République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud MAMMERI, Tizi-Ouzou

Faculté de Génie Electrique et d'Informatique
Département d'Automatique

Mémoire de Fin d'Etudes

En vue de l'obtention du diplôme

D'Ingénieur d'Etat en Automatique

Thème

Conception et réalisation d'un bras manipulateur à six degrés de liberté autonome assisté par la vision artificielle

Proposé par

Dirigé par : **Mr Mellah Rabah**

Présenté par :

Mr OUKACI Slimane
Mr OUAZAR Yahia

Soutenu le : 07 / 07 /2011

Promotion 2011

Résumé

Le travail présenté dans ce mémoire concerne la conception et réalisation d'un robot à six degrés de liberté dont l'actionnement des servomoteurs qui contrôlent les différentes articulations, peut être commandé par ordinateur ou assisté par la vision. On a commencé par une présentation générale du robot et donner ses modélisations géométriques directe et inverse, et présenter les actionneurs et différents capteurs utilisés avant d'entamer la commande assistée par la vision et enfin exposer les résultats obtenus après plusieurs essais expérimentaux effectués sur le robot.

Remerciements

Nous tenons à remercier en premier lieu « ALLAH » Le Tout Puissant, Qui nous a donné la force, le courage et la volonté pour mener à bien ce modeste travail.

Nous tenons à exprimer notre gratitude à Monsieur Rabah Mellah pour son suivi.

Nos remerciements vont aussi à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Dedicaces

Je dédie ce modeste travail

A mes très chers parents

A mes chers frère et sœurs

A toute ma famille

A mon binôme

A tous mes amis

A tous ceux qui me sont chers

Slimane

Je dédie ce modeste travail

A ma très chère Grand-mère (YAYA)

A la mémoire de mon Grand-père

A mes très chers parents

A mes chères sœurs

A toute ma famille

A mon binôme

A tous mes amis

A tous ceux qui me sont chers

Yahia

Sommaire

<u>Chapitre III</u> : Modélisation géométrique

Chapitre IV : Les actionneurs et capteurs utilisés

Chapitre V: Vision par ordinateur

Liste des tableaux

Chapitre I Présentation du robot

Chapitre III Modélisation géométrique

Chapitre IV Actionneurs et capteurs

Liste des figures

Chapitre I Présentation du robot

Chapitre II La structure mécanique

Chapitre III Modélisation géométrique

Chapitre IV Les actionneurs et capteurs

Chapitre V Vision par ordinateur

Chapitre VI Applications et commande

Introduction

En 1921, Karel Capek a introduit le mot «robot» dans son livre « Rossum's Universal Robots ». Le terme a été ensuite utilisé par Isaac Asimov dans un court récit publié en 1942. Durant les années 1940 et 1950, deux évolutions majeures ont permis la conception de robots modernes : la commande numérique et les téléopérateurs.

Les premiers téléopérateurs ont été développés au début des années 40s, La Commande numérique a été inventée à la fin des années 1940 et début des années 1950. Il s'agit d'une méthode de contrôle des axes de machine-outil au moyen d'un codage numérique. La première machine à commande numérique a été présentée en 1952 au Massachusetts Institute of Technology (MIT), dont les suivantes recherches ont abouti à l'élaboration d'un langage de programmation pour machines-outils (APT) qui a été conçu pour être utilisé dans la fabrication assistée par ordinateur (FAO).

En 1961, le premier robot industriel nommé UNIMATE a été installé à General Motors, développé par Joe Engelberger et George Devol,

Aujourd'hui, les bras robotiques sont largement utilisés dans la fabrication industrielle, ceci a permis sans aucune doute que leurs capacités et leurs performances ont augmenté grâce à l'amélioration des mécanismes, des contrôleurs, des développements de logiciels, des capteurs, des systèmes d'entraînement, et des matériaux utilisés.

Les différentes définitions d'un robot selon les normes JIRA, RIA et ISO

• Définition d'un robot industriel par la JIRA (Association Japonaise de Robotique Industrielle)

Un système capable d'accomplir des tâches, en tout ou en partie, habituellement dévouées aux humains.

• Définition d'un robot industriel par la RIA (Robot Institute of America)

Un manipulateur reprogrammable multi fonctionnel conçu pour déplacer des matériaux, des pièces, des outils ou tout autre dispositif spécialisé au moyen d'une série de mouvements programmés et d'accomplir une variété d'autres tâches.

• **Définition selon l'ISO (International Standard Organization)**

Une machine formée par un mécanisme incluant plusieurs degrés de libertés, ayant souvent l'apparence d'un ou plusieurs corps terminant par un poignet capable de tenir des outils, des pièces ou un dispositif d'inspection.

I-1- Présentation générale du robot I-Kernel

Le robot que nous avons conçu et réalisé dans le cadre de ce projet, est un bras à six degrés de liberté de classe D (Autonome) à servomoteurs RC dont les articulations sont de type rotoïde.

Le projet est composé de deux parties : le corps du robot et son système de commande.

I-1-1- Pourquoi le nom I-Kernel ?

I-kernel est un mot composé de deux parties :

I : représente la première lettre du mot Intelligent.

Kernel : c'est la traduction anglo-saxonne du mot noyau.

Le tout veut dire que le robot agit avec un certain niveau d'intelligence.

I-1-2-Le corps du robot

Le bras manipulateur est le composant mécanique du système et comporte six axes de rotation, dont chacun contrôlé par un servomoteur à courant continu associe à un potentiomètre, comme la montre la figure ci-dessous.

En outre, le robot est aussi équipé de deux caméras CMOS et d'une pince à commande électrique comportant deux capteurs QTC de pression.

Figure 1-1 Robot I-Kernel

I-1-3-Le système de contrôle

Notre système de contrôle est composé d'un ordinateur muni d'un logiciel d'interfaçage et une carte de commande des servomoteurs nommée Arduino Méga.

Toutes les informations appropriées aux entrées/sorties des différents actionneurs du robot sont traitées par ce système de commande doté d'un logiciel I-Kersoft et d'une carte de commande. Le logiciel I-Kersoft reçoit les informations venant des deux caméras, pour synthétiser en temps réel les ordres de commande à transmettre via un port USB vers la carte de commande en vue de commander en PWM les actionneurs.

I-2- Caractéristiques techniques

Dans cette partie, on trouve toutes les principales caractéristiques du robot et ses capacités à communiquer avec l'environnement extérieur.

I-2-1- Caractéristiques du bras manipulateur

Axes	6 axes de révolution
Motorisation des axes	Servomoteurs RC a courant continu
Respectabilité en position	± 0.1 mm
Commande de l'effecteur	commande électrique (2 positions)
Limites angulaires des articulations	Voir figure ci-dessous
Poids	3 kg
Capteur de pression	QTC
Alimentation électrique pour chaque actionneur	6V/1.5A DC
Camera	Webcams CMOS Logitech 1.3 Méga pixel port USB à focal manuel
Matériaux	Aluminium et plexiglas

Tableau 1-1 Caractéristiques du bras manipulateur

A partir de la structure mécanique, on peut placer les repères des articulations du robot conformément à la convention de Denavit-Hartenberg, comme le montre la figure ci-desous.

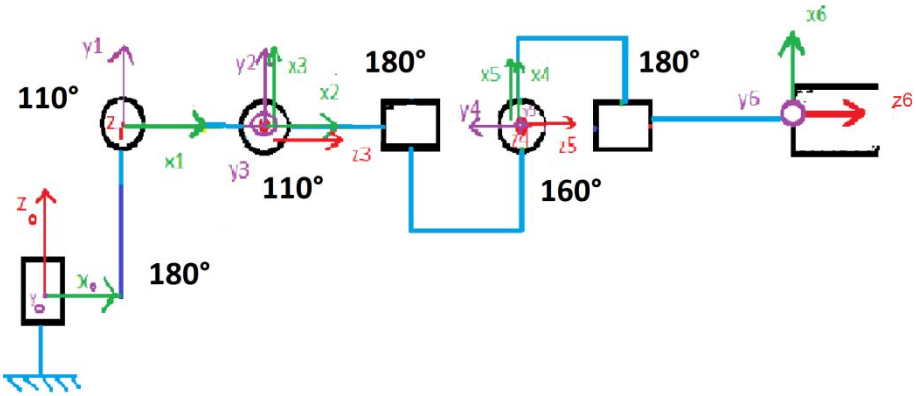


figure1-2 amplitudes de chacun des axes.

I-2-2- Caractéristiques du système de commande

Le système de commande est composé de deux parties : l'ordinateur et une carte de commande des servomoteurs.

I-2-2-1- Ordinateur

Microprocesseur	Intel Pentium Dual Core cpu T3400 2.17 GHz
Type de l'OS	32bits
RAM	4Go
Carte graphique	NVIDIA GeForce 9200M GE 512 GPU
Système d'exploitation	Linux ubuntu 10.04
Langage de programmation	C/C++/JAVA
E/S	Ports USB

Tableau 1-2 Caractéristiques du système de commande : ordinateur

I-2-2-2- Carte de commande des servomoteurs

Il s'agit d'une carte à base de microcontrôleur Atmel très utilisée dans le domaine de la robotique pour son aptitude à commander des actionneurs tels que les servomoteurs et les moteurs pas à pas. Ses caractéristiques techniques sont représentées dans le tableau suivant :

Microcontrôleur	Atmel ATMega 1280
Tension de fonctionnement	5 V
Tensions de sortie recommandées	7-12 V
Tensions de sortie limite	6-20 V
Broches digitales E/S	54 (dont 14 fournissent une PWM en sortie)
Broches analogique d'entrées	16
Courant continu par broche E/S	40 mA
Courant DC pour les broches 3.3 V	50 mA
Mémoire flash	128 KB dont 4 KB employés par le Boot loader
SRAM	8KB
EPROM	4KB
Fréquence d'horloge	16 MHz

Tableau 1-3 Caractéristiques de la Carte de commande des servomoteurs

I-3- Domaine d'utilisation du robot I-Kernel

D'après la description donnée auparavant, le robot I-Kernel peut effectuer plusieurs tâches telles que l'assemblage, le soudage, la peinture et le tri.

La tâche choisie pour notre robot est le tri et ce en raison des matériaux utilisés au niveau d'une partie de la structure mécanique à savoir le plexiglas et non pas de l'aluminium qui est trop lourd pour les actionneurs utilisés.

I-4- Fonctionnement du robot

- Le robot est initialisé de telle manière que chaque servomoteur de chaque partie est initialisé à un angle donné comme suit :

  Base : $\theta_1 = 90°$.
  Epaule : $\theta_2 = 90°$.
  Le coude : $\theta_3 = 90°$.
  Poignet : $\theta_4 = 90°$.
  $\theta_5 = 0°$.
  $\theta_6 = 90°$.

Ce qui nous donne la position (0, 0, Z) des deux caméras dans l'espace de tâches, comme c'est montré dans la figure ci-dessous :

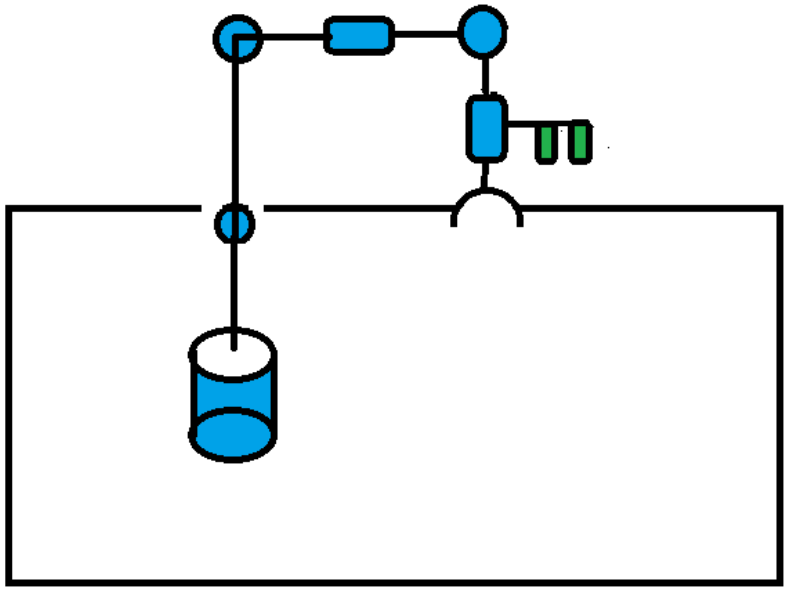


Figure 1-3. État initial du robot

- Les deux caméras vérifient si l'objet désiré est dans le champ d'atteignabilité.

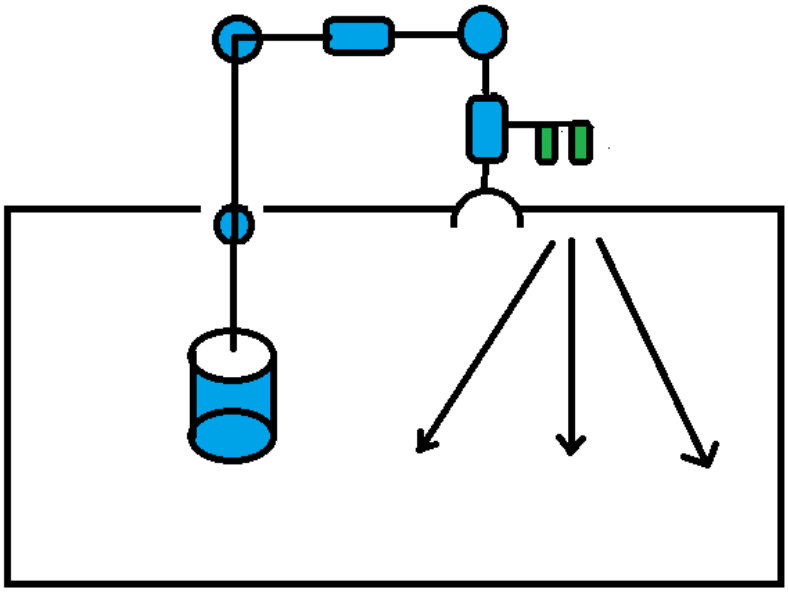


Figure 1-4 Recherche de l'objet

- Une fois l'objet détecté, l'intervention de la stéréovision nous permet d'extraire les paramètres X, Y, Z de l'objet.

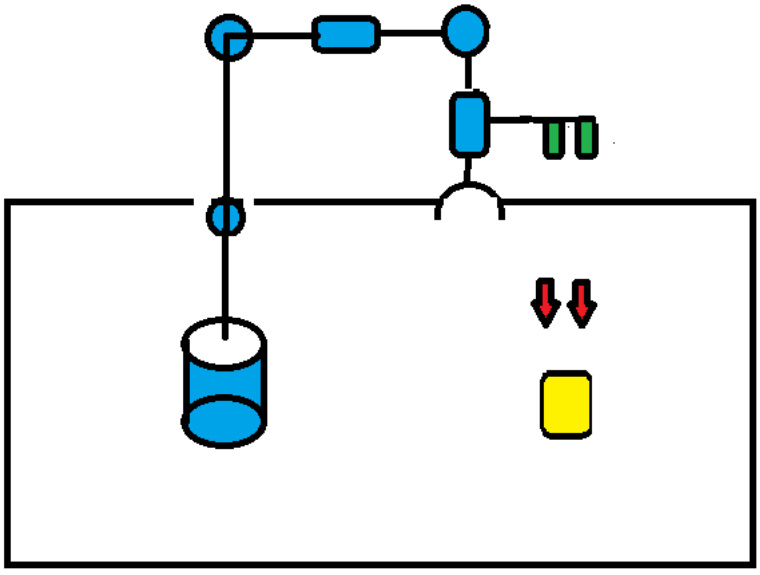


Figure 1-5 Objet détecté

- Les paramètres extraits sont envoyés vers la carte de commande.

- Grâce au modèle géométrique inverse, les paramètres précédents sont traduits en angles à effectuer par chacun des servomoteurs pour que la pince atteigne la position souhaitée.

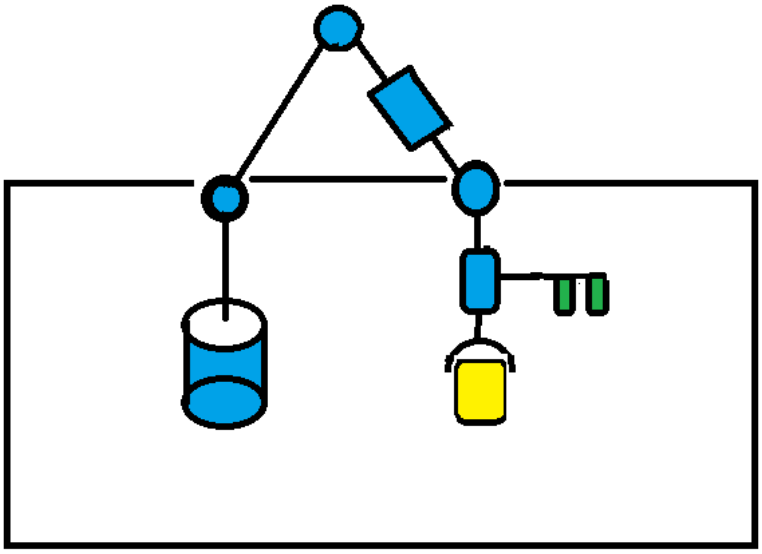


Figure 1-6 Cible atteinte par le robot

- Une fois que la pince atteint la position de l'objet, elle se ferme jusqu'à ce que les deux capteurs à ses extrémités envoient une valeur préprogrammée.

- Le robot revient à sa position initiale en prenant l'objet.

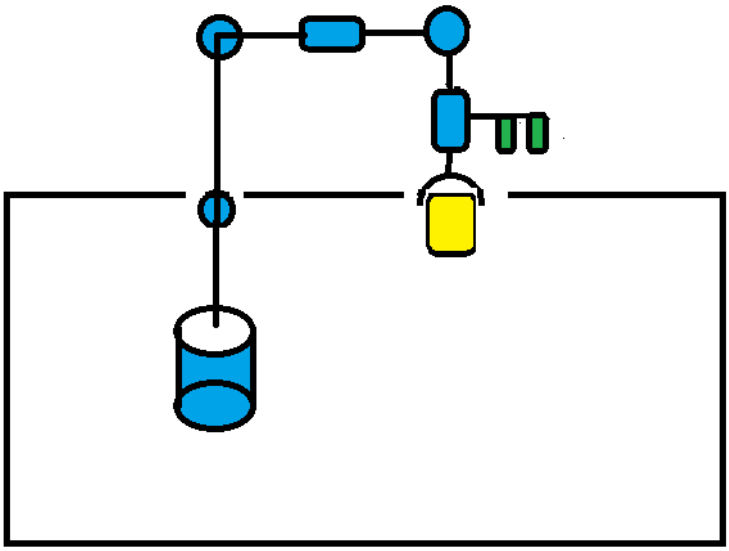


Figure 1-7 Positon initial on prenant l'objet

- Le robot dépose l'objet à un endroit préprogrammé et revient à sa position de départ.

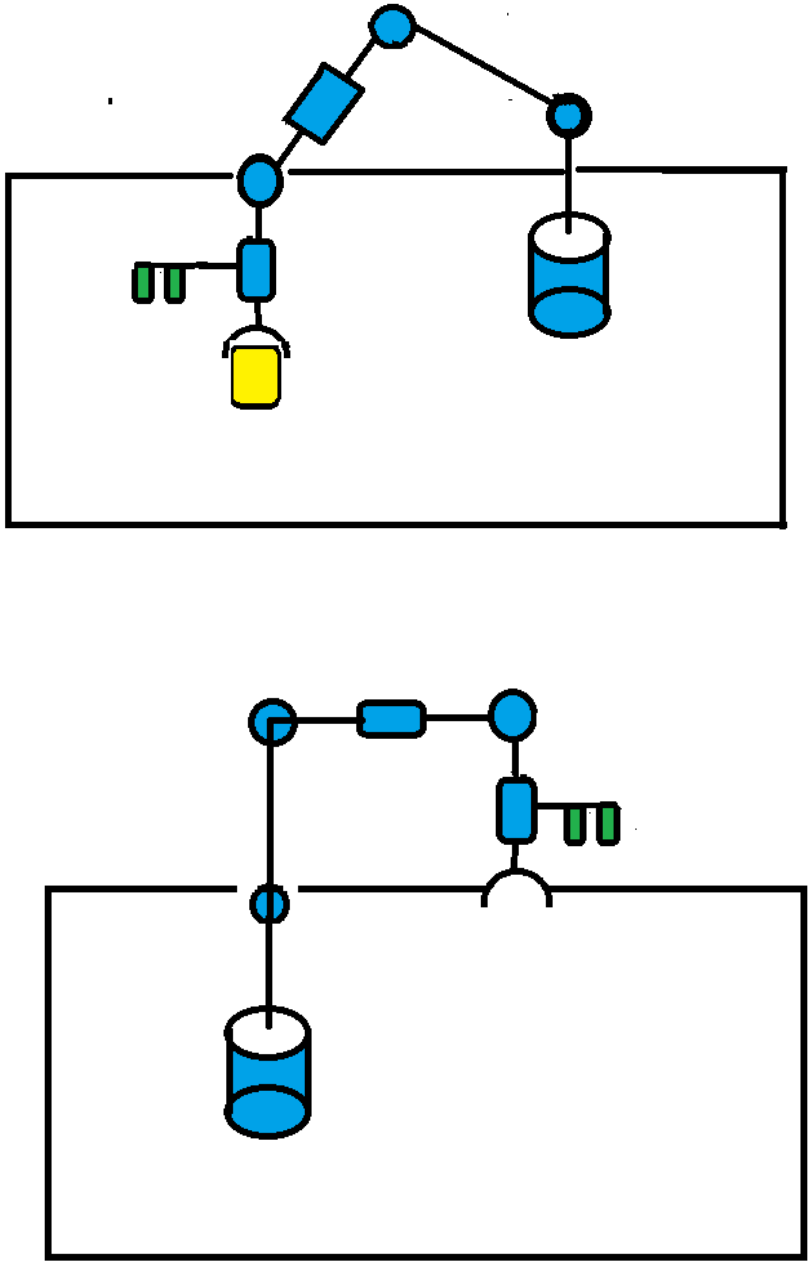


Figure 1-8 Phase finale de la tache

II-1- Introduction

Tous les robots partagent les dispositifs d'une structure mécanique sous une certaine forme, la structure d'un bras manipulateur à six degrés de liberté (anthropomorphe) est mécanique et peut s'appeler une chaîne cinématique (sa fonctionnalité est d'apparenter au squelette d'un bras humain). Cette chaîne est constituée des segments (ses os), des actionneurs (ses muscles) et d'articulations qui sont reliées à celles qui les précédent et a celles qui les succèdent. En effet, la structure mécanique constitue le lien physique entre un repère de référence et un repère de tâches.

II-2- Structure mécanique du robot I-Kernel

II-2-1- Matériaux utilisés

Nous avons opté pour l'Aluminium en raison de sa résistance à l'oxydation , sa faible densité et sa facilitée d'usinage, ainsi que le plexiglas comme solution de remplacement sur certaines parties de la structure mécanique.

II-2-2- Structure mécanique articulée

Nous avons conçu une structure mécanique conformément à un modèle de 6 degrés de liberté dont l'image et le modèle CAO sont donnés par les figures ci-dessous.

