{T_i}}{\partial q_k} J_i^i \dot{T_i^T}) = tr(\sum_{k<i<n} \frac{\partial T_i}{\partial q_k} J_i^i \dot{T_i^T})) \tag{1.30}$$

D'où :

$$\frac{d\left(\frac{\partial K}{\partial \dot{q}}\right)}{dt} = tr(\sum_{k<i<n} (\frac{\partial \dot{T_i}}{\partial q_k} J_i^i \dot{T_i^T}) + \frac{\partial T_i}{\partial q_k} J_i^i \ddot{T_i^T}))) \tag{1.31}$$

L'équation du mouvement sera :

$$\tau = tr(\sum_{k<i<n} (\frac{\partial T_i}{\partial q_k} (J_i^i \ddot{T_i^T} + m_i s_i^i g^T)) \tag{1.32}$$

τ_i : représente l'ensemble des forces et couples généralisés appliqués a la $i^l ieme$ articulation.

L'écriture matricielle de l'équation dynamique du mouvement du bras manipulateur est la suivante :

$$\Gamma(q, \dot{q}, \ddot{q}) = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) \tag{1.33}$$

On notera :

•G(q)= $\Gamma^{gravité}(q)$
Avec :

$$\Gamma_k^{gravité} = g^T \gamma_k T_k \sum_{k<i<N} T_i^k m_i s_i^i \tag{1.34}$$

•M(q)=$\Gamma^{inertie}(q, \ddot{q})$
Avec :

$$\tau_k^{inertie} = \sum_{k<i<N} \sum_{1<l<i} tr(\Gamma_k T_i J_i^i T_i^T \Gamma_l^T) \ddot{q_l}) \tag{1.35}$$

• C(q)=$\Gamma^{charistoffel}(q, \dot{q})$
Avec :

$$\Gamma_k^{charistoffel} = \sum_{k<i<N} \sum_{1<l=\lambda<i} tr(\Gamma_k T_i J_i^i T_i^T \delta_{l,\lambda}^T \dot{q_l} \dot{q_\lambda}) \tag{1.36}$$

On notera aussi :

$$\gamma_k^j = T_{k-1}^j \delta_k T_j^{k-1} \tag{1.37}$$

Et δ est tel que :

$$\frac{\partial T_i^j}{\partial q_k} = T_{k-1}^j \delta_k T_i^{k-1} \tag{1.38}$$

Et finalement :

$$\delta_{l,\lambda} = \begin{cases} \gamma_\lambda \gamma_l \text{ si } \lambda < l \\ \gamma_l \gamma_\lambda \text{ si } \lambda gel \end{cases} \tag{1.39}$$

Remarque : L'application de cette méthode sur le cas qui nous intéresse, à savoir le robot SCARA, est détaillée dans l'annexe A.

1.5 Présentation de la structure existante du robot SCARA

1.5.1 Le robot Scara

La structure dont nous disposons au sein du LCP est un robot SCARA à trois degrés de liberté de type RRP, il est constitué de deux articulations rotoides et d'une articulation prismatique.

Les éléments constituant cette structure sont :

- Une base,
- Deux tiges d'aluminium,
- Une barre d'aluminium sur laquelle est montée l'articulation prismatique,
- Deux moteurs à courant continu,
- Une pince de saisie avec un moteur intégré.

1.5.2 Les actionneurs

La structure dont nous disposons possède deux types d'actionneurs, à savoir les moteurs à courant continu et les vérins pneumatiques.
Dans ce qui suit on donnera brièvement la caractéristique de chacun de ces deux actionneurs.

a. Les moteurs

Les moteurs installés au niveau des articulations rotoides sont des moteurs à courant continu à aimants permanents munis d'encodeurs incrémentaux optiques ayant une résolution de 500CPR (compte par tour). Ce sont des moteurs de référence GM9236 du fabricant américain Pittman. Les caractéristiques de ce type de moteurs sont résumées

Paramétre	valeur	unité
Constante du couple	$48.8*10^{-3}$	N.m
Constante de la force contre électromotrice	$48.8*10^{-3}$	V.s/rad
Inertie du moteur	$7.06*10^{-6}$	Kgm^2
Constante de temps électrique	1.06	ms
Constante de temps mécanique	8.5	ms
Masse du moteur	391.2	g
Résistance	2.49	ω
Inductance	2.63	mH
Tension d'alimentation	24	V
Rapport de réduction	65.5	-

TABLE 1.1: Caractéristiques du moteurs PITTMAN GM9236 [8]

dans le tableau (1.1) [8].

FIGURE 1.7: Les moteurs PITTMAN.

1.5.3 L'interface UMAC

L'UMAC (*Universal Motion and Automation Controller*) est un système modulaire en format 3U de la firme DELTA TAU, bâtit autour d'un contrôleur PMAC (Programmable Multi Axis Controller) utilisant la technologie des DSPs pour le calcul des trajectoires de mouvement et pour la commande des moteurs électriques[8].

Un DSP (*Digital Signal Processor*) est un type spécial de processeurs optimisé pour des opérations mathématiques rapides et répétées. Il est très souvent utilisé dans le domaine de l'audio, de la vidéo et du control de mouvement où il fournit une solution plus rentable qu'en utilisant un microprocesseur multi-usage[19].

L'UMAC est constitué de plusieurs cartes montées sur un châssis et offrant la possibilité d'extension, ses cartes sont :

- La Turbo PMAC CPU,
- La carte d'alimentation,
- La carte des entrées/sorties,
- La carte d'interface d'axes[2].

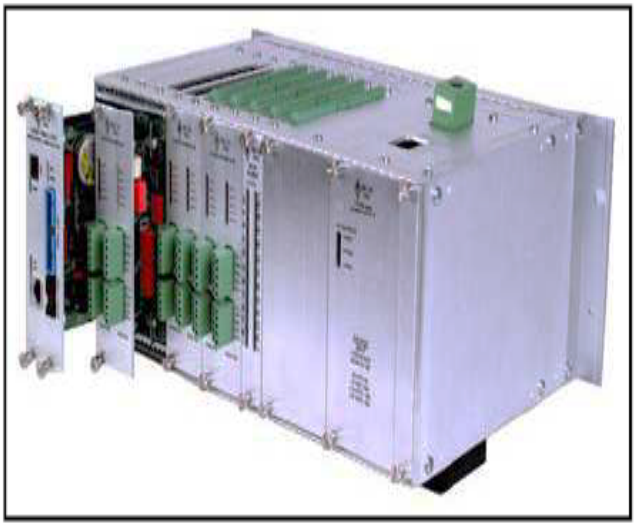

FIGURE 1.8: Le système Turbo UMAC.

a. La Turbo PMAC CPU

La Turbo PMAC CPU est l'unité de traitement de l'UMAC, c'est une combinaison entre un processeur, une interface de circuiterie pour le mouvement (DSPGATE),des entrées /sorties (IOGATE) et une interface de communication avec le PC[19].

Grâce à la technologie des DSPs, la PMAC CPU offre des solutions de mouvement intéressantes. En effet, elle permet le control de 32 axes, 16 systèmes de coordonnées, l'implémentation d'algorithmes cinématiques directs et inverses et aussi une communication simultanée avec l'interface logicielle[2]. les caractéristiques et les performances de la Turbo PMAC CPU sont exposées dans le tableau (1.2)

a.1 Le processeur :
La Turbo PMAC CPU utilise un processeur *MOTOROLA* de la famille des DSP 56300[19]. Ce processeur à une architecture de Harvard caractérisée par la séparation entre mémoire données et mémoire programmes, ce qui offre la possibilité d'accéder aux données et aux instructions du programme en même temps (en un cycle d'horloge) et permet donc d'avoir de grandes vitesses d'exécution[20].

Caractéristique	spécifications
Fréquence d'horloge	20MHZ
Temps de cycle	55ms par control d'axe
Algorithme de control	PID avec un feedforward en vitesse et en accélération
Précision sur la vitesse $+/-1$ compte d'encodeur	0.005heightPrécision sur la position
Encodeurs	4 à 16 encodeurs incrémentaux a quadrature numérique

TABLE 1.2: Caractéristiques et performances de la Turbo PMAC CPU[9].

a.2 Les mémoires :

Deux types de mémoires sont utilisés par la Turbo PMAC CPU :

1. **Les mémoires SRAM :**
   [Les mémoires SRAM (Static RAM) sont utilisées comme mémoire active. Ce type de RAMs est plus rapide et plus robuste que les mémoires DRAM (Dynamic RAM).Comme le DSP de *MOTOROLA* utilise l'architecture de *HARVARD* où les mémoires réservées pour le stockage des programmes et celles réservées pour le stockage des données sont séparées, on distingue alors deux configurations de SRAM : la première utilisée pour les programme de $128K$ organisée en mots de $24bits$, et la deuxième utilisée pour les données de $64K$ organisée en mots de 48 bits [19].

2. **Les mémoires Flash :**
   La mémoire non volatile du Turbo PMAC est contenue dans une mémoire flash. Ces mémoires sont relativement lentes alors la Turbo PMAC ne les utilise pas dans les opérations en cours, des copies de leurs contenu sont chargées dans les SRAMs pour que le processeur y ait accès rapidement[19].

a.3 La DSPGATE :

La DSPGATE contient toute la logique qui fournit l'interface entre la CPU et les canaux de mouvement. Un canal de mouvement est une structure hardware de la Turbo PMAC, il représente l'ensemble des interfaces et des registres pour un amplificateur, un encodeur ou pour les flags[19].

La DSPGATE est constituée de 4 canaux avec 32 registres, sa principale tâche consiste en :

-La conversion analogique/numérique,

-La réception des signaux des capteurs (comptage des impulsions venant des encodeurs),
-La notation des fins de course des moteurs,
-L'envoie des signaux de commande délivrés par la CPU aux amplificateurs [20].

- **a.4 L'IOGATE :**
  L'IOGATE est utilisée pour accéder aux différentes entrées/sorties de l'UMAC, elle fournit 48 points entrées/sorties adressés en registres de 6 bits chacun[19]. elle permet l'adaptation des signaux E/S digitaux avec isolation optique ainsi que leur sauvegarde dans des registres pour que la CPU puisse les utiliser[20].

a.5 L'interface de communication :

L'UMAC supporte plusieurs moyens de communication :

- Communication par port USB (1.1 ou 2.0),
- Communication par port Ethernet (TC/IP ou UDP/IP Protocol),
- Communication par port série RS232 /422.

b. La carte d'alimentation

La carte d'alimentation permet d'alimenter toutes les autres cartes par la conversion de la source du courant alternatif ($85V - 240V$) pour obtenir une tension continue de $15V$ avec un courant maximal de $1.5A$ pour alimenter les circuits analogiques, et de $5V$ avec un courant maximum de$14A$ pour alimenter les circuits numériques[8].

c. Les Amplificateurs

Le rôle des amplificateurs consiste à rendre les signaux de commande délivrés par la DSPGATE exploitables par les moteurs, pour ce faire ils sont dotés d'une boucle de régulation de courant.L'UMAC dispose de huit amplificateurs montés sur deux cartes, de tension d'alimentation maximale de $40VDC$et d'un courant maximum de $3A$[20].

1.5.4 Le Software

Pour la programmation de l'interface UMAC, un logiciel spécialement conçu par la firme Delta tau pour la programmation de leurs différentes cartes DSP est utilisé. Ce logiciel est le *Pewin32Pro* [8].

a. Le logiciel Pewin32pro :

Le logiciel PEWIN32PRO est le programme exécutif de PMAC sous Windows,il offre un environnement software pour le développement et la maintenance des applications utilisant le contrôleur Turbo PMAC.

L'interface graphique de *Pewin32pro* dispose d'un terminal pour la communication directe avec le PMAC ,d'un éditeur de texte pour la saisie des programmes, ainsi que de plusieurs visualisations telles que les positions et les vitesses des articulations,l'état des moteurs et des encodeurs ou encore l'état des différentes variables[21].

b. Les variables :

Par soucie de faciliter la programmation et la configuration des applications, et pour ne pas s'encombrer avec les adresses, le logiciel *Pewin32pro* utilise des variables propres à lui et bien spécifiques. Ces variables se divisent en quatre classes : I, P, Q et M.

- **Les variables I :** Ce sont des variables qui ont des significations prédéfinit, elles servent à l'initialisation et a la configuration du PMAC.
- **Les variables P :** Ce sont des variables utilisées lors du traitement des données, c'est des variables globales ;
- **Les variables Q :** Ce sont des variables qui ont le même rôle que les variables P, à la seule différence que ce sont des variables locales ;
- **Les variables M :**Ce sont des pointeurs qui permettent un accès facile à la mémoire interne de l'interface et aux registres d'entrées/sorties[20][2].

c. Les différents types de programmes

c.1. Programmes de mouvement

Dans ce type de programme, les mouvements à effectuer sont décrits par une suite d'instructions [20] ces instructions sont écrites dans un langage qui est à mi-chemin entre le langage de haut de niveau tel que le pascal et le langage G-code (RS274). La saisie des programmes se fait avec n'importe quel éditeur de texte et sont ensuite chargés dans le PMAC grâce à l'interface logicielle *Pewin*. La Turbo PMAC CPU peut gérer jusqu'à 256 programmes de mouvement en même temps[22].

c.2. Programmes PLC

Les programmes *PLCs* ont la même construction que les programmes de mouvement mais n'exécutent aucun déplacement[20].Les PLCs *(Programmable Logic Controller)* appelés ainsi car ils remplissent les même fonctions qu'un hardware PLC, sont idéaux pour

des taches asynchrones aux séquences du programme de mouvement[22]. Le PMAC gère 32 programmes PLC qui sont généralement utilisés pour :

-Le changement des gains,
-L'envoie des messages,
- La configuration hardware,
-La commande des actions,
-Le démarrage/l'arrêt des programmes de mouvement[2].

La saisie d'un programme PLC se fait de la même manière que les programme de mouvement, son exécution se fait par la commande $ENABLEPLCN$ dans le terminal de commande, avec n le nombre correspondant au programme exécuté (entre 0 et 31)[22].

d. Élaboration d'un programme de mouvement

Un programme de mouvement PMAC est élaboré pour faire passer les données a la routine de génération de trajectoires qui calcule la série des positions consécutives du mouvement et génère les commandes des servomoteurs a chaque cycle[22].

La procédure suivie lors de l'écriture et le chargement d'un programme de mouvement est la suivante :

- **Étape1 :** Définir le système de coordonnée et les différents axes,
- **Étape2 :** Créer les parenthèses OPEN et CLOSE respectivement au début et à la fin du programme pour l'ouverture et la fermeture des buffers,
- **Étape3 :** Sélectionner le mode du mouvement,
- **Étape4 :** Sélectionner le mode absolu ou le mode incrémental,
- **Étape5 :** Configurer la vitesse, l'accélération et les caractéristiques temps,
- **Étape6 :** Programmer les mouvements,
- **Étape7 :** Charger le programme dans l'UMAC,
- **Étape8 :** Exécuter le programme de mouvement en introduisant les commandes suivantes dans le terminal de PEWIN32PRO :

1. $pJ/$ **:**Configuration du moteur p,
2. n **:**Configuration du système de coordonnée n,
3. Bm **:** Définir le programme a exécuté en entrant le numéro$m$ du programme,
4. R **:** Exécuter le programme.

d.1. Les modes de mouvement :

1. **Le mode « LINEAR » :**
   C'est un mouvement ayant un profil de vitesse linéaire [20]. Dans ce mode, le mouvement est définit par son temps de monté TM et son temps d'accélération TA [22].

Le profil de vitesse d'un tel mode est présenté par la figure (1.9).

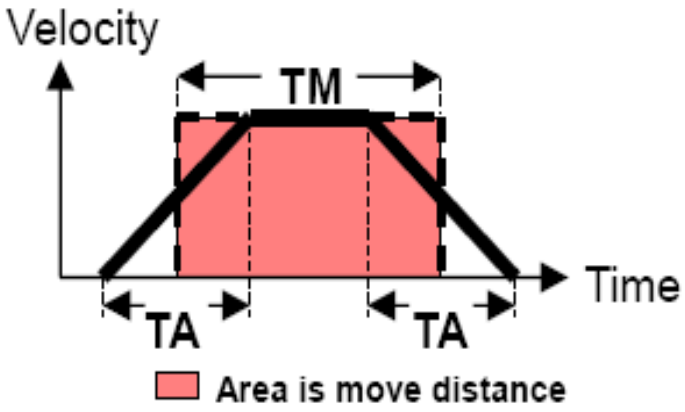

FIGURE 1.9: Profil de vitesse d'un mode linéaire[2].

2. **Le mode « CIRCLE » :**

Utilisé pour effectuer un mouvement entre deux points en suivant une trajectoire circulaire (arc de cercle), ce mode présente deux variantes : *CIRCLE1* et *CIRCLE2*. La différence entre *CIRCLE1* et *CIRCLE2* est le sens de rotation : *CIRCLE1* sens horaire et *CIRCLE2* sens trigonométrique[20].

La syntaxe du mode circulaire est donnée par :

- $CIRCLE$ (1 ou 2)
- TM (temps de mouvement) ou F (la vitesse)
- X« data » Y « data» R «data» ; donner la position finale (mode *(ABS)* ou *(INC)* et le rayon du raccordement ou bien : X « data» Y « data» I «data» J «data» ; donner la position finale (mode *(ABS)* ou *(INC)* et le centre du cercle (mode ABS ou INC).

3. **Le mode « SPLINE » :**

Le mouvement SPLINE permet le raccordement d'une série de mouvement avec une interpolation cubique, pour éviter un changement brusque de vitesse et d'accélération[20].

4. **Le mode « PVT » :**

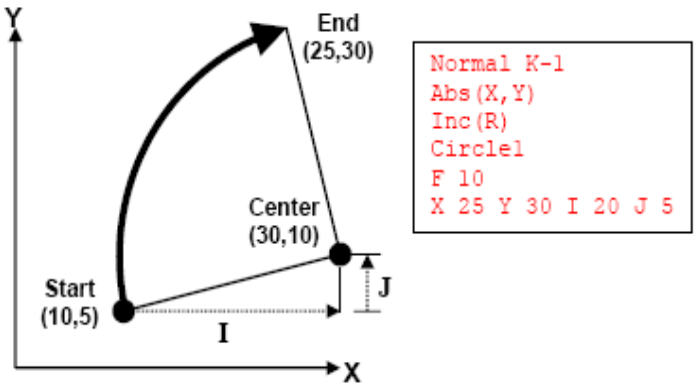

FIGURE 1.10: Profil de vitesse d'un mode circle.

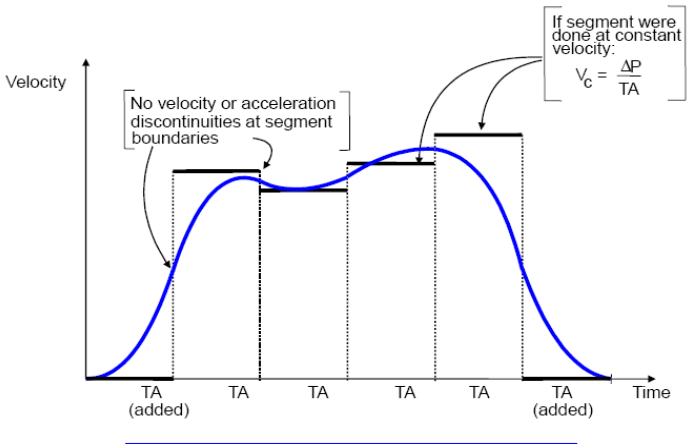

FIGURE 1.11: Profil de vitesse d'un mode SPLINE[2].

On l'utilise pour un contrôle plus direct de la trajectoire. Dans ce mode, l'utilisateur indique la position et la vitesse finales, ainsi que le temps de mouvement. Le processeur génère automatiquement le profil de vitesse optimum. Cela nécessite plus de calcul, mais permet une commande plus précise de la trajectoire[20].

d.2. Les modes de position

Deux modes de position sont possibles lors de la programmation d'un mouvement sous PEWIN32PRO :

- Le mode absolu *abs* **:**Ou le mouvement est défini par les deux positions initiale et finale,

- Le mode incrémental *Inc* **:** Ou le mouvement est défini par la position initiale et le déplacement

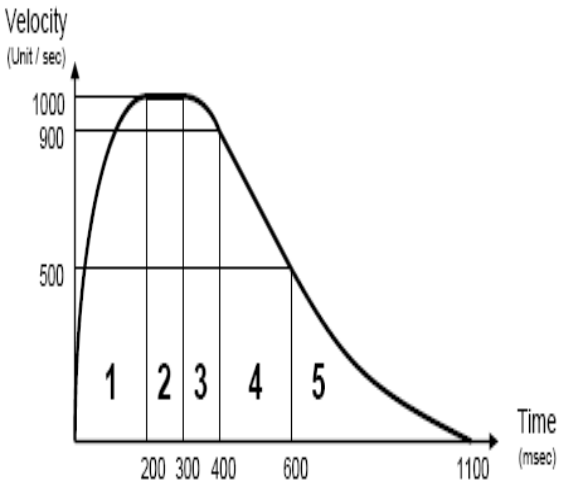

FIGURE 1.12: Profil de vitesse d'un mode PVT.

e. La régulation PID :

La commande des articulations du robot par l'interface UMAC se fait grâce à un régulateur PID intégré. Cette régulation nécessite en premier lieu, l'implémentation des modèles géométrique et cinématique du robot dans l'interface UMAC puis la configuration des différents gains du PID.

Dans ce qui suit, on exposera la manière de saisir les différents modèles du robot puis on donnera le schema du régulateur PID de l'interface UMAC, ses différents paramètres ainsi que leur configuration.

e.1. Implémentation des modèles géométrique et cinématique dans l'interface UMAC

Les différents modèles du robot sont implémentés dans l'UMAC grâce à des programmes spéciaux forward-kinematic (programme géométrique direct) et inverse-kinematic (programme géométrique inverse). Ils sont considérés comme des sous-programmes des programmes de mouvements, ils permettent le passage de coordonnées cartésiennes aux coordonnées articulaires et vis versa. Ils sont exécutés automatiquement aux temps appropriés une fois qu'ils sont activés (en affectant la valeur 1 à la variable $Isx50$). On trouve également le programme géométrique inverse pour le mode PVT qui permet l'utilisation des calculs cinématiques (modèle cinématique)[20].

e.2. Le Régulateur PID implémenté sous UMAC

Un algorithme de commande PID en association avec une boucle de compensation Feedforward et d'un filtre éjecteur, est implémenté dans l'interface. Le schéma de cette

régulation est donné par la figure 3.18.

Ce régulateur PID est caractérisé par les les paramètres suivants :

- **Kp :**$Ixx30$, Gain de l'action proportionnelle,
- **Kd :** $Ixx31$, Gain de l'action dérive,
- **Kvff :**$Ixx32$, gain de la vitesse feedforward,
- **Ki :** $Ixx33$, Gain de l'action intégrale,
- **IM :**$Ixx34$, mode integration,
- **Kaff :**$Ixx35$, gain de l'accélération feedforward[2].

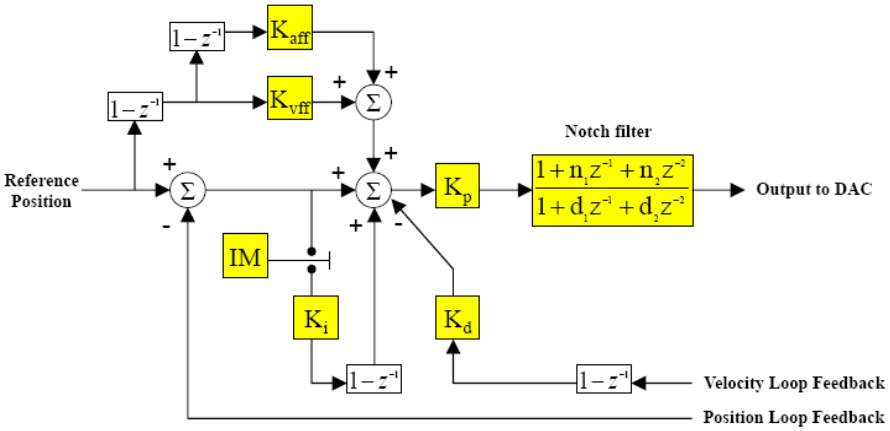

FIGURE 1.13: Schema du regulateur PID sous PEWIN32 [2].

1.6 Conclusion

Ce chapitre nous a permis de traiter la théorie des bras manipulateurs, en ce qui concerne la stratégie de réalisation d'une tâche, dont nous avons détaillé les étapes à savoir : la perception , la génération de mouvement et la commande. Nous avons par la suite calculé le modèle dynamique en nous basant sur le formalisme de *Lagrange-Euler*. Et en fin, nous avons donné une description du bras manipulateur et de l'interface de commande utilisés.

Chapitre 2

État de l'art sur l'asservissement visuel

2.1 Introduction

Dans le domaine de la robotique, la tendance est maintenant à l'utilisation de capteurs extéroceptifs sans contact et notamment des capteurs de vision dans la boucle de commande, c'est la commande référencée vision ; une telle approche peut se révéler très utile pour la commande des bras manipulateurs notamment en milieu industriel pour par exemple : compenser les erreurs de positionnement, la saisie d'objets se déplaçant sur un convoyeur ou plus généralement permettre au robot de s'adapter à son environnement[23].

Dans ce chapitre on commencera tout d'abord par définir le principe de l'asservissement visuel, puis on donnera quelques notions de base sur le traitement d'image, un domaine qui est étroitement lié à la commande référencée vision, par la suite on traitera les procédures d'intégration de la vision dans la boucle de commande en l'occurrence, la fonction de tâche, la modélisation de la camera et le calibrage. Enfin, on va exposer les différentes techniques d'asservissement visuel qu'il est possible d'utiliser pour positionner, grâce au retour visuel, l'outil terminal d'un bras manipulateur par rapport à son environnement.

2.2 L'asservissement visuel

L'intégration de la vision dans la boucle de commande a donné naissance à une nouvelle discipline en robotique, l'asservissement visuel. Il consiste à contrôler les mouvements d'un système robotique en utilisant des informations visuelles, notées s, issues d'un ou plusieurs capteurs de vision, embarqués ou non sur le système[15].L'asservissement visuel est définit comme étant le domaine qui permet d'apporter des solutions à deux problèmes :

-La possibilité d'extraire les informations suffisantes de l'image donnée par le capteur de vision,

-L'implémentation d'algorithmes de contrôle assez simples pour pouvoir travailler à une fréquence compatible avec celle du capteur de vision ; mais aussi une robustesse acceptable vis-à-vis des erreurs inévitables entre l'environnement et l'image fournie par la vision.

La première étape de l'asservissement visuel consiste à extraire de l'image fournie par le capteur de vision, des formes caractéristiques qui constituent l'objectif à atteindre appelées primitives, la deuxième étape est le design d'une commande qui assurera la convergence vers la configuration correspondante à l'image de l'objectif, et ceci en débutant d'une configuration initiale quelconque.

De nombreux chercheurs se sont penchés sur ce problème dès le début des années 70, mais leurs travaux ont été limité par les contraintes technologiques à savoir la puissance de calcule des processeurs de l'époque, ce n'est qu'en 1982 que des travaux plus formelle ont été fait, ils ont exposé deux types de problèmes : le premier étant le choix et l'extraction des primitives à utiliser, et le deuxième étant l'analyse et la synthèse des commandes[23].