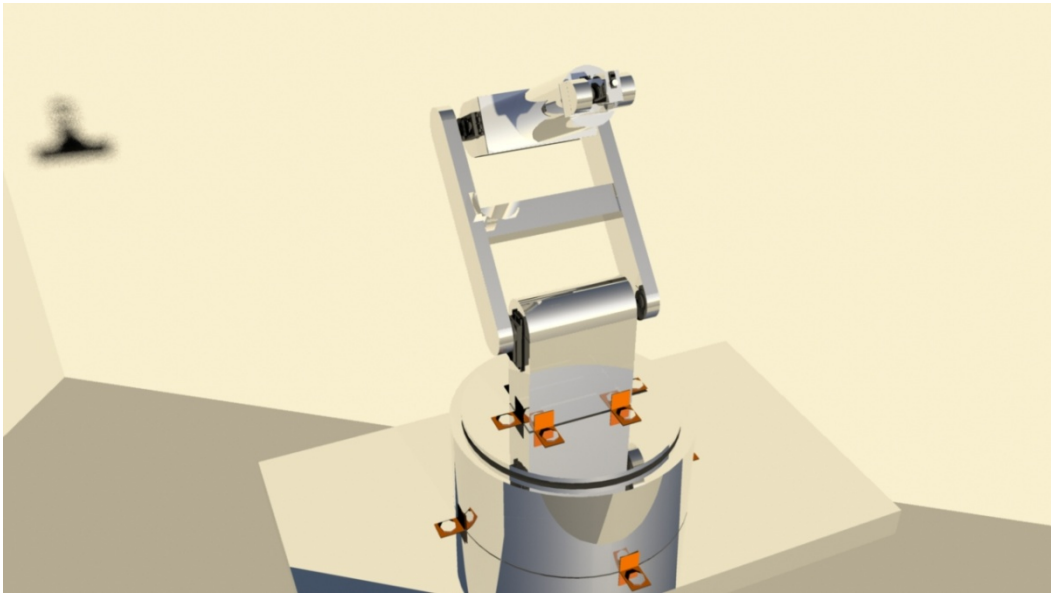


Figure 2-1 Modèle CAO du robot vue perspective 1

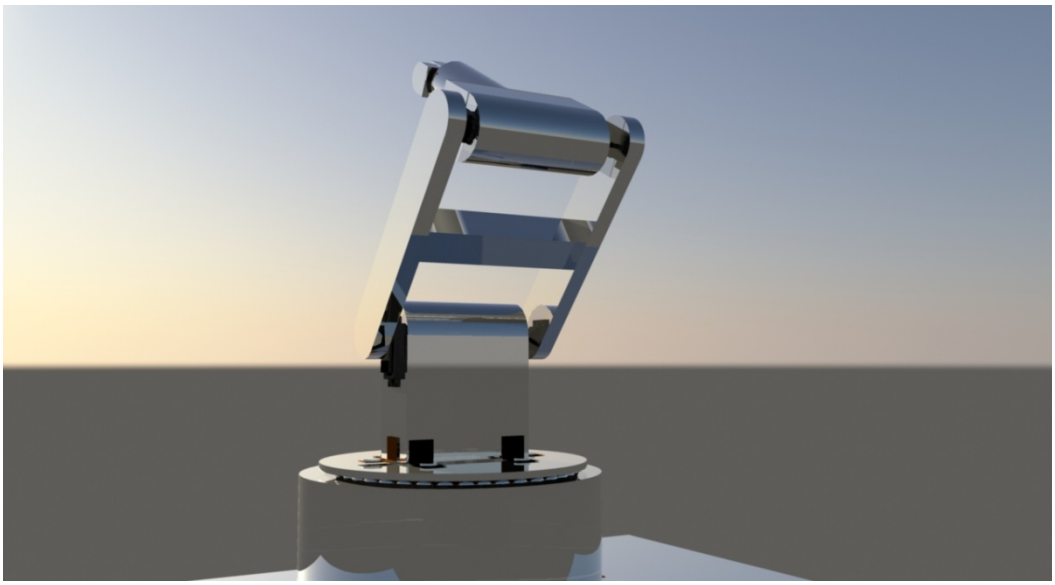


Figure 2-2 Figure 2-1 Modèle CAO du robot vue perspective 2

II-2-2-2- Structure mécanique

En tenant compte des contraintes mécanique à savoir le poids de la structure en aluminium nous avons opéré des changements sur les matériaux de façon à adapter la structure aux capacités des actionneurs utilisés. Ainsi l'image du robot modifié est montrée ci-dessous.

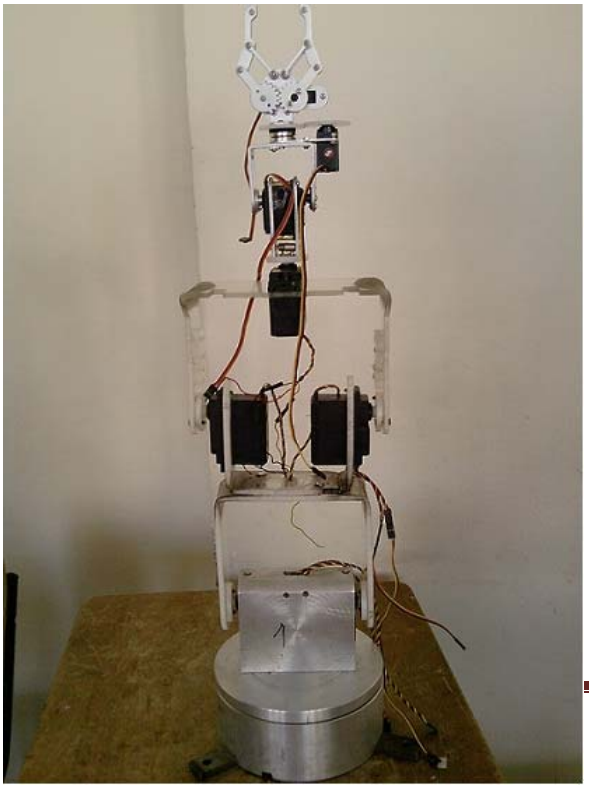


figure 2-4 Nouvelle structure mécanique du Robot I-Kernel Vues de face et de profile

II-3-Différentes parties du robot

Le bras est constitué de plusieurs parties dont toutes les articulations sont rotoides :

II-3-1-La base : elle est composée de deux parties. Une partie fixe et une partie mobile.

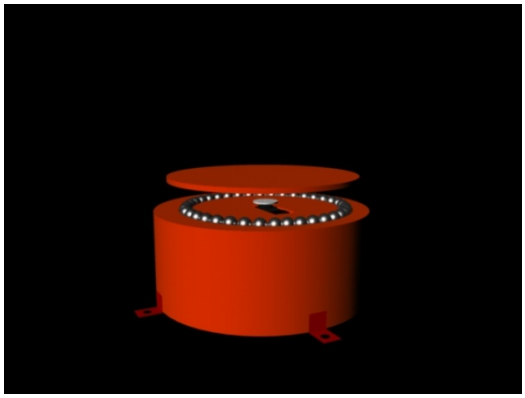


Figure 2-4-1 Modèle CAO de la base complète

II-3-1-1- La partie fixe

C'est un cylindre de diamètre et de hauteur sont respectivement 180 mm et 110mm, supporte l'ensemble du bras. Pour accueillir des billes et créer un système de roulement qui permet d'équilibrer le palonnier de l'actionneur tout en évitant les frottements avec la partie mobile, une cavité a été usinée de telle manière à avoir une ouverture pour le passage du servomoteur HS-805BB de dimension 66x34 mm et de façon à ce que l'arbre du moteur vienne exactement au centre de la pièce, comme le montre les figures ci-dessous.

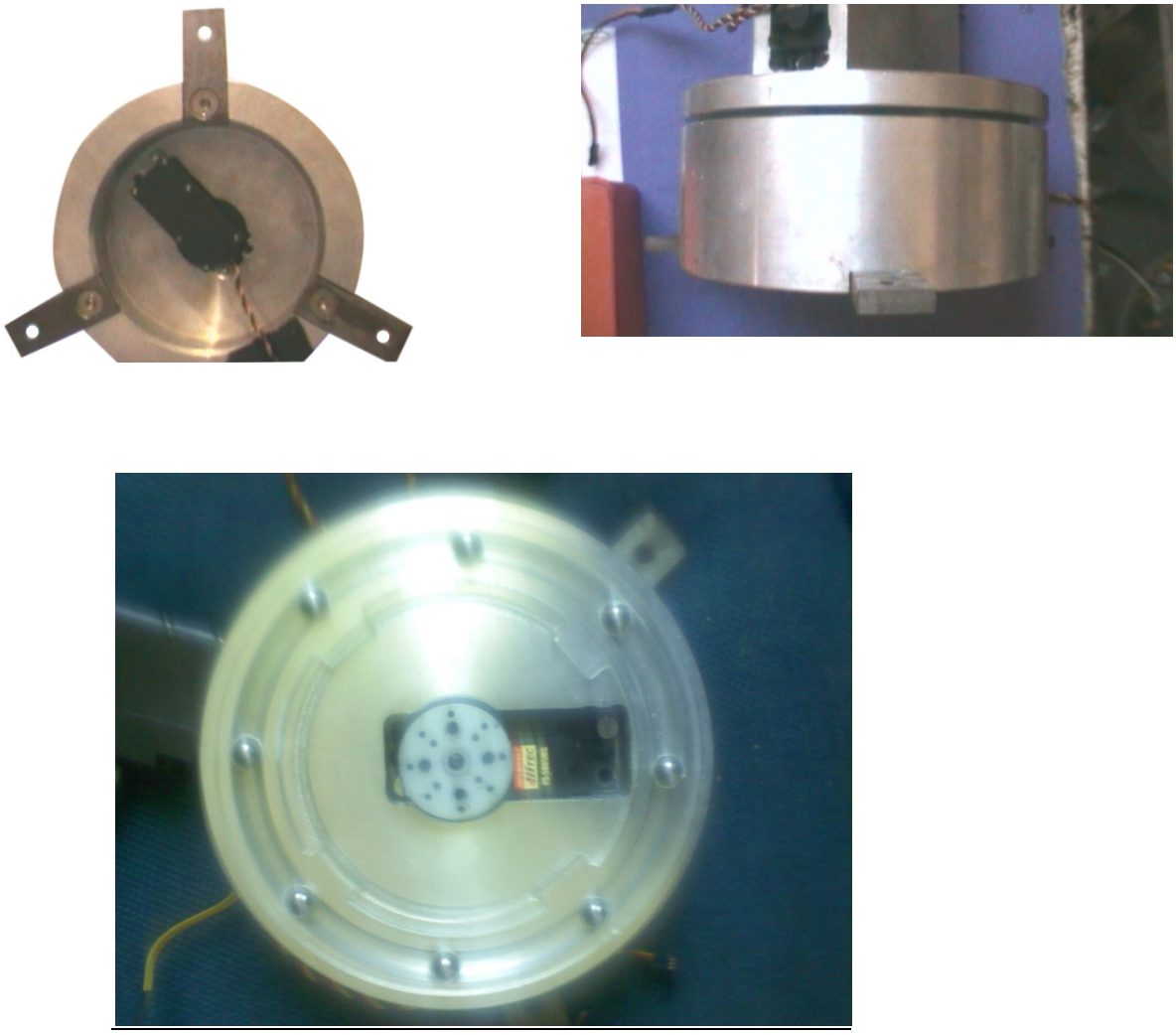


Figure 2-5 Vues de dessous, de face et de dessus de la base fixe du robot

II-3-1-2- La partie mobile

Il s'agit d'une base tournante sous forme d'un disque de diamètre 180mm et d'une épaisseur de 9mm, qui se pose sur les billes de la base fixe et se relie à l'actionneur de cette dernière à l'aide de visses accueillies dans des taraudages pour ne pas gêner l'épaule qui se pose dessus. La figure suivante illustre les deux vues de et dessus et de dessous de la base tournante.

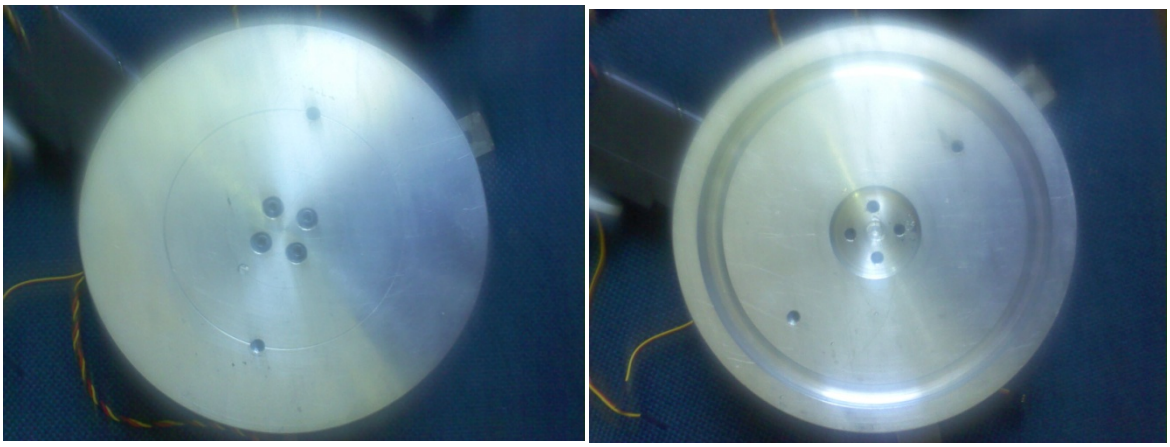
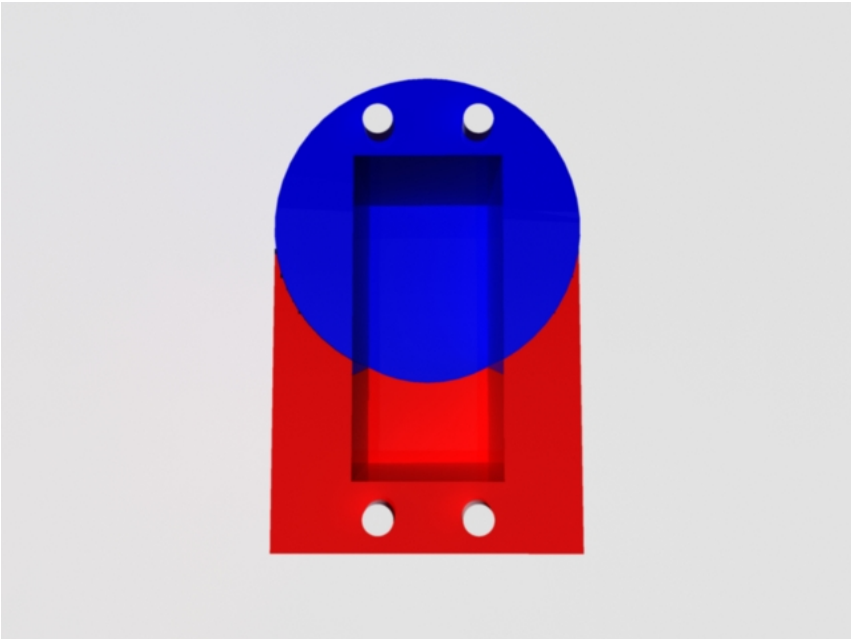


Figure 2-6 Vues de dessus de dessous de la base tournante du robot

II-3-2- L’épaule

C’est une pièce usinée en aluminium de 119x97x51 mm dont l’utilité est de fixer les servomoteurs de manière à ce qu’ils soient bien stables et leurs arbres bien parallèles. Les figures ci-dessous montrent les vues appropriées à l’épaule.



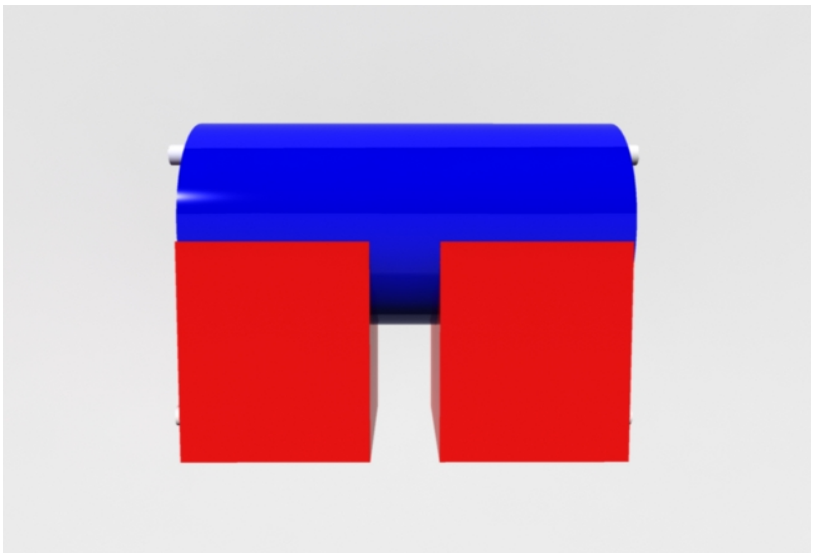


Figure 2-7 Vues de profile et de face du Modèle CAO de l'épaule

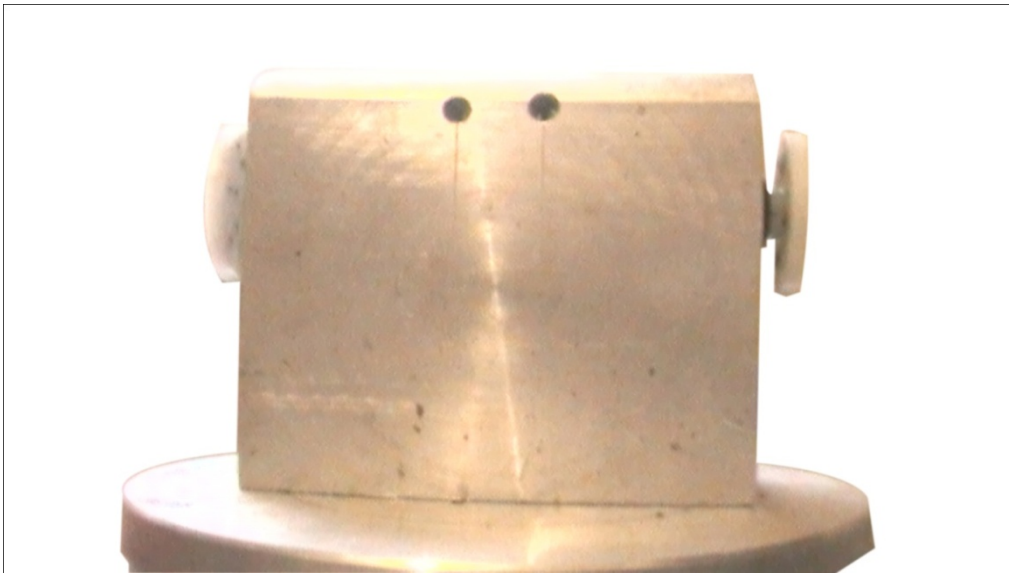


Figure 2-8 Vues de profile et de face de l'épaule

II-3-3- Les bras

C'est une pièce en plexiglas formée par deux parties en U de largeurs respectivement 155mm et 125 mm. Ces deux parties sont disposées en forme de H d'une longueur de 270 mm et d'épaisseur de 6mm et dont les extrémités inferieures sont fixées aux palonniers de l'épaule et les deux extrémités supérieures supportent les palonniers des deux servomoteurs du coude.

Figure 2-9 Vue de face des deux bras

II-3-4- Le coude

C’est une pièce en plexiglas en forme de U de dimensions 190x150 mm dont les deux extrémités sont fixées aux palonniers des servomoteurs, une encoche a été créée dans sa base pour accueillir le servomoteur de façon à ce que l’axe de ce dernier soit perpendiculaire à l’axe de l’articulation qui précède cet actionneur, ce fait le début du poignet, comme indiqué par les figures suivantes.

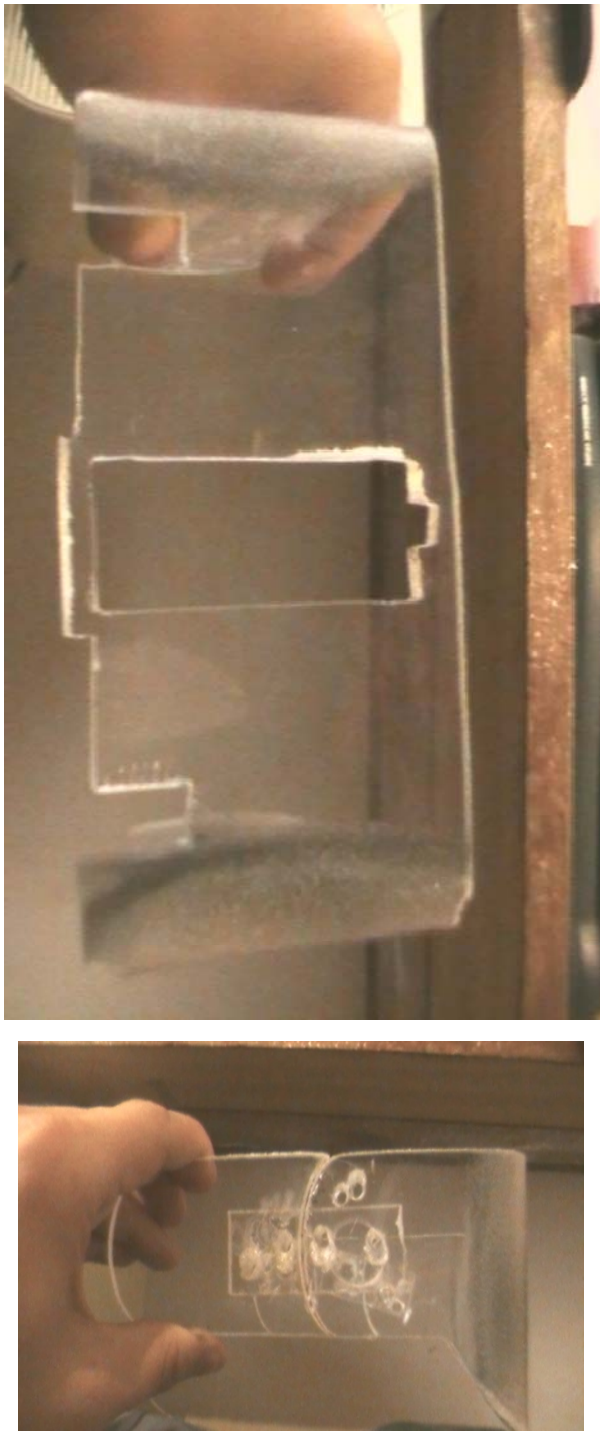


Figure 2-10 Vues de face et de profile du coude

II-3-5- Le poignet

C’est une pièce commercialisée par LynxMotion, qui est composée de trois parties articulées en aluminium dont les axes sont concourants pour former la rotule.

La première partie représentée par la figure ci-dessous, est composée de deux barres rectangulaires de mêmes dimensions ayant des supports qui assurent leur parallélisme. Elle est dotée d’une encoche où vient s’encastrer le premier servomoteur se dimensions sont de 73x23 mm.

Figure 2-11 Vue de dessus de la première partie du poignet

La deuxième partie est sous la forme <U> de 62x59 mm, dont les deux extrémités son fixées aux palonniers du servomoteur précédent, comme le montre la figure suivante :

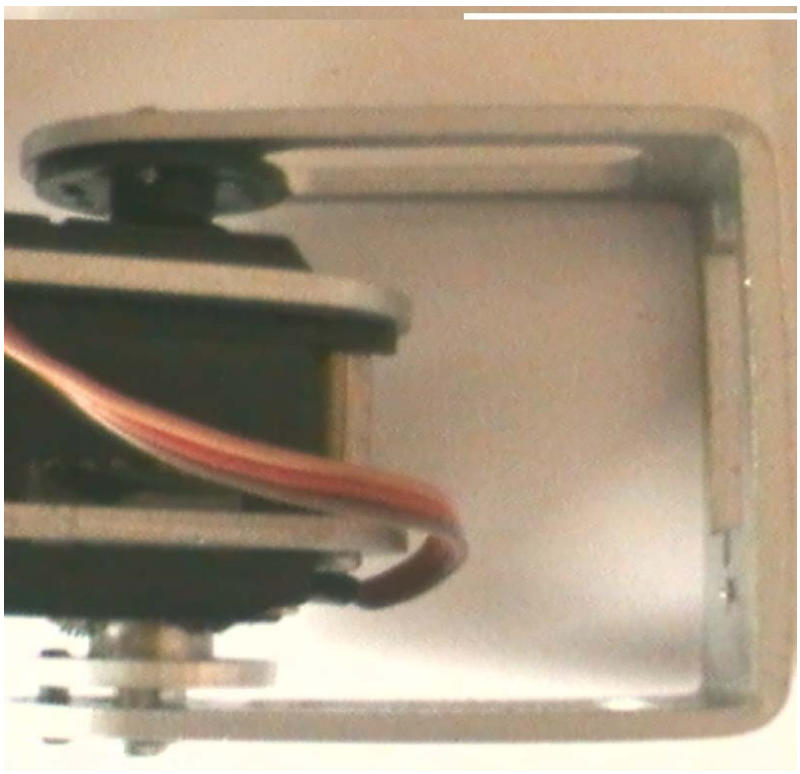


Figure 2-12 Vue de dessus de la deuxième partie du poignet

La troisième partie a une forme particulière comme l'indique cette figure :

Figure 2-13 Vue de face de la troisième partie du poignet.

Elle est munie d'un roulement fixé dessus pour être entraînée par l'engrenage du servomoteur.

II-3-6- L'organe terminal

C'est pince commercialisée par DAGU à actionnement électrique à fermeture complète et ouverture maximale de 50mm. Son rôle est de saisir l'objet désiré pour l'application de sa tâche.

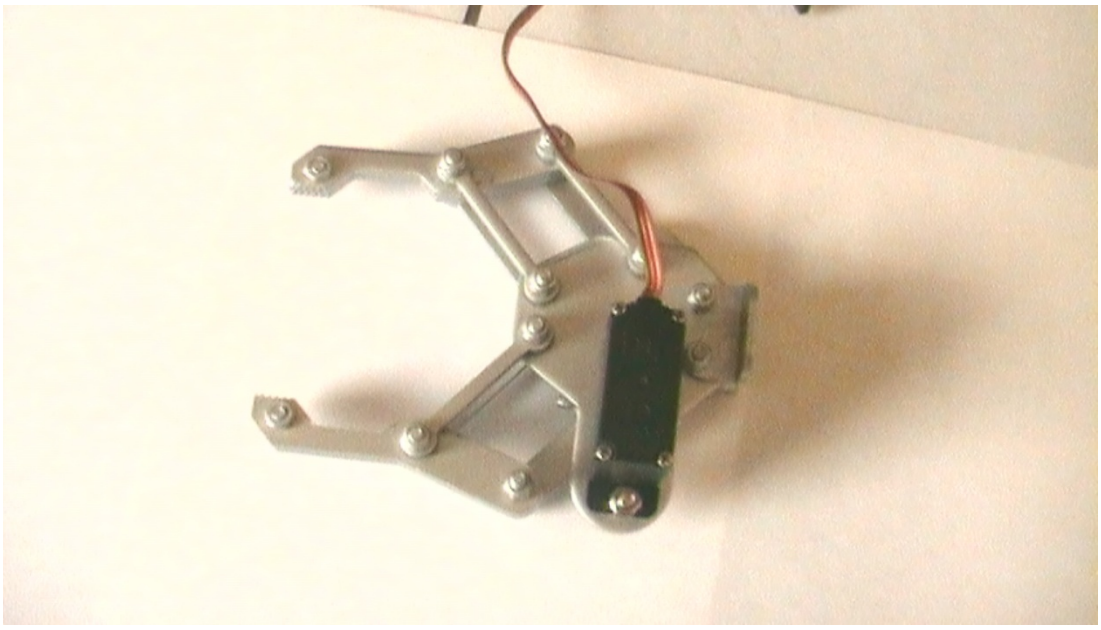


Figure 2-14 Vue de dessous de la pince

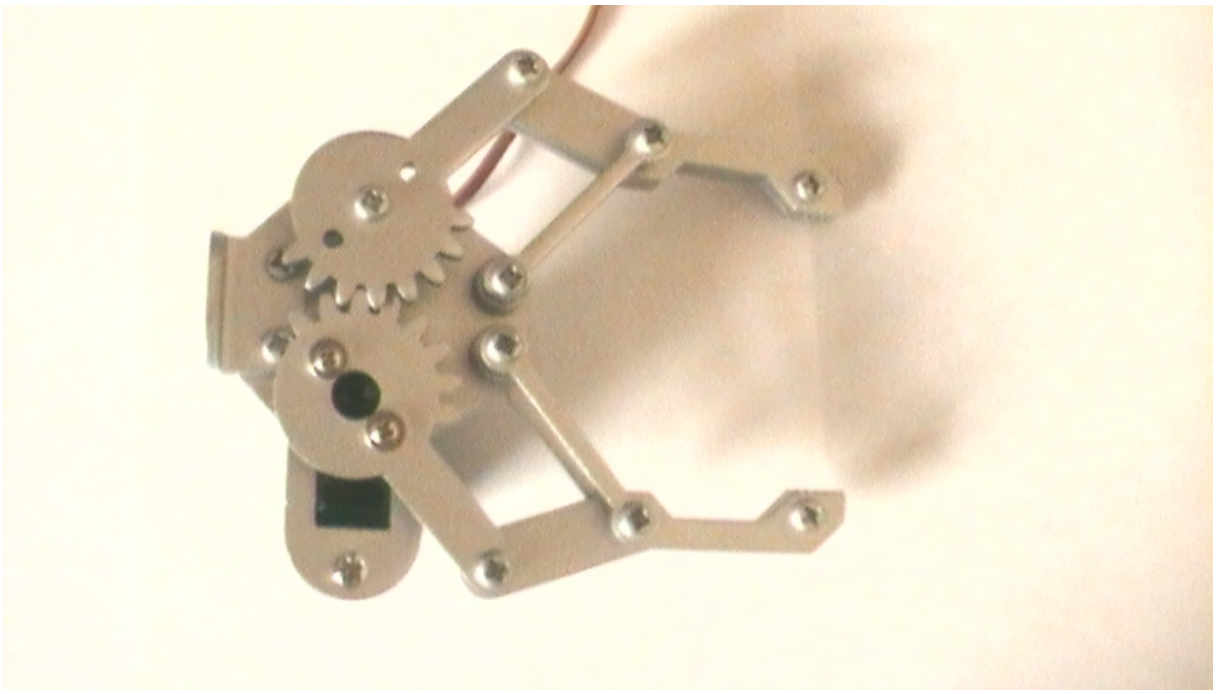


Figure 2-15 Vue de dessus de la pince

II-3-7- Les supports de webcams

C'est une pièce conçue pour fixer les deux webcams perpendiculairement à la 2eme partie du poignet de manière à pouvoir optimiser le champ de vision.

II-4- Conclusion

Ce que nous pouvons conclure à l'issue de ce chapitre, c'est que la robotique est une science multidisciplinaire nécessitant la collaboration de plusieurs champs d'expertise comme le génie mécanique, le génie électrique, l'informatique et les sciences fondamentales, une collaboration grâce à laquelle la robotique est désormais partout, quant au robot I-Kernel, le commander et parvenir à lui faire exécuter des tâches passe par des étapes cruciales en l'occurrence la modélisation et la maîtrise de son système de contrôle, ce qui sera l'objet des prochains chapitres.

III-1- Introduction

La commande du robot nécessite le calcul du modèle géométrique directe (MGD) ainsi que le modèle géométrique inverse (MGI).

Le modèle géométrique directe décrit les situations de l'organe terminal en fonction des variables articulaires du robot, et inversement le calcul du modèle géométrique inverse revient à exprimer les coordonnées articulaires du robot en fonction des coordonnées opérationnelles.

III-2- Modèle géométrique directe du robot I-Kernel

III-2-1- Définition

Etant données les positions articulaires (distance, respectivement angle pour une articulation prismatique respectivement. rotoïde), trouver l'attitude de l'organe terminal par rapport à la base.

Son calcul consiste à exprimer le vecteur des coordonnées opérationnelles X du robot en fonction du vecteur des coordonnées articulaires.

$$X=f(q)$$

III-2-2- Principe

- Fixer des repères à chaque corps du robot.
- Calculer les matrices homogènes entre chaque corps.
- Calculer la matrice homogène entre base et organe terminal.

Le robot est constitué d'un chaînage de *7* corps liés entre eux par 6 articulations rotoïdes. A chaque corps, on associe un repère Ri. Les repères sont numérotés de 0 à *6*. La ième articulation dont la position est notée qi, est le point qui relie les corps i-1 et i.

III-2-3- Convention de Denavit-Hartenberg (DH)

Méthode destinée à systématiser la modélisation de n'importe quel type de robot série. Ses principaux avantages sont :

- Simplification maximale du modèle géométrique.
- Etablissement d'une norme reconnue par tous.

On peut représenter l'attitude d'un repère R_i par rapport à un repère R_{i-1} à l'aide de 4 paramètres uniques à condition de fixer 2 contraintes :

- DH1 : l'axe x_i de Ri est perpendiculaire à l'axe z_{i-1} de R_{i-1}.
- DH2 : l'axe x_i coupe l'axe z_{i-1}.

Décomposition en 4 transformations élémentaires :

1. Rotation autour de z d'un angle θ_i.

2. Translation le long de z d'une longueur d_i.
3. Translation le long de x d'une longueur a_i.
4. Rotation autour de x d'angle αi.

Comme ces transformations sont faites par rapport au repère courant, on a :

$$DH_{i-1,\,i} = R_{\theta i z i-1\,,\theta i)} T_{\theta i z i-1\,,di)} T_{\theta i\, xi\,,ai\,)} R_{\theta i x i\,,\alpha i)}$$

Dans notre cas le schéma de la représentation géométrique et le placement des repères du robot I-Kernel est illustré par la figure ci-dessous :

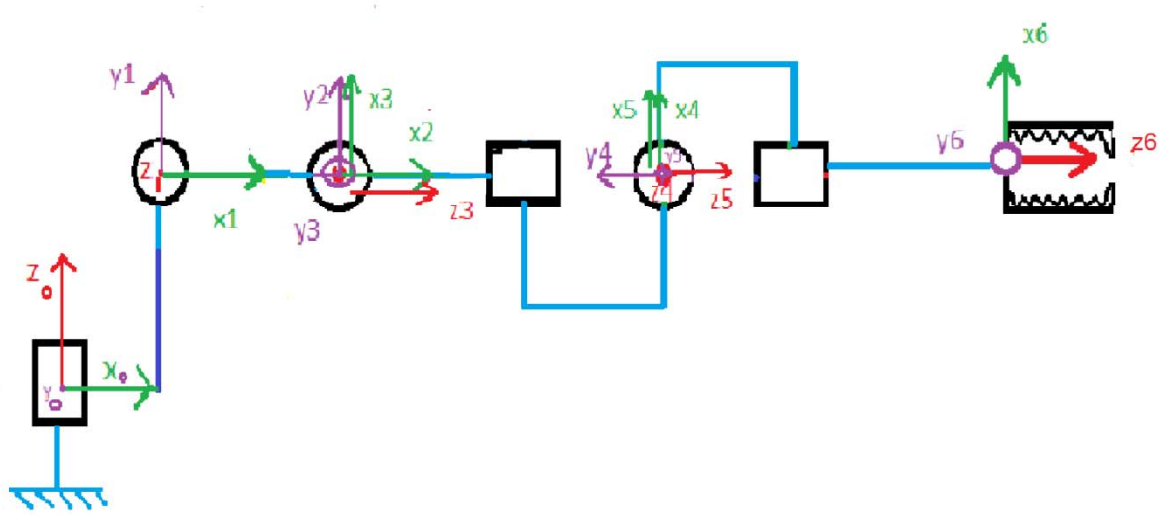


Figure 3-1 Schéma de la représentation géométrique et placement des repères du robot

III-2-4- Calcul du modèle géométrique directe du robot I-Kernel

Le tableau suivant représente tout les paramètres géométriques du robot I-Kernel :

Corps/paramètres	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	12 cm	θ_1
2	6cm	0	0	θ_2
3	0	$\pi/2$	0	θ_3
4	0	$-\pi/2$	15cm	θ_4
5	0	$\pi/2$	0	θ_5
6	0	0	5cm	θ_6

Tableau 3-1. Paramètres géométriques du robot

Afin de faciliter l'implémentation du modèle géométrique et la commande du robot nous optons pour une décomposition comme suit :

1. les 3 premiers axes.
2. les 3 derniers axes = poignet.

III-2-4-1-Les trois premiers axes

Le tableau suivant représente les paramètres géométriques des trois premiers axes du robot I-Kernel :

Corps	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	12cm	θ_1
2	6cm	0	0	θ_2
3	0	$\pi/2$	0	θ_3

Tableau 3-2. Paramètres géométriques des 3 premiers axes

$$DH_{i-1,i}=\begin{bmatrix} c\Theta_i & -s\Theta_i c\alpha_i & s\Theta_i s\alpha_i & a_i c\Theta_i \\ s\Theta_i & c\Theta_i c\alpha_i & -c\Theta_i s\alpha_i & a_i s\Theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On a

$$M_{03}=\begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Avec:

$r_{11}= ½*C(-\theta_3+\theta_1-\theta_2)+ ½*C(\theta_3+\theta_1+\theta_2)$
$r_{12}= S(\theta_1)$
$r_{13}= -½*S(-\theta_3+\theta_1-\theta_2)+ ½*S(\theta_3+\theta_1+\theta_2)$
$r_{21}= ½*S(\theta_3+\theta_1+\theta_2)+ ½*S(-\theta_3+\theta_1-\theta_2)$
$r_{22}= -C(\theta_1)$
$r_{23}= -½*C(\theta_3+\theta_1+\theta_2)+ ½*C(-\theta_3+\theta_1-\theta_2)$
$r_{31}= S(\theta_2+\theta_3)$
$r_{32}=0$
$r_{33}= -C(\theta_2+\theta_3)$

$T x= ½*a_2*C(\theta_1-\theta_2) + ½ *a_2*C(\theta_1+\theta_2) +a_1*C(\theta_1)$
$T y= ½*a_2*S(\theta_1+\theta_2) + ½ *a_2*S(\theta_1-\theta_2) +a_1*S(\theta_1)$
$T z=a_2*S(\theta_2) +d_1$

Vérification du modèle pour le cas particulier $\theta_{1,2} = 0, \theta_3 = \pi/2$

$$M_{03}\ (\theta_{1,2} = 0, \theta_3 = \pi/2)=\begin{bmatrix} 0 & 0 & 1 & a_2 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Vérification du modèle pour le cas particulier $\theta_{2,3}= \pi/2, \theta_1 = 0$

$$M_{03}\ (\theta_{2,3}= \pi/2, \theta_1 = 0)=\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & d_{1+}a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Remarque

Étant donnée la lourdeur des calculs du modèle géométrique, il peut être utile d'avoir recours à un logiciel de calcul symbolique (ex Maple).

Sous Matlab avec le toolbox Symbolic, le calcul de M_{03} s'effectue de la manière suivante:

III-2-4-2- Les trois derniers axes

Les trois axes sont concourants = Rotule.

Le tableau suivant représente les paramètres géométriques des trois derniers axes du robot I-Kernel :

```
>> syms t1 t2 t3 a1 a2 a3 d1

>> M01=[cos(t1) 0 sin(t1) a1*cos(t1); sin(t1) 0 cos(t1) a1*sin(t1);0 1 0 d1; 0 0 0 1]

>> M12=[cos(t2) sin(t2) 0 a2*cos(t2); sin(t2) cos(t2) 0 a2*sin(t2); 0 0 1 0; 0 0 0 1]

>> M23= [cos (t3) 0 sin (t3) 0; sin (t3) 0 cos (t3) 0; 0 1 0 0; 0 0 0 1]

>> M03=simple (M01*M12*M23)

M03 = [1/2*cos (t3+ t1t2) + 1/2*cos (t3+t1+t2), sin (t1) ,1/2*sin (t3+ t1t2) +

1/2*sin (t3+t1+t2) ,1/2*a2*cos (t1t2) + 1/2*a2*cos (t1+t2) +a1*cos (t1)]

[1/2*sin (t3+t1+t2) +1/2*sin (t3+ t1t2), cos (t1) ,1/2*cos (t3+t1+t2) +1/2*cos (t3+ t1t2),

1/2*a2*sin(t1+t2)+1/2*a2*sin(t1t2)+ a1*sin(t1)] [sin(t2+t3),0,cos(t2+t3),a2*sin(t2)+d1]

[0, 0, 0,1]
```

Corps	a_i	α_i	d_i	θ_i
4	0	-π/2	15	θ_4
5	0	π/2	0	θ_5
6	0	0	5	θ_6

Tableau 3-3. Paramètres géométriques des 3 derniers axes

$$DH_{i-1,i}=\begin{bmatrix} c\Theta_i & -s\Theta_i c\alpha_i & s\Theta_i s\alpha_i & a_i c\Theta_i \\ s\Theta_i & c\Theta_i c\alpha_i & -c\Theta_i s\alpha_i & a_i s\Theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On aura donc :

$$M_{3,4}=\begin{bmatrix} C\theta_4 & 0 & -S\theta_4 & 0 \\ S\theta_4 & 0 & C\theta_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{4,5}=\begin{bmatrix} C\theta_5 & 0 & S\theta_5 & 0 \\ S\theta_5 & 0 & -C\theta_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{5,6}=\begin{bmatrix} C\theta_6 & -S\theta_6 & 0 & 0 \\ S\theta_6 & C\theta_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On obtient :

$$M_{36}=\begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Avec :

$r_{11}= C\theta_4 C\theta_5 C\theta_6 - S\theta_4 S\theta_6$

$r_{12}= -S\theta_6 C\theta_4 C\theta_5 - S\theta_4 C\theta_6$

$r_{13}= C\theta_4 S\theta_5$

$r_{21}= S\theta_4 C\theta_5 C\theta_6 + C\theta_4 S\theta_6$

$r_{22}= -S\theta_6 S\theta_4 C\theta_5 + C\theta_4 C\theta_6$

$r_{23}= S\theta_4 S\theta_5$

$r_{31}= -S\theta_5 C\theta_6$

$r_{32}= S\theta_5 S\theta_6$

$r_{33}= C\theta_5$

$T_x = d_6 C\theta_4 S\theta_5$

$T_y = d_6 S\theta_4 S\theta_5$

$T_z = d_6 C\theta_5 + d_4$

III-2-4-3- Le modèle géométrique directe complet du robot I-Kernel

On déduit le modèle géométrique complet du robot : $M_{06}= M_{03}*M_{36}$

$$M_{06}=\begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$r_{11}=(½ C(-\theta_3+\theta_1-\theta_2)+ ½ C(\theta_3+\theta_1+\theta_2)) (C(\theta_4) C(\theta_5) C(\theta_6)-S(\theta_4) S(\theta_6)) +S(\theta_1) (S(\theta_4) C(\theta_5) C(\theta_6) + C(\theta_4) S(\theta_6)) -(-½ S(-\theta_3+\theta_1-\theta_2)+ ½ S(\theta_3+\theta_1+\theta_2)) S(\theta_5) C(\theta_6)$

$r_{21}=(½ C(-\theta_3+\theta_1-\theta_2)+ ½ C(\theta_3+\theta_1+\theta_2)) (- C(\theta_4) C(\theta_5) S(\theta_6)-S(\theta_4) C(\theta_6)) +S(\theta_1) (- S(\theta_4) C(\theta_5) S(\theta_6) + C(\theta_4) C(\theta_6)) +(-½ S(-\theta_3+\theta_1-\theta_2)+ ½ S(\theta_3+\theta_1+\theta_2)) S(\theta_5) S(\theta_6)$

$r_{13}=-(½\ C(-\theta_3+\theta_1-\theta_2)+½\ C(\theta_3+\theta_1+\theta_2))\ C(\theta_4)\ S(\theta_5)-S(\theta_1)\ S(\theta_4)\ S(\theta_5)+(-½\ S(-\theta_3+\theta_1-\theta_2)+½\ S(\theta_3+\theta_1+\theta_2))\ C(\theta_5)$

$T_x=-(½\ C(-\theta_3+\theta_1-\theta_2)+½\ C(\theta_3+\theta_1+\theta_2))\ C(\theta_4)\ S(\theta_5)\ d_6-S(\theta_1)\ S(\theta_4)\ S(\theta_5)\ d_6+(-½\ S(-\theta_3+\theta_1-\theta_2)+½\ S(\theta_3+\theta_1+\theta_2))(C(\theta_5)\ d_6+d_4)+½\ a_2\ C(\theta_1-\theta_2)+½\ a_2\ C(\theta_1+\theta_2)$

$r_{21}=(½\ S(\theta_3+\theta_1+\theta_2)+½\ S(-\theta_3+\theta_1-\theta_2))(C(\theta_4)\ C(\theta_5)\ C(\theta_6)-S(\theta_4)\ S(\theta_6))-C(\theta_1)(S(\theta_4)\ C(\theta_5)\ C(\theta_6)+C(\theta_4)\ S(\theta_6))-(-½\ C(\theta_3+\theta_1+\theta_2)+½\ C(-\theta_3+\theta_1-\theta_2))\ S(\theta_5)\ C(\theta_6)$

$r_{22}=(½\ S(\theta_3+\theta_1+\theta_2)+½\ S(-\theta_3+\theta_1-\theta_2))(-C(\theta_4)\ C(\theta_5)\ S(\theta_6)-S(\theta_4)\ C(\theta_6))-C(\theta_1)(-S(\theta_4)\ C(\theta_5)\ S(\theta_6)+C(\theta_4)\ C(\theta_6))+(-½\ C(\theta_3+\theta_1+\theta_2)+½\ C(-\theta_3+\theta_1-\theta_2))\ S(\theta_5)\ S(\theta_6)$

$r_{23}=-(½\ S(\theta_3+\theta_1+\theta_2)+½\ S(-\theta_3+\theta_1-\theta_2))\ C(\theta_4)\ S(\theta_5)+C(\theta_1)\ S(\theta_4)\ S(\theta_5)+(-½\ C(\theta_3+\theta_1+\theta_2)+½\ C(-\theta_3+\theta_1-\theta_2))\ C(\theta_5)$

$T_y=-(½\ S(\theta_3+\theta_1+\theta_2)+½\ S(-\theta_3+\theta_1-\theta_2))\ C(\theta_4)\ S(\theta_5)\ d_6+C(\theta_1)\ S(\theta_4)S(\theta_5)\ d6+(-½\ C(\theta_3+\theta_1+\theta_2)+½\ C(-\theta_3+\theta_1-\theta_2))(C(\theta_5)\ d_6+d_4)+½\ a_2\ s(\theta_1+\theta_2)+½\ a_2\ S(\theta_1-\theta_2)$

$r_{31}=S(\theta_2+\theta_3)(C(\theta_4)\ C(\theta_5)\ C(\theta_6)-S(\theta_4)\ S(\theta_6))+C(\theta_2+\theta_3)\ S(\theta_5)\ C(\theta_6)$

$r_{32}=S(\theta_2+\theta_3)(-C(\theta_4)\ C(\theta_5)\ S(\theta_6)-S(\theta_4)\ C(\theta_6))-C(\theta_2+\theta_3)\ S(\theta_5)\ S(\theta_6)$

$r_{33}=-S(\theta_2+\theta_3)\ C(\theta_4)\ S(\theta_5)-C(\theta_2+\theta_3)\ C(\theta_5)$

$T_z=-S(\theta_2+\theta_3)\ C(\theta_4)\ S(\theta_5)\ d_6-C(\theta_2+\theta_3)(C(\theta_5)\ d_6+d_4)+a_2\ S(\theta_2)+d_1$

III-3- Modèle géométrique inverse du robot I-Kernel

III-3-1- Définition

Le modèle géométrique direct est une fonction qui permet d'exprimer l'attitude de l'organe terminal par rapport à la base en fonction du vecteur des coordonnées articulaires q. Cette attitude peut être exprimée au moyen de la matrice homogène M_{06}. Elle peut également être exprimée au moyen d'un vecteur à 6 coordonnées p. Par exemple, en utilisant la décomposition roulis, tangage, lacet, on a :

$$P=\begin{bmatrix} Tx(q) \\ Ty(q) \\ Tz(q) \\ \theta r(q) \\ \theta t(q) \\ \theta l(q) \end{bmatrix} = G_d(q)$$

Soit G_i une fonction telle que : $q=G_i\ (p)$

La fonction G_i est appelée Modèle Géométrique Inverse de robot.

Le modèle géométrique inverse est donc une fonction qui permet d'exprimer les coordonnées articulaires q du robot à partir des coordonnées opérationnelles.

Remarque

- Le modèle géométrique inverse G_i dépend du choix de la décomposition de la rotation.
- Il existe souvent, pour une même décomposition de la rotation, plusieurs configurations q du robot qui donnent l'attitude p.
- Pour un même robot, il y a plusieurs modèles géométriques inverses alors qu'il n'y a qu'un seul modèle géométrique direct.
- A toute attitude p, il n'est pas toujours possible d'associer une configuration q du robot : cela dépend de l'espace de travail du robot et du nombre de DDL.