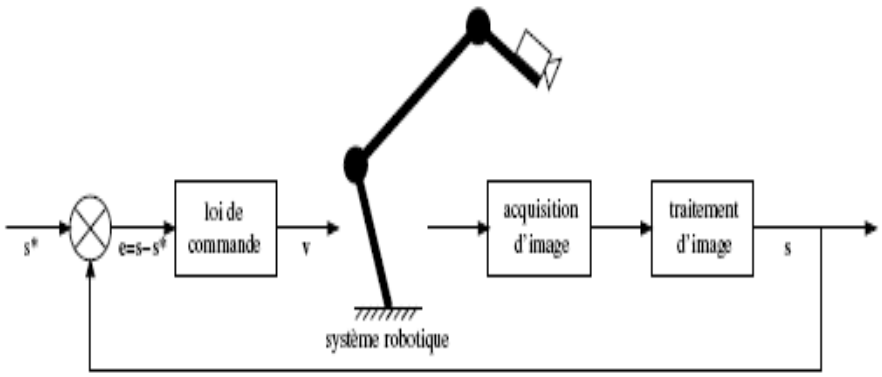

FIGURE 2.1: Schéma de principe des stratégies de commande référencées vision[3].

2.3 Notions du traitement d'images

Le traitement d'images est un thème de recherche situé entre l'informatique et le traitement du signal[24],né de l'idée et de la nécessité de remplacer l'observateur humain par la machine[25]. L'objectif de ce paragraphe est de présenter la notion d'image ainsi que les différentes opérations d'analyse d'images telles que la détection de contours, la segmentation et le traitement haut niveau.

2.3.1 Notions de l'image

- L'image

une image est une représentation planaire d'une scène ou d'un objet situé en général dans un espace tridimensionnel.Son élaboration résulte de la volonté de proposer une entité observable par l'œil humain.Ceci explique d'une part son aspect planaire et d'autre part le fait que l'information élémentaire associée à chaque point de l'image soit transcrite en niveau de gris ou en couleur[25].

- L'image numérique

Une image numérique est définie comme un signal fini bidimensionnel échantillonné à valeurs quantifiées dans un certain espace de couleurs. Elle est constituée de points (pixels)[24]. Par Pixel on désigne le plus petit élément d'une surface d'affichage auquel on peut associer individuellement une couleur et une intensité.Dans le cas d'un écran monochrome,le pixel s'identifie à un point et dans le cas d'un écran couleur ,il est constitué de trois points de couleur différente (rouge,vert,bleu)[26].

- Image en niveau de gris

Le niveau de gris est la valeur de l'intensité lumineuse en un point. Une image en niveau de gris est une image composée de points gris plus ou moins foncés[27],donc autorise un dégradé de gris entre le noir et le blanc. En général, on code le niveau de gris sur un octet (8 bits) soit 256 nuances de dégradé[24].

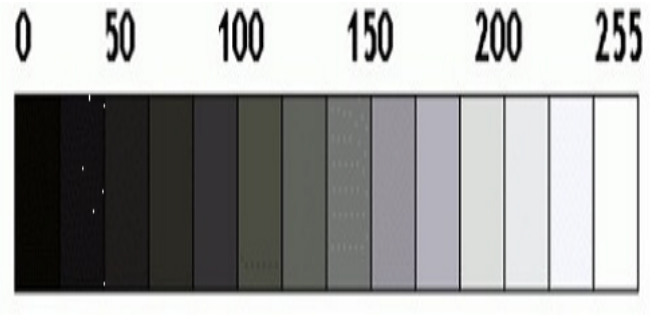

FIGURE 2.2: Les niveaux de gris.

- Caractéristique de l'image

1. **Définition et Résolution :**
   On appelle définition le nombre de points (pixel) constituant l'image, c'est-à dire le nombre de colonnes de l'image que multiplie son nombre de lignes.
   La résolution est le nombre de points par unité de surface, exprimé en points par

pouce PPP,un pouce représentant 2.54cm [6].

2. **L'histogramme :**
   L'histogramme représente la répartition des pixels en fonction de leur niveau de gris. Il fournit diverses informations comme les statistiques d'ordre et permet d'isoler des objets [28].Par convention un histogramme représente le niveau d'intensité en abscisse en allant du plus foncé au plus clair 2.3.L'histogramme est un outil privilégié en analyse d'images car il représente un résumé simple[6].Ainsi l'histogramme d'une image en 256 niveaux de gris sera représenté par un graphique possédant 256 valeurs en abscisses,et le nombre de pixels de l'image en ordonnées.

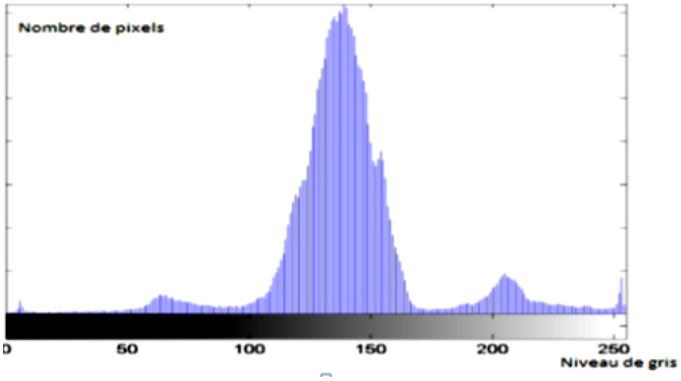

FIGURE 2.3: Histogramme d'une image.

2.3.2 Les techniques de traitement d'images

Selon le niveau d'abstraction,les techniques de traitement d'images peuvent être classées en trois catégories :

- Les processus de bas niveau, qui nécessitent très peu d'informations sur le contenu des images.
Il s'agit de processus qu'on peut regrouper sous l'appellation pré traitement d'images et extractions d'indices ;
- Les processus de niveau intermédiaire, qui englobent les techniques de segmentation d'images en régions ;
- Les processus de haut-niveau, qui peuvent nécessiter des informations sur le contenu des images, et dans lesquels on trouve, la reconstruction tridimensionnelle (modélisation de l'environnement), la reconnaissance de formes, l'analyse de mouvements,... etc[6].

a. Les Prétraitements

1. **Seuillage**
   Une image numérique, pour pouvoir être exploitée a généralement besoin d'être simplifiée, c'est le but du seuillage. Ce dernier consiste à transformer l'image codée sur $6, 8$ ou 16 bits en une image binaire ou les pixel à 1 correspondent aux objets et les pixels à 0 au fond de l'image [29].

   Le seuillage permet de sélectionner les parties de l'image qui intéressent l'opérateur, par exemple 2 types de grains (noirs et blancs) dans un mélange,pour cela il suffit de fixer un seuil ou plus selon lequel ou lesquels, on affecte au pixel en cours la couleur noire ou blanche [30].Deux types de seuillage existent :

   - **Seuillage simple**
     Consiste à fixer un seuil entre 0 et 255, et décider par la suite d'affecter la couleur noire ou blanche à un pixel de l'image selon que le niveau de gris est inférieur ou supérieur au seuil.
   - **Seuillage par Hystérésis**
     Dans cette méthode,on choisi deux seuils A et B tels que $A < B$. En parcourant l'image initiale, on crée une nouvelle en affectant la couleur noire au pixel donc le niveau de gris est soit inférieur à A soit supérieur à B et la couleur blanche ailleurs. Ce type de seuillage permet de mettre en évidence un ensemble précis de pixels où leur niveau de gris est entre A et B [30].

2. **Filtrage**
   Il consiste à appliquer une transformation à l'ensemble ou une partie d'une image numérique en utilisant des filtres[30] afin d'améliorer le contraste entre les zones de l'image à conserver après seuillage, et celles que l'opérateur juge inutiles [29].Son principe est de modifier la valeur des pixels d'une image, généralement dans un but d'améliorer son aspect,il s'agit de créer une nouvelle image en se servant des pixels de l'image d'origine. Sa qualité sera améliorée [6].
   f On distingue généralement les types de filtres suivants [30] :

   - **Filtre passe-haut**
     Atténue les composantes de basse fréquence de l'image et permet d'accentuer les détails et le contraste.il est représenté par *le gradient* et le *Chapeau haut de forme.*

   - **Filtre passe-bas**
     Consiste à atténuer les composantes de l'image ayant une fréquence haute ainsi que le bruit de l'image. On l'appelle aussi *filtre de lissage.*Ces filtres peuvent être linéaires ou non linéaire,on cite par example le filtre *Gaussien* qui est un opéra-

teur de lissage utilisé pour estomper les détails et le bruit.

b. Les techniques de segmentations

La segmentation d'une image binaire a la même finalité que le seuillage ; d'ailleurs, certains auteurs classent le seuillage parmi les méthodes de segmentation. Elle consiste à séparer l'image en plusieurs zones. La segmentation la plus simple consiste à séparer les particules convexes apparues comme connexes à la suite des opérations précédentes[29].

FIGURE 2.4: L'image originale.

FIGURE 2.5: L'image segmentée.

Il existe de nombreuses méthodes de segmentation que l'on peut regrouper en quatre principales classes d'algorithmes [30] :

- Segmentation basée sur les régions (region-based-segmentation) ;
- Segmentation basée sur une approche globale de l'image(seuillage,histogramme,approches basées sur le nuage couleur...etc ;
- Segmentation basée sur les contours(edge-based-segmentation) ;
- Segmentation basée sur la coopération entre les trois premières segmentations.

- Détection de contours

Plusieurs techniques de détection de contour existent dans la littérature. Cependant, comme un contour est défini comme étant une zone de forte discontinuité des pixels d'une image, ces méthodes se basent toutes sur le même principe, qui est celui du gradient.

Parmi les filtre les plus utilisés ,on peut citer :

- Masque de Prewitt,
- Masque de Sobel,
- Masque de Kirsch,

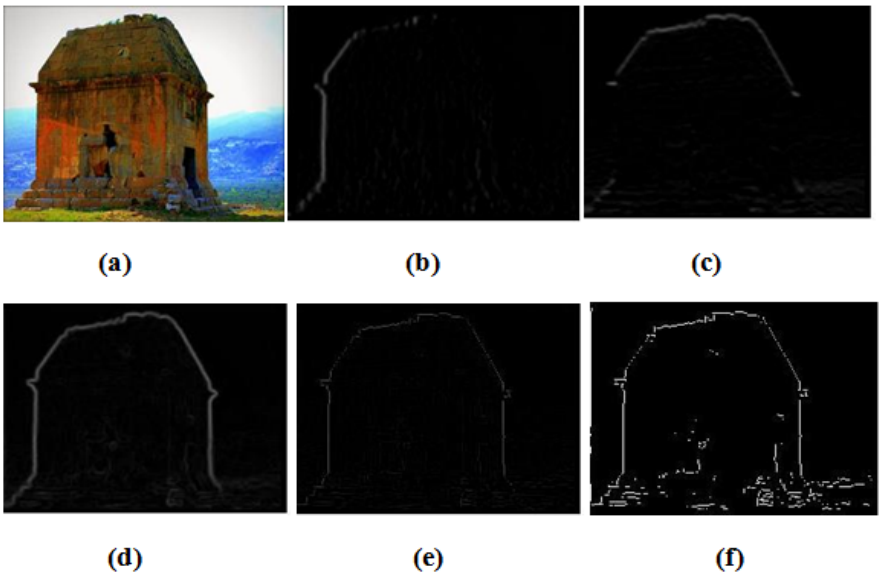

FIGURE 2.6: Détection de contour avec Canny :(a) image originale, (b) image après dérivation suivant X, (c) image après dérivation suivant Y, (d) image après application du gradient, (e) image après seuillage, (f) image après hystérésis.

- Filtre de SHEN et CASTAN,
- Le Filtre de Canny.

La détection de contour par la méthode *Canny* commence par la convolution de l'image avec une gaussienne d'un sigma donné. Après convolution, les dérivées suivant X et suivant Y sont calculées, ce qui permet d'avoir le gradient de l'image.

Une fois le gradient obtenu, un seuillage est appliqué à limage. Ceci a pour but d'éliminer les pixels qui ne constituent pas un maximum local. Finalement, un operateur hystérésis est appliqué à l'image, ce qui a pour effet de marquer chaque pixel comme appartenant ou non au contour. La figure (2.6) montre les différents résultats obtenus par application de l'algorithme Canny à une image donnée.

c. Les techniques de haut niveau

Une fois les informations extraites sous forme d'un ensemble d'entités, la dernière étape consiste à prendre une décision en fonction de ces informations.Selon l'application visée, cette décision peut être une simple valeur booléenne (l'image fait partie de la catégorie visée ou non). Ou bien une commande moteur (le cas de l'asservissement et de l'évitement d'obstacles).ou un ensemble plus complexe de traitements (mise à jour d'une carte par exemple)[6].

2.4 Retour visuel

L'utilisation du retour visuel dans les boucles de commande a été initialement proposé par *Weiss*[31] et nommé *look and move algorithm*.Sa principale idée est de réguler l'erreur sur les modèles internes des systèmes en utilisant un feedback visuel pour mesurer la position de l'outil terminal du robot et de l'objet considéré [3].

La vision si elle a permis d'envisager de nombreuses perspectives en robotiques, a bien évidement amené de nombreux problèmes théoriques concernant notamment les points suivants :

- Le choix des informations à prendre en compte pour réaliser la tâche donnée,
- L'identification de la liaison entre le capteur et l'effecteur terminal du robot,
- La modélisation des capteurs,
- L'élaboration et l'étude de lois de commande en boucle fermée intégrant des informations visuelles [32].

2.4.1 Formalisme de fonction de tâche

Assurer une tâche en robotique par asservissement visuel requiert la sélection de primitives visuelles appropriées et l'élaboration d'une loi de commande en boucle fermée. La première permet de définir une fonction de tâche avec des propriétés qui assurent que la tâche désirée sera satisfaite, la seconde permet de réguler cette fonction de tâche.

Si s représente les K informations visuelles extraites de l'image, la fonction de tâche sera donnée par :[1]

$$e(q,t) = s(q,t) - s^*(t) \tag{2.1}$$

Des conditions ont été posées par *Samson* dans [33] pour que le problème de la régulation de la fonction de tâche soit bien posé :

a. La fonction de tâche doit posséder la propriété de l'unicité de la trajectoire $qr(t)$ solution de l'équation :

$$e(q,t) = 0. \tag{2.2}$$

Or, la fonction de tâche étant non linéaire en q, la résolution de cette équation peut conduire à la définition de plusieurs trajectoires solutions distinctes. Pour éviter cela, une condition initiale q_0 est introduite, telle que : qr(0)=q_0 vérifie :

$$e(q_0,0) = 0 \tag{2.3}$$

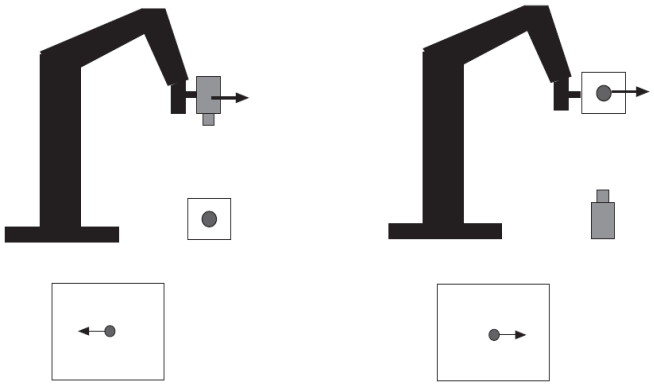

FIGURE 2.7: Configuration camera/robot :(a) configuration eye-in-hand (b) configuration eye-to-hand [1].

Notons qu'une trajectoire vérifiant cette propriété est dite trajectoire idéale. Cependant, il se peut qu'aucune solution n'existe, excepté la condition initiale, ou au contraire, qu'il y en ait une infinité de solutions.

b. D'une part, il est nécessaire que la régulation de la fonction de tâche $e(q,t)$ à zéro impose la convergence de la trajectoire du robot vers la trajectoire solution désirée, et d'autre part qu'une petite variation de $e(q,t)$ n'induise pas une grande variation de la configuration q du robot.
Lorsque toutes les conditions requises sont satisfaites, la fonction de tâche est dite *Admissible* , ce qui permet alors la synthèse de lois de commande efficaces [34].

2.4.2 Configuration camera/robot

On distingue généralement deux types de configurations entre la camera et le robot :

-La première consiste à utiliser une camera fixe qui observe la scène, cette configuration est dite Eye-to-hand (voire figure 2.7.a). Dans cette configuration, les informations visuelles permettent de commander le robot mais le déplacement de celui-ci n'interagit pas sur la position de la camera.

-La seconde possibilité consiste à embarquer la camera sur le manipulateur, donc la camera va évoluer dans la scène avec l'organe qu'elle contrôle. Cette configuration est dite Eye-in-hand (voire figure 2.7.b) [16].

2.4.3 Graphe du système avec camera embarquée

L'utilisation d'une camera embarquée nécessite la définition de plusieurs repères pour exploiter pleinement l'information provenant du système de vision [4].la figure

(2.8) présente tout les repères et les transformations qui permettent de modéliser le système.

Définissons les repères :

- R_w : Repère monde ;
- R_b : Repère lié a la base du bras manimulateur.Il est considéré fixe et confondu avec le repére monde,
- R_c : Repère lié à la camera,
- R_P : Repère lié au poignet du bras manipulateur,
- R_t : Repère lié à l'organe terminal,
- R_o : Repère lié à l'origine de l'objet,
- R_p :Repère lié à la position de prise de l'objet.

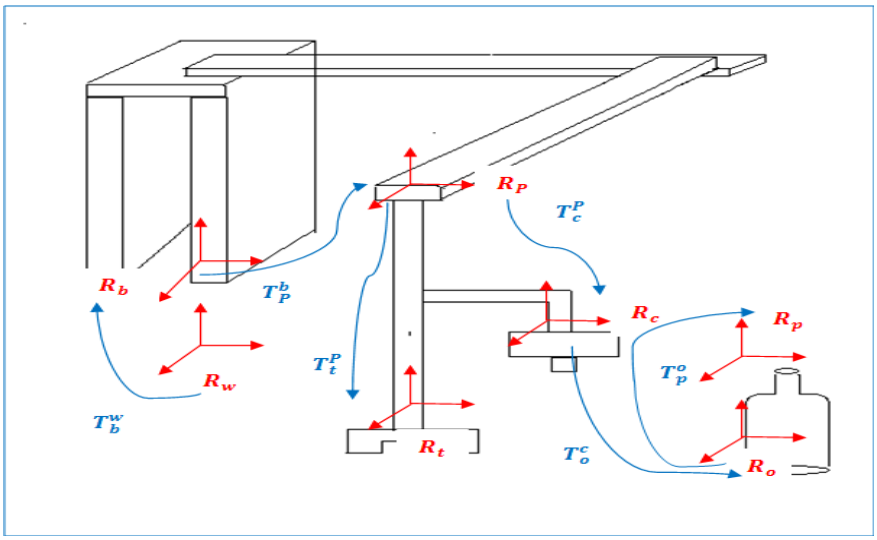

FIGURE 2.8: Graphe des repères du système avec camera embarquée(cas d'un robot SCARA).

Les transformations qui interviennent :

-T_b^w : Transformation qui lie la base au monde. Lorsque le repère de base et le repère monde sont confondus, cette matrice est égale à la matrice identité,
-T_P^b : Transformation qui lie la base du robot à son poignet, elle est donnée par le modèle géométrique direct du robot,
-T_t^P :Transformation qui définit la position de l'organe terminal,
-T_c^P :Transformation correspondant aux paramètres extrinsèques de la caméra, elle lie la caméra au Repère poignet. Lorsque on parle d'une configuration caméra embarquée, la détermination de cette transformation est appelée *calibrage Eye in hand* dans la littérature,
-T_o^c :Transformation calculée par le système de vision, elle fournit la position de l'objet à partir des Informations visuelles. Cette transformation est dite calcul de pose,

-T_p^o :Transformation qui lie le repère de l'objet et la position de prise. [4].

2.4.4 Configuration Eye-in-hand

Comme déjà mentionné, dans une configuration *eye-in-hand*,la camera est directement attachée à une articulation du robot, généralement l'outil terminal. La position de la camera change en fonction de la position des articulations du bras et l'image de l'objet peut être utilisée pour calculer la matrice de transformation objet-outil terminal et guider ainsi la tâche d'asservissement visuel.

Ceci est basé sur le fait que la camera est solidement attachée à l'organe terminal et sa position par rapport à celui-ci est constante. Il est possible cependant que dans certaines configurations, la camera soit placée sur une articulation intermédiaire du bras.

Cette configuration a l'avantage que quand la camera approche l'objet, la précision augmente,car la résolution dans l'image augmente.

L'inconvénient majeur est que lorsque le bras est en mouvement, la pose de l'objet change,et même des objets stationnaires doivent être suivis dans l'image.Ceci amène toujours le risque que l'image de l'objet sorte du champ de vision de la camera.

Ce problème dérive de la mécanique même du robot lorsqu'il est en mouvement, comme les vibrations. Un autre problème surgit du fait que placer une camera sur le bras modifie les paramètres dynamiques de celui-ci, mais ce problème peut être minimisé en utilisant une camera à poids négligeable [3].

2.4.5 Calibrage hand-eye

Dans le domaine de l'asservissement visuel, on cherche à commander le mouvement 3D de la camera à partir des informations visuelles. En effet dans le cas d'une camera embraquée celle-ci possède les mêmes degrés de liberté que l'effecteur terminal du bras manipulateur, on accède ainsi à des informations sensorielles directes sur l'interaction qui relie le robot à son environnement. La connaissance de la relation camera-effecteur va nous permettre alors d'obtenir précisément le mouvement correspondant du bras manipulateur.

Cette relation pourrait être exprimée soit dans le repère de l'effecteur, soit par rapport à la base du robot si l'on connait le modèle géométrique direct de ce dernier [16]. Le problème de calibrage camera-effecteur consiste à déterminer la transformation rigide entre le repère de la camera et le repère de l'effecteur. En terme d'équations, c'est la recherche d'une matrice homogène de passage X entre la base de l'effecteur (représentée par le repère R_P) et la base de la camera (représentée par le repère R_c) .

La matrice de passage X représente une rotation R_X et une translation T_X, sa forme générale est donnée par :

$$X = \begin{pmatrix} R_X & T_X \\ 0 & 1 \end{pmatrix} \tag{2.4}$$

L'ensemble camera-outil terminal est supposé rigidement fixé. Il y a deux principales approches pour estimer la transformation X. la première et la plus classique est la formulation $AX = XB$. Cette approche est basée sur *faire bouger l'effecteur et observer le mouvement de la camera* [35]. On place l'ensemble effecteur-camera en deux positions de l'espace et 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_P^1 à R_P^2) [16].

Une autre approche pour résoudre le problème de calibrage est de chercher la solution de $AX = YB$. Cette approche est une estimation simultanée de la transformation *hand-eye* et de la pose du robot dans le système de coordonnées du repère de la scène. X représente la transformation hand-eye qui est inconnue et Y représente la transformation entre le repère de la scène et la base du robot.

Nous avons choisit de traiter la première approche $AX = XB$, car la deuxième, du fait qu'elle comporte la transformation scène-base du robot, exige une calibration chaque fois que la scène change [35].

• **Solution de** $AX = XB$

$$\begin{pmatrix} R_A & t_A \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_X & t_X \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_X & t_X \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_B & t_B \\ 0 & 1 \end{pmatrix} \tag{2.5}$$

De cette égalité matricielle on tire le système d'équations :

$$\begin{cases} R_A R_X = R_X R_B \\ R_A t_X + t_A = R_X t_B + t_X \end{cases} \tag{2.6}$$

Cette approche compte deux étapes. La première consiste à calculer la rotation de $X(R_X)$ en effectuant seulement des translations avec le robot.Quand le mouvement n'inclue aucune rotation,l'équation 2.6 devient :

$$R_A t_X + t_A = R_X t_B + t_X \tag{2.7}$$

$$X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.8}$$

Donc on trouve :

$$t_A = R_X t_B \tag{2.9}$$

Quand on effectue au moins trois translations selon différents axes, l'équation (2.9) conduit à un système de 9 équations linéaires avec 9 inconnus qui peut être résolu en utilisant les moindres carrés.
La seconde étape consiste à calculer la translation de $X(t_X)$, ceci est obtenu en effectuant seulement des rotations selon différents axes. Quand seules des rotations sont faites, l'équation (2.6) devient :

$$R_A t_X + t_A = R_X t_B + t_X \tag{2.10}$$

$$T_B = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.11}$$

$$R_A t_X + t_A = t_X \tag{2.12}$$

En effectuant seulement une rotation selon chaque axe, l'équation (2.12) peut s'écrire sous la forme d'un système d'équations linéaires à 3 inconnus. Sa résolution peut alors être faite grâce à la méthode des moindres carrés [35].

2.5 Modèle de la camera

La situation d'une camera est définie par la position de son centre optique C et l'orientation de son axe optique.Le centre optique est lié à l'origine du repère de la camera R_c et l'axe optique est confondu avec l'axe Z_c de ce repère [4].

Différentes techniques de modélisation d'une camera sont utilisées dans le cadre des travaux en vision artificielle, cependant, le model sténopé ou trou d'épingle *(Pinhole camera model)* est le plus couramment utilisé en vision par ordinateur car il permet de modéliser finement la plupart des capteurs projectifs et de simplifier les équations mises en jeu [6].
Le model sténopé est constitué d'un plan appelé plan rétinien dans lequel l'image se forme à l'aide d'une projection perspective [5], ce plan est placé sur l'axe optique à une distance f du centre optique où f est appelée *distance focale de la camera*. Ce model suppose que, du moment où on utilise une focale de faible dimension [15], tous les rayons de lumière reflétés par l'objet passent par le centre optique C, formant

une image perspective de la scène dans le plan rétinien. A ce moment, le point focal peut être placé devant ou derrière le plan rétinien. Dans le cas où le centre optique est placé devant le plan rétinien, l'image obtenue est une projection inverse de la scène. Dans l'autre cas, l'image obtenue est une projection de la scène[5].

Physiquement, le plan image est le plan $U^{'}V^{'}$ situé à une distance focale de $f^{'}$, mais pour faciliter l'analyse des images, nous utiliserons un plan image virtuel UV situé à une distance focale f en avant du centre optique C(voir figure 2.9). La projection du point de l'objet sur le plan virtuel produit des points qui génèrent une image équivalente ayant la propriété d'être dans le même sens que l'objet [4].

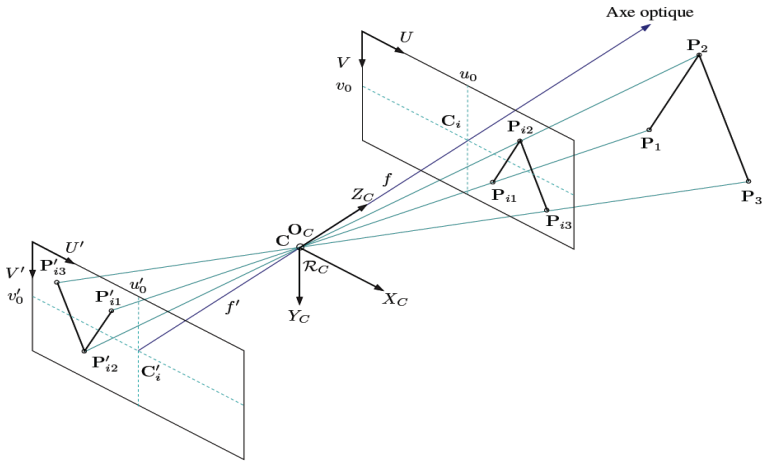

FIGURE 2.9: Modèle sténopé d'une camera avec le plan réel $U^{'}V^{'}$ et le plan virtuel UV [4].