III-3-2- Principe

Le modèle géométrique inverse du robot I-Kernel, qui est un robot 6 axes anthropomorphe peut être décomposé en 2 étapes :

-1. Trouver les positions q_1 q_2 q_3 des 3 premiers axes telles que le centre de la rotule coïncide avec la position désirée.
-2. Trouver les positions q_4 q_5 q_6 des axes du poignet de manière à avoir la bonne orientation de la pince.

Cette décomposition est rendu possible par le fait que la position du centre de la rotule n'est pas affectée par la rotation des 3 derniers axes.

III-3-3- Calcul du modèle géométrique inverse

III-3-3-1- Les trois premiers axes

Première étape : trouver q_1 q_2 q_3 tels que le centre de la rotule ait pour coordonnées $[o_x\ o_y\ o_z]^T$ dans le repère de base.

La figure ci-dessous montre le placement des repères sur les trois derniers axes :

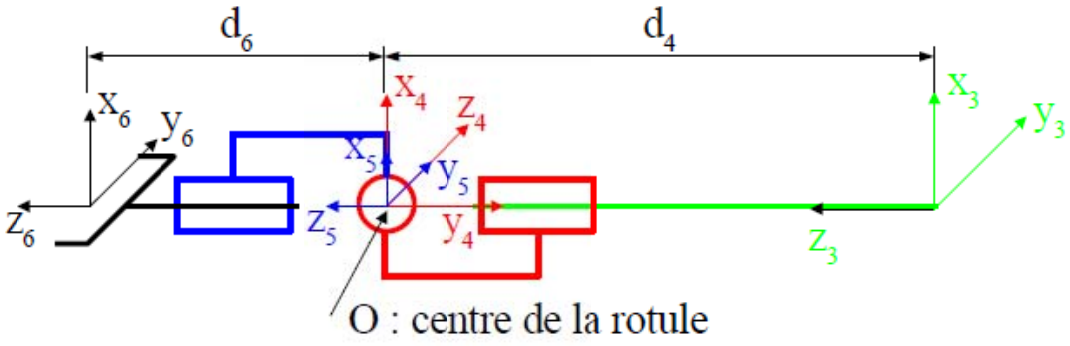
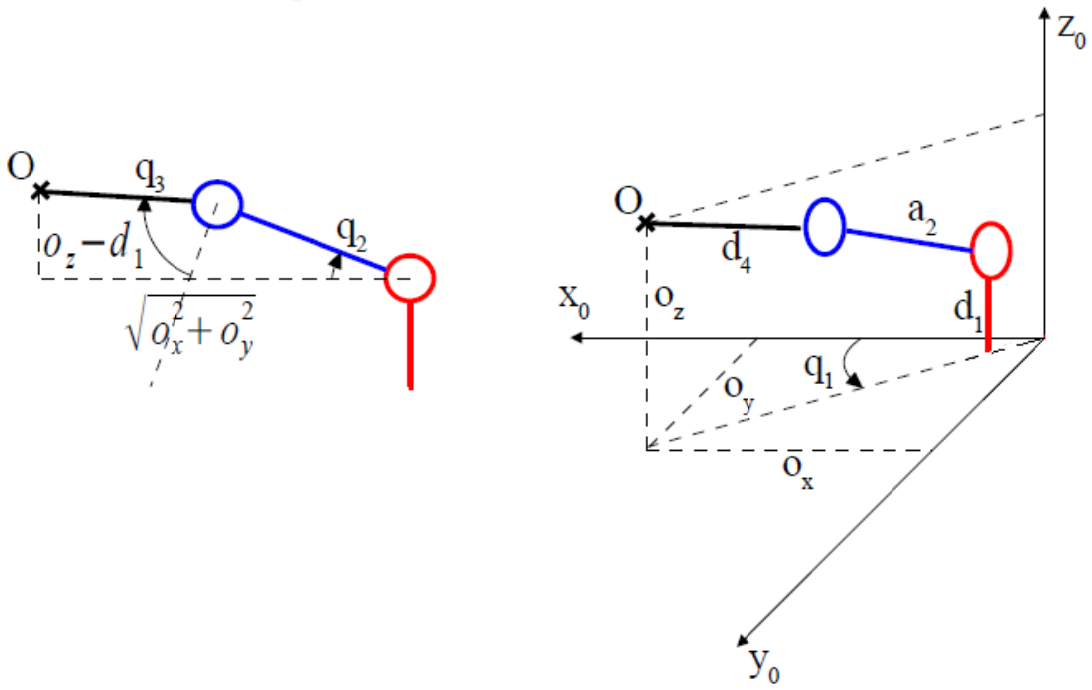


Figure 3-2 Placement de repères pour les trois derniers axes



On a : ${}^{6}O = [0\ 0\ -d_6\ 1]^{T}$

D'où : ${}^{0}O = M_{06}\ {}^{6}O = [O_x\ O_y\ O_z\ 1]^{T}$

Avec M_{06} : C'est l'attitude à atteindre pour la pince.

On a: $q_1 = \arctan 2(O_y, O_x)$

$q3 = \pm \operatorname{acos}(D)$

Avec:

$$D = \frac{(Oz-d1)^2+\sqrt{Ox^2+Oy^2}}{2a2\ d4} - \frac{a2^2-d4^2}{2a2\ d4}$$

Remarque:

● q_1 existe à condition que o_x et o_y soient ≠ 0 : ceci traduit le fait que si le point O est situé sur l'axe 1, la position de cet axe est indéterminée.

● q_2 et q_3 existent à condition que D < 1 :

$$\Rightarrow \sqrt{((Oz-d1)^2)+\sqrt{Ox^2+Oy^2}} < a2+d4$$

Cette condition traduit le fait que le bras doit être assez long pour pouvoir atteindre la position désirée.

On peut aussi déduire que :

q_2= arctan 2 (k_1 (o_z−d_1) − k_2 ($\sqrt{ox^2 + oy^2}$) , k_2 (oz−d1) + k $_1$ ($\sqrt{ox^2 + oy^2}$))

Avec : k_1=a_2+d_4 sin(q_3), k_2=d_4 cos(q3).

III-3-3-2- Les trois derniers axes

On a donc obtenu q_1 q_2 et q_3, il reste à déterminer l'orientation du poignet. Soit M_{03} (q_{123}) la matrice homogène (connue) de transformation entre la base et le repère R_3 et M_{36} (q_{456}) la matrice homogène (inconnue) de transformation entre le repère R_3 et le repère R_6.

On a : M_{03} M_{36} = M_{06}

D'où M36 = M^{-1}_{03} M_{06}

On connaît donc M_{36}, avec : M_{36} = $\begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Or, on a : M_{36} = $\begin{bmatrix} Cq_4Cq_5Cq_6 - Sq_4Sq_6 & -Cq_4Cq_5Sq_6 - Sq_4Cq_6 & -Cq_4Sq_5 & -Cq_4Sq_5d_6 \\ Sq_4Cq_5Cq_6 + Cq_4Sq_6 & -Sq_4Cq_5Sq_6 + Cq_4Cq_6 & -Sq_4Sq_5 & -Sq_4Sq_5d_6 \\ -Sq_5Cq_6 & Sq_5Sq_6 & Cq_5 & d_4+d_6Sq_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Si $q_5 \neq 0+k\pi$ alors : q_4= arctan2 (±r_{23}, ±r_{13})
q_5= arctan2 (±$\sqrt{(}r^2_{31}$ +r^2_{32}), r_{33})
q_6= arctan2 (±r_{32}, ±r_{31})

Cas particulier

q_5= 0: q_4= 0
q_5= 0
q_6= arctan2 (r_{21}, r_{11})

q_5= π: q_4= 0
q_5= π
q_6= arctan2 (r_{12}, r_{22})

III-3-3-3- Remarques

● Il y a deux (2) possibilités pour $\{q_1\ q_2\ q_3\}$ et pour $\{q_4\ q_5\ q_6\}$. Il y a donc quatre (4) possibilités pour $\{q_1\ q_2\ q_3\ q_4\ q_5\ q_6\}$. Ce qui signifie qu'il y a quatre (4) configurations du robot qui satisfont à l'attitude désirée de l'organe terminal.

● Il y a 3 singularités dans la détermination du modèle géométrique inverse :
- – lorsque le point O est sur l'axe 1,
- – lorsque le point O est inaccessible (bras trop court),
- – lorsque $q_5 = k\pi$.

III-4- Conclusion

Dans ce chapitre nous avons calculé le modèle géométrique direct ainsi que le modèle géométrique inverse du robot I-Kernel, Pour ce faire nous avons décomposé le robot en 2x3 articulations, ainsi le modèle géométrique des trois premières articulations désigne la position à atteindre par le bras, et le modèle des trois dernières articulations contrôle la façon avec laquelle la pince attrapera l'objet.

Introduction

La structure mécanique seule ne satisfait pas à la définition d'un robot autonome, car les actionneurs et les capteurs interviennent dans sa réalisation, ainsi un système de commande a été introduit afin de répondre à la définition. Dans ce chapitre nous avons décrit et cité les actionneurs et capteurs que nous avons utilisé pour réaliser ce robot.

IV-1- Les actionneurs

IV-1-1-Introduction

Dans une machine ou un système de commande, semi automatique ou automatique, un actionneur est l'organe de la partie opérative qui, dés qu'il reçoit un ordre de la partie commande via un éventuel pré-actionneur, convertit l'énergie qui lui est fournie en un travail utile à l'exécution de tâches, éventuellement programmées, d'un système automatisé.

En d'autres termes, un actionneur est l'organe fournissant la force nécessaire à l'exécution d'un travail ordonné par une unité de commande.

Il existe donc plusieurs familles d'actionneurs, pneumatiques, hydrauliques et électriques.

Le choix des actionneurs d'un robot est généralement une tâche fastidieuse. Il faut dans un premier temps déterminer avec suffisamment de précision le travail qui sera accompli par chacun des actionneurs. Évidemment, pour pouvoir valider un choix, il faut connaître les performances que nous voulons atteindre.

Pour notre projet le choix le plus approprié est d'utiliser des servomoteurs RC pour cause de :

1- le fil du signale (commande) à faible courant peut être raccordé directement à une sortie du microcontrôleur. Pas besoin de circuit d'interface.
2- On peut commander l'arrêt, la marche, le sens de rotation et la vitesse du servomoteur à l'aide d'un seul fil. Economie d'E/S
3- le servomoteur tourne à la bonne vitesse pour notre robot
4- le servomoteur offre un couple important sous un volume réduit.

IV-1-2- Les servomoteurs RC

IV-1-2-1-Présentation et caractéristiques générales

Les Servomoteurs RC (radio commandé) sont souvent utilisés dans beaucoup d'applications industrielles parce qu'ils sont très compacts et moins très chers. Un servomoteur est composé d'un moteur DC, d'une boite de vitesse, d'un capteur de retour de position (habituellement un potentiomètre) ainsi que électronique approprié à sa commande. Le servomoteur peut être contrôlé par un signal externe à large impulsion de modulation (PWM). La figure suivante illustre sa vue d'ensemble.

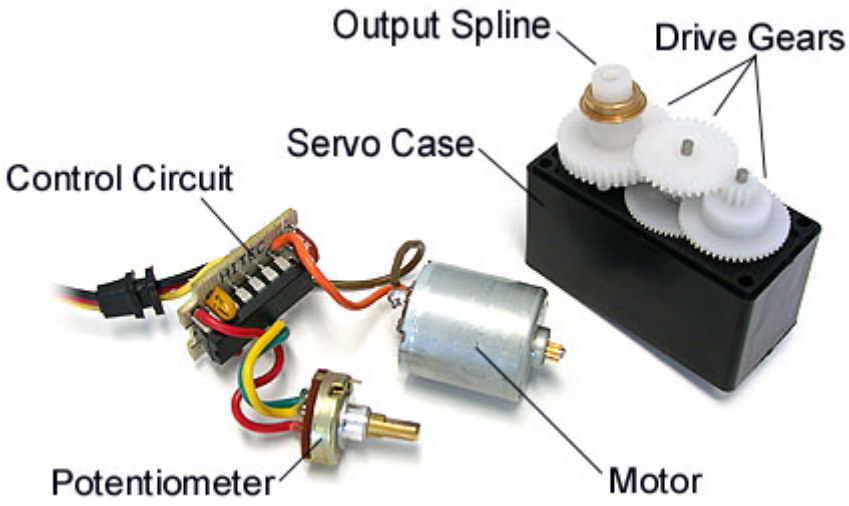


Figure 4-1 Vue d'ensemble d'un servomoteur RC

Si un signal rencontre le servomoteur « timingrequirements », cela défini habituellement une position cible d'entrée pour l'électronique de commande du servomoteur. La carte électronique du servomoteur compare la position de l'arbre avec la position en entrée, en essayant de trouver la position de l'arbre qui correspond. Le signal de contrôle de la position est un signal carré continu, comme le montre cette figure.

Figure 4-2 protocoles des signaux de commande des servomoteurs RC

Les servomoteurs RC sont des actionneurs très courants en robotique et en construction de modèles. Les servomoteurs RC sont composés essentiellement d'un petit moteur DC, d'un engrenage de réduction de vitesse et d'un circuit électronique contenant un pont H. Habituellement le rotor du servomoteur bouge jusqu'à une certaine position et essaye de la conserver.

La position du rotor dépend du signal de contrôle reçu par les servomoteurs RC. En fonction du type de servomoteurs, l'angle de rotation maximum du moteur peut changer. Les Servomoteurs RC qui tournent de manière constante sont rares. Dans ce cas le signal de contrôle ne détermine pas l'angle de révolution mais la vitesse de rotation. Les Servomoteurs RC "hack" sont bien plus courants, dans ce cas le potentiomètre de retour de position est remplacé par deux résistances fixes et le résistor mécanique (butée) qui limite la rotation à 180°. Une caractéristique très importante du servomoteur est son ratio poids-puissance.

Le signal de contrôle du servomoteur est une impulsion spécifique en rapport avec le signal modulé (PWM), en vue de détermine la position du rotor, comme indiquée par la figure ci-dessous. La période du signal est de 20 ms (50 Hz) et la largeur du pic est de 1 ms – 2 ms. En effet, 1 ms marque l'une des positions extrêmes et 2 ms marque la seconde, 1.5 ms marque la position centrale du rotor du servomoteur.

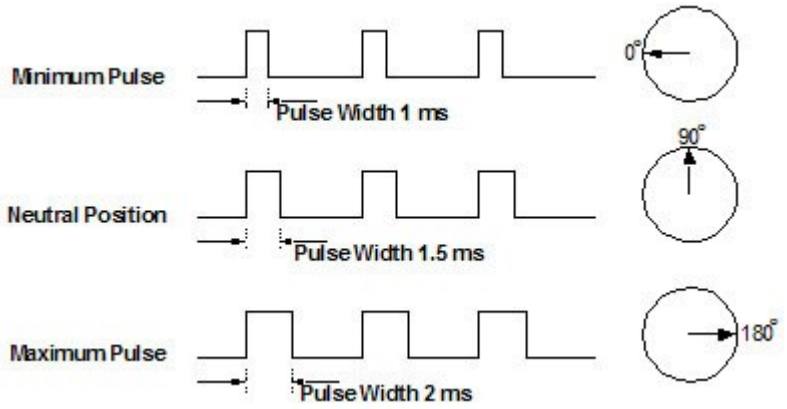


Figure 4-3 relation PWM/Position de l'arbre

Le servomoteur traditionnel est aussi connu sous le nom de servomoteur analogique. Parce que dans les dix dernières années beaucoup de servomoteurs RC numériques sont devenus courants. La différence entre les deux est que dans les servomoteurs RC analogiques, le moteur est contrôlé par un signal PWM d'entrée de 50 Hz, alors que dans les servomoteurs RC numériques, le moteur est contrôlé par un microcontrôleur avec une fréquence plus grande. Le signal d'entrée est le même pour les servomoteurs RC digitaux mais une fréquence de modulation du moteur plus grande permet de déterminer la position plus rapidement et avec plus de précision.

IV-1-2-2- Fonctionnement d'un servomoteur

Pour faire bouger l'axe de sortie il faut lui envoyer un signal sur le fil signal sous forme d'une impulsion. C'est la largeur de cette impulsion qui détermine l'angle de rotation de l'axe de sortie. La durée de l'impulsion peut être comprise entre 0.9 et 2.1ms, 1.5ms correspond au centre.
Petit détail qui a son importance: cette impulsion doit être répétée toutes les 20ms.
Remarque : La rotation des servomoteurs ne depasse pas les 180°.

IV-1-2-3- Rôle du potentiomètre

Le potentiomètre sert à l'asservissement de position. L'axe du potentiomètre est solidaire de l'axe de sortie, donc le potentiomètre tourne en même temps que l'axe de sortie. La résistance aux bornes du potentiomètre varie donc en fonction de la position de l'axe. La résistance du potentiomètre est en rapport avec la durée du signal.
C'est le rôle de l'électronique de faire la relation entre la durée de l'impulsion et la résistance aux bornes du potentiomètre.

IV-1-2-4- Variation de la vitesse d'un servomoteur

On fait varier la vitesse du servomoteur en jouant sur la durée de l'impulsion, plus la durée de l'impulsion est proche de 1.5ms plus la vitesse de rotation est faible, et le système réducteur de vitesse sert tout simplement à réduire la vitesse de rotation de l'axe de sortie.

Les moteurs à courant continu tournent à des vitesses trop élevées et donc ils perdent en précision contrairement aux servomoteurs RC.

IV-1-3- Les servomoteurs RC utilisés

Pour la réalisation de notre projet nous avons utilisé deux types de servomoteurs RC : Un numérique et Huit analogiques.

IV-1-3-1- Les servomoteurs RC analogiques

HITEC-HS-805BB

Pour les servomoteurs RC de l'épaule et du coude, nous avons utilisé des servomoteurs RC de grande taille donc de puissance importante et ce pour fournir un couple adéquat pour soulever le poids de la structure. Leur référence est HITEC-HS-805BB et il est montré sur la figure ci-dessous :

Figure 4-4 HITEC-HS-805BB

Ses dimensions en cm sont montrées dans la figure ci-dessous :

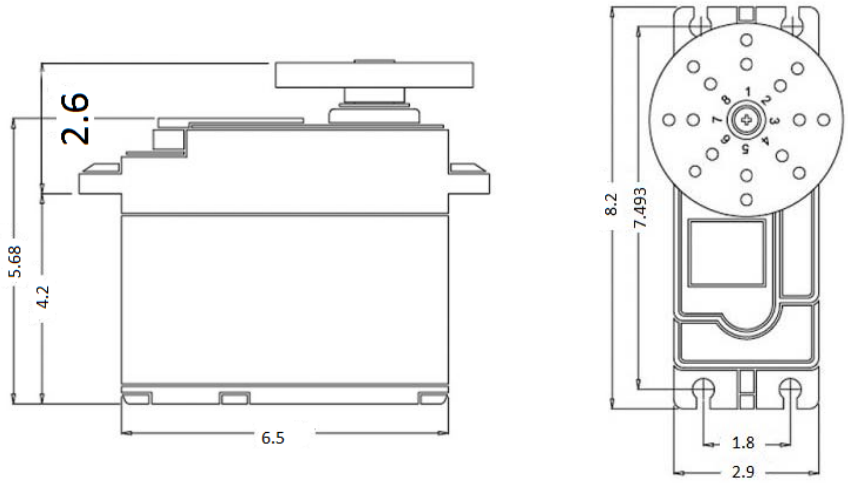


Figure 4-5 les dimensions du servomoteur HITEC-HS-805BB

Les caractéristiques techniques de ce modèle sont synthétisées dans le tableau suivant :

Système de contrôle	Contrôle par largeur d'impulsion, neutre a 1500 us
Impulsion requise	Signal carré de 3-5V crête à crête
Courant de drainage	8 mA au repos et 700 mA en fonctionnement sans charge, la consommation peut atteindre et dépasser 1.2A
Tension d'opération	4.8 – 6 volt
Température de fonctionnement	-20 à +60 degrés C
Vitesse de fonctionnement à 4.8 V	60°/0.19s sans charge
Vitesse de fonctionnement à 6 V	60°/0.14s sans charge
Couple à 4.8V	19.8 Kg. Cm
Couple à 6V	24.7 Kg. Cm
Angle de déplacement	90° pour une variation des impulsions de 800 us
Type de moteur	3 Pole Ferrite
Type de roulement	roulement à bills double
Type d'engrenage	Nylon dure
Poid	152 g

Tableau 4-1 Les caractéristiques techniques du servomoteur HITEC-HS-805BB

HITEC-HS-485HB:

Ce servomoteur est utilisé dans la 3eme partie du poignet et la pince.

Figure 4-6 HITEC-HS-485HB

Ses dimensions sont montrées dans la figure ci-dessous :

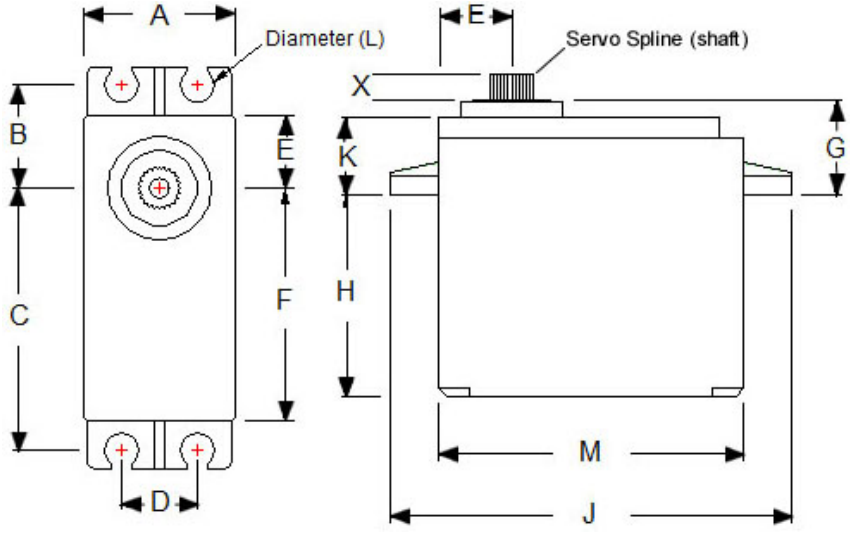


Figure 4-7 les dimensions du servomoteur HITEC-HS-485HB

Où:

A = .780” (19.82mm)
B = .547” (13.9mm)
C = 1.352” (34.35mm)
D = .394” (10mm)
E = .394” (10mm)
F = 1.181” (30mm)
G = .472” (12mm)
H = 1.102” (28mm)

J = 2.09” (53.1mm)
K = .384” (9.75mm)
L = .174” (4.42mm)
M = 1.575” (40mm)
X = .126” (3.2mm)

Ses caractéristiques techniques sont synthétisées dans le tableau suivant :

Système de contrôle	Contrôle par largeur d'impulsion, neutre a 1500 us
Impulsion requise	Signal carré de 3-5V crête à crête
Courant de drainage a 4.8 V	150 mA sans charge
Courant de drainage a 6 V	180 mA sans charge
Tension d’opération	4.8 – 6 volt
Température de fonctionnement	-20 à +60 degrés C
Vitesse de fonctionnement à 4.8 V	60°/0.22s sans charge
Vitesse de fonctionnement à 6 V	60°/0.18s sans charge
Couple à 4.8V	4.8 Kg. Cm
Couple à 6V	6 Kg. Cm
Angle de déplacement	90° pour une variation des impulsions de 800 us
Type de moteur	3 Pole Ferrite
Type de roulement	roulement à bills
Type d’engrenage	Nylon dure
Poids	45 g

Tableau 4-2 Les caractéristiques techniques du servomoteur HITEC-HS-485HB.

DGServo S09NF STD :

Ce servomoteur est utilisé dans la première partie du poignet, sa dimension est 41x38x20mm .

Ses caractéristiques techniques sont synthétisées dans le tableau suivant :

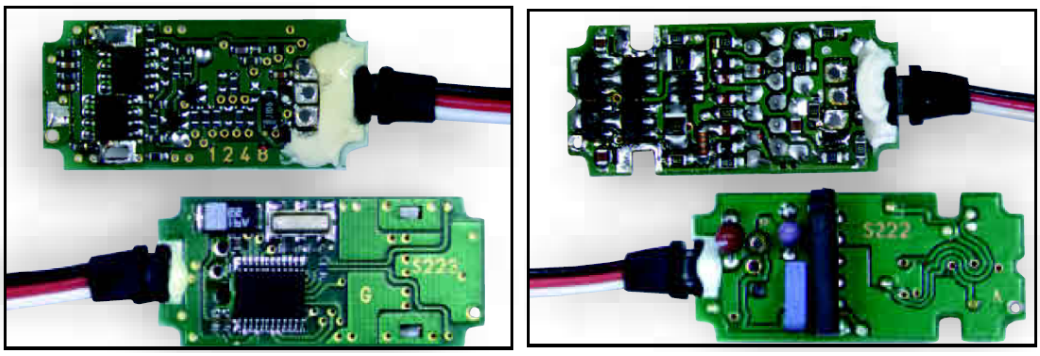
Système de contrôle	Contrôle par largeur d'impulsion, neutre a 1500 us
Impulsion requise	Signal carré de 3-5V crête à crête
Courant de drainage	8 mA au repos et 700 mA en fonctionnement sans charge, la consommation peut atteindre et dépasser 1.2A
Tension d'opération	4.8 – 6 volt
Température de fonctionnement	0 à +60 degrés C
Vitesse de fonctionnement à 4.8 V	60°/0.15s sans charge
Vitesse de fonctionnement à 6 V	60°/0.18s sans charge
Couple à 4.8V	3.5 Kg. Cm
Couple à 6V	5 Kg. Cm
Angle de déplacement	90° pour une variation des impulsions de 800 us
Type de moteur	3 Pole Ferrite
Type de roulement	roulement à bills
Type d'engrenage	engrenage métallique
Poids	38g

Tableau 4-3 Les caractéristiques techniques du servomoteur DGServo S09NF STD

IV-1-3-2- Servomoteurs digitaux RC

Les servomoteurs RC digitaux ont des avantages opérationnels significatifs par rapport aux servomoteurs standards RC.

Les servomoteurs digitaux RC sont les même que les servomoteurs standard (analogique), excepté une différence dans leurs microcontrôleurs, qui analysent les signaux entrants du récepteur et commandent les moteurs.



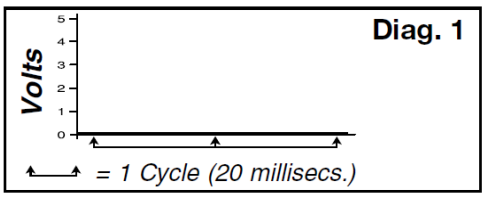
Servomoteur digital servomoteur analogique

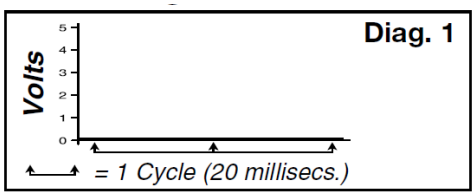
Figure 4-8 Les circuits électroniques des deux types de servomoteurs RC

Sa commande s'effectue alternativement à la puissance initiale du servomoteur, réduisant la période transitoire, et faisant augmenter la résolution et développant une puissance énorme.

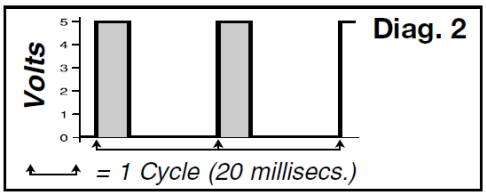
Les trois diagrammes illustrés dans la figure ci-dessous montrent les différences qui sont entre les deux servomoteurs pendant les deux cycles «Marche/Arrêt» durant une période de 20ms.

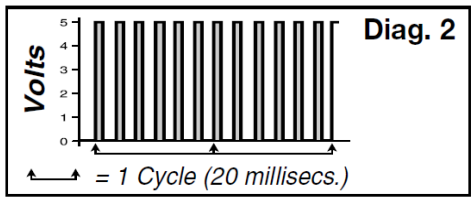
Servomoteur standard(analogique) servomoteur digital



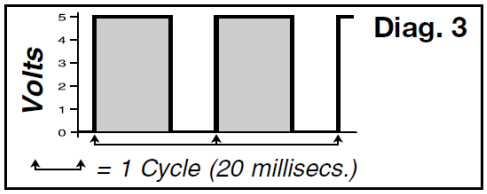


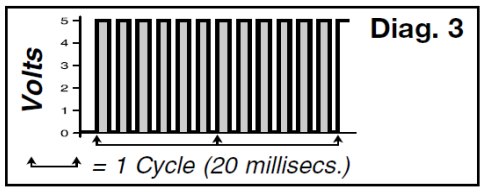
Diag.1 - sans imputions.





Diag.2 – avec des impulsions étroites .





Diag.3 –avec des larges impulsions.

Figure 4-9 diagrammes de comparaison entre servomoteur numérique et analogique

Le microcontrôleur du servomoteur digital envoie des impulsions au moteur à une fréquence sensiblement plus haute. Ceci signifie que, par opposition au moteur recevant 50 impulsions/sec, il reçoit maintenant 300 impulsions/sec. Ceci signifie également que non seulement le moteur répond plus rapidement aux commandes, mais augmente ou diminue la puissance.

L'accélération/décélération peut être transmise au servomoteur fréquemment. Ceci donne aux servomoteurs digitaux un temps de réponse plus court, plus vite et une accélération/décélération plus douce, et une meilleure puissance.

Sur notre robot nous avons opté d'utiliser le servomoteur digitale HS-5805MG de la gamme HITEC pour la base mobile en raison que c'est la partie qui doit supporter tout le poids.

HS-5805MG :

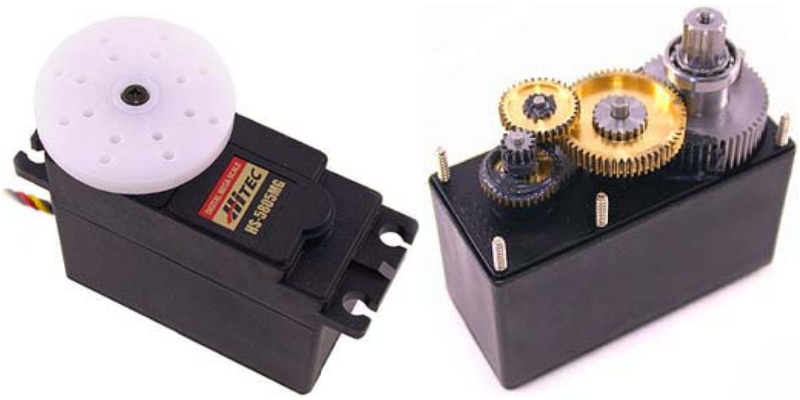


Figure 4-10 HS-5805MG

Ses dimensions sont montrées dans la figure ci-dessous :

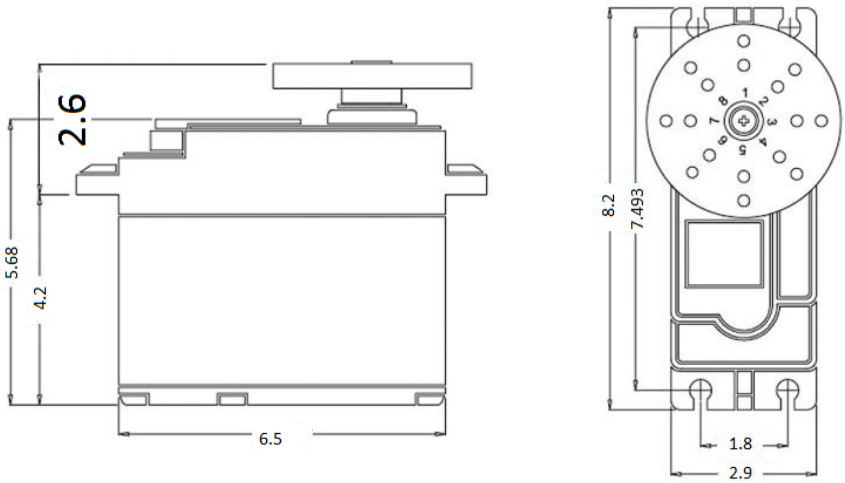


Figure 4-11 les dimensions du servomoteur HS-5805MG

Ses caractéristiques techniques sont synthétisées dans le tableau suivant :

Système de contrôle	Contrôle par largeur d'impulsion, neutre a 1500 us
Impulsion requise	Signal carré de 3-5V crête à crête
Courant de drainage	8 mA au repos et 700 mA a 4.8 et 830mA a 6V en fonctionnement sans charge, la consommation peut atteindre et dépasser 1.8A
Tension d'opération	4.8 – 6 volt
Température de fonctionnement	-20 à +60 degrés C
Vitesse de fonctionnement à 4.8 V	60°/0.19s sans charge
Vitesse de fonctionnement à 6 V	60°/0.14s sans charge
Couple à 4.8V	19.8 Kg. Cm
Couple à 6V	24.7 Kg. Cm
Angle de déplacement	45° pour une variation des impulsions de 400 us
Type de moteur	3 Pole Ferrite
Type de roulement	double roulement à bills
Type d'engrenage	4 engrenages métalliques
Poid	197 g

Tableau 4-4 Les caractéristiques techniques du servomoteur HS-5805MG

IV-2- Les capteurs

Afin de donner à notre robot un certain aspect d'autonomie et plus exactement d'intelligence, nous nous somme optés à utiliser deux capteurs :

IV-2- 1 Webcam

Une webcam est une caméra conçue pour être utilisée comme un périphérique d'ordinateur, et qui produit une vidéo dont la finalité n'est pas d'atteindre une haute qualité, mais de pouvoir être transmise en direct au travers d'un réseau, typiquement Internet.

Une webcam peut se connecter à un ordinateur via :

- Le port USB,
- Plus rarement par le port FireWire,
- Ou par les ports parallèles ou séries (anciens modèles, abandonné du fait du trop bas débit),

- Sur un réseau Ethernet (haut de gamme, permet d'avoir une distance plus longs) ou Wi-Fi (sans fil),
- Ou grâce à une carte d'acquisition vidéo interne ou externe à l'ordinateur,
- Via un convertisseur : entrée analogique sortie numérique USB,
- Via un serveur vidéo : entrée analogique sortie numérique Ethernet.

Les webcams possèdent un capteur, qui peut être CCD, ou CMOS, les webcams à capteurs CMOS étant généralement moins chères et de qualité moindre qu'un capteur CCD. Certaines webcams sont équipées de lentilles en verre que l'on retrouve dans les appareils photo. Pour améliorer l'image, les webcams sont souvent couplées à un système logiciel d'interpolation, ayant pour but d'afficher une image détaillée à partir d'une image de faible qualité en créant des pixels intermédiaires dont la couleur est calculée par comparaison avec les pixels adjacents.

Dans notre projet nous avons utilisés deux webcams identiques de prototype donné par la figure ci-dessous. La gamme Logitech modèle c160, 1.3 Méga pixel, focal réglable manuellement, technologie CMOS.

Figure 4-12 Webcam Logitech c160

IV-2- 3 Capteur de pression (QTC)

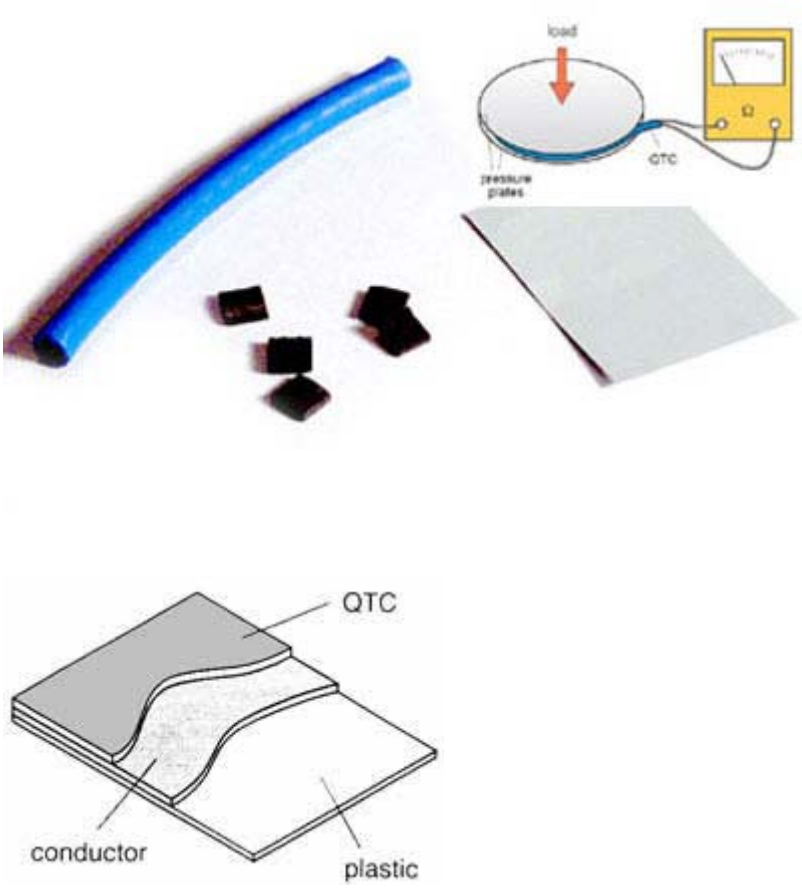


Figure 4-13 vue d'ensemble du capteur QTC

Les composites à effet tunnel QTC (Quantum Tunnelling Composite) ont été développés par Peratech (UK) pour des applications en commutation et en détection.

Ces matériaux, constitués de particules métalliques dans un liant élastomère de type silicone, modifiant leurs propriétés électriques sous l'effet de la pression : ils passent lentement de l'état d'isolant à celui de conducteur métallique comme c'est illustrer ci-dessous.

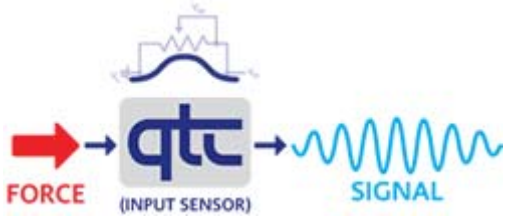


Figure 4-14 : changement d'état du capteur : isolant/conducteur en appliquant une force

Contrairement aux plastiques chargés de particules de carbone, les QTC sont des isolants presque parfaits lorsque aucune contrainte ne s'exerce (1012 ohms contre quelques milliers

d'ohms pour les composites carbone) et ont le niveau de conductivité d'un métal (1 ohm contre quelques centaines pour les composites carbone) quand la pression est suffisante.

Les QTC sont aussi plus sensibles et la pression nécessaire pour produire un changement de résistance est bien moindre. La pression peut s'exercer par un simple toucher, par flexion, par torsion, par traction. La figure ci-dessous illustre la relation force/résistance.

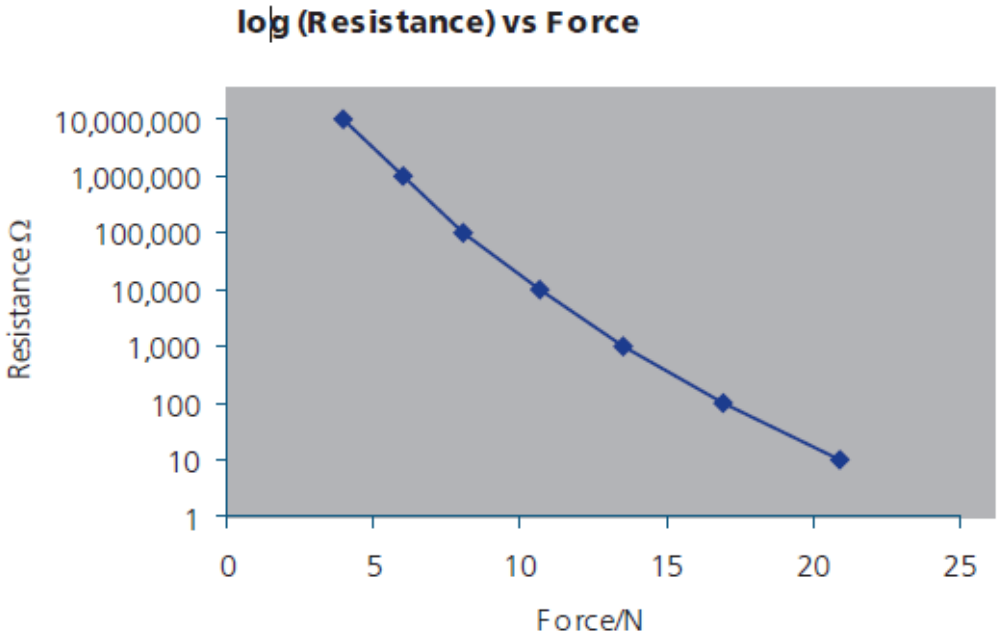


Figure 4-15 Relation Force/Resistance

Dans un composite QTC, le mécanisme principal de la conduction est l'effet tunnel, lié à la forme des charges métalliques comme le montre cette figure :

Figure 4-16 image de synthèse de l'effet tunnel

Dans les polymères chargés de carbone, les particules sont arrondies et lisses : elles sont en contact les unes avec les autres et créent ainsi un chemin de conduction, par un mécanisme dit de percolation.

Dans un QTC, les particules métalliques sont hérissées d'aspérités qui sont mouillées par le silicone : elles ne se touchent donc pas, même si la charge est dense.

Les pointes à la surface permettent de concentrer des charges électriques. Ceci produit une augmentation localisée du champ électrique qui réduit l'effet barrière de l'élastomère et permet l'apparition de la conductivité.

Quand un QTC est comprimé, les particules entrent en contact et l'épaisseur de la barrière isolante est réduite. La transition isolant/conducteur suit une courbe régulière et reproductible, et l'augmentation de la conductivité est exponentielle.

Les QTC peuvent être moulés pour obtenir une taille, une épaisseur ou une forme quelconque, ce qui permet d'obtenir toutes sortes de produits tels que, par exemple, commutateurs marche/arrêt, commutateurs proportionnels, microrupteurs, connexions électriques sans étincelles ou fusibles réarmables. Ces circuits peuvent être utilisés dans des pavés numériques, des moyens de défilement simples, des joysticks, des commandes d'éclairage et de moteurs, et comme un capteur de sensibilité tell qu'on la utiliser dans la pince de notre robot.

Les caractéristiques techniques de ce capteur sont données dans le tableau ci-dessous :

Largeur	3.6 mm
Longueur	3.6 mm
Epaisseur	1 mm
Poids	0.04g
Densité	4.0 g/cm^3
Plage de force	0 N - 100 N
Durée de vie	Plus de 1, 000,000 compressions
Température de fonctionnement	-20ºC to 120ºC
Plage d'humidité	0 – 100%
Résistivité mécanique	7 x 10^{12} Ohm cm
Plage de résistance typique	Inferieure a 10^{12} Ohms, supérieure a 1 Ohm
Tension de fonctionnement	0 – 40 V
Courant maximale	10A

Tableau 4-5 Les caractéristiques techniques capteur QTC

V-1- Introduction

Une des particularités des êtres vivants est de pouvoir acquérir des images, via l'œil, comme une information, puis de pouvoir l'interpréter via le cerveau. L'enjeu de la vision par ordinateur est de permettre à un ordinateur de "voir", c'est à dire, comme l'homme, de récupérer l'information par l'intermédiaire d'un dispositif d'acquisition d'image puis de l'exploiter.

Ainsi, l'ordinateur sera alors capable de reconnaitre des formes ou encore de séparer une image en différentes zones distinctes et cohérentes.

V-2- Définition

La vision par ordinateur est la science qui développe les bases théoriques et algorithmiques, ainsi que extraction et analyse des informations utiles, à partir d'une image observée ou une séquence d'images.

Ces informations peuvent être liées à la reconnaissance d'un objet connu, la description en trois dimensions d'un objet inconnu, la position et l'orientation ou la mesure de toute propriété spatiale d'un objet, telle que la distance entre deux de ses points distingués ou le diamètre de la section circulaire.

Dans notre projet, nous voulions extraire la position d'un objet considéré comme connu en se basant sur la stéréovision et la reconnaissance d'objet. Pour ce faire, nous avons opté sur l'utilisation d'une bibliothèque en C/C++ nommée OpenCV (voir annexe).

V-3- Stéréovision

La stéréovision est l'un des sujets clés de la recherche en vision par ordinateur actuellement. On peut la décrire on prenant deux points de vue d'un objet dans un monde en 3D, en comparant la distance entre les pixels correspondants dans les deux images et en prenant cette distance comme une mesure de la distance réelle entre l'objet et la caméra.

Ces informations de distance sont récupérées par un algorithme de stéréo dense dont le résultat est une carte de disparité.

Une carte de disparité est une image 2D où la couleur de chaque pixel représente la distance entre ce point et la caméra. En d'autres termes, les pixels lumineux sont proches et les pixels sombres sont loin de la camera.

Les étapes qui entrent en jeu dans le calcul de la position d'un objet connu en stéréovision sont :

1. La calibration du capteur stéréoscopique
2. La mise en correspondance stéréo dense
3. la triangulation

V-3-1- Capteur stéréoscopique

On distingue deux capteurs stéréoscopiques :

V-3-1-1- Capteur stéréoscopique actif

En combinant une camera avec une source de lumière afin de mesurer les coordonnées tridimensionnelles de points sur la surface d'un objet.

Un faisceau laser (lumière En stéréovision monochromatique) éclaire une partie de la scène qui correspond à l'intersection du plan lumineux et de la scène. La figure 5-1 illustre un capteur stéréoscopique actif.

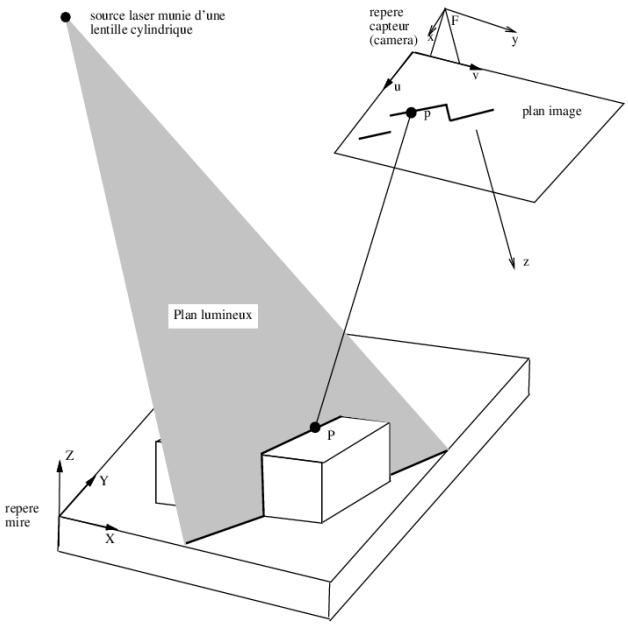


Figure 5-1 Capteur stéréoscopique actif.

V-3-1-2- Capteur stéréoscopique passif

Il s'agit de deux cameras qui observent la même scène, on récupère ainsi deux projections de chaque point de la scène.

Considérons le schéma de la figure ci-dessous, qui représente deux cameras, à chaque camera est associé un repère. Soit P un point de la scène et soient p et p' ses deux projections dans les deux images gauche et droite. Nous pouvons donc décrire l'équation de la droite passant par le centre focal F de la camera de gauche et le point P. De même on peut écrire l'équation de la droite passant par F'et p'. Le point p de la scène se trouve à l'intersection de ces droites. Afin de pouvoir calculer cette intersection et donc déterminer la position de P, il faut pouvoir exprimer les deux équations de la droite Fp et F'p' dans le même repère.

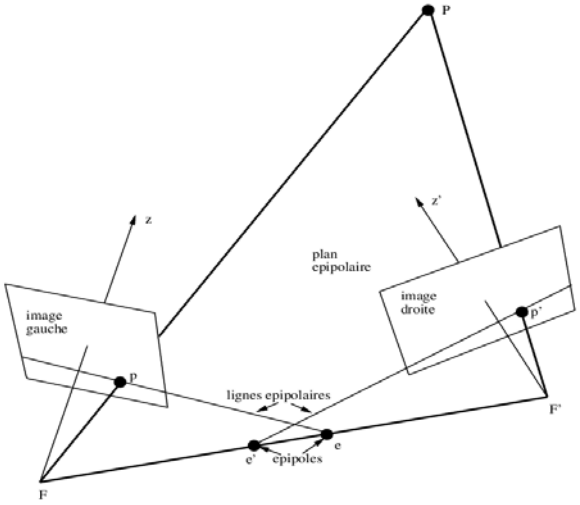


Figure 5-2 Capteur stéréoscopique passif.