2.5.1 Matrice de projection perspective

Nous nous intéressons à la projection et aux propriétés géométriques du système. Le point P est défini par les coordonnées (exprimées dans le repère de la camera) :

$$P \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.13}$$

Les coordonnées du point P_i correspondent à la projection sur le plan image UV du point P, elles s'expriment dans le repère caméra par :

$$P_i \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{2.14}$$

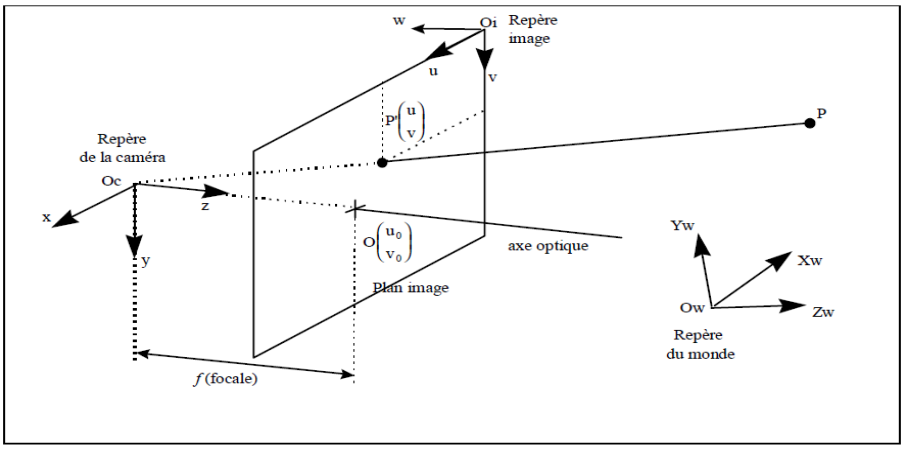

FIGURE 2.10: Représentation d'une projection perspective [5].

Où x_i ety_i représentent les coordonnées dans le plan image :

$$\begin{cases} x_i = f\dfrac{x}{z} \\ y_i = f\dfrac{y}{z} \\ z_i = f \end{cases} \tag{2.15}$$

En utilisant la représentation par coordonnées homogènes, le système s'écrit :

$$s\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} sx_i \\ sy_i \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.16}$$

Où s représente la profondeur du point dans l'image.
La matrice de projection ρ est alors définie par :

$$\rho = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.17}$$

Cette matrice peut être décomposée en un produit $\rho = k_p\rho_0$ [4]
avec ;

$$k_p = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.18}$$

$$\rho_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.19}$$

2.5.2 Transformation affine camera/image

Les points images sont mesurés en pixel dans un repère bidimensionnel uv associé à l'image. Afin de pouvoir écrire la matrice de transformation du repère caméra au repère image, nous devons introduire les paramètres suivants :

-k_u, le facteur d'échelle horizontal ($pixels/mm$),
-k_v, le facteur d'échelle vertical ($pixels/mm$).
-u_0,v_0 et w_0, sont les coordonnées de Oc dans le repère image (mesurées en Pixel)[5].

Nous introduisons en plus les paramètres λ_u et λ_v qui sont respectivement la largeur et la hauteur du pixel car, comme le montre la figure 2.11, les pixels d'une image sont rarement carrés [4].

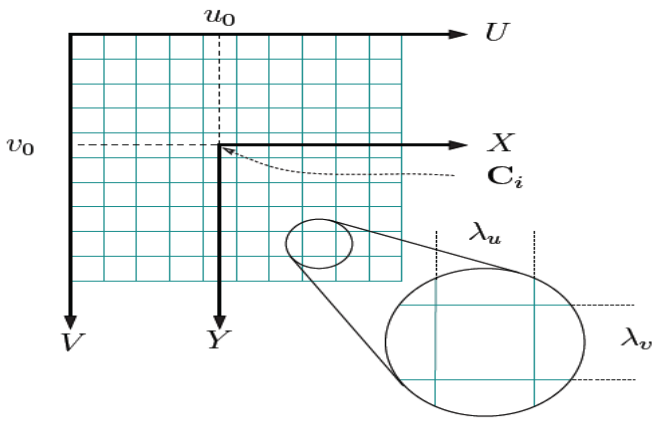

FIGURE 2.11: maillage en pixel et caractéristiques d'une image

$$\begin{cases} k_u = \dfrac{1}{\lambda_u} \\ k_v = \dfrac{1}{\lambda_v} \end{cases} \tag{2.20}$$

La transformation du repère caméra au repère image s'écrit [5] :

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \begin{pmatrix} k_u & 0 & 0 \\ 0 & k_v & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \\ w_0 \end{pmatrix} \tag{2.21}$$

C'est une transformation affine représentant un changement d'échelle et une translation. La composante w étant toujours nulle, on peut ignorer la troisième ligne et écrire cette transformation sous la forme suivante :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{2.22}$$

La matrice K_A définit une transformation affine :

$$K_A = \begin{pmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.23}$$

2.5.3 Paramètres intrinsèques

En faisant une projection perspective suivi d'une transformation affine, on abouti à :

$$\begin{cases} u = k_u f \dfrac{x}{z} + u_0 \\ v = k_v f \dfrac{y}{z} + v_0 \end{cases} \tag{2.24}$$

En multipliant les matrices K_A,K_P,ρ_0, on abouti à une représentation par matrice homogène ;

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} \alpha u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.25}$$

Où :

$$\begin{cases} \alpha_u = f k_u \\ \alpha_v = f k_v \end{cases} \tag{2.26}$$

Ce modèle de la caméra sténopé ne comporte que des paramètres intrinsèques (α_u,α_v,u_0,v_0) et suppose :

-Un montage mécanique parfait,
-Les pixels ont une forme rectangulaire,
-Les lentilles ne présentent aucune distorsion [4].

2.6 Obtention de la consigne S^*

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 [16].
Quelle que soit la nature des informations visuelles choisies, la consigne S^* s'obtient très généralement en fixant à priori la situation à atteindre entre la camera et l'objet d'intérêt, ou encore par apprentissage :

- Dans un premier cas, si S comprend des informations visuelles, leur valeur désirée s'obtient aisément si un modèle de l'objet est disponible : il suffit d'appliquer les équations de projection perspectives pour calculer la position de l'objet dans l'image. Cependant, toute erreur sur les paramètres de modélisation de la camera, sur le modèle de l'objet et éventuellement le calibrage camera/effecteur aura pour conséquence que, quand la valeur de S sera égale à S^*, la situation réellement atteinte sera différente de celle spécifiée, en raison du biais introduit par les erreurs de modélisation.

- L'obtention de la consigne par apprentissage consiste à amener dans une phase préalable le robot à sa position désirée par rapport à l'objet, d'acquérir l'image correspondante et de calculer la valeur de S^* exactement de la même façon que pour les futurs calculs de $S(t)$. En présence d'erreurs de modélisation, on se retrouve avec le cas paradoxale où la consigne et les mesures sont biaisés, mais où la situation atteinte après convergence est correcte aux seules erreurs de mesures prés[15].

2.7 Techniques d'asservissement visuel

2.7.1 Asservissement visuel 3D

Dans l'asservissement visuel 3D, la référence est exprimée sous la forme d'une attitude, notée r^*,d'un repère lié à la cible vue par la caméra par rapport à un repère lié au robot.Cette mesure s'obtient grâce aux primitives extraites de l'image et à un modèle géométrique de la cible, mais elle est très sensible aux incertitudes sur ce modèle ainsi qu'aux erreurs de calibrage de la caméra [15].

La tâche à réaliser s'exprime alors sous la forme d'une situation de référence $s^* = r^*$ à atteindre. La commande repose ainsi sur la détermination de la situation $s = r$ de la camera, à partir des informations visuelles extraites de l'image (voir figure 2.12).

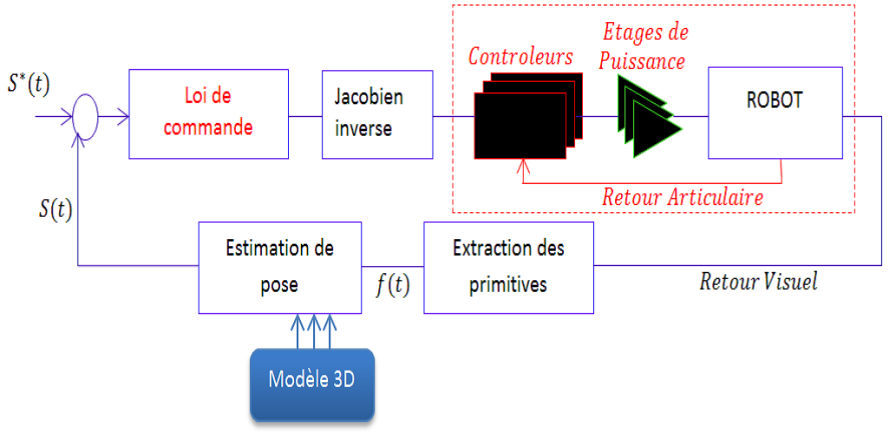

FIGURE 2.12: Schéma asservissement visuel 3D [6].

De nombreuses méthodes permettent d'estimer la situation d'une camera par rapport à un objet à partir de l'image perçue de cet objet. Elles reposent très généralement sur la connaissance à priori d'un modèle (3D ou 2D) de l'objet et des paramètres intrinsèques de la camera. Ces méthodes utilisent des informations visuelles de différentes natures, telle que des points, des droites, des ellipses résultant de la projection de cercles ou de sphères ou encore des objets cylindriques [6].

En général, l'avantage principal de cette approche est que la trajectoire du système caméra/robot est commandée directement dans un repère cartésien. Ceci permet une planification de trajectoires plus facile. Cependant, particulièrement dans le cas de la configuration *eye-in-hand*, les repères visuels utilisés pour l'évaluation de la position peuvent sortir de l'image. Si la caméra est grossièrement calibrée, les positions courantes et désirées de la caméra ne seront pas estimées exactement, ce qui mènera à de faibles performances[4].

2.7.2 Asservissement visuel 2D

Les techniques de l'asservissement visuel 2D *(image-based visual survoing)* utilisent, elles, directement les informations visuelles s extraites de l'image, c'est-à-dire qu'elles ne nécessitent pas la phase d'estimation de r^* contrairement à l'asservissement visuel 3D.Les lois de commande consistent alors à contrôler le mouvement de la camera afin que les mesures dans l'image $s(t)$ atteignent une valeur désirée s^* , voire suivent une trajectoire spécifiée $s^*(t)$ [7]. Des tâches typiques comme le suivi et le positionnement sont alors accomplies en réduisant l'erreur entre un ensemble de primitives but et un ensemble de primitives courantes du plan image (voir figure 2.14) [4].
Une primitive est une forme géométrique élémentaire (point, segment de droite,

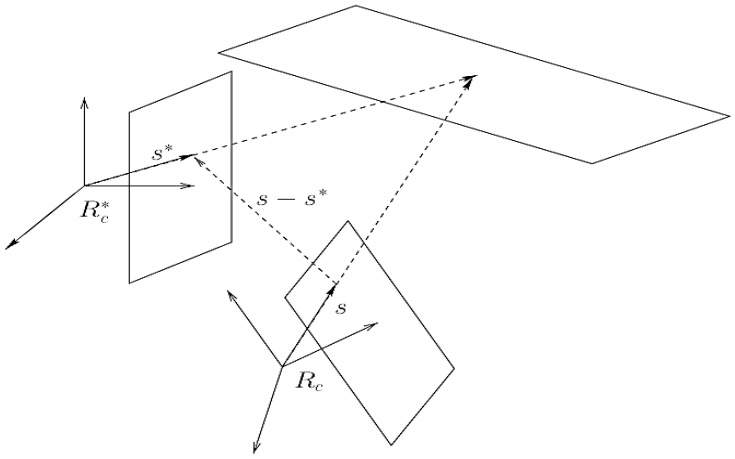

FIGURE 2.13: Principe d'asservissement visuel.

portion d'ellipse, ...etc.). Elle sert à modéliser la projection d'un objet dans le plan image. Les premiers travaux traitant de l'asservissement 2D ont utilisé 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 requérir un temps de traitement d'image faible. Il faut noter que ce choix de primitives simples et faciles à extraire était motivé par les limitations existantes à l'époque, en termes de ressources de calcul [15].

L'asservissement visuel 2D implique le calcul de la matrice d'interaction qui relie les coordonnés du repère de la scène à celles du repère de la caméra. Le calcul de la matrice d'interaction exige la connaissance des paramètres intrinsèques et extrinsèques de la caméra.

Cette méthode est considérée comme une commande très robuste par rapport aux erreurs de calibration de la caméra et du robot. La calibration faible affecte seulement le taux de convergence de la loi de commande dans le sens où un temps plus long est nécessaire pour atteindre la position désirée [4].

2.7.3 Asservissement visuel 2D 1/2

Chaumette a proposé cette méthode dans le cadre de la thèse d'*Ezio Malis* en 1998 [36].Cette approche est à mi-chemin entre les approches 2D et 3D. Elle évite leurs inconvénients respectifs : contrairement à l'approche 3D elle ne nécessite pas de modèle géométrique 3D de l'objet. Par rapport à l'approche 2D, elle assure la convergence de la loi de commande dans tout l'espace de la tâche [4].

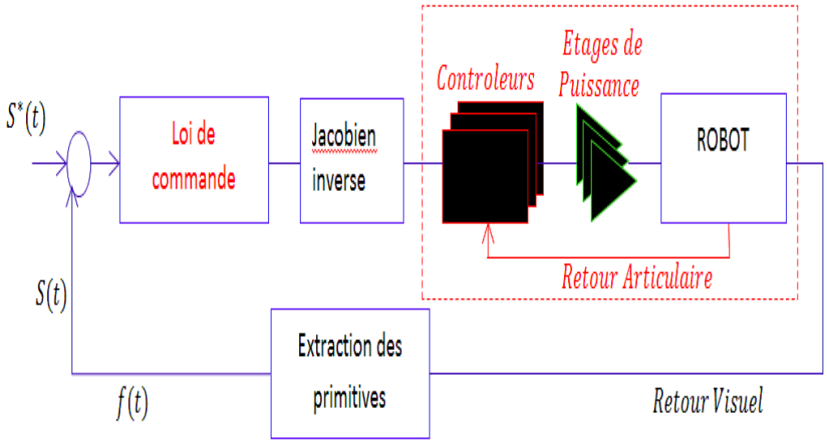

FIGURE 2.14: Schéma asservissement visuel 2D [6].

Cette approche a été qualifiée de méthode 2D 1/2 car les informations utilisées comme mesures de consignes sont, pour certaines d'entre elles, exprimées directement dans l'image et, pour les autres, exprimées dans le repère de la camera (voir figure 2.15).

Elle offre la possibilité de séparer les boucles d'asservissement en rotation et en translation de la caméra, ce qui permet :

-Un fort découplage de la loi de commande ;
-Un contrôle partiel dans l'image permettant de conserver la cible en permanence dans le champ de vision de la camera ;
-Une étude de la stabilité et du domaine de convergence de la loi de commande .

L'avantage majeur de cette approche est que la connaissance d'un modèle géométrique 3D de l'objet n'est plus nécessaire, ce qui lui donne un domaine d'application très important à partir du moment où le motif à atteindre est préalablement déterminé, par exemple lors d'une phase d'apprentissage. La seule information 3D utilisée dans la commande est la profondeur désirée approximative d'un point de l'objet, soit un besoin d'informations à priori bien moindre qu'en asservissement visuel 3D et 2D[7].

2.8 Asservissement visuel 2D

L'asservissement visuel 2D correspond à l'utilisation des informations visuelles directement par la boucle de commande. Par exemple, dans [1], le positionnement de la caméra par rapport à la cible est réalisé en utilisant le formalisme de la fonction de tâche [4].

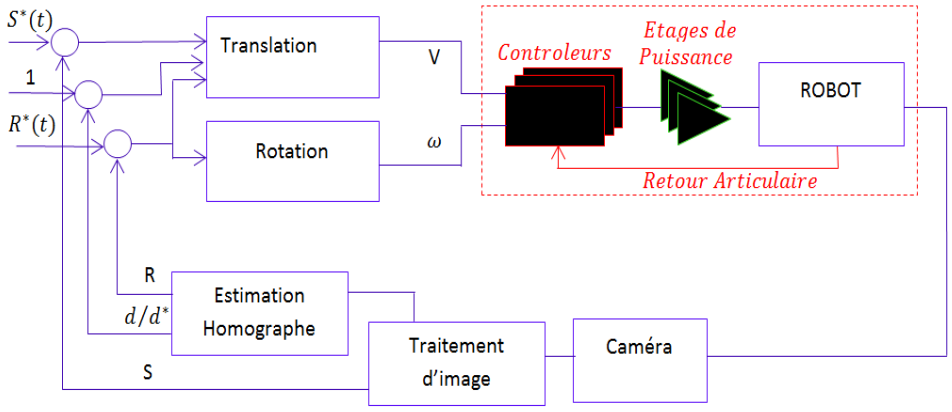

FIGURE 2.15: Schéma asservissement visuel 2D 1/2.

Donc dans le cadre d'un asservissement visuel 2D, le choix des informations visuelles et l'obtention de la relation caractérisant leur variation sont deux points fondamentaux. Les informations visuelles $S(q,t)$ caractérisent les mesures effectuées au moyen de la camera, en fonction de la configuration q du robot et du temps pris en tant que paramètre indépendant [15].

Dans ce paragraphe on traitera en premier lieu des points fondamentaux d'un asservissement 2D à savoir la modélisation des informations visuelles et l'interaction camera/environnement, puis on évoluera vers la régulation de la fonction de tâche qui permettra la commande du bras manipulateur.

2.8.1 Modélisation des informations visuelles

Dans le but d'être prise en compte dans un asservissement visuel, un ensemble S de k informations visuelles doit être défini par une application différentiable de l'espace euclidien $SE3$ vers R_3.

$$S = S(P(t)) \tag{2.27}$$

Où $P(t)$ est un élément de l'espace $SE3$, décrit l'attitude à un instant t entre la camera et son environnement. De la, seul le mouvement de la camera ou de l'objet qu'elle perçoit peut modifier la valeur des informations visuelles.

La différentielle de S nous permet de définir comment les variations des informations visuelles se rapportent au mouvement relatif entre la camera et la scène.

En dérivant 2.27, on obtient :

$$\dot{S} = \frac{\partial S}{\partial P} = L_s V_c \tag{2.28}$$

Avec ;

- L_s : est une matrice $k * l$, notée matrice d'interaction,où l représente le nombre de degré de liberté du bras,

- V_c : vitesse instantanée relative entre la camera et la scène, exprimée dans le repère R_C de la camera en son origine C.

On notera par la suite la vitesse de translation de l'origine du système de coordonné, et par ω la vitesse angulaire, tel que $V_c = (v, \omega)$. Si R_C^O décrit la matrice de rotation du repère R_O ,lié à l'objet, au repère R_C de la camera.On a par définition :

$$[\omega]_x = \dot{R}_C^O (R_C^0)^T = -\dot{R}_O^C (R_O^C)^T = R_O^C \dot{R}_O^C \tag{2.29}$$

Avec,
$[\omega]_x$: Matrice symétrique de pré-produit vectoriel associée à ω [1].

a. configuration Hand-Eye

Si on considère une camera montée sur l'effecteur terminal du bras manipulateur observant un objet statique, la relation entre S et le vecteur des vitesses articulaires $\dot{q}$ peut facilement s'obtenir :

$$\dot{S} = \frac{\partial S}{\partial P} = L_s v = L_s V_n^c J_n^n(q) \dot{q} \tag{2.30}$$

où,
$V_c = L_s V_n^C J_n^n$: est le jacobien des informations visuelles et où :

$J_n^n(q)$ est le jacobien du robot exprimé dans le repère de l'effecteur terminal.
V_n^C est la transformation du torseur cinématique du repère de la camera vers le repère de l'effecteur terminal. Cette matrice qui reste constante dans le cas où la camera est rigidement liée à l'effecteur terminal du bras manipulateur est donnée par :

$$A = \begin{pmatrix} R_n^c & [t_n^c]_x R_n^c \\ 0_3 & R_n^c \end{pmatrix} \tag{2.31}$$

où,

R_n^C et t_n^C sont, respectivement, la matrice de rotation et le vecteur de translation du repère de la camera vers le repère de l'effecteur terminal. Les éléments de cette matrice de transformation peuvent être estimés en utilisant les méthodes du calibrage Hand-eye exposés précédemment.

Il est à noter que les techniques d'asservissement visuel sont généralement robustes vis-à-vis des erreurs de modélisation, soit de la matrice de transformation soit du jacobien du robot.

Plus généralement, si la camera observe un objet en mouvement, la différentielle de S est donnée par :

$$\dot{S} = \frac{\partial S}{\partial t} + L_s V_n^c J_n^n(q)\dot{q} \tag{2.32}$$

où,
$\frac{\partial S}{\partial t}$ Représente la variation de S due au mouvement propre de l'objet (qui est généralement inconnu). Dans le cas peu probable où le mouvement de l'objet est connu, et donné par exemple par son torseur cinématique V_0 dans R_C, on a [1] :

$$\dot{S} = L_s V_n^c J_n^n(q)\dot{q} + L_s V_0 \tag{2.33}$$

2.8.2 Interaction camera/environnement

Comme énoncé auparavant, le choix des informations visuelles et l'obtention de la relation les liant au mouvement de la camera sont deux aspects fondamentaux de l'asservissement visuel 2D. Cette relation, obtenue par dérivation des informations sensorielles S par rapport à la situation r de la camera, est définie par une matrice appelée matrice d'interaction [6] ou Jacobien de l'image par analogie évidente avec le jacobien du robot [7].
La matrice L_s représente le lien entre le mouvement de la caméra et l'évolution des indices visuels. Elle dépend non seulement de la nature des informations visuelles choisies, mais aussi de la situation de la caméra par rapport à l'objet observé, ou, plus précisément de la distance entre la caméra et la cible observée (appelée communément la profondeur) [15]. Cette matrice joue, donc un rôle essentiel dans l'élaboration des diverses lois de commande possibles [7].
Différentes méthodes existent pour caractériser la matrice d'interaction. Ainsi, dans certains travaux, des méthodes d'apprentissage ont été utilisées pour l'estimation en ligne de cette matrice. Dans ce cas, les auteurs traitent des informations visuelles de type points, et c'est directement le produit des matrices $L_s J_n^n(q)$ qui est estimé. Toutefois, avec cette approche il n'est pas possible de démontrer la stabilité des lois de commande associées.
D'autres travaux ont adopté le choix de formuler une expression analytique de la matrice d'interaction, suivant le modèle géométrique de la cible et les primitives utilisées. Ainsi, *Chaumette* a proposé une méthode générale de calcul analytique de la matrice d'interaction pour différentes primitives géométriques simples, telles que des points, des cercles, des sphères, des ellipses, etc, ou des moments de l'image [15]. Nous traiterons dans ce qui suit les deux cas : primitives de type point et primitives de type droite.

a. Cas de primitives de type point

Le modèle mathématique d'une camera est défini par une projection perspective,

de sorte qu'un point M avec les coordonnées (X, Y, Z) projeté dans le plan image est un point m de coordonnées (x, y) avec :

$$\begin{aligned} x &= f\frac{X}{Z} \\ y &= f\frac{Y}{Z} \end{aligned} \tag{2.34}$$

En différentiant cette équation on obtient les variations dans l'image des coordonnées (x, y) en fonction de la vitesse $\dot{M}$ des coordonnées du point M

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \frac{f}{z} & 0 & -f\frac{x}{z^2} \\ 0 & \frac{f}{z} & -f\frac{y}{z^2} \end{pmatrix} \dot{M} \tag{2.35}$$

La vitesse $\dot{M}$ est donnée en fonction du torseur cinématique V par la relation.

$$\dot{M} = [-I_3 \; [M]_x]V \tag{2.36}$$

Avec,
$[M]_x$ Représente la matrice symétrique de pré-produit vectoriel associée à M.

L'équation 2.35 devient alors :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = L_{xy}v \tag{2.37}$$

Où,

$$L_{xy} = \begin{pmatrix} -\frac{f}{Z} & 0 & \frac{x}{Z} & \frac{xy}{f} & -(f + \frac{x^2}{f}) & y \\ 0 & -\frac{f}{Z} & \frac{y}{Z} & (f + \frac{y^2}{f}) & -\frac{xy}{f} & -x \end{pmatrix} \tag{2.38}$$

Il est à noter que les termes introduits par le mouvement angulaire dépendent seulement des mesures de x et y d'autre part, les termes introduit par le mouvement de translation sont inversement proportionnels à la profondeur du point 3D. Cet effet est valable pour tous les types de primitives visuelles [1].

Les algorithmes de traitement d'images exigent des mesures exprimées en pixels. Si on ignore les distorsions non linéaires, dues par exemple à l'utilisation d'une camera à focal de courte longueur d'objectif, la relation entre les coordonnées $(u.v)$ exprimées en pixels et les coordonnées (x, y) métriques est donnée par la matrice la transformation affine camera/image[1].

$$k_A = \begin{pmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.39}$$

A partir de cette transformation on pourra écrire :

$$\begin{pmatrix} x \\ y \end{pmatrix} = K_A^{-1} \begin{pmatrix} u \\ v \end{pmatrix} \tag{2.40}$$

En différentiant (2.40) et en remplaçant dans (2.37) on aura l'expression de la matrice d'interaction exprimé en coordonnée pixel dans l'image L_s

$$L_s = \begin{pmatrix} k_u & 0 \\ 0 & k_v \end{pmatrix} L_{xy} \tag{2.41}$$

L_s est généralement dite *matrice jacobien de l'image*.Finalement il reste à choisir le nombre d'informations visuelles. Il a été montré que le nombre des informations visuelles à choisir doit être de sorte que la dimension du vecteur S soit supérieur au nombre de degrés de liberté du bras manipulateur. D'autres recherches montrent que pour éviter d'avoir une perte de rang de la matrice d'interaction, le nombre minimum d'informations visuelles à choisir doit être d'au moins 3 [16]. Le vecteur des informations sensorielles s est alors défini par : $s = (x_1, y_1, \ldots.., x_k, y_k)^T$, où chaque couple représente les coordonnées métriques du point Pi de la cible, projeté dans l'image. La matrice d'interaction globale correspondant à la cible observée est constituée alors par la superposition des matrices de tous les points de la cible considérée, ce qui donne :

$$L_s = \begin{pmatrix} L_{p_1} \\ . \\ . \\ L_{p_k} \end{pmatrix} \tag{2.42}$$

Où chaque ligne $L_{(P_i)}$ correspond à la matrice d'interaction d'un point Pi . Enfin, différents types de primitives peuvent être construits sur la base de points : centres de gravité, segments, polygones, cercles, etc. [15].

b. Cas de primitives géométrique 2D

Il est aussi possible de calculer la matrice d'interaction relative aux informations visuelles construites à partir de primitives géométriques. Ceci est fait tout simplement en définissant les équations représentant :

La nature de la primitive et sa configuration dans la scène :

$$h(X, Y, Z, P_1, P_2, \ldots, P_n) = 0 \tag{2.43}$$

La projection de la primitive dans le plan image :

$$g(x, y, p_1, p_2, \ldots, p_l) = 0 \tag{2.44}$$

La relation entre la primitive 3D et son image :

$$\frac{1}{Z} = u(x, y, P_1, P_2, \ldots, P_l) \tag{2.45}$$

Comme exemple, nous allons considérer le cas d'une primitive de type *droite* [1].Dans un univers 3D, une droite est représentée par l'intersection de 2 plans non parallèles (voir figure 2.16), cette représentation est donnée dans le référentiel lié a la camera sous la forme :

$$h(X, Y, Z, A_1, ..., C_2) \begin{cases} h_1 = A_1X + B_1Y + C_1Z + D_1 \\ h_2 = A_2X + B_2Y + C_2Z + D_2 \end{cases} \tag{2.46}$$

Projetant ces deux équations dans le plan image en utilisant les formules de projection perspectives pour une distance focale $f = 1$:

$$\begin{cases} x = \dfrac{X}{Z} \\ y = \dfrac{Y}{Z} \end{cases} \tag{2.47}$$

De h_1 on obtient la fonction :

$$\frac{1}{Z} = Ax + By + z \tag{2.48}$$

Avec ;

$$\begin{cases} A = -\dfrac{A_1}{D_1} \\ B = -\dfrac{B_1}{D_1} \\ C = -\dfrac{C_1}{D_1} \end{cases} \tag{2.49}$$

Et de h_2, on obtient l'équation de la droite $2D$, notée D, résultant de la projection dans le plan image de droite 3D :

$$ax + by + c = 0 \tag{2.50}$$

Avec ;

$$\begin{cases} a = A_2 \\ b = B_2 \\ c = C_2 \end{cases} \tag{2.51}$$

Comme le choix des paramètres (a, b, c) n'est pas minimal, on préfère passer aux coordonnées polaires (ρ, θ) [1].