V-3-2- Calibration du capteur stéréoscopique

Calibrer géométriquement une caméra permet de faire la correspondance entre les points de la scène observée et les points dans le plan du capteur. La plupart des méthodes de calibration utilisent le modèle du sténopé pour décrire le modèle d'une caméra.
C'est une approche très intéressante, car elle permet de modéliser la plupart des capteurs projectifs et aussi de simplifier l'estimation des paramètres du modèle. Le principe du modèle de sténopé est montré par la figure ci-dessous :

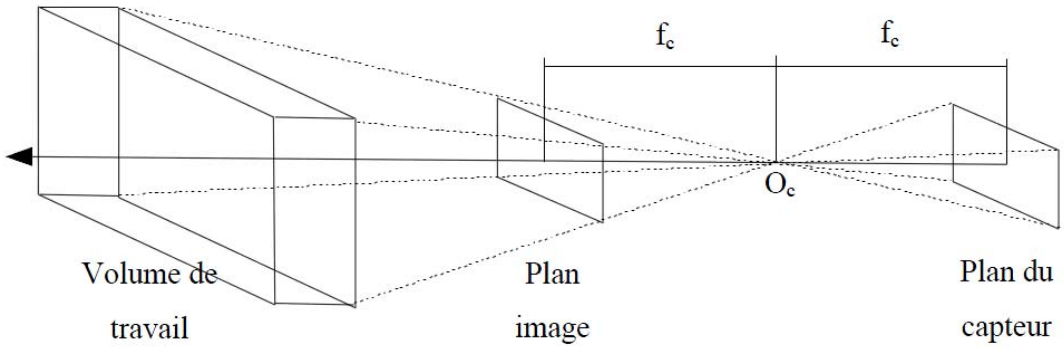


Figure 5-3 modèle du sténopé.

Les caméras ont certains paramètres qui définissent comment des objets du monde réel sont projetés sur le plan image. Les quatre paramètres les plus importants sont la longueur focale selon les axes X et Y et le déplacement possible du centre de l'image sur l'axe optique. La longueur focale est idéalement la distance entre le centre de projection des deux

cameras C1 et C2 et le plan de l'image R1 et R2. Ces paramètres sont résumés dans une matrice homogène, appelée la matrice intrinsèque ou interne de la camera :

$$A = \begin{pmatrix} fx & 0 & cx \\ 0 & fy & cx \\ 0 & 0 & 1 \end{pmatrix}$$

On a une autre matrice appelée matrice extrinsèque ou externe, qui définit la transformation de ce système de coordonnées en celui de la camera, elle contient les rotations et les translations selon x, y, z.

$$E = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ces deux matrice forment le modèle mathématique de la camera.

Quand on travaille avec des caméras et en particulier les webcams, souvent il y a une certaine distorsion de l'image en raison des imperfections de la caméra. Les deux distorsions les plus importantes sont la distorsion radiale causée par la lentille qui est plus sphérique que l'objectif parfaitement parabolique qui sera l'idéal, et la distorsion tangentielle, provoquée par la lentille qui n'est pas totalement parallèle au plan de l'image. Ces distorsions peuvent être annulées par une simple projection mathématique.

Les cinq paramètres nécessaires pour annuler la distorsion peuvent être calculés si nous pouvions trouver une trame de points qui se trouve sur des lignes droites dans le monde réel. A cet effet, souvent un échiquier sur une surface plane est utilisé.
Basé sur la trame déformée prise d'un échiquier, on pourrait déterminer à la fois les distorsions et en utilisant les paramètres déterminés mathématiquement pour reconstruire toutes les images originales prises par la caméra.

Dans notre cas nous travaillions avec la stéréovision, qui est tributaire de la relation spatiale entre les deux caméras. En utilisant l'algorithme proposé par Zhang, nous pouvons automatiser ce processus en visualisant l'échiquier par les deux cameras sous différents angles. Ainsi, un simple algorithme de détection de coins peut reconnaître les intersections noire et blanche des cases de l'échiquier.

L'échiquier peut être utilisé en toute taille N x M. De cette façon, pour chaque image contenant un échiquier, nous avons N x M points pour lesquels nous connaissons les coordonnées des images des deux caméras. En utilisant les disparités de ces points entre les deux caméras, on peut estimer la matrice de rotation / translation (appelés E) qui transforme le système de coordonnées de la caméra de gauche au système de coordonnées de la caméra de

droite ou vice versa. En utilisant cette transformation, nous pourrions transformer les coordonnées physiques d'un objet vu par la caméra de gauche aux coordonnées physiques tel que vu par la caméra de droite. La fonction de calibrage stéréoscopique d'OpenCV calcule une matrice (appelée F), qui pourrait transformer les coordonnés des pixels de l'image de gauche a celle de droite, ou vice versa.

V-3-3- Géométrie épipolaire et de rectification

La géométrie épipolaire est utilisée dans la stéréovision pour limiter l'espace de recherche afin de trouver les points correspondants dans les deux images. Un point X dans l'espace 3D est vu dans la vue de gauche que nous appellerons l'image source, comme une ligne dont les points extrêmes sont le point focal de la caméra de gauche O_L et le point X. Par contre, dans la vue de droite, que nous appellerons l'image de recherche, il vu comme une ligne appelée ligne épi-polaire, (voir figure 4-4)

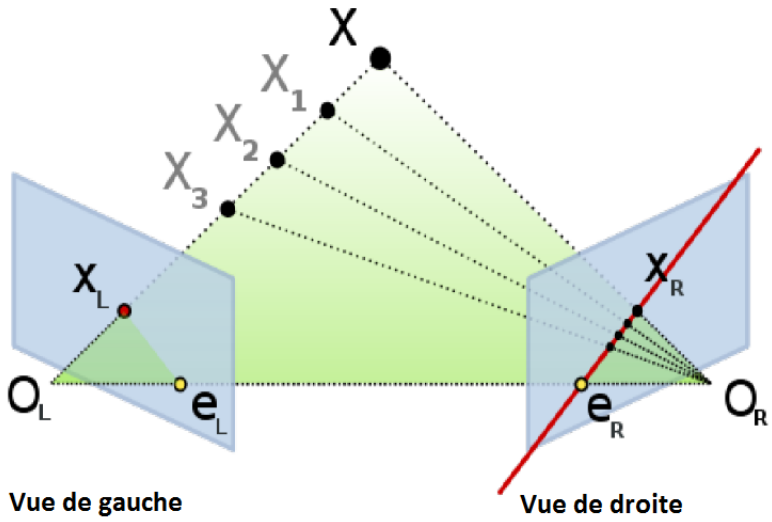


Figure 5-4 Géométrie épipolaire.

En tenant compte des matrices internes et externes des deux caméras et un point x, nous pouvons générer une ligne épipolaire correspondante à ce point dans l'image de recherche, ce qui limite l'espace de recherche à cette ligne. Ceci signifie que pour chaque pixel de l'image source, nous devons calculer la ligne épi-polaire correspondante à l'image de recherche. Il est possible de transformer les images de telle manière que les lignes épi-polaires soient parallèles et horizontales, et ce processus est appelé la rectification.

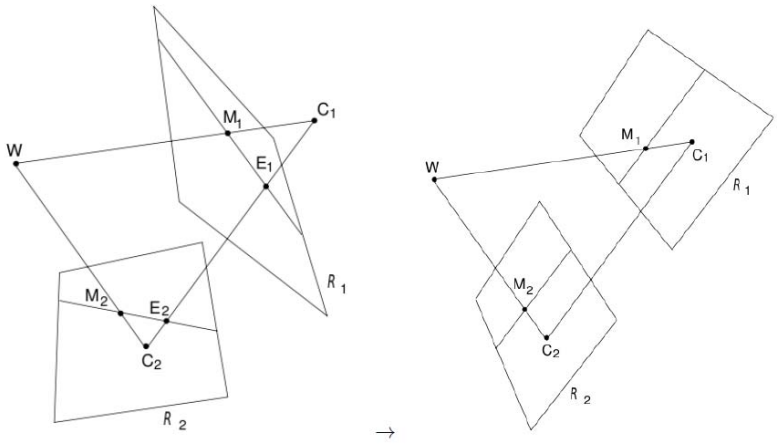


Figure 5-5 processus de rectification.

La figure ci-dessus illustre le processus de rectification les points E_1 et E_2 appelés les points épi-polaires des deux images. Toutes les lignes épi-polaires se croisent au point épipolaire d'une image donnée.

Cependant, quand les plans des rétines des deux caméras se trouvent sur le même plan, les points épi-polaires se déplacent à l'infini, et les lignes épipolaires dans une image deviennent parallèles les unes par rapport aux autres et sont parallèles à la ligne de base dont les extrémités sont les centres optiques des deux plans de la rétine. Cela signifie que si la base est parallèle à l'axe des X, les lignes épi-polaires sont horizontales.

V-3-3-1 Algorithme

Nous allons donner les étapes de base de l'algorithme de rectification dans ce qui suit :

1. D'abord les deux matrices des deux caméras sont séparément factorisées en trois parties:
   - La matrice interne A_0.
   - La matrice de rotation R_o qui donne la rotation entre le repère de référence de la camera et celui de l'espace 3D.
   - La matrice de translation t_0 entre le repère de référence de la camera et celui de l'espace 3D.

   Cela signifie $P = A_0 [R_o | t_0]$.

2. La nouvelle matrice de rotation R_n est construite, cette matrice permet de s'assurer que dans les nouveaux systèmes de référence des caméras, l'axe des x est parallèle à la ligne de base.

3. La nouvelle matrice interne A des caméras est construite.

4. Les nouvelles matrices des caméras sont maintenant P_{n1} = A [R | Rc1] et P_{N2} = A [R | Rc2]. Les centres optiques sont inchangés.

5. Une cartographie est créée, celle-ci associe les plans d'image originale de P_{o1}, P_{o2} à P_{n1} et P_{n2}. Parce qu'en général les pixels dans les nouvelles images ne correspondent pas aux positions des anciennes images, l'interpolation bilinéaire est utilisée pour combler les lacunes.

V-3-4 La mise en correspondance stéréo dense

La Stéréo dense combine les deux images rectifiées et prend la position des pixels de l'image gauche à l'endroit des pixels correspondant dans l'image droite, Avec cette méthode, on calcule la distance du pixel de la caméra. La profondeur est ensuite traduite en une carte de profondeur où la couleur des points proches de la caméra, sont presque blanc alors que les points qui sont éloignés tendent vers la couleur noire.

Les points entre les deux sont représentés en échelle de gris, qui s'assombrissent plus en s'éloignant de la caméra, le résultat est montré par la figure ci-dessous.

Figure 5-6 : mise en correspondance stéréo dense.

Il existe plusieurs algorithmes de stéréo dense :

- Graph Cut
- Believe Propagation
- Region Based / Block Matching
- programmation dynamique
- Block Matching
- Semi Global Block Matching

V-3-4-1 Mise en correspondance (Matching)

L'objectif principal d'un algorithme de stéréo dense est de mettre en correspondance un point d'une image avec un autre point qui lui correspond dans une autre image, pendant la mise en correspondance l'algorithme doit effectuer des taches. En effet, il compare les lignes épi-polaires des deux images pixel par pixel, pour chaque pixel d'une ligne il faut qu'il trouve son contrepartie sur la ligne épi-polaire correspondante dans l'autre image.

Souvent cet algorithme rencontre des obstacles, tels que :

- Le désordre des pixels.
- L'occlusion des pixels, comme c'est illustrer dans la figure ci-dessous, dans les deux exemples, où le pixel P_O est visible pour la caméra O_L, mais il est occlus dans la vue de caméra O_R,

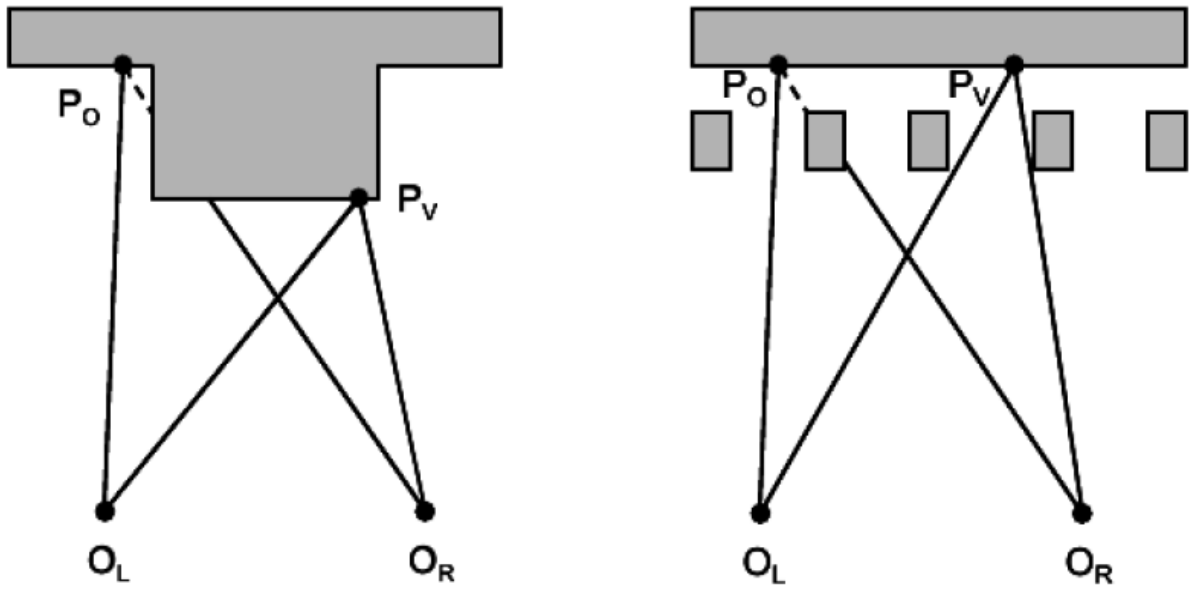


Figure 5-7 obstacles en mise en correspondance.

Dans notre projet on a opté pour l'algorithme Semi Global Block Matching .

V-3-4-2 Algorithme du Semi Global Block Matching

Cet algorithme est basé sur l'idée d'utiliser la moyenne des données des blocks de pixels comme énergie des pixels et ensuite mettre en place un éventail de calcul sur les coûts de toutes les disparités allant de la disparité minimale à la disparité maximale et ceci pour tous les pixels.

Les étapes de cet algorithme sont :

1- calcul du coût.
2- Agrégation du coût.

3- Calcul de disparité.
4- le raffinement des disparités.

Cet algorithme est implémenté dans la bibliothèque OpenCV, il est plus précis mais lent par rapport aux algorithmes citées auparavant, on doit régler des paramètres dans cet algorithme pour avoir une carte de profondeur optimal.

Les paramètres a réglées sont les suivants:

- le minimum et le maximum des disparités
- SAD size : c'est la taille des blocs pour le calcul de l'énergie de l'algorithme.
- PreFilterCap : est la valeur de la fonction de coût.
- disp12MaxDiff : est la valeur maximale pour la vérification de la disparité gauche-droite.
- speckleWindowSize : définit la région maximale qui est considérée comme chatoiement/ crête.
- speckleRange : définit la différence de disparité qui est considéré comme chatoiement.

V-3-5 Triangulation

Apres avoir effectué les trois premières étapes de la stéréo vision, on peut calculer la positon (x,y,z) d'un point P se trouvant dans l'espace 3D.

La triangulation fait usage d'un certain nombre de variables, les centres des rétines des deux caméras (C_1, C_2), la longueur focale (F) , les plans images (R_1,R_2), et les deux points (P_1, P_2) qui sont la projection du point P dans les deux images, V_1 et V_2 sont les distances horizontales entre P_1 et C_1,P_2 et C_2 respectivement et b la distance en mm qui sépare C_1 de C_2 comme le montre la figure ci-dessous.

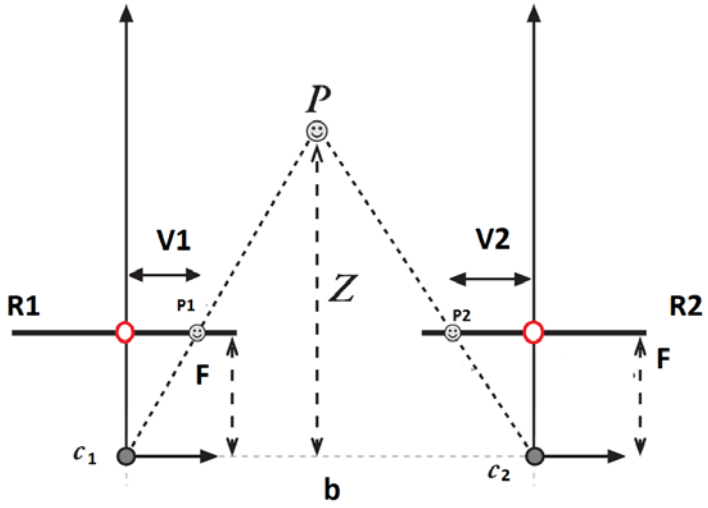


figure 5-8 Relation géométrique.

La disparité (D) des points P_1 et P_2 de l'image gauche et de l'image droite peut être calculée en prenant la différence entre V_1 et V_2. En utilisant cette disparité, on peut calculer la distance réelle du point dans l'environnement réel à partir des deux images. Les formules suivantes peuvent être déduites de la forme géométrique ci-dessus:

$$z = \frac{b \cdot F}{D}$$

$$x = \frac{V_1 \cdot z}{f}$$

$$y = \frac{y_1 \cdot z}{f}$$

Où y_1 est la position de P_1 selon l'axe des Y dans l'image gauche.

V-4 Reconnaissance d'objet

La reconnaissance d'objets en vision par ordinateur est la tâche de trouver un objet donné dans une image ou une séquence vidéo. Les humains reconnaissent une multitude d'objets dans des images avec peu d'effort, malgré le fait que l'image des objets peut varier quelque peu dans les points de vue différents, même avec beaucoup de différentes tailles / échelles ou quand ils sont avec une rotation.

Les objets peuvent même être reconnus quand ils sont partiellement obstrués à la vue, cette tâche est encore un défi pour les systèmes de vision par ordinateur en général.

La bibliothèque OpenCV offre nativement une démonstration très intéressante pour la détection de visages. Cette détection est basée sur un système de détection des visages appelé HaarTraining, qui est un système basé sur le travail Viola & Jones, ou ce qu'on appelle la

détection d'objets par descripteurs bas-niveau. Nous avons essayé d'adapter ces méthodes, ou ce qu'on appelle un classifié, pour reconnaitre des boites cubiques où une marque de chaussures est illustrée sur leurs faces supérieures à la place des visages.

La première étape est la création d'une base de donnée d'échantillons « positifs » et d'échantillons « négatifs » afin de permettre au système de faire des comparaisons statistiques entre ces échantillons et l'image actuelle acquise par la camera ce qui permet au système de reconnaitre notre objet.

1. Images Positives :

Nous avons besoin de collecter des images positives qui contiennent les objets d'intérêt.
Intel mentionne qu'ils ont utilisé plus de 5000 modèles positifs de visages en face frontale, et que plus de 5000 modèles autres positifs de visages en face frontale, qui ont été dérivées à partir de 1000 visages d'origine.

On peut obtenir des images d'objets réels à partir de plusieurs sources sur internet ou on peut les créer nous-mêmes. Il faut veiller néanmoins à ce que ces images soient les plus différentes possibles en changeant la surface et l'éclairage sous un large éventail de conditions.

De plus, l'emplacement de la caméra doit varier pour avoir plusieurs angles de l'objet. Sur chaque image positive, nous devons « marquer » les objets qui nous intéressent par un cercle ou un rectangle qui pourra être identifié durant les prochains traitements , à savoir que ces modèles devront être divisés en deux groupes, un pour l'apprentissage (environ 4000 modèles) et un pour le test (environ 1000 modèles).

2. Images négatives :

Pour les images négatives il suffit de veiller à ce qu'elles ne contiennent pas l'objet ou des objets leur ressemblant.
Un minimum de 5000 modèles négatifs est exigé, avec une préférence pour les images non-compressée tel que BMP.
A savoir que ces modèles devront être divisés en deux groupes, un pour l'apprentissage (environ 4000 modèles) et l'autre pour le test (environ 1000 modèles).

3. Apprentissage :

Une fois que les modèles positifs et négatifs sont prêts, on peut lancer le processus d'apprentissage du système.
Le processus d'apprentissage vise à doter le système de la capacité à détecter l'objet sous différents angles et différents environnements. Ceci est fait à travers l'application haartraining fournie avec OpenCV.

Le système analyse les deux types de modèles en plusieurs itérations, un minimum d'une

vingtaine, pour améliorer sa précision, puis, générera un fichier XML qui lui servira de base de décision par la suite.

Ce processus est extrêmement lent et consommateur de ressources, il devra donc être réaliser sur des machines adéquates dont la configuration minimale est :

- 2GHz en processeur, de préférence dual core pour ajouter plus de puissance.
- 2Go en Mémoire.

4. Détection

Le classificateur peut détecter les objets sur des images statiques ou sur un flux vidéo.

Si les résultats sont insatisfaisants, il suffit alors de supprimer le fichier XML de la base et refaire au système un nouvel apprentissage. On peut aussi avoir besoin de détecter plusieurs objets, dans ce cas, il suffit de faire plusieurs sessions d'apprentissage, une pour chacun de ces objets.

V-5 Conclusion

Nous avons présentés dans ce chapitre une brève description sur la reconnaissance d'objet et l'approche la stéréovision passive, qui est utilisée dans plusieurs domaine telle que la reconstruction 3D de l'environnement, le calcul de la position d'un objet connu en 2D (notre cas) ou 3D, reste que cette approche est moins précise, par rapport à la stéréovision active qui est considéré actuellement comme la meilleure solution pour obtenir de bons résultats au niveau de la vision par ordinateur ou bien la vision robotique.

VI-1 Introduction

De manière générale, les applications robotiques considérées, consistent à déplacer l'effecteur d'un robot pour qu'il atteigne une cible. Dans notre projet, nous avons commandé la pince du robot en lui envoyant la position x,y,z de l'objet qu'on a extraite par l'intermédiaire de la stéréovision, de façon a ce que la pince puisse atteindre cet objet et l'attraper, ce chapitre inclut toutes les étapes utilisées pendant la pratique ainsi que l'application et la commande du robot, pour cela nous allons dialoguer sur le logiciel que nous avons développé puis sur l'utilisation de la carte Arduino qui commande les servomoteurs RC.

VI-2 Robot/système de commande

Le plan de notre projet se présente comme suit

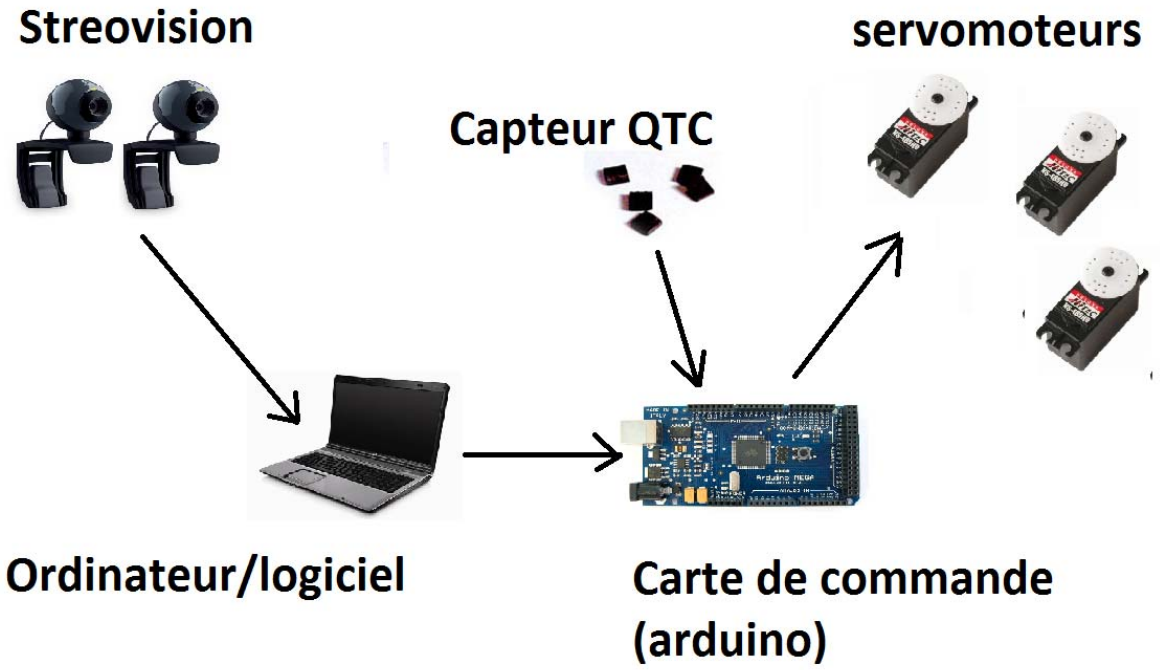


Figure 6-1 : Vue d'ensemble du projet

VI-3 Logiciel I-Kersoft

Le logiciel que nous avons programmé en C/C++ en mode console sous Linux ubuntu 10.04 est composé de deux parties : Partie vision et partie transmission.