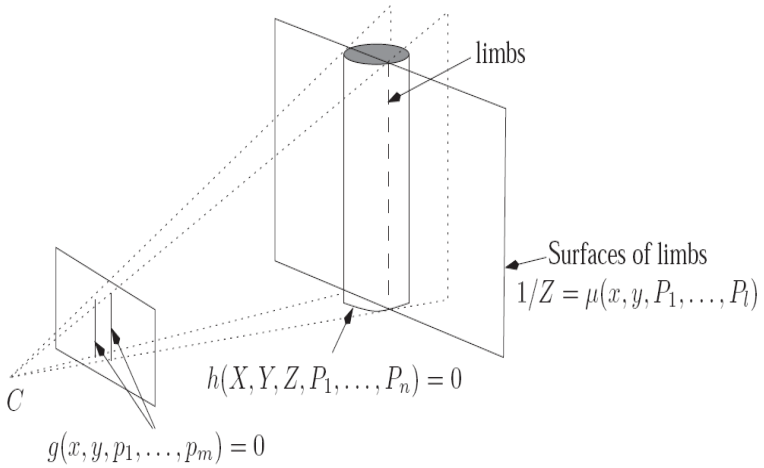

FIGURE 2.16: projection d'une primitive résultante de l'intersection d'un plan virtuel et d'un cylindre dans le plan image.

– • **Représentation polaire**

En représentation polaire, l'équation de la droite est donnée par :

$$xcos(\theta) + ysin(\theta) - \rho = 0 \tag{2.52}$$

Où ;

$$\begin{cases} \theta = arctan(\frac{b}{a}) \\ \rho = -\frac{c}{\sqrt{(a^2 + b^2)}} \end{cases} \tag{2.53}$$

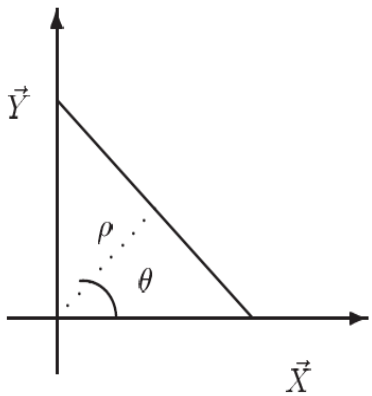

FIGURE 2.17: Representation polaire d'une ligne 2D.

En différentiant l'équation (2.52),qui correspond à l'hypothèse que l'image d'une

ligne droite reste une ligne droite quelque soit le mouvement de la camera on aura :

$$\dot{\rho} + (xsin(\theta) - ycos(\theta)) = \dot{x}cos(\theta) + \dot{y}sin(\theta) \tag{2.54}$$

En se basant sur l'équation (2.52), x est écrit en fonction de y si $cos(\theta) \neq 0$ (ou y en fonction de x si ce n'est pas le cas) l'équation (2.54) peut alors s'écrire en utilisant les résultats des équations :

$$\dot{\rho} + (\rho(tan(\theta))\dot{\theta} - y\frac{\dot{\theta}}{cos(\theta)}) = K_1 V + yK_2 V \tag{2.55}$$

Avec ;

$$\begin{cases} K_1 = [\lambda_1 cos(\theta)\lambda_1 sin(\theta) - \lambda_1 \rho sin(\theta)cos(\theta) - \dfrac{\rho^2}{cos(\theta)}] \\ K_2 = [\lambda_2 cos(\theta)\lambda_2 sin(\theta) - \lambda_2 \rho - \rho - \rho(tan(\theta)\dfrac{1}{cos(\theta)}] \end{cases} \tag{2.56}$$

où ;

$$\begin{cases} \lambda_1 = -\dfrac{A_\rho}{cos(\theta)} - C \\ \lambda_2 = Atan(\theta) - B \end{cases} \tag{2.57}$$

Immédiatement on aura :

$$\begin{cases} \dot{\rho} = (K_1 + \rho(sin(\theta))K_1)V \\ \dot{\theta} = -cos(\theta)k_2 V \end{cases} \tag{2.58}$$

D'où ;

$$\begin{cases} L_\rho = [\lambda_\rho cos(\theta)\ \lambda_\rho sin(\theta)\ -\lambda_\rho \rho\ (1+\rho^2)sin(\theta)\ -(1+\rho^2)cos(\theta)\ 0] \\ L_\theta = [\lambda_\theta cos(\theta)\ \lambda_\theta sin(\theta)\ -\lambda_\theta \rho\ -\rho cos(\theta)\ -\rho sin(\theta) - 1] \end{cases} \tag{2.59}$$

Avec [1] ;

$$\begin{cases} \lambda_\rho = -A_\rho cos(\theta) - B_\rho sin(\theta) - C \\ \lambda_\theta = -Asin(\theta) + Bcos(\theta) \end{cases} \tag{2.60}$$

Des résultats pour plus de primitives complexes (cercles, sphères, cylindres... etc.) sont donnés dans [32] rendant ainsi possible l'utilisation des informations visuelles 2D associées à ces primitives dans un asservissement visuel.

2.8.3 Régulation de la fonction de tâche

Réaliser une tâche en robotique par asservissement visuel requiert tout d'abord la sélection des informations visuelles appropriées puis, l'élaboration d'une loi de commande. La première phase se résume à définir la fonction de tâche avec des propriétés qui assurent que la tâche choisie soit réalisée, la deuxième consiste à réguler cette fonction de tâche[1].
Dans ce paragraphe, on traitera tout d'abord de la définition de la fonction de tâche qui nous permettra d'atteindre notre objectif puis, on donnera la démarche classique de l'élaboration de la loi de commande.

a. Fonction de tâche

Si on utilise un ensemble de k informations visuelles, la forme de la fonction de tâche est :

$$e(P(t)) = C(s(p(t)) - s^*) \tag{2.61}$$

$s(p(t))$: est la valeur courante des informations visuelles sélectionnées.
s^* : est la valeur que s doit atteindre pour que la fonction de tâche soit réalisée pour les méthodes d'obtention de s^*, se référer au paragraphe 2.6.
C : est une matrice de dimension $l * k$, avec l le nombre de degrés de liberté du robot, dite : *matrice des combinaisons*, définie de sorte que les composantes de e soient indépendantes et contrôlent les N degrés de liberté du bras manipulateur [1].

b. Synthèse de la loi de commande

Comme il a été cité plus haut, l'asservissement visuel 2D consiste schématiquement en la régulation de la fonction de tâche 2.61 :

Où C représente la matrice de combinaison déjà définie, l'introduction de cette matrice peut perturber la convergence des indices visuels, en introduisant des minimas locaux. En effet, la régulation de $e(t)$ vers 0 n'assure que la convergence de Cs vers Cs^*, ce qui ne garantie pas forcement la convergence de s vers s^*[6].

Avec le formalisme de fonction de tâche, la régulation à zéro de l'erreur visuelle correspond à un mouvement du capteur visuel vers sa position désirée. Une approche pour réaliser ce mouvement consiste à imposer une vitesse de commande assurant une décroissance exponentielle de cette erreur avec un gain λ :

$$\dot{e}(q) = -\lambda(e(q)) = -\lambda(C(s - s^*)), \lambda > 0 \tag{2.62}$$

Afin que cette loi de commande soit complètement définie, un choix de la matrice C s'impose.

Cette approche constitue la commande classique adoptée pour la boucle de vision, mais du moment que le but est d'amener les erreurs sur les indices visuelles dans le plan image vers zéro, d'autre lois de commande peuvent être envisagées : la commande par mode glissant, le régulateur PID, la commande floue... etc. la synthèse de ces lois de commande fera l'objet du chapitre suivant.

c. Calcul de la matrice d'interaction

La synthèse de la commande repose sur l'élaboration d'une méthode de calcul explicite de cette matrice souvent associée à des primitives géométrique simples, telles que des points, des droites, des cercles, des ellipses, ou encore des moments de l'image. Dans ce contexte, les lois de commande synthétisées dépendent de la nature des informations visuelles choisies. Par conséquent la tâche robotique à réaliser est elle aussi dépendante de l'objet observé par la présence ou non d'informations visuelles dont on est capable de calculer la matrice d'interaction associée [6].
La question est donc de savoir comment mettre à jour cette matrice dans la boucle d'asservissement. Deux cas de figure sont donc possibles, selon que $L(s,z)$ est calculée hors ligne ou en ligne [15] :

- $\widehat{L} = L(s,\widehat{z})$, c'est à dire que l'on calcule à chaque itération la valeur courante de la matrice d'interaction. Une estimation des paramètres $\widehat{z}$ doit alors être réalisée en ligne.
- $\widehat{L} = L(s^*,z^*)$, c'est-à-dire que la matrice choisie est constante et correspond à la configuration désirée. Une valeur de la profondeur à la position désirée même très approximative est alors nécessaire.

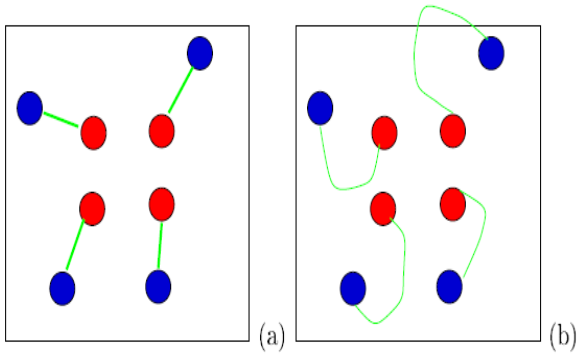

FIGURE 2.18: (a).trajectoire attendu dans l'image en choisissant$\widehat{L} = L(s,\widehat{z})$, (b). trajectoire possible dans l'image en choisissant $\widehat{L} = L(s*,z*)$ [7].

Ces deux méthodes présentent chacune des avantages mais aussi des inconvénients ; Le choix de $\widehat{L} = L(s,\widehat{z})$ contraint fortement la trajectoire que doivent suivre dans l'image les informations visuelles pour atteindre leur but. Par exemple

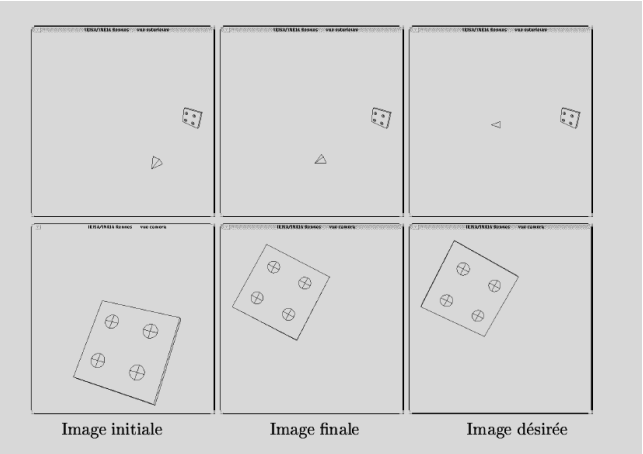

FIGURE 2.19: convergence vers le minimum local en utilisant $L = L(s, z)$[7].

si l'ont choisit des points, ce choix implique que ceux-ci doivent aller simultanément en ligne droite vers leur position désirée (voir Fig 2.18.a). Cependant ce comportement souhaitable dans l'image peut parfois entrainer des mouvements de la camera inadéquats (correspondants à une position de singularité) voir même impossibles à réaliser. La camera peut ainsi atteindre un minimum local de l'erreur dans l'image à partir duquel il est impossible de rejoindre le minimum global correspondant à une erreur nulle (voir Fig (2.19).

Le choix de $\widehat{L} = L(s^*, z^*)$ lui, ne contraint quasiment pas la trajectoire dans l'image (voir Fig 2.18.b). Le comportement découplé et la positivité de (L^+). L n'étant assuré que dans le voisinage de la position désirée. Du coup, et même si nous ne pouvant l'expliquer, il s'avère que la commande est moins sujette au problèmes des mouvements irréalisables et des minimas locaux, la trajectoire dans l'image calculée par la li de commande permettant d'éviter ces configurations défavorables (voir 2.20) ; par contre, il est tout à fait possible que l'objet sorte du champs de vue de la camera au cours du positionnement. (Pour y rentrer, mais plus tard, ensuite...).

Finalement, dans les deux cas, il est possible de rencontrer une singularité de la matrice d'interaction ou pour du moins, des positions ou son conditionnement est très faible, entrainant la encore soit une instabilité de la commande, soit un échec dans la convergence du système [7].

2.9 Conclusion

Dans ce chapitre, nous avons exposé un état de l'art sur l'asservissement visuel. Les généralités sur la vision en robotique nous ont permis d'aborder les principes généraux de l'utilisation de cette faculté dans ce domaine. Après quoi, les

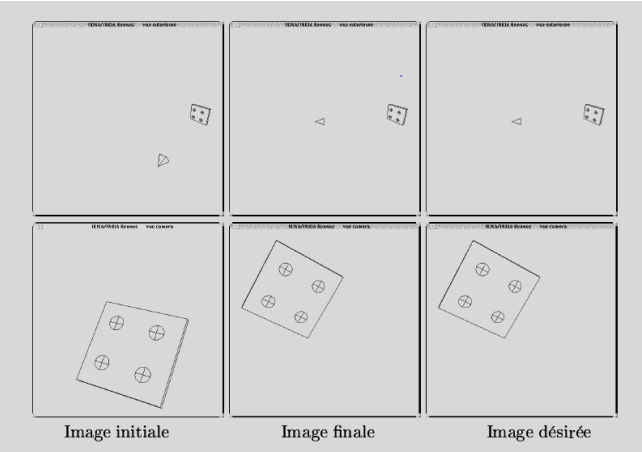

FIGURE 2.20: convergence vers le minimum global en utilisant $L = L(s^*, z^*)$[7].

différentes techniques utilisées en asservissement visuel ont été abordées. Par la suite, nous avons détaillé le principe de la technique d'asservissement $2D$ qui est la technique adoptée pour ce travail.

Chapitre 3

Synthèse de la commande référencée vision pour le robot manipulateur

3.1 Introduction

En se basant sur le principe de fonction de tâche, plusieurs applications par asservissement visuel peuvent être réalisées sur un bras manipulateur. Dans le cadre de notre étude, on a choisit deux tâches à savoir, le positionnement par rapport à quatre points et le suivi de ligne.

Dans ce chapitre on présentera en premier lieu la configuration de la vision sur le robot puis, on exposera les différents résultats obtenus en simulation pour chacune des tâches, et ce pour trois lois de commande différentes : la commande classique, le régulateur PI et la commande par mode glissant. Une comparaison des performances de chaque loi de commande est donnée à la fin de ce chapitre.

3.2 Application de l'asservissement 2D au robot SCARA

Pour l'application de la vision, on a travaillé sur une structure simplifiée à deux degrés de liberté à liaisons rotoides.
Pour la configuration camera/robot, on a adopté la configuration eye-in-hand. La camera a était rigidement liée à l'extrémité de la dernière membrure du bras. Avec la configuration adoptée, la camera ne dispose d'aucun degré de liberté.
La *Robotic Toolbox* de *Matlab* nous a permis de faire une représentation graphique du bras, ainsi que de simuler la configuration de la camera sur le bras, cette représentation est donnée par la figure 3.1. Dans ce schéma, on peut très bien distinguer les liaisons du bras (en bleu), et la schématisation de la configuration de la camera (en rouge).

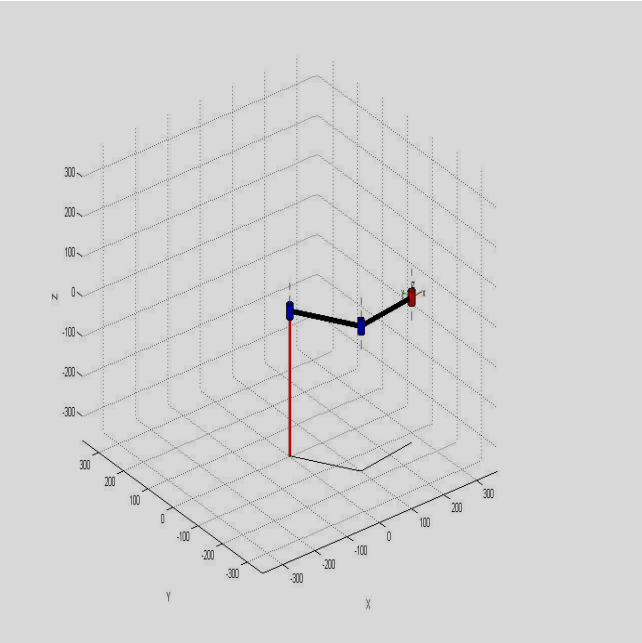

FIGURE 3.1: Représentation graphique du robot SCARA et de la configuration de la camera.

3.3 Matrice d'interaction

Le calcul de cette matrice a était détaillé dans le chapitre II. Cependant,ces calculs doivent être adaptés à notre application à savoir au robot SCARA à deux degrés de liberté.

Nous avons toujours par projection perspective :

$$P_i = \begin{pmatrix} x = f\dfrac{X}{Z} \\ y = \dfrac{Y}{Z} \end{pmatrix} \tag{3.1}$$

X,Y et Z sont les coordonnées d'un point M quelconque de l'objet dans le repère de la camera, et x et y sont les coordonnées métriques de sa projection perspective dans le plan image.

En différentiant cette fois ci l'équation (3.1) on obtient :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \frac{f}{Z} & 0 & 0 \\ 0 & \frac{f}{Z} & 0 \end{pmatrix} \dot{M} \tag{3.2}$$

Ceci est obtenu du fait que pour un robot SCARA la profondeur z est fixe pour des primitives prise dans un plan parallèle a la scène.

En exprimant la vitesse $\dot{M}$ en fonction du torseur cinématique V ,on aura

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = L_{xy} V \tag{3.3}$$

Avec ;

$$\begin{pmatrix} -\frac{f}{Z} & 0 & 0 & 0 & -f & y \\ 0 & -\frac{f}{Z} & 0 & f & 0 & -x \end{pmatrix} \tag{3.4}$$

Finalement, comme déjà démontré, la jacobienne de l'image s'obtient par :

$$L_s = \begin{pmatrix} k_u & 0 \\ 0 & k_v \end{pmatrix} \tag{3.5}$$

Par souci de compatibilité des dimensions des matrices dans l'expression **2.30**, on considère le jacobien de l'image sous sa forme réduite, notée L_r . Cette matrice sera prise de dimension $2x2$, elle ne comptera que les translations suivant x et suivant y comme suit :

$$L_r = \begin{pmatrix} -\frac{f}{Z} & 0 \\ 0 & -\frac{f}{Z} \end{pmatrix} \tag{3.6}$$

3.4 Calibration de la camera

3.4.1 Procédure de calibration

Comme il a été mentionné dans le chapitre II,la calibration ou étalonnage de la camera est une étape nécessaire afin d'identifier ses paramètres intrinsèques. Afin d'accorder moins de temps au problème de calibration qui, à lui seul, représente un champ de recherche à part entière, nous avons procédé à l'étalonnage au moyen de l'application *Camera calibration toolbox* sous Matlab disponible sur internet.

La calibration d'une camera exige l'utilisation d'une mire spéciale dite *mire de calibration* dont on prendra plusieurs images à l'aide de la camera à calibrer, ces images fourniront la base de données nécessaires à la toolbox de *Matlab* pour calculer les paramètres intrinsèques de la camera.

La mire utilisée doit satisfaire quelques exigences :
- Elle doit contenir des formes faciles à reconnaitre et dont la position est très bien connue selon le référentiel environnement ;
-Elle doit comporter des éléments non coplanaires [37].
Plusieurs variantes de mires se rencontrent dans les applications, mais la mire le plus simple reste un simple échiquier, c'est d'ailleurs ce type de mire qu'on a

utilisé pour l'étalonnage de notre camera.

La procédure d'étalonnage commence en prenant au moins 20 images de la mir posée sur le plan horizental.les images doivent être prises depuis différents angles, sans perdre de vue l'ensemble de la mire.la figure 3.2 montre dix images parmi celles utilisées pour l'étalonnage.

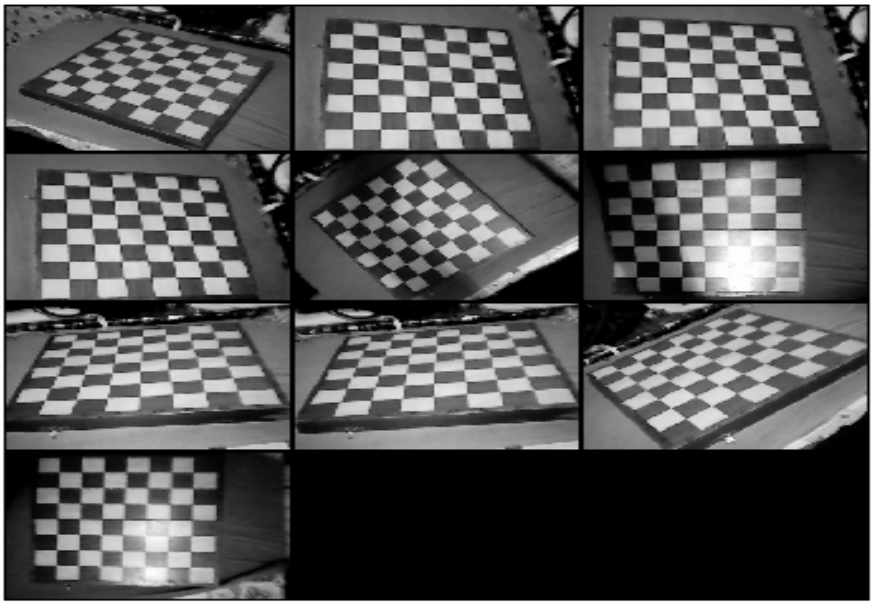

FIGURE 3.2: images parmi les 20 utilisées pour l'étalonnage de la camera.

La seconde étape consiste à utiliser un détecteur de primitives pour trouver le gradient du contraste entre les carrés de l'échiquier. La Toolbox de *Matlab* offre une interface visuelle élégante pour détecter et marquer les coins comme montré par la figure (3.4).un repère est rattaché à l'un des coins de l'échiquier avec l'orientation comme montrée par la figure (3.3). Les coordonnées de chaque primitive par rapport au repère ainsi déterminés sont alors stockées. Un système linaire d'équations à 8 inconnus doit être résolu, donnant la position de chaque point dans le plan image et la position réelle de chaque coin correspondant dans l'échiquier ;

La précision de la procédure d'étalonnage dépend de la précision des mesures des dimensions de l'échiquier. Une règle à suivre est que la mire de calibration doit être prise avec des tolérances d'un ordre, une ou deux fois inférieur à la précision désirée pour la calibration [38].

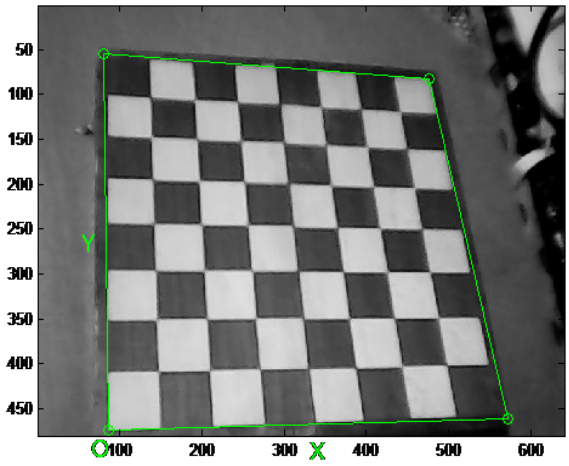

FIGURE 3.3: Association des repères X et Y à l'image.

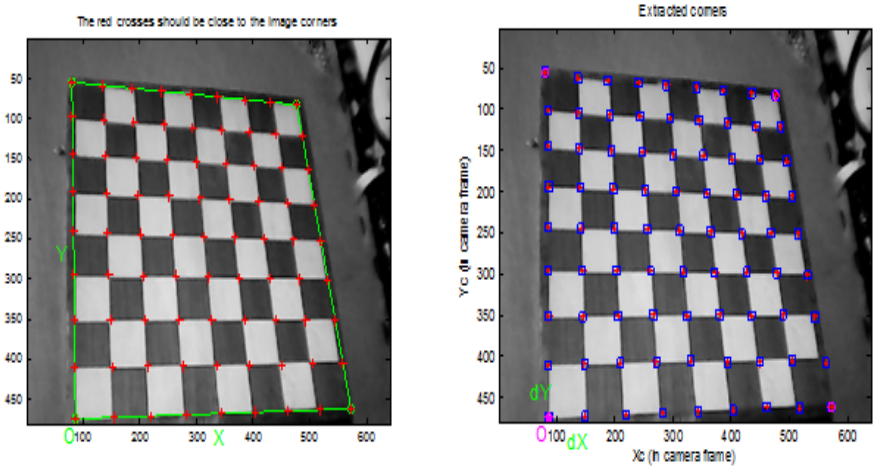

FIGURE 3.4: Détection et marquage des coins.

3.4.2 Résultats de la calibration

Lors de la calibration, une suite de 30 images a était utilisée. Pour chaque position de prise, les paramètres extrinsèques de l'outil terminal sont enregistrés. Les différentes poses peuvent être mise en évidence grâce à leur signification réelle à savoir en terme de position de la camera par rapport à la mire comme dans la figure 3.5 ; mais aussi, en considérant la camera comme fixe, c'est alors la mire qui changera de position par rotation devant la camera comme montré dans la figure (3.6).