VI-3-1 Partie vision

Après avoir ajusté manuellement nos deux webcams de façon à ce qu'elles soient parallèles, et régler leurs focaux de façon qu'ils soient approximativement identiques, nous avons appliqué les étapes de la stéréovision comme nous l'avons expliqué dans le chapitre précédent.

La partie vision du logiciel contient quatre sous programmes en mode console :

A - Premier sous programme

Capture de paires d'images gauche et droite simultanément:

Nous avons utilisé un échiquier de 10x7 carreaux, nous avons pris soixante cinq paires d'images de l'échiquier en niveau de gris par les deux cameras avec différentes positions (translation et rotation), comme s'est demandé dans la fonction de la calibration stéréoscopique d'OpenCV, les deux fonctions ci-dessous nous permettent de capturer deux images au même temps.

```
cvSaveImage( imagenameL, image_grayL);
cvSaveImage( imagenameR, image_grayR);
```

Le résultat est comme suit :

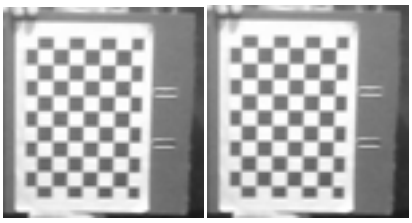


Figure 6-2 Captures simultanées des deux images gauche et droite

B - Deuxième sous-programme

Calibration et rectification :

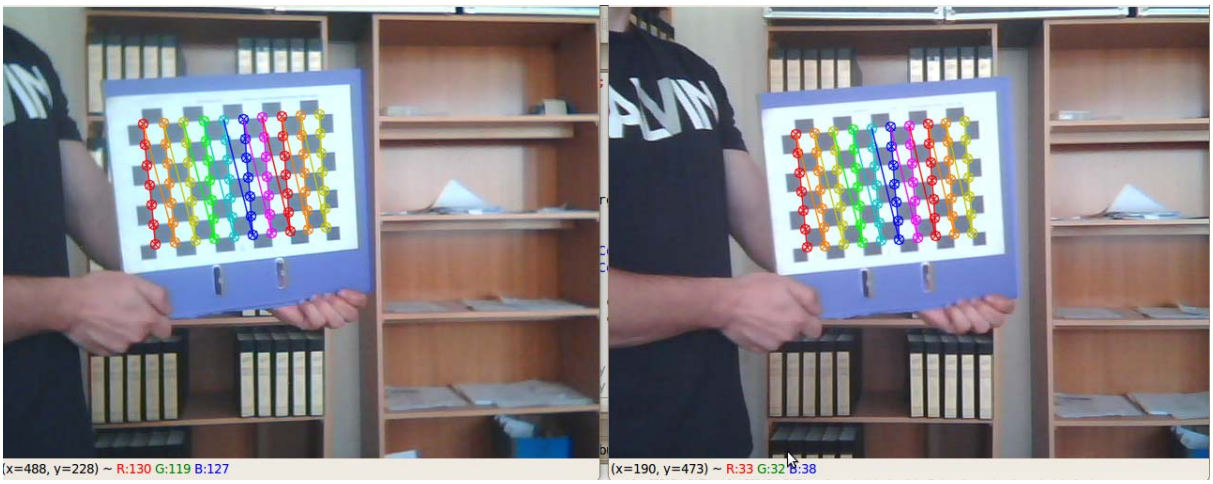
Les deux webcams sont physiquement et approximativement alignées de telle sorte que chaque webcam ait essentiellement le même champ de vision que l'autre. Cela permet d'éviter les problèmes épipolaires et de maximiser la zone de chevauchement stéréo tout en minimisant la distorsion de projection.

Nous avons mis ces paires d'images dans un dossier spécifique où nous avons créé un fichier texte nommé images.txt contenant en premier le nombre de case horizontales et verticales de l'échiquier et les noms des paires d'images par alternance (image gauche x et image droite x), qui sont utilisés pour calibrer les caméras, et ensuite rectifier les images comme c'est montré sur la figure ci-dessous :

```
10 7
left000.bmp
right000.bmp
left001.bmp
right001.bmp
left002.bmp
right002.bmp
left005.bmp
right005.bmp
left006.bmp
right006.bmp
left007.bmp
right007.bmp
left008.bmp
right008.bmp
left009.bmp
right009.bmp
left010.bmp
right010.bmp
left011.bmp
right011.bmp
left012.bmp
right012.bmp
left013.bmp
right013.bmp
left014.bmp
right014.bmp
```

Figure 6-3 Contenus du fichier image.txt

Le programme lit d'abord la taille de l'échiquier et les paires d'images gauches et droites, ensuite il détecte les coins de l'échiquier en se basant sur les sous-pixels des coins détectés, après il définit les points principaux qui constituent les coins où tous les échiquiers peuvent être trouvés (différentes translation et rotation de l'échiquier), le résultat de ce processus est affiché comme suit:



Camera gauche camera droite

Figure 6-4 Détection de coins de l'échiquier

En tenant compte de cette liste de points trouvés dans les images de l'échiquier, le programme appelle la fonction cvStereoCalibrate () pour calibrer les webcams.

Le résultat de cette fonction de calibration nous donne la matrice de la caméra _M , le vecteur de distorsion _D pour les deux caméras, et aussi la matrice de rotation _R, le vecteur de translation _T , la matrice essentielle _E, et la matrice fondamentale _F. Ces résultats sont sauvegardés dans deux fichiers différents, extrinsics.yml et intrinsics.yml, comme c'est illustré ci-dessous :

```
%YAML:1.0
R: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 9.9988902878869390e-01, -1.2074563670773128e-02,
       8.7255383881445839e-03, 1.2050261133178541e-02,
       9.9992338128778269e-01, 2.8324478158997242e-03,
       -8.7590704201271040e-03, -2.7269884796299166e-03,
       9.9995792022424990e-01 ]
T: !!opencv-matrix
   rows: 3
   cols: 1
   dt: d
   data: [ -4.3445421499685324e+00, -3.0759385635786923e-01,
       -1.0474609267317361e+00 ]
```

Figure 6-5 extrinsics.yml

```
%YAML:1.0
M1: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 6.462480405545654 8e+02, 0., 3.231317154126469 4e+02,
0.,
       6.462480405545654 8e+02, 2.400727059822820 9e+02, 0., 0.,
1. ]
D1: !!opencv-matrix
   rows: 1
   cols: 5
   dt: d
   data: [ 2.386411539633786 8e-01, -4.019193406579779 6e-02,
0., 0., 0. ]
M2: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 6.462480405545654 8e+02, 0., 3.270475683148394 5e+02,
0.,
       6.462480405545654 8e+02, 2.408355240395547 7e+02, 0., 0.,
1. ]
D2: !!opencv-matrix
   rows: 1
   cols: 5
   dt: d
   data: [ 3.296578764035490 4e-01, -3.132956327800410 8e-01,
0., 0., 0. ]
```

Figure 6-6 intrinsics.yml

Ensuite le programme s'exécute et met les deux cameras en marche et en mode vidéo afin que le processus soit en temps réel et à une fréquence de 30 image par seconde.

Pour évaluer la précision du calibrage, le programme doit vérifier de près si un point d'image de gauche est sur la même ligne épipolaire du même point de l'image droite.

Pour ce faire, le programme doit annuler les distorsions déformantes les points d'origine des deux images en utilisant la fonction cvUndistortPoints (). Pour calculer cette relation point-ligne nous avons utilisé la fonction cvComputeCorrespondEpilines (), afin de calculer le produit des points avec les lignes (dans le cas idéal, le résultat est nul), la distance absolue cumulée sous forme des erreurs.

Le programme calcule une carte de rectification en temps réel en utilisant l'algorithme de Bouguet avec la fonction d'OpenCV nommée cvStereoRectify().

Les images rectifiées sont ensuite calculées en utilisant cvRemap (). Dans notre cas, les lignes sont dessinées entre les paires d'images de façon à ce que les images corrigées soient alignées. Le résultat est montré par la figure ci-dessous, où nous pouvons voir que les images originales sont en grande partie corrigées de haut en bas.

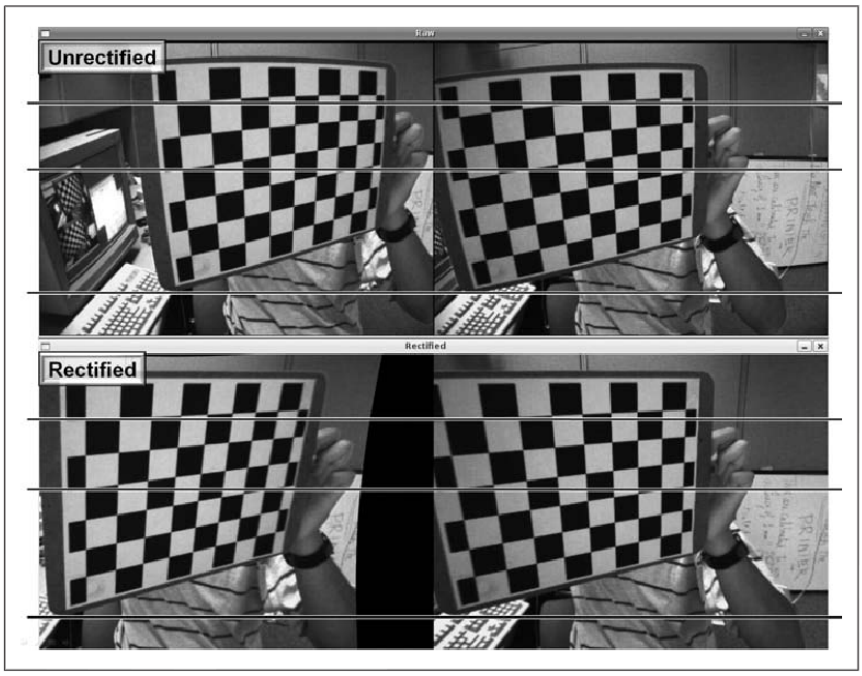


Figure 6-6 Images avant et après rectification

C - Troisième sous programme

Après que le programme ait rectifié les images, il doit initialiser l'algorithme SGBM qui se trouve dans la bibliothèque OpenCV comme une classe, on commence par la déclaration de l'objet comme suit :

```
StereoSGBM sgbm;
```

Ensuite on doit régler les paramètres de notre algorithme pour avoir une carte de disparité optimale. Après plusieurs tests, nous nous sommes satisfaits des réglages suivants :

```
sgbm.preFilterCap = 63;

sgbm.SADWindowSize =7;

sgbm.P1 = 8*cn*sgbm.SADWindowSize*sgbm.SADWindowSize;

sgbm.P2 = 32*cn*sgbm.SADWindowSize*sgbm.SADWindowSize;

sgbm.minDisparity = 0;

sgbm.numberOfDisparities =96;

sgbm.uniquenessRatio = 0;

sgbm.speckleWindowSize =100;

sgbm.speckleRange = 32;

sgbm.disp12MaxDiff =0;
```

sgbm.fullDP =true;

Après avoir introduit manuellement ces paramètres dans le code de notre programme, on peut alors calculer la carte de disparité en utilisant la fonction :

sgbm(imgL, imgR, disp) ;

Où imgL et imgR sont les images rectifiées des deux cameras en temps réel, et disp est le résultat sortant qui représente la carte de disparité au niveau de gris, comme le montre la figure ci-dessous.

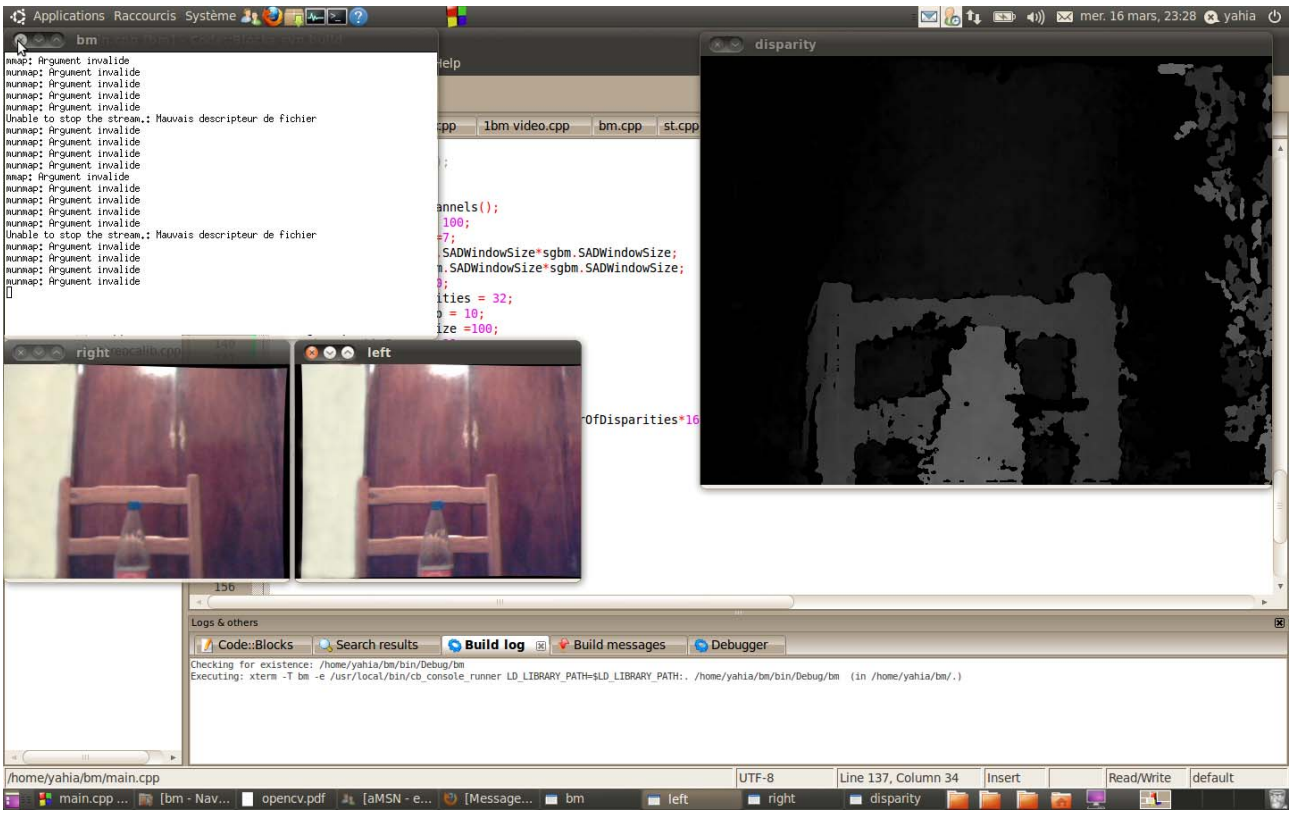


Figure 6-7 Carte de disparité

D – Quatrième sous-programme

Détection d'objet et triangulation **:**

Détection d'objet

Pour la détection d'objet nous avons suivi les étapes proposées dans la bibliothèque OpenCV comme suit :

Collections d'images

Images négatives

Nous avons collecté 5000 images sur lesquelles notre objet n'apparait pas. Après les avoir mises dans un dossier nommé négative, nous avons créé un fichier texte nommé negative.txt contenant les noms des fichiers des images négatives comme suit:

```
negative/neg-0002.jpg
negative/neg-0003.jpg
negative/neg-0004.jpg
negative/neg-0005.jpg
negative/neg-0006.jpg
negative/neg-0007.jpg
negative/neg-0009.jpg
negative/neg-0010.jpg
negative/neg-0011.jpg
negative/neg-0012.jpg
negative/neg-0013.jpg
negative/neg-0015.jpg
negative/neg-0016.jpg
negative/neg-0019.jpg
negative/neg-0020.jpg
negative/neg-0022.jpg
negative/neg-0023.jpg
negative/neg-0024.jpg
negative/neg-0025.jpg
negative/neg-0026.jpg
negative/neg-0027.jpg
negative/neg-0028.jpg
negative/neg-0029.jpg
negative/neg-0030.jpg
negative/neg-0031.jpg
negative/neg-0034.jpg
negative/neg-0035.jpg
```

Figure 6-8 Une partie du contenu du fichier negative.txt

Images positives

Nous avons fait la même chose pour les images positives. Néanmoins, à cause de la lourdeur du calcul, nous nous somme contentés de n'utiliser que huit images où le logo que nous avons choisi comme objet à détecter apparait dans différentes positions en deux dimensions ; cette figure montre les images positives:

Figure 6-9 Images positives

Le fichier texte est comme suit :

positive/adidas1.jpg
positive/adidas2.jpg
positive/adidas3.jpg
positive/adidas4.jpg
positive/adidas5.jpg
positive/adidas6.jpg
positive/adidas7.jpg
positive/adidas8.jpg

Apprentissages par échantillons (de référence)

OpenCV est dotée d'une fonction qui permet de faire des apprentissages à partir des images sans appliquer les distorsions. Cette fonction est cvCreateTestSamples, elle s'exécute lorsque les options, -info et -vec sont spécifiées. En effet nous avons d'abord convertis les deux fichiers texte en extension .dat :

```
find ../../data/negative/ -name '*.jpg' > negative.dat
find ../../data/positive/ -name '*.jpg' > positive.dat
```

Ensuite :

```
$perl createtestsamples.pl positive.dat negative.dat tests 1000 "./createsamples -bgcolor 0 -bgthresh 0 -maxxangle 1.1 -maxyangle 1.1 -maxzangle 0.5 maxidev 40"
$ find tests/ -name 'info.dat' -exec cat \{\} \; > tests.dat
```

Ce qui nous donne une fusion des images positives et négatives afin d'avoir une quantité de 1000 images positives dans différents environnements des images négatives.

Application de l'algorithme HAARTRAINING

Cette application est décrite par cette instruction :

$ haartraining -data haarcascade -vec samples.vec -bg negative.dat -nstages 20 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 7000 -nneg 3019 -w 20 -h 20 -nonsym -mem 512 -mode ALL

Générer un fichier XML

Ce fichier doit contenir toutes les caractéristiques de l'objet sous forme d'un fichier XML, pour ce faire nous exécutons l'application suivante :
$ convert_cascade --size="20x20" haarcascade haarcascade.xml

Pour cause de lourdeur de calcul, nous avons opté pour remplacer le logo par un visage, vu qu'OpenCV offre un exemple déjà calculé. Pour implémenter cet objet dans notre programme nous avons fait appel à la classe CascadeClassifier pour créer un objet nommé cascade comme suit :
CascadeClassifier cascade1;

Ensuite nous avons chargé le fichier xml offert par OpenCV « haarcascade_frontalface_alt.xml » dans le corps du programme.

Triangulation

Après que le logiciel ait détecté l'objet dans les deux cameras, nous l'avons programmé de façon à ce que l'objet détecté soit encadré dans les images des deux cameras, ensuite nous avons calculé le centre du cadre qui est le point P dont on veut extraire la position en stéréovision, après le calcul de positon du point P par la triangulation, le logiciel envoie les valeurs de x,y,z de l'objet en temps réel vers la carte de commande Arduino.

VI-3-2- Partie transmission

VI-3-2-1- Interfaçage logiciel/Arduino

Afin d'envoyer la position calculée de l'objet par notre logiciel vers la carte de commande Arduino, nous avons opté pour la bibliothèque libSerial en C++ qui nous permet d'accéder aux ports séries sur les systèmes POSIX, ses fonctions nous permettent de régler des paramètres différents du port série telle que la vitesse de transmission, la taille des caractères, le contrôle du flux sortant et entrant.

Pour cela nous allons montrer les étapes d'implémentation de cette bibliothèque dans notre programme :

Tout d'abord, nous avons importé cette bibliothèque, et nous avons définis le port qui est relié à Arduino

```
#include SerialStream.h

#include iostream

#define PORT "/dev/ttyUSB3"

SerialStream ardu;

using namespace std;

using namespace LibSerial;
```

Ensuite nous avons synchronisé le logiciel de façon à ce qu'il se communique avec la même vitesse de transmission qu'Arduino, l'algorithme est comme suit :

```
void open () {

   ardu.Open(PORT);
   ardu.SetBaudRate(SerialStreamBuf::BAUD_115200);
   ardu.SetCharSize(SerialStreamBuf::CHAR_SIZE_8);
}
```

Ensuit vient l'envoi des coordonnées x,y,z vers Arduino.

VI-4- Carte de commande Arduino

Pour déplacer la pince vers la position de l'objet, nous avons programmé la carte de commande afin qu'elle traduise les coordonnées cartésiennes x,y,z de l'objet en coordonnées angulaires (θ_i) des six articulations du robot pour que celles-ci soient traduites en signaux PWM, cette transformation de coordonnées se fait grâce aux résultats du modèle géométrique inverse calculé dans le chapitre trois.

La carte Arduino Mega peut commander douze servomoteurs RC au même temps avec une fréquence d'horloge de 16 MHz. Pour ce faire nous avons programmé cette carte afin qu'elle communique avec l'ordinateur en utilisant la même vitesse de transmission. Ainsi, la fonction de la bibliothèque Serial incluse dans l'IDE de Arduino comme suite:

```
Serial.begin(115200);
```

Ensuite pour que la carte de commande reçoive les coordonnées de l'objet en temps réel nous avons ajouté la fonction :

Serial1.read();

Nous avons utilisé la bibliothèque Servo qui permet de gérer les signaux PWM vis à vis des angles calculés grâce au modèle géométrique inverse. En outre, nous avons déclaré les servomoteurs utilisés dans notre robot et nous avons initialisé le programme de la carte avec les résultats obtenus dans la chapitre 3 comme suit :

Servo base;
Servo epaule1;
Servo epaule2;
Servo coude1;
Servo coude2;
Servo poignet1;
Servo poignet2;
Servo poignet3;
Servo pince;

$\theta_i =$
$\theta_i =$
$\theta_i =$
$\theta_i =$
$\theta_i =$
$\theta_i =$

Pour actionner ces servomoteurs la commande utilisée sur Arduino est :

servo.write(pos);

A fin que la pince puisse attraper l'objet nous avons ajouté une condition dans notre algorithme de façon à ce que la pince se ferme jusqu'à ce qu'elle ait pris l'objet et ce grâce aux capteurs QTC qui se trouvent sur les deux parties intérieurs de la pince.