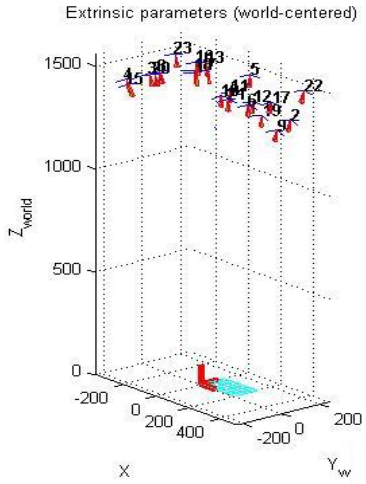

FIGURE 3.5: Différentes poses en terme de position de la camera par rapport à la mire.

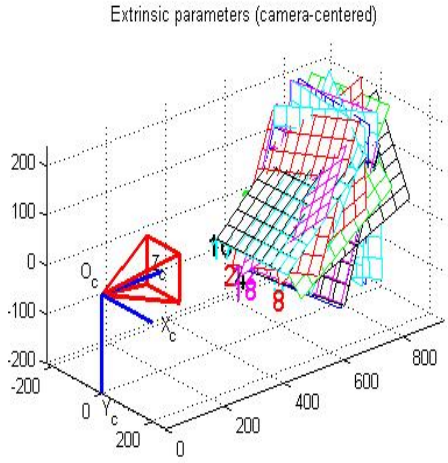

FIGURE 3.6: Différentes poses en terme de position de la mire par rapport à la camera.

Après la première exécution du programme de calibration, les erreurs sur les primitives de chaque image peuvent être estimées (voir figure (3.7),(3.8)) alors, les images dont les primitives présentent une importante erreur peuvent être supprimées et le programme de calibration relancé, ce qui permet d'affiner les résultats obtenus et d'aboutir à la précision voulu pour la calibration. Les résultats obtenus sont résumés dans le tableau (3.1).

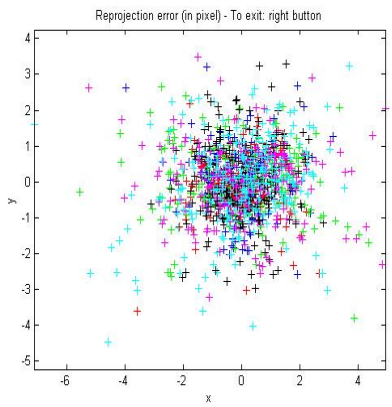

FIGURE 3.7: Projection des erreurs en pixel :$1^{ére}$ exécution du programme.

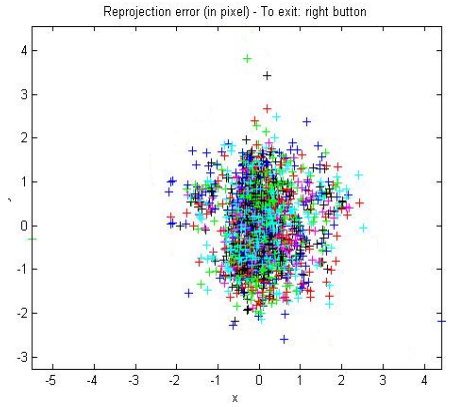

FIGURE 3.8: Projection des erreurs en pixel :plusieurs opérations sur les images.

3.5 Choix des informations visuelles

L'utilisation d'un capteur de vision pour commander le robot *SCARA* requiert un choix sur les primitives visuelles à utiliser (points, droite, ellipses, cercles,...etc.). En effet, ce choix dépend de la tâche à réaliser (suivi de trajectoire, poursuite de cibles, positionnement... etc.), et devra correspondre à une seule configuration du robot dans l'espace 3D, par exemple un choix de primitives visuelles de

Paramètre	Valeur
Plan image (donnée constructeur)	$640x480pixels$
Distance focale (donnée constructeur)	$3.85mm$
Facteur d'échelle (α_u α_v)	[965.23 958.42) ±(7.86786 7.81066)
Centre optique (u_0 v_0)	(319.50 239.50)±(0.00 0.00)
Inclinaison/angle entre les axes pixeliques	[0.00] ±[0.00] , $angledesaxesenpixel = 90.00 \pm 0.00 degrees$
Distorsion	(-0.17 0.32 -0.00 0.01 0.00) (0.05 0.43 0.00 0.00 0.00)
Erreur pixel	$\begin{pmatrix} 0.96967 & 0.81925 \end{pmatrix}$

TABLE 3.1: Paramètres intrinsèques de la camera estimés avec la *Camera Calibration Toolbox* de Matlab.

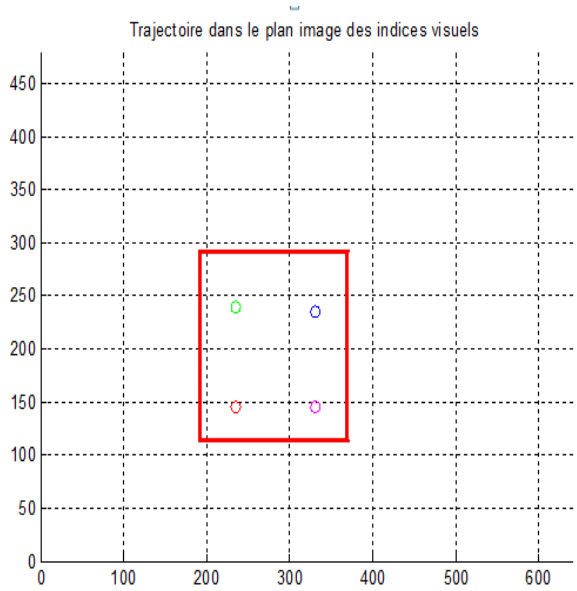

FIGURE 3.9: Configuration désirée des informations visuelles dans le plan image pour le cas d'un positionnement par rapport à quatre points.

type points, nous contraint à choisir un nombre de points supérieur à trois pour satisfaire cette condition.

3.5.1 Cas d'un positionnement par rapport à quatre points

Notre premier objectif est de réaliser le positionnement de l'outil terminal par rapport à quatre points , donc la consigne s^* sera donnée par :

$$s^* = \begin{pmatrix} u_1 & v_1 & u_2 & v_2 & u_3 & v_3 & u_4 & v_4 \end{pmatrix}^T \tag{3.7}$$

Pour nos simulations on a choisit le vecteur s^* comme suit :

$$s^* = \begin{pmatrix} 145 & 235 & 135 & 331 & 235 & 331 & 240 & 235 \end{pmatrix}^T \tag{3.8}$$

3.5.2 Cas d'un suivi de ligne

Le deuxième objectif consiste en un suivi de ligne. Pour ce cas le vecteur d'informations visuelles S sera donné par :

$$s = \begin{pmatrix} \rho \\ \theta \end{pmatrix} \tag{3.9}$$

La droite de référence sera alors notée comme une configuration s^* à atteindre telle que :

$$s^* = \begin{pmatrix} \rho^* \\ \theta^* \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.10}$$

3.6 Commande et stabilité

La loi de commande d'un bras manipulateur peut être synthétisée à partir du modèle d'état déduit du modèle dynamique. De manière générale, cette loi de commande doit :

- Tenir compte du caractère non linéaire du système,
- Assurer la stabilité,
- Permettre le suivi des variations de consigne.

Pour le cas de la commande par retour visuel, les termes q ,$\dot{q}$ et $\ddot{q}$ doivent être exprimés en fonction des informations visuelles S, en d'autre termes, la fonction de passage de l'espace des informations visuel à l'espace articulaire f telle que $q = f(s)$ doit être connue.Or, si l'obtention de cette fonction est possible dans le cas d'un objet fixe, ce n'est plus le cas lorsqu'on considère un objet mobile à cause du fait qu'une configuration dans l'image ne correspond plus à une seule configuration articulaire.

La commande du robot par une loi qui ne dépend que des informations visuelles souffre de plusieurs insuffisances, d'une part, cette loi doit assurer la stabilité globale et les performances du système et d'autre part, elle nécessite un calcul de la fonction f à chaque itération, mis à part le fait qu'elle n'est applicable que pour le cas d'un objet fixe. Pour remédier à toutes ces insuffisances, la commande par retour visuel se fait par régulation en cascade. La boucle externe représente la boucle de vision qui fournit les consignes articulaires, et la boucle interne

représente la boucle de commande du robot, qui régule les vitesses articulaire à la valeur donnée en consigne par la boucle de vision. Le schéma complet d'une boucle d'asservissement visuel est donné par la figure (3.10).

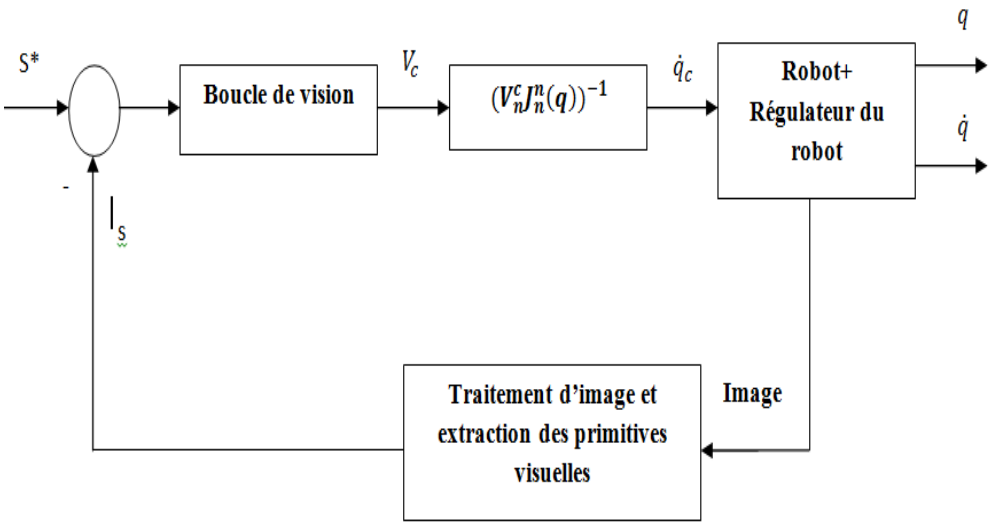

FIGURE 3.10: Schéma global de la boucle d'asservissement visuel d'un bras manipulateur.

Vu que la structure en cascade impose à la boucle du robot d'être plus rapide que celle de la vision, un bon choix de la commande du robot s'impose.

L'interface dont nous disposons utilise une régulation PID, la synthèse analytique de cette loi de commande est détaillée dans l'annexe B. Il s'agit donc de choisir les paramètres Kp, Ki et Kd qui assurent la stabilité et les performances souhaitées.

Le choix de la commande de la boucle de vision doit quant à lui, être fait de façon à assurer la convergence de la fonction de tâche tout en veillant à ce que les consignes générées ne contraignent pas le robot à des couples articulaires importants. Dans ce travail, trois lois de commande ont étaient appliquées à la boucle de vision, la comparaison des résultats de simulation de chaque loi est donnée à la fin de ce chapitre.

3.7 Synthèse des lois de commande pour la boucle de la vision

Afin d'asservir un système par retour visuel, trois architectures de commande sont possibles :

(a) **Commande séquentielle :**
Utilisée lorsque le dispositif de traitement d'image ne peut pas fournir des images à une cadence supérieure à l Hertz. Elle fonctionne selon le mode *look then move*. La Séquence consiste à acquérir un ensemble d'images puis de les traiter pour générer une trajectoire en position de l'organe terminal du robot qui sera envoyée au contrôleur bas niveau du robot.

(b) **Commande cinématique :**
La commande cinématique est un asservissement en boucle fermée qui consiste à envoyer périodiquement des consignes de vitesses articulaires au contrôleur bas niveau du robot. La cadence du rafraichissement des consignes est imposée par la fréquence d'acquisition des images et la durée des traitements nécessaires à l'extraction de l'information visuelle et du calcul de la loi de commande.

(c) **Commande dynamique :**
La commande dynamique tient compte d'une modélisation plus réaliste du comportement du robot. Dans ce cas, le système de vision commande directement les variateurs des actionneurs du robot en se substituant au contrôleur bas niveau du robot. Cette architecture permet de tirer pleinement partie des possibilités dynamiques offertes par le robot pour réaliser des asservissements visuels rapides [32].

Pour notre travail, nous avons retenu la commande cinématique, une commande qui est la plus largement utilisée et la plus traitée .Dans la littérature, plusieurs types de lois de commande ont été suggérées ;non linéaires [39] [40] ; LQR, LQR optimale [41] [42] et même H robuste [43].

Dans ce qui suit on va développer un certain nombre de loi de commande pour réguler la fonction de tâche. On commencera par la loi de commande la plus classique puis, on abordera des lois de commande plus complexes.

3.7.1 Commande classique

Comme déjà mentionné ; une approche pour la régulation de la fonction de tâche consiste à imposer une vitesse de commande assurant une décroissance exponentielle de l'erreur avec un gain λ :

$$\dot{e} = -\lambda(e(t)) \tag{3.11}$$

λ étant un réel positif non nul. Ceci correspond à une décroissance exponentielle de la fonction de tâche :

$$\dot{e} = -\lambda(e(t)) \tag{3.12}$$

En utilisant l'expression de la fonction de tâche

$$\dot{e} = C(s - s^*) \tag{3.13}$$

En injectant dans cette équation l'expression **(2.32)** et en considérant s^* fixe au cours du temps on trouve :

$$\dot{e} = C(L_s V + \frac{\partial s}{\partial t}) \tag{3.14}$$

Pour obtenir le comportement de décroissance exponentielle décrit par l'équation 3.12, on écrit finalement :

$$\dot{e} = -\lambda(e(t)) = CL_s V + \frac{\partial e}{\partial t} \tag{3.15}$$

On a alors l'expression du torseur cinématique V_c utilisée comme entrée de commande du système :

$$V_c = (CL_s)^{-1}(-\lambda(e(t)) - \frac{\partial e}{\partial t}) \tag{3.16}$$

En pratique, l'interaction entre le capteur visuel et son environnement ainsi que les variations de la fonction de tâche $\frac{\partial e}{\partial t}$ dues au mouvement propre de la cible visuelle ne sont pas connues parfaitement et sont des paramètres estimés. Dans le cas simplifie ou la cible visuelle est fixe, la loi de commande considérée devient :

$$V_c = -\lambda(CL_s)^{-1}(e(t)) \tag{3.17}$$

Où le choix de la matrice C dépend du nombre d'informations visuelles considéré :

– si $k = l$, avec l le nombre de degrés de liberté du manipulateur, la matrice C peut être choisie égale à la matrice identité ; ce qui permet de générer un comportement de décroissance exponentielle pour tous les éléments du vecteur d'informations S visuelles avec la loi de commande :

$$V_c = -\lambda(L_s)^{-1}(s - s^*)^* \tag{3.18}$$

– si$k > l$, la matrice C est alors choisie de dimension lxk et de rang l, la matrice de combinaison est généralement choisit comme étant la pseudo inverse d'une estimation de la matrice d'interaction à la position désirée [44] :$C = L^+_{s=s^*}$. Une autre solution consiste à définir cette matrice comme la matrice identité de dimension k et à remplacer dans l'équation 3.18 l'inverse de la matrice d'interaction par sa pseudo inverse définie par :

$$L_s^+ = (L_s^T L_s)^{-1} L_s^T \tag{3.19}$$

Comme démontré dans [45], avec $k > l$, la loi de commande basée image (3.19) est localement asymptotiquement stable pour une estimation correcte de la matrice d'interaction L_s (c'est-à-dire lorsque $L_s^+ L_s > 0$). En revanche, sa stabilité globale n'est pas garantie et il peut exister des minimas locaux en dehors du voisinage de la configuration désirée.

3.7.2 Régulateur PI

a. Commande Proportionnelle

Le régulateur à action proportionnelle,ou régulateur P, a une action simple et naturelle,puisqu'il construit une commande $u(t)$ proportionnelle à l'erreur $e(t)$[46].

$$U(t) = K_p e(t) \tag{3.20}$$

b. Commande proportionnelle intégrale

Très répandu, il permet d'obtenir une erreur nulle grâce à un intégrateur, ainsi qu'un temps de réponse réglable, en donnant de plus à la réponse l'allure d'une évolution exponentielle[47].

L'action $u(t)$ est proportionnelle à l'erreur $e(t)$ et à l'intégrale de l'erreur :

$$u(t) = k_p e(t) + k_i \int [e(t)]\, dm \tag{3.21}$$

3.7.3 Commande par mode glissant

Le régime glissant intervient dans la définition d'une classe très importante de systèmes de commande qui sont les systèmes à structure variable.

Les systèmes à structure variable sont définis comme étant des systèmes caractérisés par une discontinuité dans leurs modèles, une discontinuité qui est due soit à une action volontaire de l'opérateur par la commutation automatique d'un ou plusieurs switchers, ou à des changements de certains paramètres de ces systèmes.

Dans cette partie, nous allons synthétiser un contrôleur par asservissement visuel 2D, basé sur la théorie des modes glissants. Ce contrôleur doit, comme pour le cas classique, garantir que les indices visuels atteignent la configuration désirée dans le plan image. L'objectif est de réaliser un positionnement référencée vision plus robuste par rapport aux erreurs de modélisation, et aux bruits des mesures et du traitement d'image. Pour cela nous allons nous baser sur le principe de fonction de tâche déjà exposé dans le chapitre précédant, ainsi que sur les principes des modes glissants dont nous allons donner un bref rappel au début de ce paragraphe.

(a) **Principe de la Commande par Modes Glissants**

L'objectif du mode glissant est de contraindre le système à atteindre une surface dite *surface de glissement* puis d'y rester à l'aide d'une commande discontinue [48]. Cette surface est définit par :

$$\sigma(x) = 0 \qquad \text{pour} \qquad u = u_{eq}$$

Tel que : u_{eq} est la commande que l'on veut imposer au système.
Pour la synthèse d'une loi de commande par mode de glissement, deux étapes sont nécessaires :

- Choix de la surface de glissement adéquate : ce choix concerne le nombre et la forme des surfaces. Le nombre des surfaces est donné par la dimension du vecteur de commande U alors que la forme des surface est déterminée en fonction des applications et des objectifs visés.
- Synthèse d'une commande discontinue de façon que la trajectoire d'état atteigne la surface de glissement puis y reste, lorsque la commande est sur la surface de glissement $u = u_{eq}$ et on est en mode continu [48].

• **Choix de la surface de glissement** Soit le système non linéaire donné par :

$$\dot{x} = f(x,t) + g(x,t)u \tag{3.22}$$

Soit $S(x,t)$: une fonction dont l'annulation permet de satisfaire l'objectif de commande, alors l'ensemble :

$$\sigma(x) = \{x \in X, \sigma(x,t) = 0\} \tag{3.23}$$

Représente un sous ensemble de X appelé *surface de glissement*.
L'objectif de commande est d'annuler l'erreur de réglage, alors la surface de glissement qui satisfait cette condition est donnée par :

$$\sigma(x) = \left(\lambda + \frac{d}{dt}\right)^{r-1} e(t) \tag{3.24}$$

Tel que :

– X : Le vecteur d'état ;
– $e(t)$: L'erreur de réglage ;
– λ : Une constante positif de glissement ;
– r : Le degré relatif du système.

La seule solution pour $\sigma(x) = 0$ est $e(t) = 0$ ce qui vérifie l'objectif de commande [48].
Maintenant que la surface de glissement est définie, il faut synthétiser une commande qui permettra d'amener la trajectoire d'état vers cette surface et ensuite la maintenir au voisinage de cette dernière quel que soit les perturbations et les incertitudes.

Avant de procéder à la synthèse de la loi de commande, il faut d'abord assurer son existence, en effet pour que la convergence vers la surface de glissement ait lieu, il faut satisfaire la condition suivante : $\dot{\sigma}\sigma < 0$, appelée : ***condition d'attractivité***.
La commande à synthétiser est la somme de deux commandes u_{eq} et u_n. u_{eq} est la commande équivalente, elle correspond aux valeurs que prend la commande une fois sur la surface de glissement, c'est la commande proposée par *Filippov* et elle est obtenue en considérant :

$$\dot{\sigma} = 0$$

Pour $\sigma(x,t) = 0$, on a :

$$\dot{x} = f(x,t) + g(x,t)u_{eq} \tag{3.25}$$

Alors :

$$\dot{\sigma} = \frac{\partial \sigma}{\partial x}\left(f(x,t) + g(x,t)u_{eq}\right) = 0 \tag{3.26}$$

L'unique solution de cette équation est :

$$u_{eq} = -\left(\frac{\partial \sigma}{\partial x}g(x,t)\right)^{-1}\frac{\partial \sigma}{\partial x}f(x,t) \tag{3.27}$$

u_n : si u_{eq} sert à maintenir la variable à contrôler sur la surface de glissement, u_n sert à forcer le système a atteindre la surface de glissement, elle est donnée par la condition d'attractivité, son expression est :

$$u_n = -K\,sign\left(\sigma(x,t)\right) \tag{3.28}$$

(b) **Phénomène du *chattering* ou du broutement**

- **Problème du *Chattering***
  La condition d'attractivité stipule que :

$$\sigma\dot{\sigma} < 0$$

Donc :

$$\dot{\sigma} = -K sign\,(\sigma)$$

Ceci, combiné aux erreurs sur le zéro conduit à de fortes oscillations de la trajectoire d'état autour de la surface de glissement, ce qui se traduit pratiquement par des vibrations, c'est le phénomène de broutement *(Chattering)*.

Ce phénomène est très néfaste pour trois raisons :

- Les oscillations peuvent exciter les modes à hautes fréquences ;
- Importantes pertes énergétiques ;
- Importante sollicitation des actionneurs.

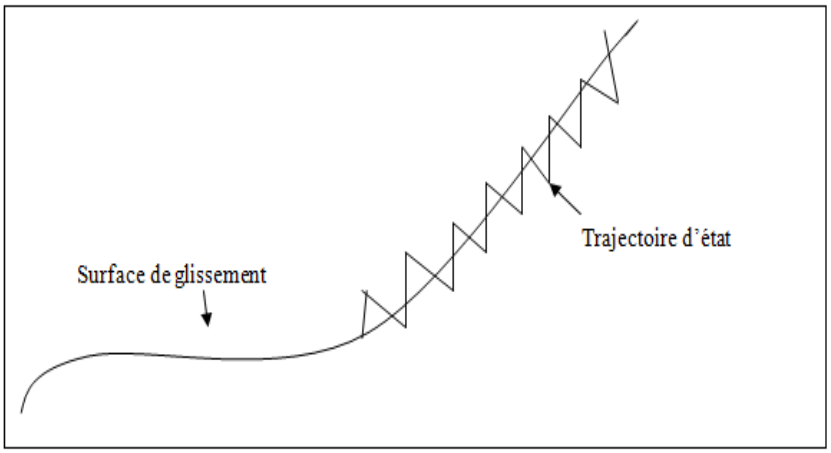

FIGURE 3.11: Phénomène de broutement

(c) **Synthèse de la loi de commande par mode glissant**

Le choix évident pour la surface de glissement σ est de la prendre égale à la fonction de tâche comme il a été procédé dans [15] :

$$\sigma(q,t) = C(s(q,t) - s^*) \tag{3.29}$$

Où C est la matrice de combinaison utilisée dans le cas de la commande classique.Sur cette surface, les objectifs de commande sont atteints. En effet si $\sigma = 0$ alors $s = s^*$.

Avant toute chose, il est nécessaire de déterminer l'expression de la dérivée de la surface de glissement :

$$\dot{\sigma} = C\dot{s} = CL_s V_n^C J_n^n(q)\dot{q} + C\frac{\partial s}{\partial t} \tag{3.30}$$

La commande équivalente est calculée à partir de la condition d'existence du régime glissant à savoir ($\sigma = 0$) et ($\dot{\sigma} = 0$) ceci donne :

$$\dot{q}_e = -(CL_s V_n^C J_n^n(q))^{-1} C\frac{\partial s}{\partial t} \tag{3.31}$$

Mais comme le calcul de la matrice d'interaction est basé sur les mesures des indices visuels s et de la profondeur z, cette dernière est le plus souvent déduite d'un estimateur, faute de moyen spécifique de mesure. A partir de là, on obtient une estimation de la matrice d'interaction réduite, notée L_s .De plus, le terme $\frac{\partial s}{\partial t}$ quantifie théoriquement le mouvement propre de l'objet par rapport à la caméra. Si l'objet est complètement immobile, alors il est mis à zéro. Nous considérons tout au long de cette étude uniquement une estimation de ce terme, que nous notons $\frac{\partial s}{\partial t}$. A partir de ces hypothèses, l'expression du contrôle équivalent devient :

$$\dot{q}_e = -(CL_s V_n^C J_n^n(q))^{-1} C\frac{\partial s}{\partial t} \tag{3.32}$$

le terme discontinu est exprimé par :

$$q = -k_s sign(\sigma) \tag{3.33}$$

Où le gain k_s est dans ce cas une matrice diagonale $2x2$ Enfin, l'expression finale de la loi de commande par modes glissants référencée vision peut être exprimée comme suit :

$$\dot{q}_s = -(CL_s V_n^C J_n^n(q))^{-1} C\frac{\partial s}{\partial t} - k_s sign(\sigma) \tag{3.34}$$

Pour notre cas, nous avons considéré un objet fixe ce qui veut dire que le terme $\frac{\partial s}{\partial t}$ sera nul. La loi de commande utilisée est alors réduite à l'expression suivante :

$$\dot{q}_s = -k_s sign(\sigma) \tag{3.35}$$

On rappelle que $\dot{q}_{s)}$ est un vecteur de dimension deux. Contrairement au cas de la commande classique où le choix des gains se fait arbitrairement par la technique dite essai-erreur, les gains du contrôleur par régime glissants (élément de la diagonale de la matrice k_s) doivent garantir la condition de glissement ($\sigma\dot{\sigma} < 0$) afin d'assurer la stabilité globale du système et la convergence de la loi de commande.

- Calcul du gain du contrôleur mode glissant

L'étude de la stabilité se fait par le choix d'une fonction de *Lyapunov* définie positive :

$$V = \frac{1}{2}\sigma^T\sigma = \frac{1}{2}\sum \sigma_i^2 \tag{3.36}$$

Où σ_i :représente la i^{eme} composante de σ
Cette fonction étant toujours positive,le régulateur a mode glissant est asymptotiquement stable si on assure la négativité absolue de la dérivée de la fonction de *Lyapunov*,cette dérivée s'écrit :

$$\dot{V} = \sum \sigma_i \dot{\sigma_i} \tag{3.37}$$

Afin de garantir la négativité de cette fonction dérivée et avoir ainsi une convergence en temps fini du système on doit assurer la η attractivité en vérifiant l'inégalité suivante :

$$\sigma_i \dot{\sigma_i} \leq \eta * \mid \sigma_i \mid \tag{3.38}$$

Où σ_i est une constante strictement positive, servant à régler la vitesse de convergence du système

Dans cette inégalité , le seul inconnu est le gain K_s.Les valeurs des éléments diagonaux de cette matrice peuvent ainsi être calculés à partir des deux systèmes d'inégalité définies à partir de (3.38)

3.8 Simulations de la boucle de vision

les différentes lois de commande synthétisées sont testées en simulation sous Matlab pour ce test , nous avons pris comme paramètres intrinsèques de la camera,les résultats de la calibration et comme position initiale du bras la configuration initiale $q1 = q2 = 0.8rad$.

3.8.1 Cas d'un positionnement par rapport à 4 points

- **Commande classique**
  La simulation de la commande avec un gain $\lambda = 2$ a donnée les résultats présentés dans les figure (3.12),(3.13) respectivement pour les deux fonctions de tâche et pour l'évolution des vitesses articulaires.

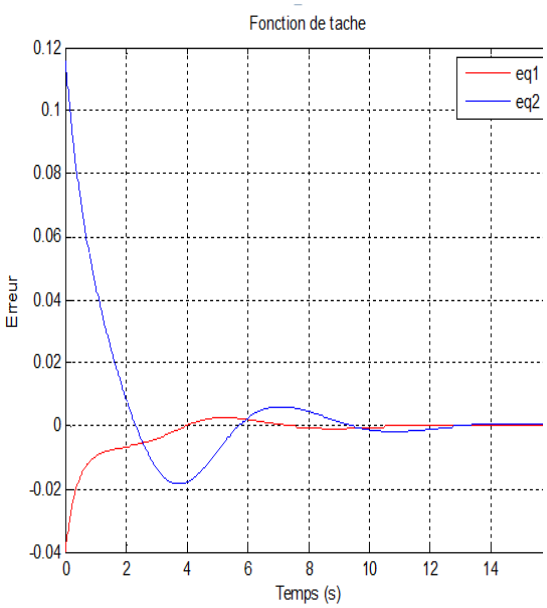

FIGURE 3.12: Les fonctions de tâche :commande classique

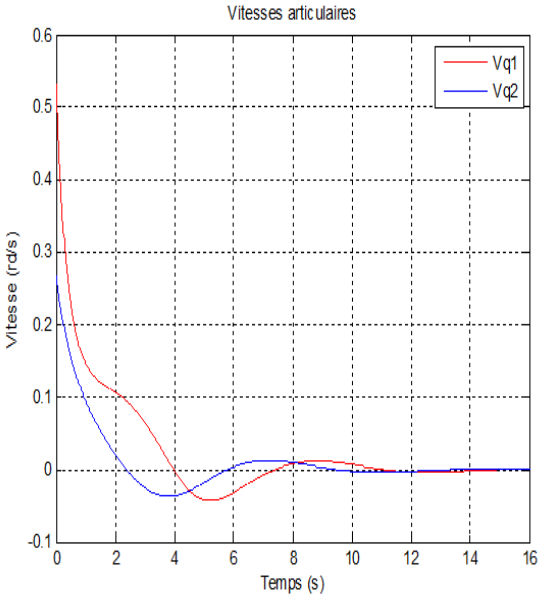

FIGURE 3.13: Les vitesses articulaires :commande classique.

L'évolution des indices visuelles est donnée par la figure (3.14),avec la position initiale est schématisée en bleu et la position désirée en rouge.

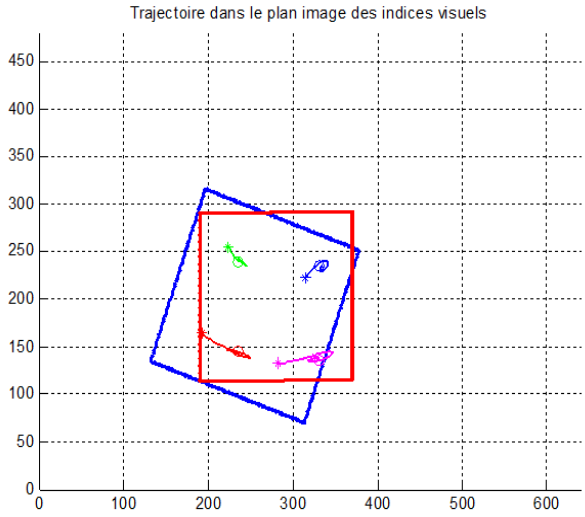

FIGURE 3.14: Trajectoire des indices visuelles dans le plan image :Commande classique.

- Interprétation

D'après les résultats précédents,on constate que l'objectif de commande a été atteint vu que la fonction de tâche converge vers zero ainsi que les vitesse articulaires ,cependant on remarque que l'évolution des indices visuelles de leurs positions initiales à leurs positions désirés ne se fait pas d'une manière optimale.

- **Régulateur PI**

Pour le régulateur PI ,les résultats de simulation sont donnés par les figures(3.15),(3.16) ,(3.17).

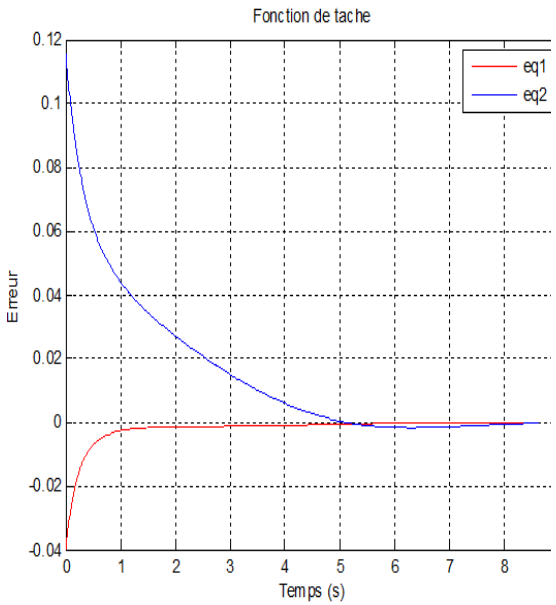

FIGURE 3.15: Les fonctions de tâche :Réglage PI.

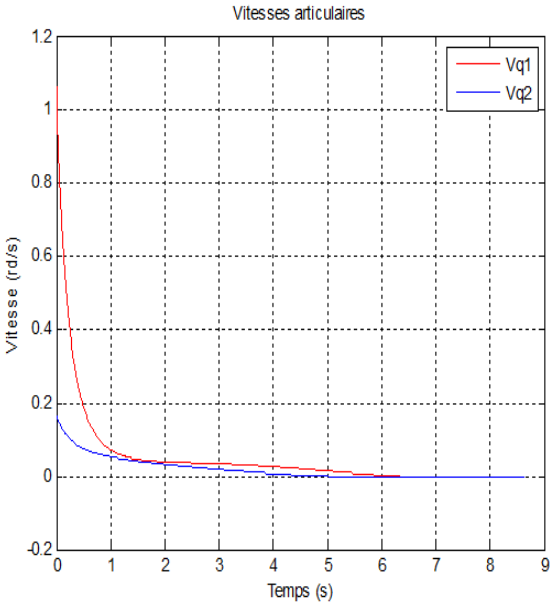

FIGURE 3.16: Les vitesses articulaires :Réglage PI.

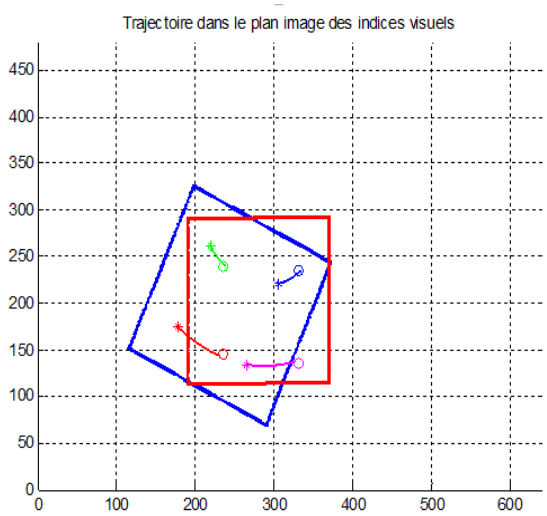

FIGURE 3.17: Trajectoire des indices visuelles dans le plan image :Réglage PI.

- Interprétation

Pour cette loi de commande ,la convergence vers zéros de la fonction de tâche (3.15)ainsi que des vitesses articulaires (3.16) se fait en $6s$,les courbes obtenues sont lisses ce qui se traduit par des trajectoires optimales des indices visuels.la figure (3.17) montre les convergences en lignes droites vers les positions désirées.

- **Commande par mode glissant**

La simulation de la commande par mode de glissement , en choisissant comme surface de glissement la fonction de tâche a donné les courbes (3.18) pour la fonction de tâche et (3.19) pour les vitesse articulaires,tandis que la figure (3.20) donne l'évolution des indices visuelles.

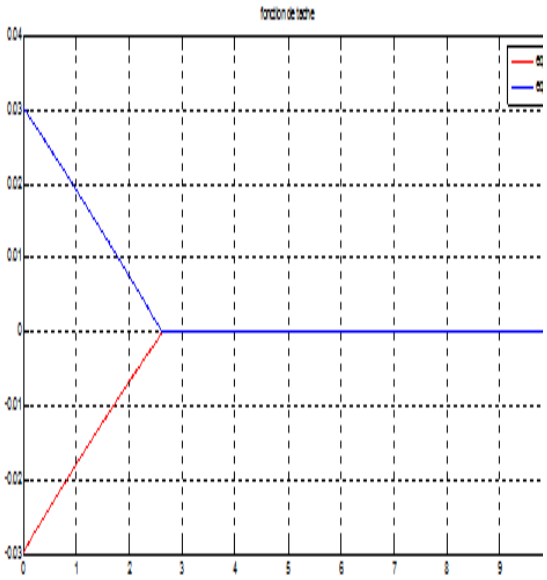

FIGURE 3.18: Les fonctions de tâche :commande par mode glissant.

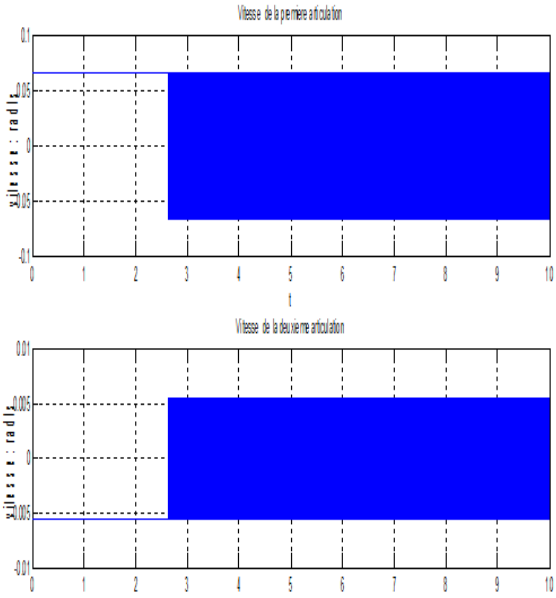

FIGURE 3.19: Les vitesses articulaires :commande par mode glissant.

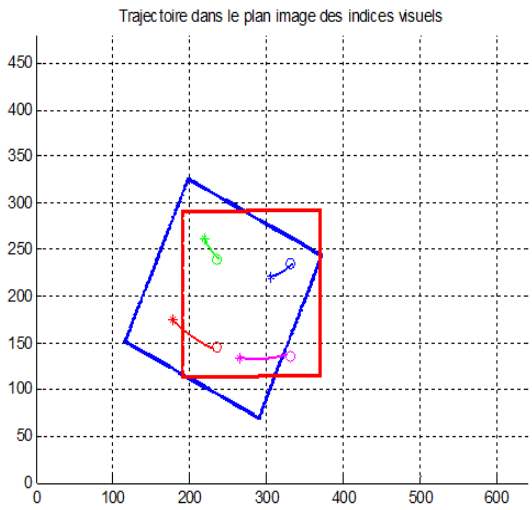

FIGURE 3.20: Trajectoire des indices visuelles dans le plan image :commande par mode glissant.

- Interprétation

Cette surface ainsi considérée assure la convergence de la fonction de tâche en un temps $3s$ (3.18) mais présente l'inconvénient d'avoir un fort chatering des

vitesses articulaires (3.19).en effet à partir de $t = 3s$,la vitesse oscille entre des valeurs de ± 0.5 pour la première articulation et de ± 0.05 pour la deuxième articulation.

Et en choisissant comme surface de glissement la fonction $\sigma = \dot{e} + \lambda(e)$,La simulation de la commande a donné les courbes (3.21) pour la fonction de tâche , (3.22) pour les vitesse articulaires et (3.23) pour l'évolution des indices visuelles.

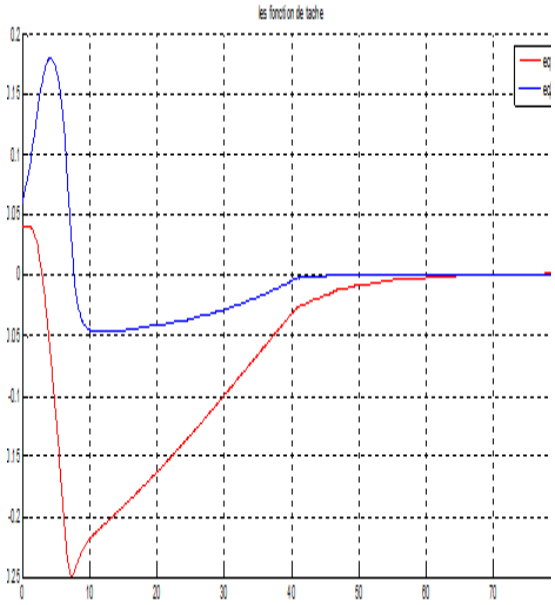

FIGURE 3.21: Les fonctions de tâche :commande par mode glissant.

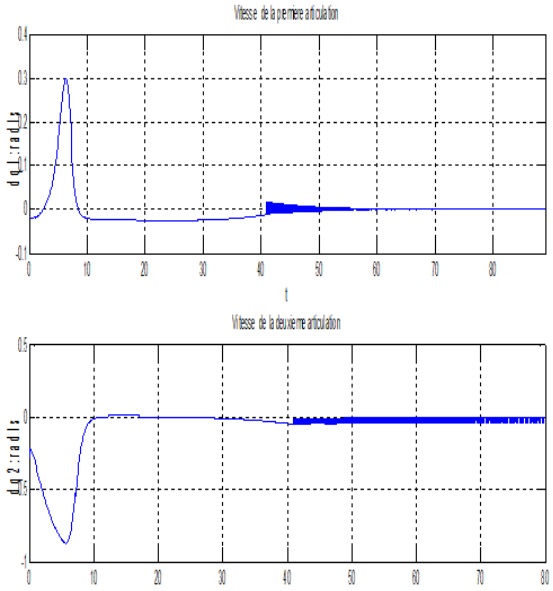

FIGURE 3.22: Les vitesses articulaires :commande par mode glissant.

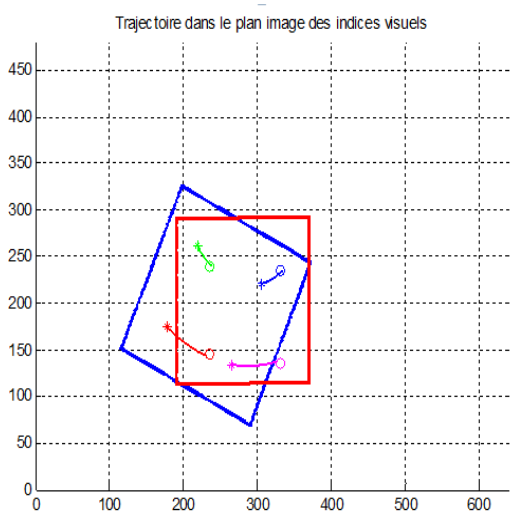

FIGURE 3.23: Trajectoire des indices visuelles dans le plan image :commande par mode glissant.

- Interprétation

Pour cette surface de glissement, la convergence de la fonction de tâche 3.21 se fait en un temps plus important que pour la première surface mais fournit une meilleure dynamique des vitesses articulaires 3.22.Même si le phénomène de chattering persiste, il est soit à amplitude décroissante dans le cas de la premier vitesse articulaire soit constant mais de faible amplitude pour le cas de la deuxième vitesse.

Remarque :

cette surface donne de meilleur résultats que la première, ceci s'explique par le fait qu'elle inclut dans son expression la dérivée de l'erreur dans ce cas le régime glissant contraint non seulement l'erreur à aller vers zero mais aussi ces dynamique.

3.8.2 Cas d'un suivi de ligne

- **Commande classique**

  la simulation de la commande classique pour le suivi d'une ligne donnée par $\rho = 0$, $\theta = 0$, avec une position initiale du bras $q1 = q2 = 0.8$,a permis de tracer l'évolution de la fonction de tâche (3.24) ainsi que celle des deux erreurs :sur le rayon et sur l'angle (3.25).Les vitesses articulaires sont représentées par la figure (3.26).

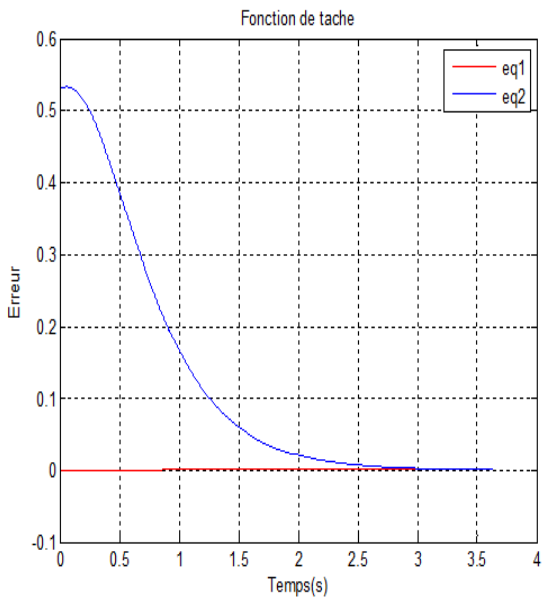

FIGURE 3.24: Les fonctions de tâche :Commande classique.

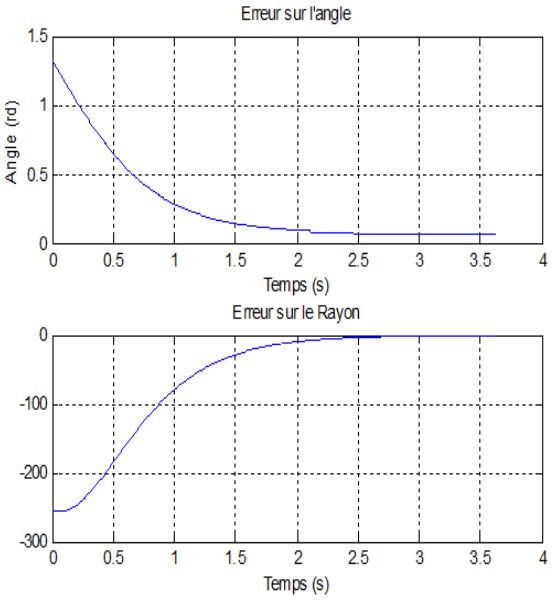

FIGURE 3.25: Erreurs rayon,angle :commande classique.

- Interprétation

On constate d'après la figure (3.25)que la convergence de la ligne détectée vers sa position désirée dans le plan image se fait en $3s$ .Lorsque la fonction de tâche

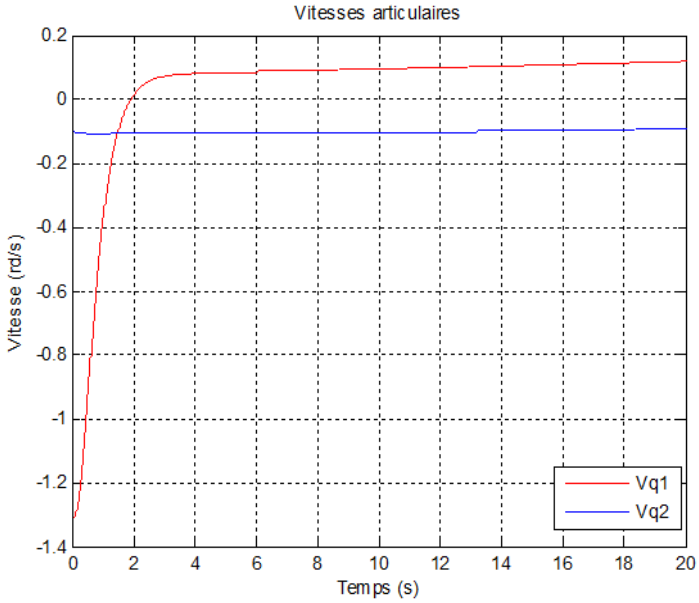

FIGURE 3.26: Les vitesses articulaires :commande classique

converge vers zeros (3.24),les vitesses articulaires se stabilisent à leurs valeurs désirées pour le suivi de ligne (3.26)

- **Régulateur PI**
  pour le régulateur PI,les résultats de simulations sont donnés par (3.27),(3.28),(3.29) qui correspondent respectivement à la fonction de tâche,les erreurs sur le rayon et l'angle et enfin les vitesses articulaires.

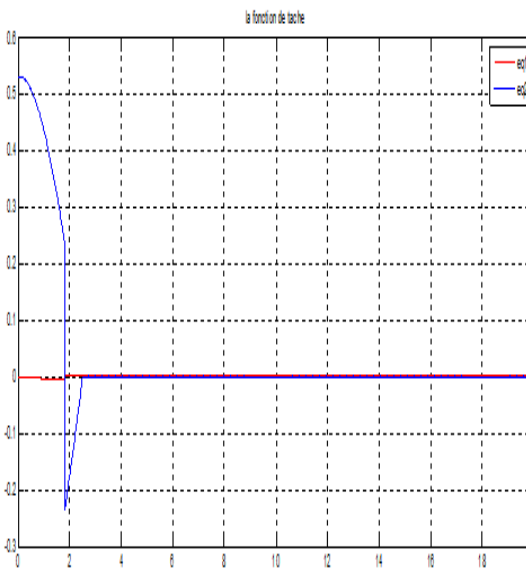

FIGURE 3.27: Les fonctions de tâche :Réglage PI.

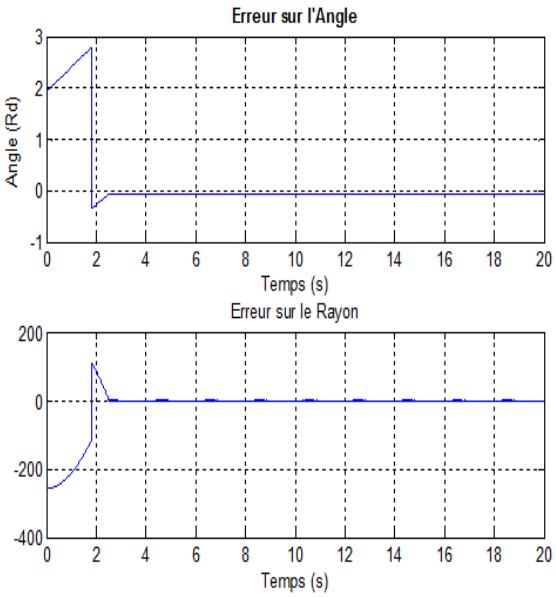

FIGURE 3.28: Erreurs rayon,angle :Réglage PI.

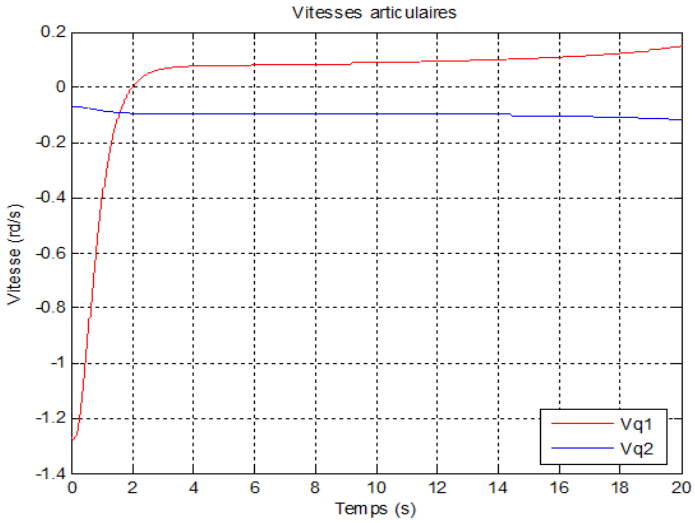

FIGURE 3.29: Les vitesses articulaires :Régulateur PI

- **Commande par mode glissant**

  En simulant la tâche de suivi de ligne avec la commande par mode de glissement ($\sigma = C(s - s^*)$) , on a obtenu les résultats présentés par les figures (3.30),(3.31).

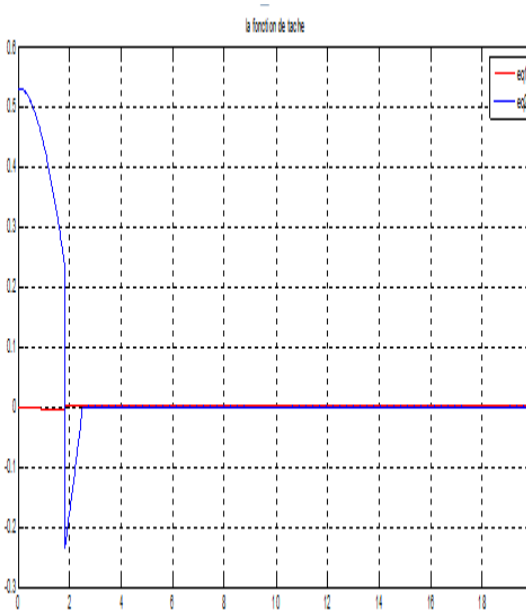

FIGURE 3.30: Les fonctions de tâche :commande par mode glissant.

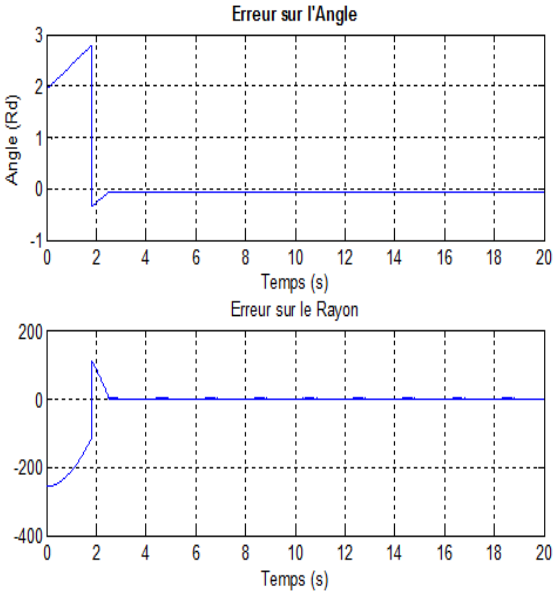

FIGURE 3.31: Erreurs rayon,angle :commande par mode glissant.

- Interprétation

Comme pour le positionnement par rapport à quatre points,cette commande assure la convergence de la fonction de tâche (3.30) mais présente un chattering sur les vitesses articulaires 3.32.

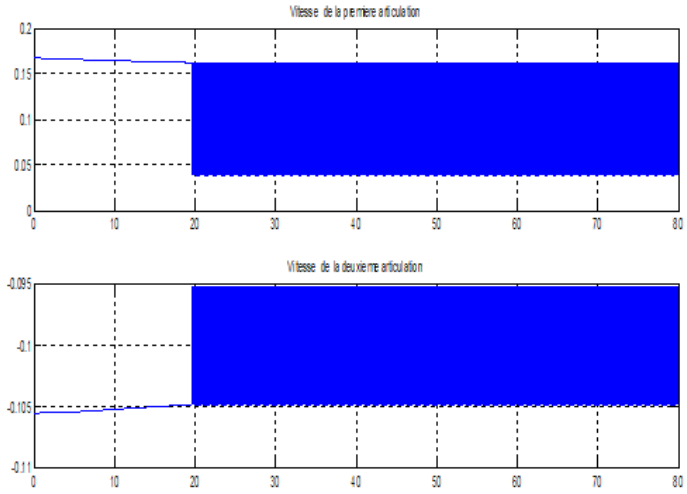

FIGURE 3.32: Les vitesses articulaires :commande par mode glissant

3.9 Résultats de simulation de la commande du robot

La boucle de vision renvoie juste les consignes des vitesses articulaires, la boucle du robot doit, à partir de ces consignes, fournir les couples de commande aux actionneurs qui feront déplacer l'outil terminal vers la position désirée. Ainsi pour la validation de nos lois de commande, des simulations de la boucle complète ont étaient faite, le couple de commande fournit par la boucle du robot est appliqué aux modèle dynamique, l'erreur entre les vitesses articulaires du robot et la consigne nous fournit alors une information sur la convergence du système et l'étude des couples nous permet de tester l'applicabilité de chaque loi de commande.