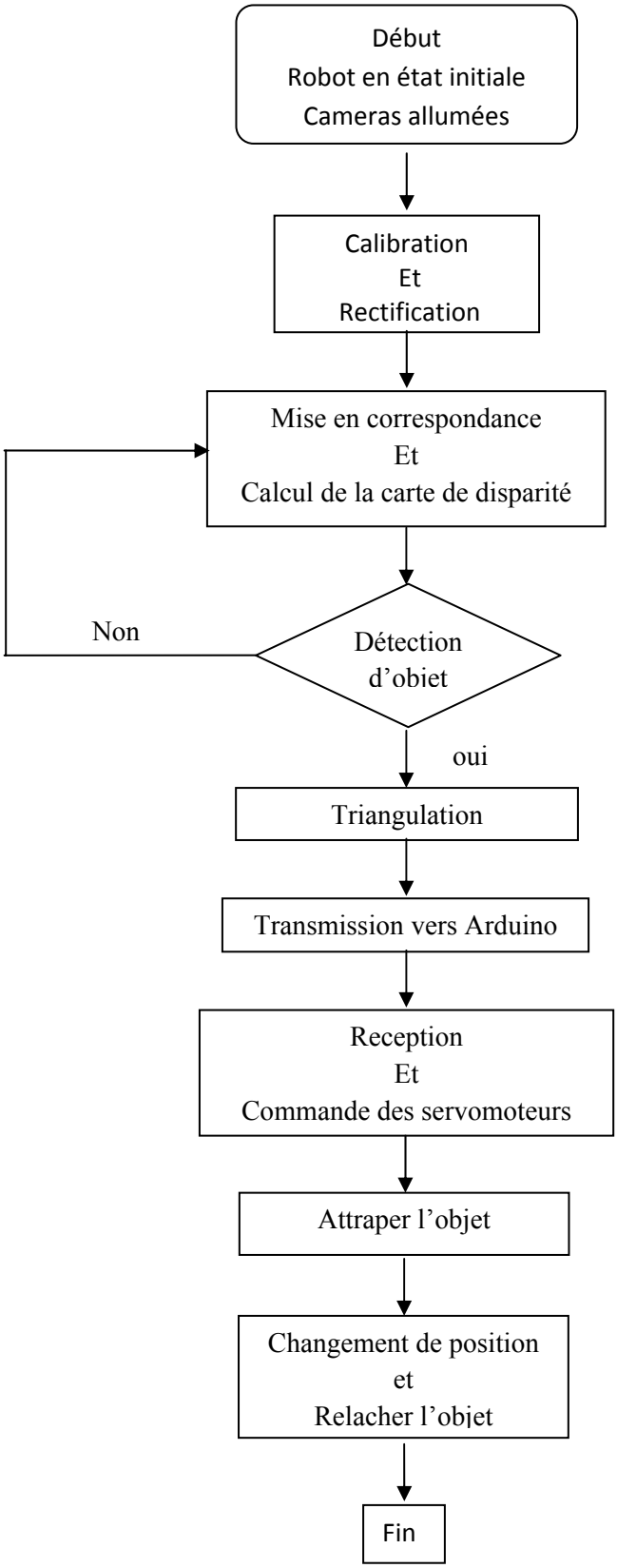
Après ça, comme c'est mentionné dans le premier chapitre la pince change de direction en allant au point ou elle doit relâcher l'objet et finalement la pince retourne à son état initial.

Remarque

Les servomoteurs sont dotés d'une boucle fermée interne illustrée dans la partie annexe, ce qui nous a posé un inconvénient au niveau de l'asservissement, car les servomoteurs ne renvoient pas la position atteinte vers Arduino.

VI-5- Organigramme logiciel/Arduino

Afin de résumer notre projet, un organigramme expliquant toutes les étapes que le logiciel et Arduino effectuent est donné comme suit :


Début
Robot en état initiale
Cameras allumées
Calibration
Et
Rectification
Mise en correspondance
Et
Calcul de la carte de disparité
Non
Détection
d'obiet
oui
Triangulation
Transmission vers Arduino
Reception
Et
Commande des servomoteurs
Attraper l'objet
Changement de position
et
Relacher l'objet
Fin

Ce mémoire est le résultat d'un travail de recherches et de conception consacré à la commande assistée par la vision du robot I-Kernel et la spécification des tâches pouvant être effectuées par ce robot.

Dans ce travail ; ont été réalisés la présentation, la structure mécanique, la modélisation géométrique, la vision par ordinateur et son application pour la commande du robot, à travers l'exécution de certaines tâches une validation expérimentale partielle est réalisée.

Dans le premier chapitre nous avons présenté de manière générale le corps du robot I-Kernel et son système de commande en donnant les caractéristiques techniques et les tâches qu'il peut effectuer. Dans le deuxième chapitre nous avons donné plus de détails sur la structure mécanique du robot et ce en donnant les schémas et les modèles CAO de chacune de ses parties ainsi que les matériaux utilisés sur chacune d'elles. Dans le troisième chapitre il a été question d'élaborer les modèles géométriques directe et inverse selon la convention de Denavit - Hartenberg. Afin que le robot accomplisse sa tâche il a fallut lui intégrer des actionneurs en l'occurrence des servomoteurs RC et des capteurs ce qui a été l'objet du chapitre quatre de notre projet. Comme un robot n'en est un que s'il est commandé nous avons intégré une commande assistée par la vision dont on a exposé les gros titres dans le chapitre cinq avant de donner plus de détails et l'intégrer à notre système dans le chapitre six.

Un long travail a été nécessaire pour aboutir aux résultats mentionnés, car il a fallut concevoir un système en partant d'une idée personnelle qu'il fallait perfectionner au fur et à mesure qu'on avançait dans le projet, et vu le temps repartit pour cela certains résultats n'ont pas pu être confirmés au laboratoire tel que l'intégration de la vision au robot même si elle a bien fonctionné séparément.

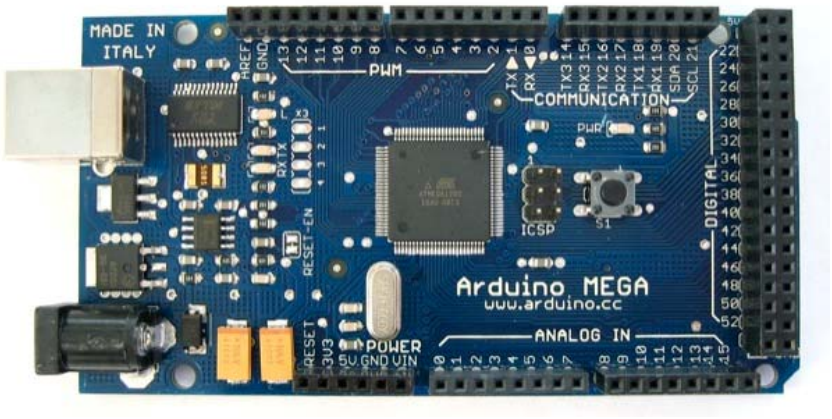
Les difficultés rencontrées au cours de ce projet se situent au niveau matériel ; en effet les servomoteurs de l'épaule n'ont pas pu soulever le poids du robot, ce qui nous a contraints à fixer cette articulation pendant les essais. Le système de commande n'étant pas assez puissant il n'a pas pu supporter l'algorithme de la stéréovision et détection d'objets en temps réel.

Pour l'intégration des capteurs QTC à la pince, une colle spéciale est nécessaire mais elle est indisponible.

Dans le prolongement direct de ce travail plusieurs études sont à envisager :

- Remplacer l'approche de la stéréovision passive par une stéréovision active.
- Utilisation de la reconnaissance d'objet par la technologie GPU.
- Modification des servomoteurs afin d'avoir un retour de position, de vitesse, accélération et de couple.
- Intégrer une commande par neuro-flou.
- Remplacer le système de commande par un système embarqué (FPGA).
- Ajouter un asservissement visuel.

Arduino Mega :



vue d'ensemble :

Arduino Mega est une carte de commande basée sur le microcontrôleur ATmega1280. Elle dispose de 54 entrées / sorties numériques (dont 14 peuvent être utilisées comme sorties PWM), 16 entrées analogiques, 4 UART (ports série matériels), un oscillateur en cristal de 16 MHz, une connexion USB, une prise d'alimentation, un connecteur ICSP et un bouton de réinitialisation.

Programmation :

Arduino Mega peut être programmée avec le logiciel Arduino , Le ATmega1280 sur Arduino Mega vient avec un bootloader qui permet de télécharger un nouveau code sans l'utilisation d'un programmeur externe. Elle communique avec le protocole STK500 .

On peut également contourner le bootloader et le programme du microcontrôleur grâce à l'ICSP (In-Circuit Serial Programming).

La programmation de cette carte est composée de deux fonctions principales :

setup () :
Cette fonction est appelée des l'exécution du programme. Elle a pour fonction d'initialiser les variables, indiquer les modes des broches, déclarer les bibliothèques, cette fonction s'exécute une seule fois, après chaque mise sous tension ou la réinitialisation de la carte Arduino.

loop () :
Cette fonction permet au programme de se répéter en forme de boucle principale .

Alimentation :

Arduino Mega peut être alimentée via la connexion USB ou avec une alimentation externe. La source d'alimentation est automatiquement sélectionnée.

La plage de tension recommandée est de 7 à 12 volts, Les broches d'alimentation sont les suivantes:

• VIN. La tension d'entrée de la carte Arduino quand c'est une source d'alimentation externe (par opposition à 5 volts de la connexion USB ou tout autre source d'énergie réglementés).

• 5V. L'alimentation régulée utilisée pour alimenter le microcontrôleur et les autres composants sur la carte. Cela peut provenir soit de VIN via un régulateur de bord, ou par l'intermédiaire d'un port USB ou d'une autre alimentation 5V régulée.

• 3V3. Une alimentation de 3,3 volts générée par un le circuit intégré FTDI, avec un courant maximal de 50 mA.
• GND. Broches de masse.

Mémoire :

Le ATmega1280 a 128 Ko de mémoire flash pour stocker le code (dont 4 Ko est utilisée pour le bootloader), 8 Ko de SRAM et 4 Ko de mémoire EEPROM (qui peuvent être lues et écrites avec la bibliothèque EEPROM).

Entrée et sortie :

Chacune des 54 broches numériques sur la Mega peut être utilisée comme une entrée ou une sortie, en utilisant les fonctions pinMode (), digitalWrite (), et digitalRead (). Elles fonctionnent sous 5 volts. Chaque broche peut fournir ou recevoir un maximum de 40 mA et a une résistance de pull-up interne (déconnecté par défaut) de 20-50 kOhms. En outre, certaines broches ont des fonctions spécialisées:

• Série: 0 (RX) et 1 (TX), 19 (RX) et 18 (TX), 17 (RX) et 16 (TX), 15 (RX) et 14 (TX). (RX) Permettent de recevoir et (TX) pour la transmission des données en série . Pins 0 et 1 sont également connectés aux broches correspondantes du FTDI USB-TTL .

• interruptions externes: 2 (interruption 0), 3 (interruption 1), 18 (interruption de 5), 19 (interruption 4), 20 (interruption de 3) et 21 (interruption 2).
Ces broches peuvent être configurées pour déclencher une interruption sur une faible valeur, un front montant ou descendant, ou un changement de valeur.

• PWM: 0 à 13. Fournissent une sortie en PWM à 8-bit avec la fonction analogWrite () .

• SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS), ces broches SPI sont également réparties sur le connecteur ICSP, qui est physiquement compatible avec la Duemilanove et Diecimila.

• I2C: 20 (SDA) et 21 (SCL). Protocole de communication I2C utilisant la librairie Wire.

Arduino Mega possède 16 entrées analogiques, chacune d'elles fournit 10 bits de résolution (1024 valeurs différentes).

• AREF. Tension de référence pour les entrées analogiques, utilisée avec analogReference.

• Reset : bouton de réinitialisation du microcontrôleur.

Communication :

Arduino Mega peut communiquer avec un ordinateur,ou une autre Arduino, ou d'autres microcontrôleurs. Le ATmega1280 fournit quatre UART, un FT232RL FTDI fournissant un port COM virtuel.

Le logiciel Arduino comprend un moniteur qui permet d'envoyer des données textuelles vers et à partir de la carte Arduino.

Les LEDs RX et TX sur la carte clignotent lorsque des données sont transmises via le circuit integré FTDI et d'une connexion USB à l'ordinateur (mais pas pour la communication série sur les broches 0 et 1).

Une bibliothèque (SoftwareSerial) permet la communication série sur l'une des broches numériques de la Mega.

Le ATmega1280 supporte également la communication I2C (TWI) et SPI. Le logiciel comprend une bibliothèque (Wire) à fin de simplifier l'utilisation des bus I2C.

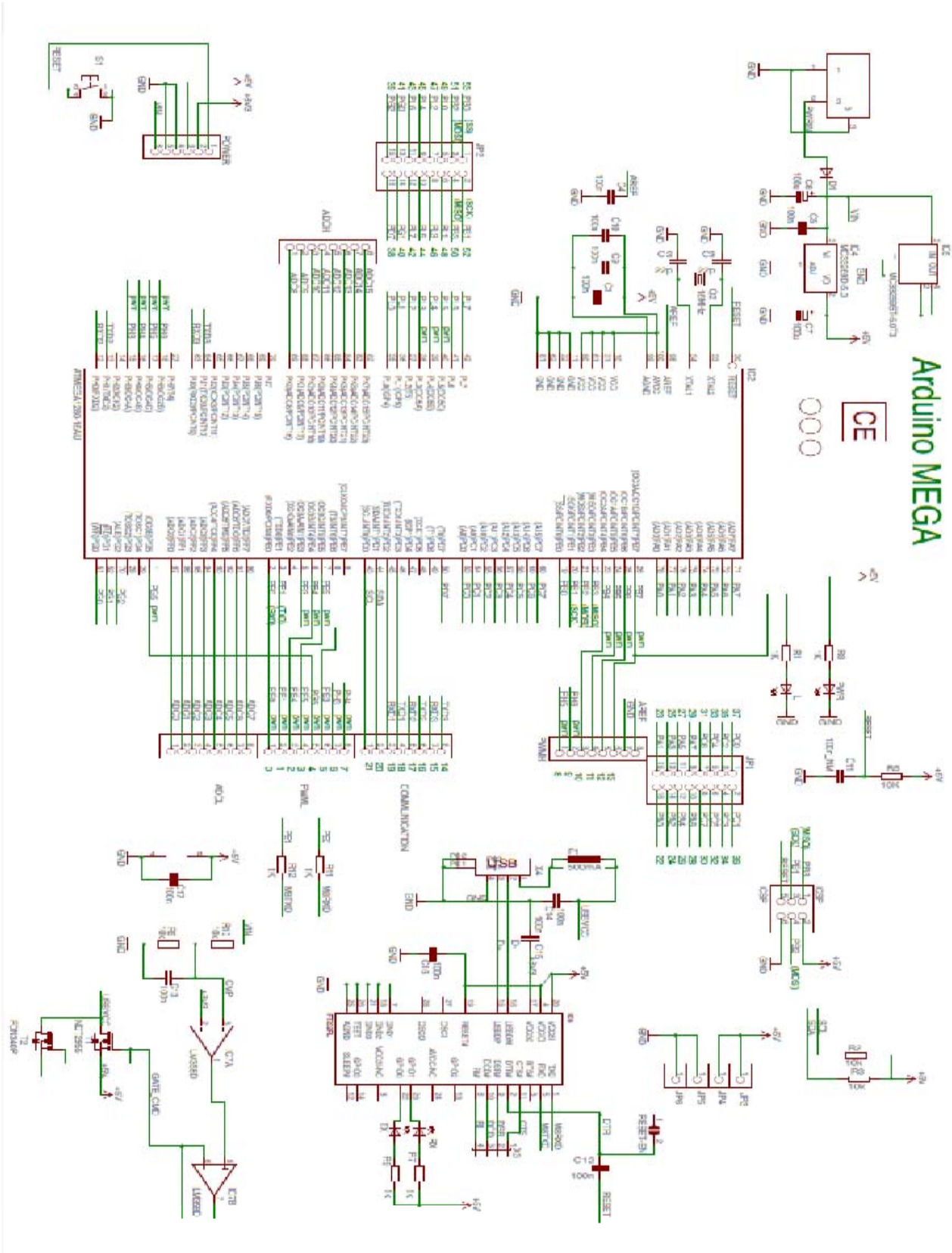
La Bibliothèque Servo :

Cette bibliothèque permet à une carte Arduino de contrôler des servomoteurs RC, elle supporte jusqu'à 48 servomoteurs, l'utilisation de la bibliothèque désactive la fonction analogWrite () .

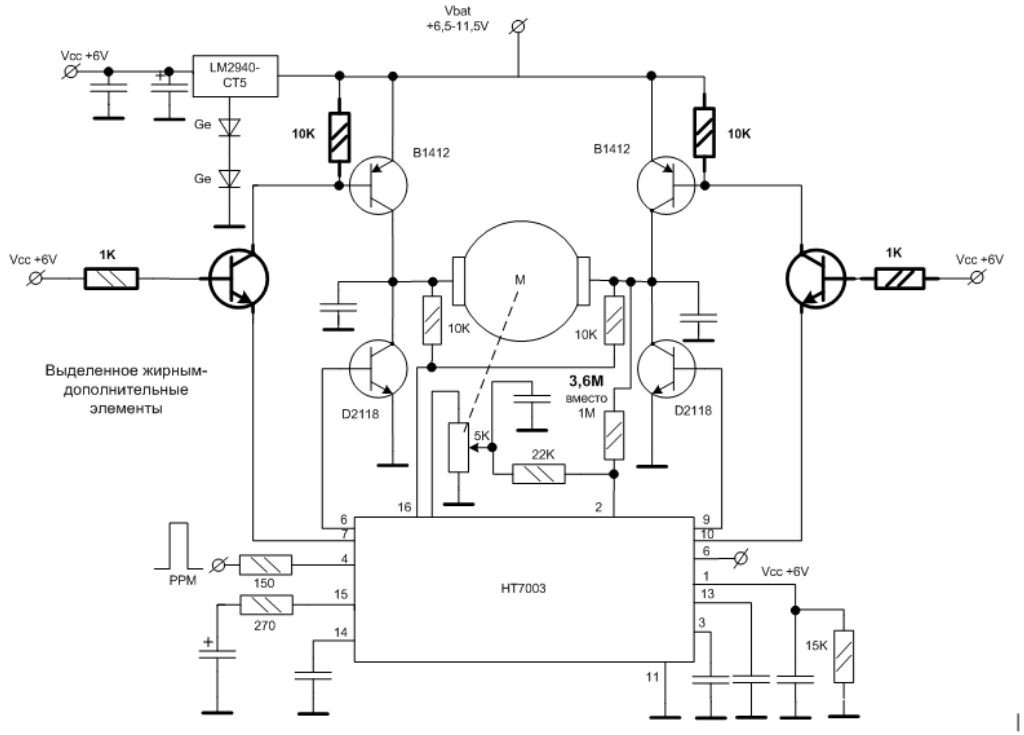
Ses fonctions sont :

- attach()
- write()
- writeMicroseconds()
- read()
- attached()
- detach()

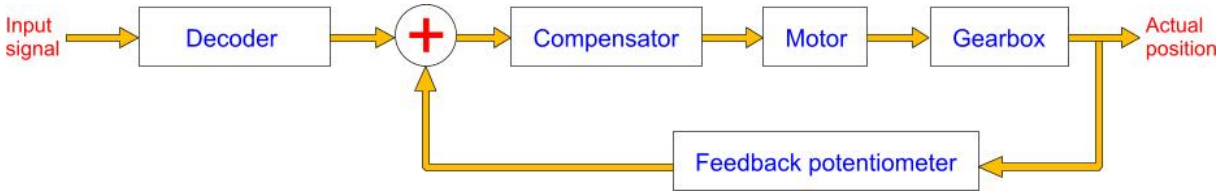
Schéma électrique :

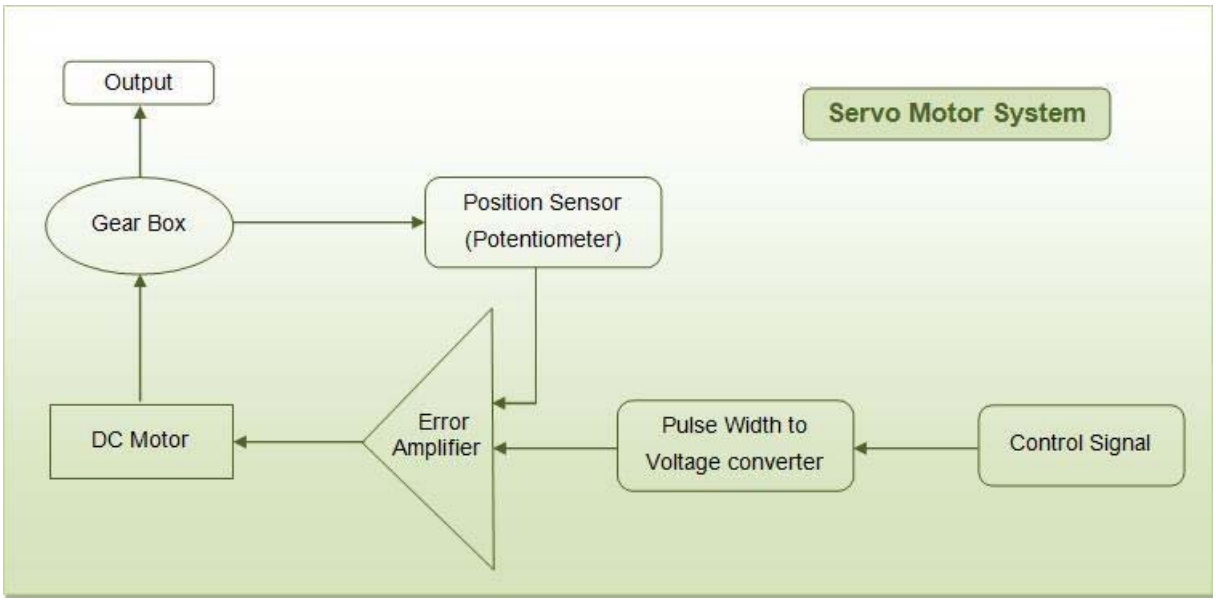


Circuit électronique d'un servomoteur de gamme Hitec:



Schémas simplifié d'un servomoteur RC :





OpenCV :

OpenCV (pour Open Computer Vision) est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel. La société de robotique Willow Garage assure le support de cette bibliothèque depuis 2010.

Cette bibliothèque est distribuée sous licence BSD.

Utilisation

Elle propose la plupart des opérations classiques en traitement bas niveau des images:

- lecture et affichage d'une image ou d'une vidéo (fichier ou caméra), écriture sur le disque ;
- calcul de l'histogramme des niveaux de gris ou d'histogrammes couleurs ;
- lissage, filtrage ;
- binarisation, segmentation en composantes connexes ;
- morphologie mathématique.

Elle met également à disposition de l'utilisateur quelques fonctions d'interfaces graphiques, comme les curseurs à glissière, les contrôles associés aux événements souris, ou bien l'incrustation de texte dans une image.

Cette bibliothèque s'est imposée comme un standard dans le domaine de la recherche parce qu'elle propose un nombre important d'outils issus de l'état de l'art en vision par ordinateur tel que :

- détection de droites, de segment et de cercles par Transformée de Hough
- détection de visages par la méthode de Viola et Jones
- cascade de classifieurs boostés
- détection de mouvement, historique du mouvement
- poursuite d'objets par mean-shift ou Camshift
- détection de points d'intérêts

- estimation de flux optique (tracker de Kanade-Lucas)
- triangulation de Delaunay
- diagramme de Voronoi
- enveloppe convexe
- ajustement d'une ellipse à un ensemble de points par la méthode des moindres carrés

Certains algorithmes classiques dans le domaine de l'apprentissage artificiel sont aussi disponibles :

- K-means
- AdaBoost
- Réseau de neurones artificiels
- Machine à vecteurs de support
- Estimateur (statistique)

- **Alain PRUSKI** "ROBOTIQUE GENERALE" edition ellipses.

- **Wisama Khalil, Etienne** Dombre "Modélisation ,identification et commande des robot" 2eme edition Hermes.

- **Jacques Gangloff** "Cours audio-visuel sur la robotique" universitee de Strasbourg.

- **Oussama Khatib** "Cours audio-visuel sur la robotique" Stanford university .

- Taib A et Temam H: Mémoire " Commande en position et en vitesse du robot trainer ED-7220C" Dirigé par Mr Mellah.

- **Gilles BUREL** "INTRODUCTION AU TRAITEMENT D'IMAGES - SIMULATION SOUS MATLAB" edition Hermes.

- **Gary Bradski** and **Adrian Kaehler** "Learning OpenCV" O'REILLY.

- **Heiko Hirschmüller** (2008) « Stereo Processing by Semi-Global Matching and Mutual Information » , IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 30(2), February 2008, pp. 328-341.

- http://opencv.willowgarage.com .

- http://www.arduino.cc

- http://usa.autodesk.com/maya

- http://www.logitech.com

- http://www.hitecrcd.com/

- http://www.lynxmotion.com/

- http://www.mindsetsonline.co.uk

- http://www.msa.ac.uk

- http://www.peratech.com

- http://www.dagurobot.com/