3.9.1 Positionnement par rapport à 4 points

Les mêmes lois de commande synthétisés ont étaient ajoutées dans la boucle de commande du robot selon le schéma de la figure (3.10), les résultats des simulations donnent les figure (3.33), (3.34) pour la commande classique, les figures (3.35), (3.36) pour le régulateur PID et (3.37),(3.38) pour la commande par mode de glissement.

- **Commande classique**

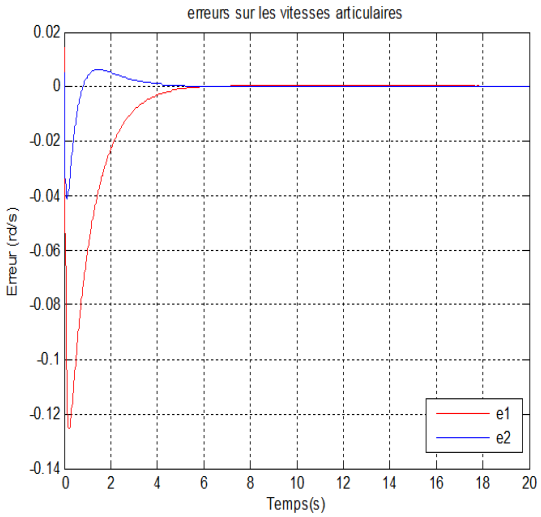

FIGURE 3.33: L'erreur sur les vitesses articulaires :commande classique.

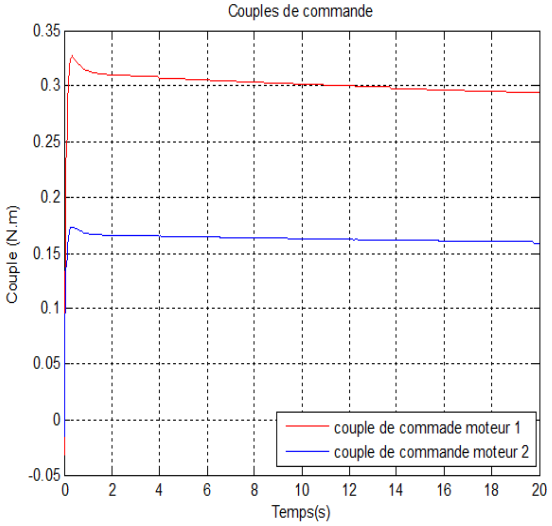

FIGURE 3.34: Les Couples de commande :commande classique.

- **Régulateur PI**

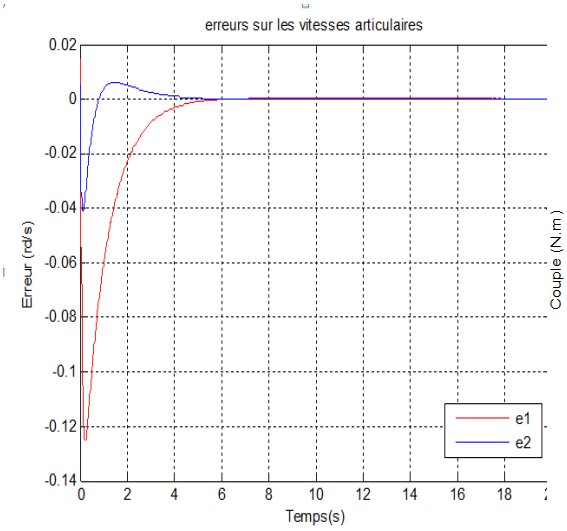

FIGURE 3.35: L'erreur sur les vitesses articulaires :réglage PI.

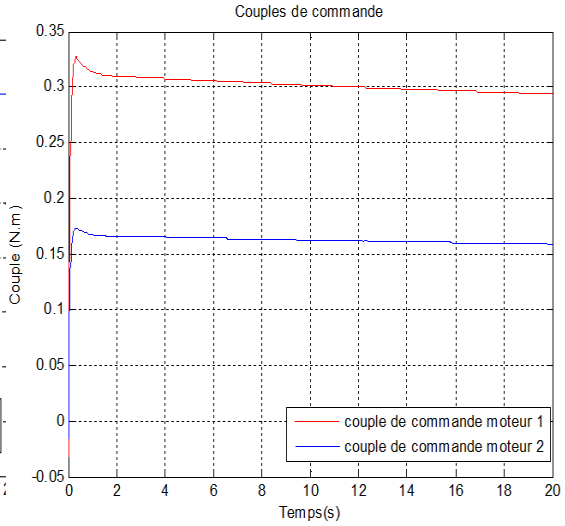

FIGURE 3.36: Les Couples de commande :Réglage PI.

- **Commande par mode glissant**

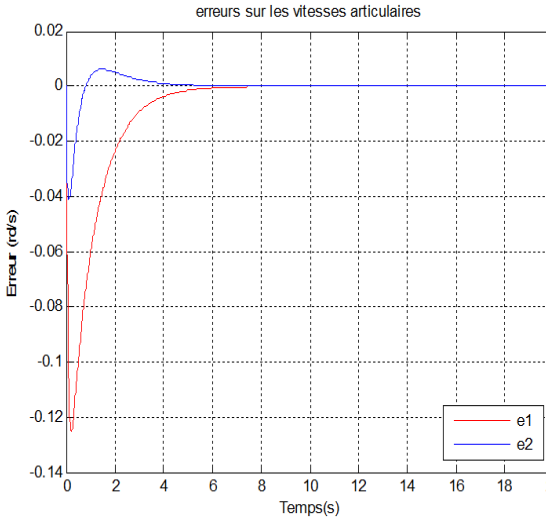

FIGURE 3.37: L'erreur sur les vitesses articulaires :commande par mode glissant.

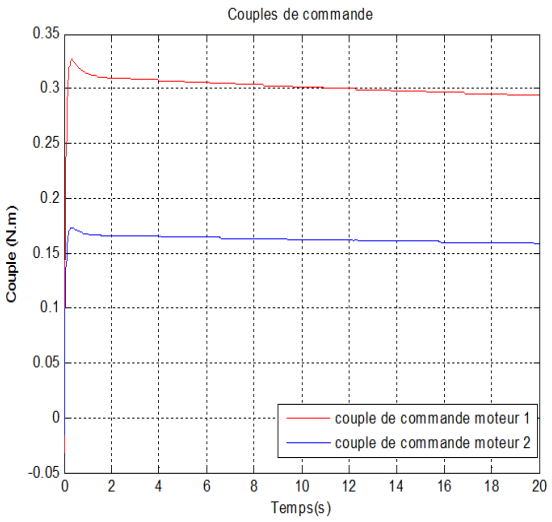

FIGURE 3.38: Les Couples de commande :commande par mode glissant.

- Interprétation

Pour les trois lois de commande on constate l'annulation de l'erreur entre les consignes de vitesses données par chacune et les vitesses à la sortie de la boucle du robot *Modele dynamique* : (3.33),(3.35),(3.37).

3.9.2 Suivi de ligne

Comme pour le positionnement par rapport à quatre points, une simulation de la boucle complète du robot pour chaque lois de commande a était faite, les résultats sont donnés par les figures 3.39 et 3.40 pour la commande classique, les figures 3.41 et 3.42 pour la commande PI et enfin par les figures 3.43 et 3.44 pour le mode glissant.

- **Commande classique**

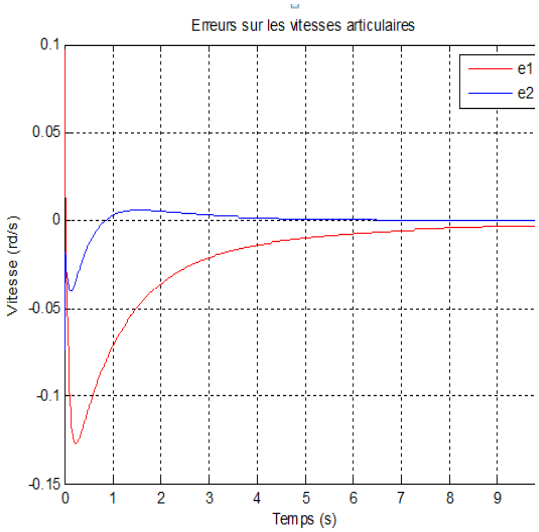

FIGURE 3.39: L'erreur sur les vitesses articulaires :Commande classique.

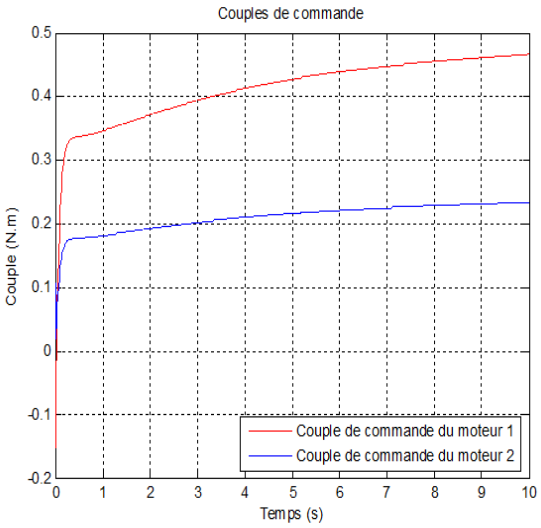

FIGURE 3.40: Les Couples de commande :commande classique.

- **Régulateur PI**

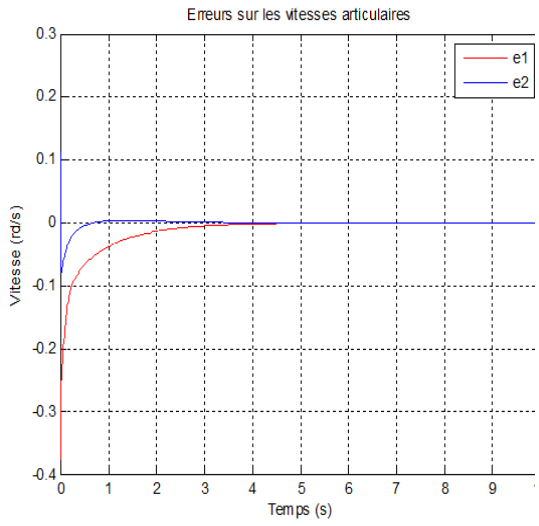

FIGURE 3.41: L'erreur sur les vitesses articulaires :Réglage PI.

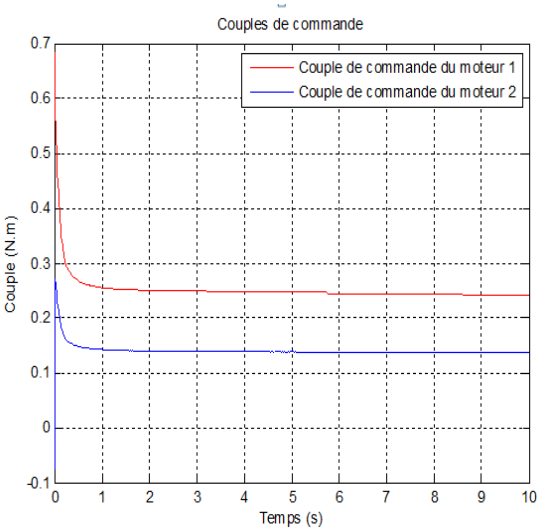

FIGURE 3.42: Les Couples de commande :Réglage PI.

- **Commande par mode glissant**

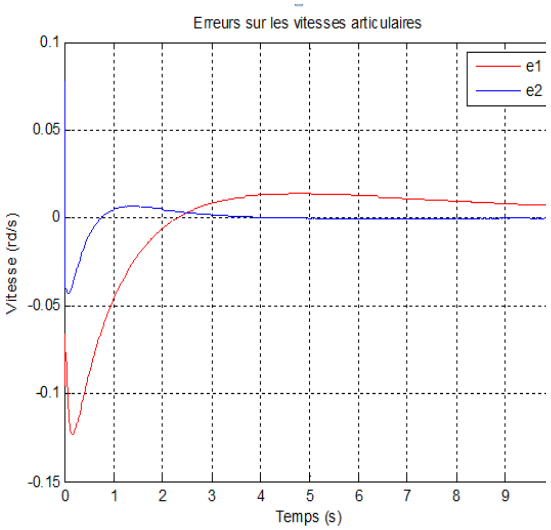

FIGURE 3.43: L'erreur sur les vitesses articulaires :Commande par mode glissant.

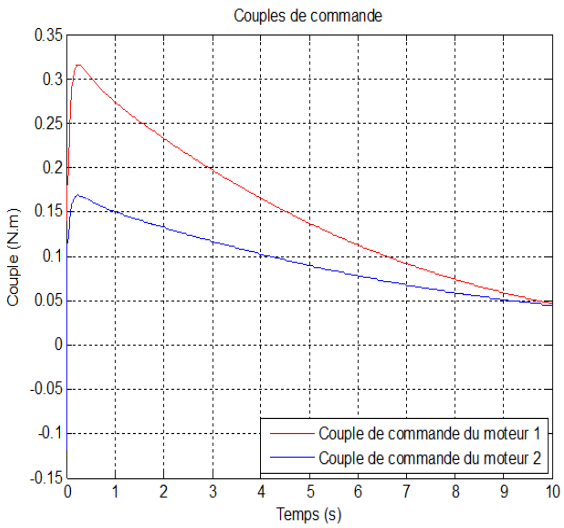

FIGURE 3.44: Les Couples de commande :Commande par mode glissant.

- Interprétation :

L'application des consignes de vitesses de la boucle de vision sur le robot représenté par son modèle dynamique,et commandé par un contrôleur PID,donne les résultats des figures (3.39),(3.41),(3.43) pour l'erreur sur les vitesses articulaires et (3.40),(3.42),(3.44) pour les couples appliqués au robot et ceci pour chaque loi de commande développée pour la boucle de vision.

3.9.3 Test de robustesse

Dans ce paragraphe,on va étudier la robustesse de l'asservissement visuel vis-à-vis des perturbation.pour ce faire ,on considéré le cas où les perturbations agissent sur les couples articulaires.Notre objectif est d'assurer la convergence du robot vers la configuration souhaitée,décrite par l'image désirée.donc l'objectif des simulations est de montrer la robustesse de l'asservissent visuel vis à vis des diverses perturbation présenté sur le système dans le cas d'un positionnement par rapport à des points et dans le cas d'un suivi de ligne.

les perturbation introduites dans ces simulations sont de l'ordre de 10 pour cent de la commande utile.

(a) **- Positionnement par rapport à 4 points**

- **Commande classique**

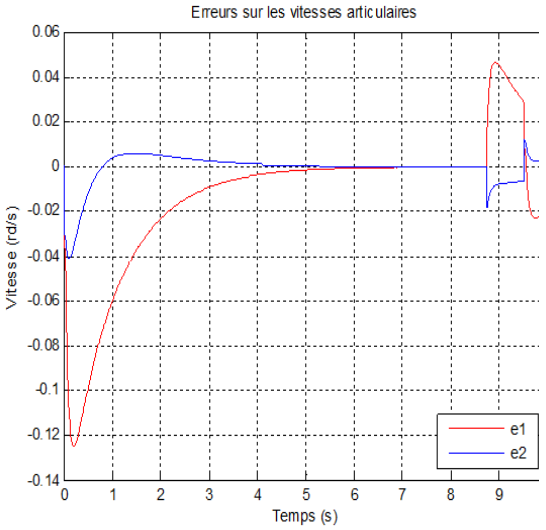

FIGURE 3.45: L'erreur sur les vitesse articulaires :commande classique.

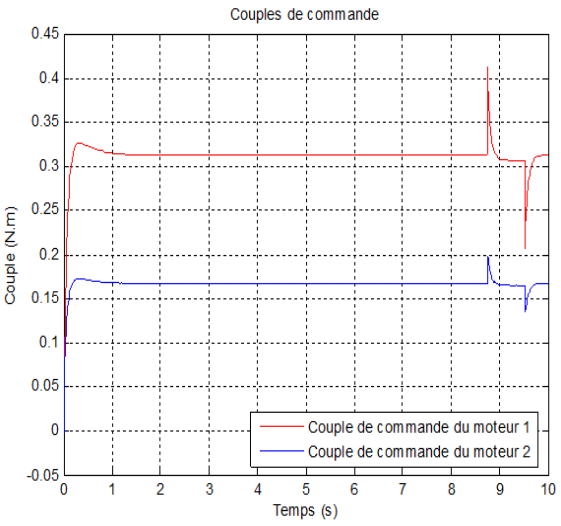

FIGURE 3.46: Couples de commande :commande classique

- **Régulateur PI**

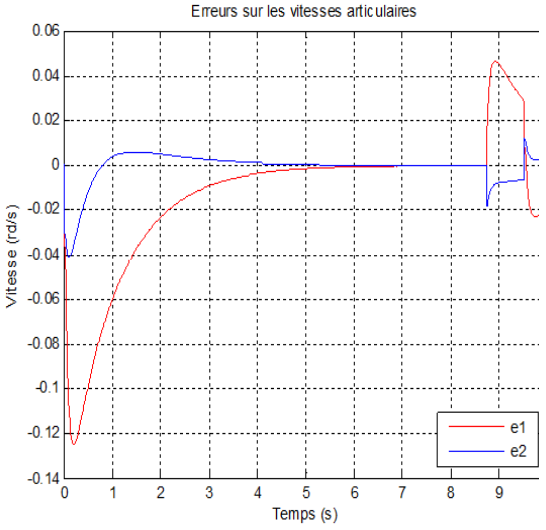

FIGURE 3.47: L'erreur sur les vitesse articulaires :Réglage PI.

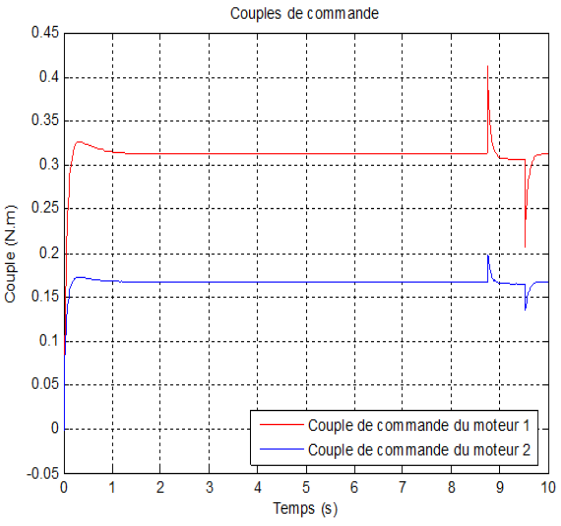

FIGURE 3.48: Couples de commande :Réglage PI .

- **Commande par mode glissant**

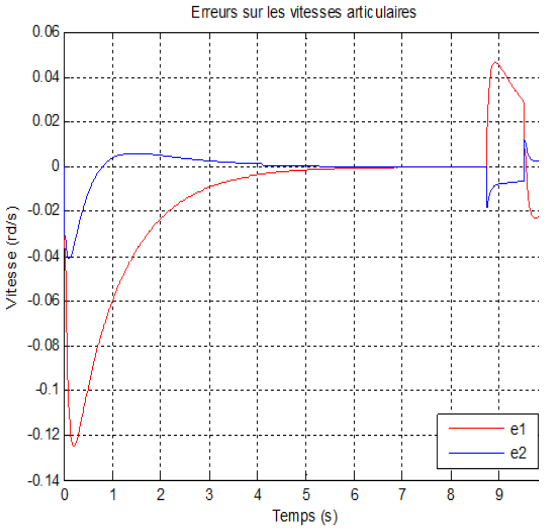

FIGURE 3.49: L'erreur sur les vitesse articulaires :mode glissant.

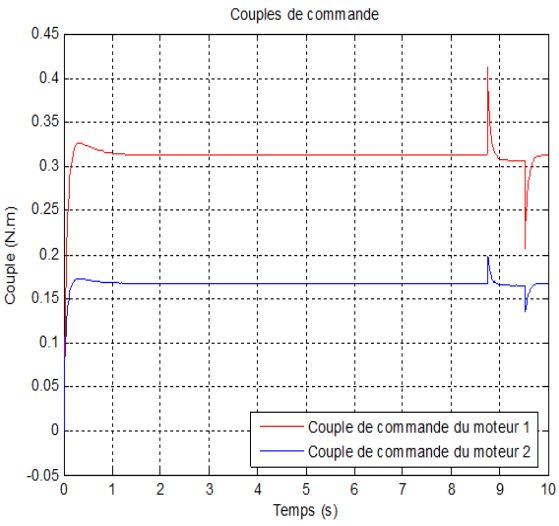

FIGURE 3.50: Couples de commande :mode glissant.

(b) **Suivi de ligne**

- **Commande classique**

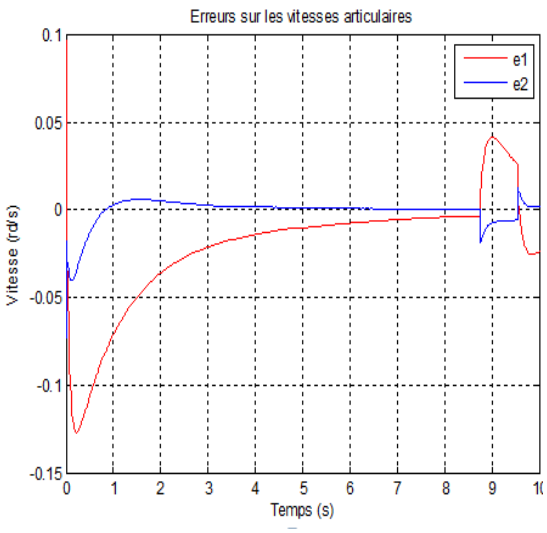

FIGURE 3.51: L'erreur sur les vitesse articulaires :commande classique.

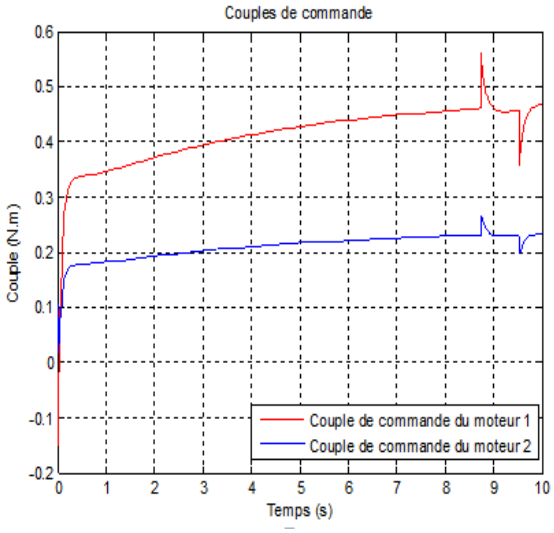

FIGURE 3.52: Couples de commande :commande classique.

- **Régulateur PI**

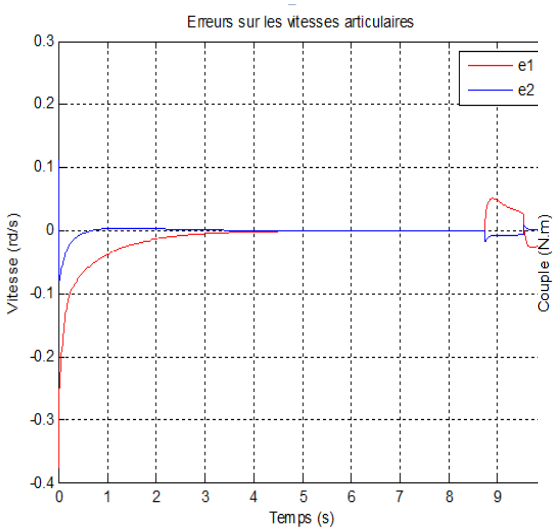

FIGURE 3.53: L'erreur sur les vitesse articulaires :Réglage PI.

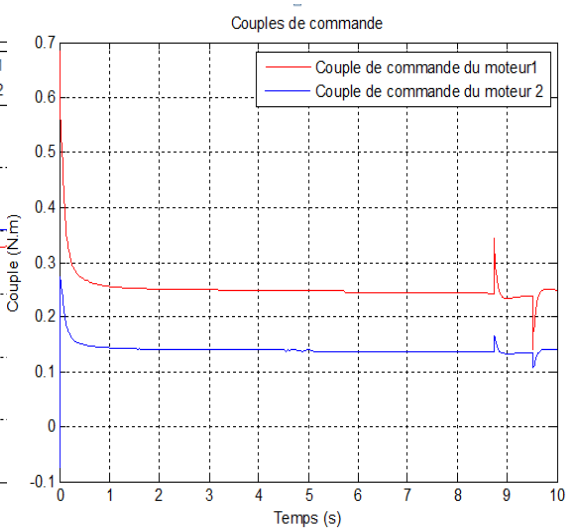

FIGURE 3.54: Couples de commande :Réglage PI.

- **Commande par mode glissant**

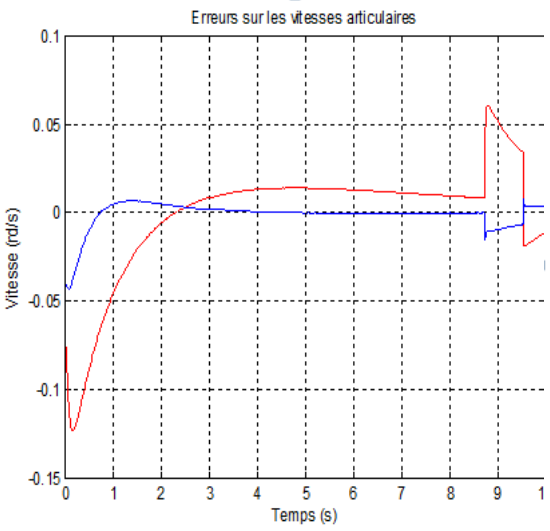

FIGURE 3.55: L'erreur sur les vitesse articulaires :commande par mode glissant.

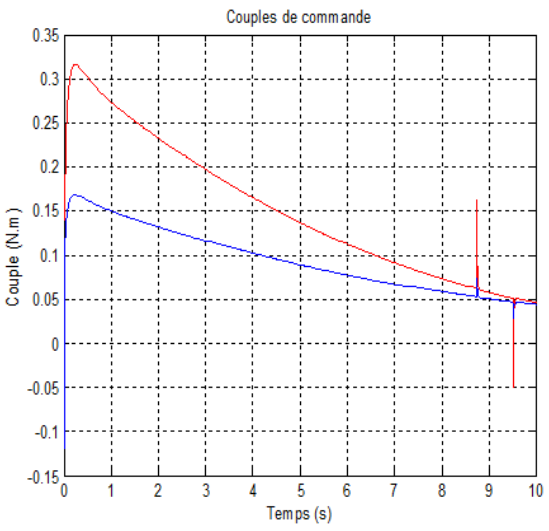

FIGURE 3.56: Couples de commande :mode glissant .

- Interprétation

les résultats de simulation présentés dans ce paragraphe confirme que l'asservissement visuel est robuste vis à vis des perturbations sur les courbes articulaire, les même performances ont été constaté dans les diverse loi de commande (commande classique ,le PI et la commande par mode de glissant) :rejet des perturbations avec couples articulaires admissible.

On conçoit donc ainsi une loi de commande robuste indépendante de la loi de commande de robot,qui ne fait qu'assurer la stabilité et la bonne poursuite de ce dernier.Cette robustesse est due au fait qu'a chaque instant,la commande visuelle

agit de manière à faire tendre l'image actuelle vers l'image désirée.

3.10 Comparaison des résultats des lois de commande

Afin de sélectionner la meilleure loi de commande à appliquer à notre système nous avons procédé d'après les résultats précédents à une comparaison des performances des lois de commande étudiées dans le cas d'un positionnement par rapport à quatre points ,cette comparaison est résume dans le tableau (3.10).

Remarque :

Dans le tableau 3.10,par $\sigma 1$ on désigne la commande par mode glissant avec la première surface et par $\sigma 2$ celle obtenue avec la deuxième surface.

	Com classique	PI	($\sigma 1$)	($\sigma 2$)
T de convergence	10S	5s	3s	50s
Erreur max/min	$0.11/-0.04$	$0.11/-0.03$	$0.23/-0.23$	$0.17/-0.23$
Allure des courbes	moins lisse	lisse	fort oscillation	très faible oscillation

Remarque :

Le tableau (3.10) résument les résultats d'application de la technique d'asservissement visuel par diverses méthodes(lois de commande :la commande classique, le PI et le mode de glissement avec les deux surface présentées précédemment).Le choix de la méthode se fait en fonction de ses performances et du temps de calcul. Dans notre cas la commande PI peut se révéler comme étant le meilleur choix car elle a assuré une excellente convergence vers les configuration désirée avec une meilleure allure des courbes ,les vitesses articulaire,les couples articulaire et la trajectoire dans le plan image, en un temps admissible.

3.11 Conclusion

Dans ce chapitre,nous avons synthétisé les lois de commandes à appliquer à la boucle de vision.Nous avons traité trois types de commande :la commande classique ,le PI et la commande par mode glissement.

Nous avons par la suite comparé les performances de chaque commande,en terme de temps de convergence,de forme de courbes et de robustesse de la boucle de commande complète.

Conclusion Générale et Perspectives

Ce présent mémoire s'est porté sur l'étude de la commande référencée vision du bras manipulateur de type SCARA à deux degrés de liberté disponible au niveau du *LCP de l'Ecole Nationale Polytechnique* . Nous nous sommes intéressées au développement de deux types de tâches par asservissement visuel à savoir ; le positionnement par rapport à quatre points et le suivi de ligne. La réalisation de ces tâches a était effectuée par la technique 2D. Nous avons entamé ce rapport par des généralités sur les bras manipulateurs, ce qui constitue un préambule avant d'aborder la présentation du système qui nous intéresse. Dans cette présentation, nous avons donné les caractéristiques de notre système ainsi que de son interface de commande et son environnement software.

Dans le deuxième chapitre, nous avons donné les notions de base d'un asservissement visuel, nous avons aussi donné quelques bases du traitement d'image, domaine qui est étroitement lié à la vision par ordinateur, et par la suite donné la modélisation du capteur de vision. Nous avons notamment abordé dans ce chapitre la notion de fonction de tâche, de matrice d'interaction et son calcul pour deux types de primitives : le point et la droite, et enfin nous avons étudié le cas particulier de la technique 2D, technique que nous avons utilisée par la suite.

Afin de choisir la meilleure loi de commande à appliquer au système, nous avons, dans le chapitre III, synthétisé et simulé sous *Matlab* trois lois de commande différentes. Tout d'abord, nous avons utilisé une loi de commande classique, qui constitue l'approche la plus intuitive puis, nous avons essayé un régulateur PI et nous avons terminé par une commande robuste par mode glissant. La comparaison des résultats de simulation de ces trois lois de commande nous ont permis de sélectionner la mieux adaptée à notre application, ceci en termes de temps de réponse, d'erreur max et de robustesse.

Pour l'implémentation d'un asservissement visuel sur notre système, nous proposons, on s'est opposé à certains problèmes :

– La possibilité d'envoyer des consignes en vitesse de manière continue vers l'interface.
– la spécificité de l'environnement software PEWIN à de simple commande de mouvement connus au préalable.
– La possibilité d'un traitement en temps réel limitée par la puissance du calculateur ainsi que par la fréquence du capteur de vision utilisés

Comme perspectives,nous proposons : L'implémentation de l'asservissement visuel sur le système réel ,L'utilisation de moyen de traitement d'image plus puissant et de camera spécialisée.L'exploitation des algorithmes d'extraction de primitives : SURF,SIFT et ORB pour des applications de vision sur le robot.

Annexe A

Modélisation géométrique

Modèle géométrique direct

Le MGD d'un bras manipulateur donne la position de son organe terminal en fonction de sa configuration : $X = f(q)$.

Le MGD du robot Scara à 4 degrés de liberté ; ayant trois articulation rotoide et une articulation prismatique ; peut être représente par la matrice homogène suivante :

$$T_0^4 = T_1^0 * T_2^1 * T_3^2 * T_4^3 \tag{3.39}$$

Où : T_i^{i-1} représente la matrice de transformation homogène définissant le repère R_i dans le repère R_{i-1},elle s'obtient par la méthode de *Denavit-Hartenberg*(D-H) :

$$T_i^{i-1} = \begin{pmatrix} c\theta_i & -s\theta_i & 0 & d_i \\ c\alpha_i s\theta_i & c\alpha_i c\theta_i & -s\alpha_i & -r_i s\alpha_i \\ s\alpha_i s\theta_i & s\alpha_i c\theta_i & c\alpha_i & r_i c\alpha_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.40}$$

Où : d_i : distance entre z_{i-1} et z_i correspondant à une rotation autour de x_{i-1},
θ_i : angle entre les axes x_{i-1} et x_i correspondant à une rotation autour de z_i ,
r_i : distance entre x_{i-1} et x_i le long de z_i.
Et :

$$\begin{cases} c\beta_i = cos(\beta_i) \\ s\beta_i = sin(\beta_i) \end{cases}$$

La représentation du robot Scara 4*dll* suivant la convention de *Denavit-Hartenberg* est donnée par la figure 3.57, les valeurs numériques sont regroupées dans le tableau 1 ci-dessous

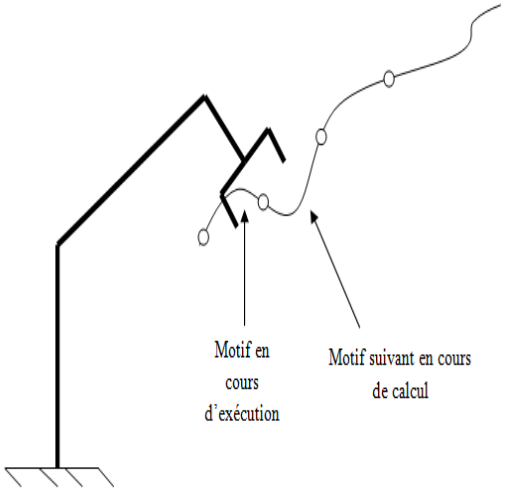

FIGURE 3.57: Système de coordonnées et paramètres de Denavit-Hartenberg affectés au Scara 4dll.

Numéro de la liaison i	σ_i	d_i	α_i	θi	r_i
1	0	0	0	θ_1	r_1
2	0	d_2	0	θ_2	0
3	0	d_3	0	θ_3	0
4	1	0	0	0	r_4

TABLE 3.2: Paramètres D-H du robot Scara RRRP

Avec : σ_i :Égal à zero ou bien un pour une articulation rotoide ou bien prismatique respectivement.
Et :

$$\begin{cases} d_2 = 186.5mm \\ d_3 = 172.5mm \\ r_1 = 385mm \\ r_{4min} = 364, r_{4max} = 415 \end{cases}$$

En utilisant la relation 3.40,on obtient les transformations homogènes $T_1^0, T_2^1, T_3^2, T_4^3$,ce qui permet le calcul de la matrice de transformation entre le repère $R_0 et R_4$,c'est la matrice homogène T_4^0 donnée dans l'équation 3.39.

$$T_4^0 = \begin{pmatrix} c_{123} & -s_{123} & 0 & d_3c_{12} + d_2c_1 \\ s_{123} & c_{123} & 0 & d_3s_{12} + d_2s_1 \\ 0 & 0 & 1 & r_4 + r_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.41}$$

Où :

$$\begin{cases} c_i = cos(\theta_i),\ c_{ij} = cos(\theta_i + \theta_j),\ c_{ijk} = cos(\theta_i + \theta_j + \theta_k) \\ s_i = sin(\theta_i),\ s_{ij} = sin(\theta_i + \theta_j),\ s_{ijk} = sin(\theta_i + \theta_j + \theta_k) \end{cases} \tag{3.42}$$

On définit le vecteur X de l'espace opérationnel comme étant la position dans l'espace cartésien $3D$ et l'orientation ϕ définie par l'angle entre les axes x_4 et x_0
Alors,le modèle géométrique de la structure est donné par le système :

$$\begin{cases} x = d_2 c_1 + d_3 c_{12} \\ y = d_2 s_1 + d_3 s_{12} \\ z = r_1 + r_4 \\ \phi = \theta_1 + \theta_2 + \theta_3 \end{cases} \tag{3.43}$$

Modèle géométrique inverse

Le MGI,comme exposé précédemment, donne la ou les configurations correspondantes à une position de l'outil terminal.
On considère le cas où le repère de l'outil terminal est disposé tel que la transformation désirée $(T_4^0)^*$ soit de la forme :

$$(T_4^0)^* = \begin{pmatrix} S_x & N_x & 0 & P_x \\ S_y & N_y & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.44}$$

par identification avec le system 3.41,on trouve :

$$\begin{cases} P_x = d_2 c_1 + d_3 c_{12} \\ P_y = d_2 s_1 + d_3 s_{12} \\ P_z = r_1 + r_4 \end{cases} \tag{3.45}$$

En résolvant le système 3.45,on aboutit à la solution :

$$\begin{cases} c_1 = \dfrac{(d_2 + d_3 c_2) P_x + (d_3 s_2) P_y}{d_2^2 + d_3^2 + 2 d_2 d_3 c_2} \\ s_1 = \dfrac{-(d_3 s_2) P_x + (d_2 + d_3 c_2) P_y}{d_2^2 + d_3^2 + 2 d_2 d_3 c_2} \\ c_2 = \dfrac{P_x^2 + P_y^2 - d_2^2 - d_3^2}{2 d_2 d_3} \\ r_4 = P_z - r_1 \end{cases} \tag{3.46}$$

Enfin ,on obtient :

$$\begin{cases} \theta_1 = arctan(\dfrac{-(d_3 s_2)P_x + (d_2 + d_3 c_2)P_y}{(d_2 + d_3 c_2)P_x + (d_3 s_2)P_y}) \\ \theta_2 = Arcos(\dfrac{P_x^2 + P_y^2 - d_2^2 - d_3^2}{2 d_2 d_3}) \\ r_4 = P_z - r_1 \end{cases} \tag{3.47}$$

Et pour $\theta_3, ontrouve$:

$$\theta_3 = arctan(\frac{S_y}{S_x}) - (\theta_1 + \theta_2) \tag{3.48}$$

Avec :

$$\begin{cases} S_x = c_{123} \\ S_y = s_{123} \end{cases} \tag{3.49}$$

On obtient ainsi le MGI de notre robot manipulateur :

$$\begin{cases} \theta_1 = arctan(\dfrac{-(d_3 s_2)P_x + (d_2 + d_3 c_2)P_y}{(d_2 + d_3 c_2)P_x + (d_3 s_2)P_y}) \\ \theta_2 = Arcos(\dfrac{P_x^2 + P_y^2 - d_2^2 - d_3^2}{2 d_2 d_3}) \\ \theta_3 = arctan(\dfrac{S_y}{S_x}) - (\theta_1 + \theta_2) r_4 = P_z - r_1 \end{cases} \tag{3.50}$$

Modélisation cinématique

Modèle cinématique direct

Le modèle cinématique direct du premier ordre d'un robot-manipulateur décrit les vitesses des coordonnées opérationnelles en fonction des vitesses articulaires ,noté :

$$\dot{X} = J(q)\dot{q} \tag{3.51}$$

Le calcul de la matrice jacobienne peut se faire en dérivant directement le modèle géométrique direct,
Ainsi ;les dérivées temporelles des variables opérationnelle du système 3.43 donnent :

$$\begin{cases} \dot{x} = -d_2 s_1 \dot{\theta_1} - d_3 s_{12} \dot{\theta_1} - d_3 s_{12} \dot{\theta_2} \\ \dot{y} = d_2 c_1 \dot{\theta_1} + d_3 c_{12} \dot{\theta_1} + d_3 c_{12} \dot{\theta_2} \\ \dot{z} = \dot{r_4} \\ \dot{\phi} = \dot{\theta_1} + \dot{\theta_2} + \dot{\theta_3} \end{cases}$$

Par conséquent,la jacobienne J est de la forme :

$$J = \begin{pmatrix} -(d_2 s_1 + d_3 s_{12}) & -d_3 s_{12} & 0 & 0 \\ d_2 c_1 + d_3 c_{12} & d_3 c_{12} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \tag{3.52}$$

Modèle cinématique inverse

Le MCI d'un bras manipulateur donne les vitesses généralisées $\dot{q}$ en fonction des vitesses opérationnelles $\dot{X}$. Comme pour notre cas$n = m$,alors pour obtenir ce modèle il suffit d'inverser la matrice jacobienne J du modèle direct Lorsque la matrice J a la forme suivante :

$$J = \begin{pmatrix} A & 0 \\ B & C \end{pmatrix} \tag{3.53}$$

Alors,son inverse est donné par la matrice :

$$\mathrm{J}^{-1} = \begin{pmatrix} A^{-1} & 0 \\ -C^{-1}BA^{-1} & C^{-1} \end{pmatrix}$$

En appliquant la relation 3.11 ,on obtient la matrice J^{-1} de notre système :

$$J^{-1} = \frac{1}{d_2 d_3 s_2} \begin{pmatrix} d_3 c_{12} & d_3 s_{12} & 0 & 0 \\ -(d_2 c_1 + d_3 c_{12}) & -(d_3 s_{12} + d_2 s_1) & 0 & 0 \\ d_2 c_1 & d_2 s_1 & 0 & d_2 d_3 s_2 \\ 0 & 0 & d_2 d_3 s_2 & 0 \end{pmatrix} \tag{3.54}$$

Modélisation dynamique

En utilisant l'équation 3.40,on obtient les transformées homogènes élémentaires de type rotoides,

$$T_1 = \begin{pmatrix} c_1 & -s_1 & 0 & l_1 c_1 \\ s_1 & -c_1 & 0 & l_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.55}$$

$$T_2 = \begin{pmatrix} 0 & 0 & 0 & l_1 c_1 \\ 0 & 0 & 0 & l_1 s_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} c_{12} & -s_{12} & 0 & l_2 c_{12} \\ s_{12} & c_{12} & 0 & l_2 s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.56}$$

Ce qui donne pour les vitesses,

$$\dot{T}_1 = \begin{pmatrix} -s_1 & -c_1 & 0 & -l_1 s_1 \\ c_1 & -s_1 & 0 & l_1 c_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.57}$$

Et :

$$\dot{T}_2 = \begin{pmatrix} 0 & 0 & 0 & -l_1 s_1 \\ 0 & 0 & 0 & l_1 c_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \dot{v_1} + \begin{pmatrix} -s_{12} & -c_{12} & 0 & -l_2 s_{12} \\ c_{12} & -s_{12} & 0 & l_2 c_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} (\dot{v_1} + \dot{v_2}) \tag{3.58}$$

Et pour les accélérations ;

$$\dot{T}_1 = \begin{pmatrix} -s_1 & -c_1 & 0 & -l_1 s_1 \\ c_1 & -s_1 & 0 & l_1 c_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \dot{v_1} + \begin{pmatrix} -c_1 & s_1 & 0 & -l_1 c_1 \\ -s_1 & -c_1 & 0 & -l_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \dot{v_1^2} \tag{3.59}$$

Et :

$$\dot{T}_2 = \begin{pmatrix} 0 & 0 & 0 & -l_1 s_1 \\ 0 & 0 & 0 & l_1 c_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \dot{v_1} + \begin{pmatrix} -s_{12} & -c_{12} & 0 & -l_2 s_{12} \\ c_{12} & -s_{12} & 0 & l_2 c_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \dot{v_1} + \dot{v_2} + \begin{pmatrix} 0 & 0 & 0 & -l_1 c_1 \\ 0 & 0 & 0 & -l_1 s_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \dot{v_1^2} + \begin{pmatrix} -c_{12} & s_{12} \\ -s_{12} & -c_1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{3.60}$$

Les masses étant supposées ponctuelles, nous avons :

$$s_i^i = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \; l_i^i \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \; pour i = 1, 2 \tag{3.61}$$

D'où ;

$$s_i^i = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \; j_i^i = m_i \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \; i = 1, 2 \tag{3.62}$$

Quant à l'accélération de la pesanteur, elle a la direction de $(\breve{}z)$, d'où ;

$$g = \begin{pmatrix} 0 \\ 0 \\ -1 \\ 0 \end{pmatrix} \tag{3.63}$$

Ce qui nous donne pour les termes $G(q)$,$M(q)$ et $B(q)$:

$$
\begin{cases}
G_1 = G_2 \\
M_{1,1} = m_1 l_1^2 + m_2(l_1^2 + 2l_1 l_2 + l_2^2) \\
M_{1,2} = M_{2,1} = m_2(l_2(l_2 + l_1 c_2) \\
M_{2,2} = m_2 l_2^2 \\
B_{1,(1,1)} = 0 \\
B_{1,(1,2)} = B_{1,(2,1)} = -m_2 l_1 l_2 s_2 \\
B_{2,(1,1)} = m_2 l_1 l_2 s_2 \\
B_{2,(1,2)} = B_{2,(2,1)} = 0 \\
B_{2,(2,2)} = 0
\end{cases}
$$

Annexe B

Représentation d'état du bras manipulateur

Comme déjà exposé dans le premier chapitre, le model dynamique du robot s'écrit sous la forme :

$$\tau(q,\dot{q}) = M(q)\dot{q} + C(q,\dot{q})\dot{q} + G(q) \tag{3.64}$$

En posant le model suivant :

$$\delta_{l,\lambda} = \begin{cases} x_1 = q \\ x_2 = \dot{q} \end{cases} \tag{3.65}$$

On peut déduire le modèle d'état du bras manipulateur, qui sera :

$$\begin{cases} \dot{x_1} = \dot{q} \\ \dot{x_2} = \dot{q} = (M(q)^{-1})(\tau(q,\dot{q},\dot{q}) - C(q,\dot{q})\dot{q} - G(q)) \end{cases} \tag{3.66}$$

Où ;

- τ : représente la commande à appliquer au robot ;
- q et $\dot{q}$:représentent les sorties à commander.

Synthèse de commande décentralisée

L'utilisation de moteurs à fort rapport de réduction minimise l'influence de l'inertie de chaque segment du bras comparée à celle du moteur, on peut ainsi considérer qu'on commande deux sous systèmes du second ordre de la forme :

$$\tau_i = M_i q_i + C_i \dot{q}_i + G_i \; i = 1, 2. \tag{3.67}$$

On pourra alors synthétiser une commande décentralisée pour 2 sous systèmes linéaires indépendants. La commande de la position articulaire q_i peut alors être assurée par un régulateur linéaire de type PID :

$$\tau_i = K_{Pi} e_i + K_{Di} \dot{e}_i + K_{Ii} \int e_i \, dT \tag{3.68}$$

Avec ; $e_i = q_{id} - q_i$: L'erreur sur la position articulaire.

K_{Pi} K_{Di} et K_{Ii} : Les gains respectivement de l'action, Proportionnelle, dérivée et intégrale. Ces gains peuvent être déterminés à l'aide des méthodes d'ajustement notamment la méthode de *Ziegler-Nichols.*

Bibliographie

[1] Wissama Khalil. Commande des bras manipulateurs. *Hermes Science*, 2002.

[2] DELTA TAU. Turbo pmac training. *Delta Tau Data Systems, Inc*, 2008.

[3] Marco Antonio Perez Cisneros. Intelligent models structures in visual servoing. *Thése de Doctorat,Institue of Science and Technology WMIST*, 2004.

[4] Ignacio Herrera Aguilar. Commande des bras manipulateur et retour visuel pour des applications à la robotique de service. *mémoire de Magister,Toulouse*, 2007.

[5] Cauchois Cynl. Calibration du capteur de vision omnidirectionnelle :syclop. *Thése DEA,Université de technologie de compiégne*, 1998.

[6] Rabah Ammour-Abderaouf Boussif. Commande référencée vision du robot mobile "robucar". *Thése d'ingéniorat,ENP,Algérie*, 2012.

[7] F Chaumette. De la perception à l'action : l'asservissement visuel ;de l'action à la perception :la vision active. *Thése Habilitation à diriger la recherche,Université Rennes I*, 1998.

[8] Lounis Sadelli-Imad Eddine Aiteur. Commande à base d'un dsp de la nouvelle structure du robot scara à 3dll. *mémoire d'ingéniorat,ENP,Algérie*, 2012.

[9] DELTA TAU. Pmac hardware reference manual. *Delta Tau Data Systems, Inc*, 1998.

[10] Wissama Khalil-Etienne Dambre. Robot manipulators modeling,performance,analysis and control. *ISTE*, 2007.

[11] Raymond Hanus. Automatique avancée 3 :asservissement et commande des robots. *Hermes Sciences publication*, 2007.

[12] Farid Oubbati. Commande d'un bras de robot manipulateur à l'aide d'une carte dspace-ds1104. *Mémoire de Magister,ENP,Algérie*, 2006.

[13] Aljendra Barrera. Advances in robot navigation. *Inteck*, 2011.

[14] Sabah Chemami. Etude des différentes lois de commande pour un robot manipulateur 6dll comportant une liaison prismatique. *Thése Magister,Université Larbi Ben Mhidi,Oum El Bouaghi*, 2009.

[15] Noureddine Ouadah. Le control visuel appliqué dans la robotique mobile. *Thése Doctorat,ENP*, 2011.

[16] Nedjma Zaidi-Hassen Taidirt. Asservissement visuel d'un bras de robot. *Thése d'ingéniorat,ENP,Algérie*, 2006.

[17] S Mechoud. Commande adaptative floue du bras de robot manipulateur puma 560 en position/orientation. *Thése d'ingéniorat,ENP,Algérie*, 2009.

[18] J Angeles. Fundamentals of robotic mechanical systems. *Third edition, Springer Science,Business Media, LLC*, 2007.

[19] DELTA TAU. Turbo pmac user manual. *Delta Tau Data Systems, Inc*, 2008.

[20] S Ben Chabane-O Douaidi. Commande par dsp de l'umac de delta tau d'un robot traceur de type scara. *Thése d'ingéniorat,ENP,Algérie*, 2011.

[21] DELTA TAU. Delta tau software catalogue. *Delta Tau Data Systems, Inc*, 2008.

[22] DELTA TAU. Pmac quick reference. *Delta Tau Data Systems, Inc*, 2002.

[23] François Chaumette-Bernard Espiau. A new approch to visual servoing in robotics. *IEEE*, 1992.

[24] IS. Initiation au traitement d'images avec matlab. *Elément d'un cours d'infirmatique scientifique,ENSBANA*, 2003.

[25] S Philipp J.P.Cocquerez Massons. Analyse d'images :filtrage et segmentation. *Elément d'un cours*, 1995.

[26] J Laumand. La robotique mobile. *Edition Hermés*, 2001.

[27] V Rouilly. Detection d'ellipce par transformation de hough. *http ://www.tsi.enst.fr/*, 2000.

[28] Antoine Manzanera. Traitement d'image et vision artificielle. *Elément de cours,ENSTA,Unité d'électrotechnique*.

[29] ecole des mines. Analyse d'image. *Elément d'un cours, Génie des Procédés , centre SPIN, Ecole des Mines de Saint-Etienne*, 2001.

[30] R K. Réalisation d'une application logiciel de commande pour saisir des objets dans un système robotique. *thése d'ingeniorat,institut nationale d'informatique,Algerie, OPTkey = RYME*.

[31] L.E Weiss. Dynamic visual servo control of robots :an adaptative image-based approach. *thése PhD, Université de Carnegie Mellon*, 1984.

[32] F Chaumette. La relation commande-vision : théorie et application à des taches robotiques. *Thése de doctorat,Univérsité Rennes I*, 1990.

[33] C Samson-M le borgne B Espiau. Robot control :the task function approach. *Clarenden Press,Oxford*, 1991.

[34] Tak Auyeung. Robot programming in c. *Thése PhD*, 2006.

[35] Thomas Kiesche. Hand-eye calibration with a stereo caméra. *Thése de Doctorat,Aalborg University Esbjerg*, 2006.

[36] E Malis-F Chaumette. 2 1/2 d visual servoing. *Transaction on robotics and automation vol.15, no.4,p.238-250*, 1999.

[37] Alain Boucher. Vision par ordinateur :paramétres de la camera et calibration. *Element d'un cours*, 2012.

[38] Too Li. Commande d'un robot de télé-échographie par asservissement visuel. *Thése de Doctorat,Université de Rennes 1*, 2013.

[39] F Reyes-R Kelly. Experimental evaluation of fixed-camera direct visual controller on a direct drive robot. *IEEE International Conference on Robotics and Automation,Vol.3 p 2327-2332*, 1998.

[40] K Hashimoto. Visual servoing :real time control of control robot manipulators based on visual sensory feedback. *World Scientific series in Robotics and Automated systems ,Vol.07*, 1993.

[41] N Papanikolopoulos P 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.09,no.1,p 14-35*, 1993.

[42] K Hashimoto-T Ebine H Kimura. Visual sevoing with hand-eye manipulator-optimal central approach. *IEEE transactions on Robotics and Automation,Vol.12,no.5,p 766-774*, 1996.

[43] D Khadraoui-R Rouveure Debain-P Martinet-P Bonton-J Gallice. Vision-based control in driving assistace of agricultural vehicles. *International journal of robotics Research,Vol.17,No.10,p 1040-1054*, 1998.

[44] B Espiau-F Chaumette P Rives. A new approach to visual servoing in robotics. *IEEE transactions on Robotics and Automation,Vol.08,no.3,p 313-326*, 1992.

[45] F Chaumette-S Nutchinson. Visual servo control,part i :basic approaches. *IEEE transactions on Robotics and Automation,Vol.12,no.5,p 82-90*, 2006.

[46] Michel Etique. Regulation automatique. *Elément d'un cours,HEIG-Vd*, 2006.

[47] M Nougaret. Principes généraux de correction. *Techniques de l'ingenieur ,R.7405.*

[48] M TAdjine. commande par mode de glissement. *Elément d'un cours,Automatique Avancée,ENP*, 2013.