République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU **G**ENIE **E**LECTRIQUE ET D'**I**NFORMATIQUE
DEPARTEMENT D'AUTOMATIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Sciences et Technologies**
Filière : **Génie électrique**
Spécialité : **Commande des systèmes**

Présenté par
mohamed LARIBI
mohamed LIMANI

Thème

Commande gestuelle d'un bras manipulateur à 4 degrés de liberté

Mémoire soutenu publiquement le 01 / 10 / 2015 devant le jury composé de :

M Redouan KARA
MCA, UMMTO, Président

M Rabah MELLAH
MCA, UMMTO, Encadreur

M Prénom NOM
Grade, Lieu d'exercice, Co-Encadreur

M Ahmed MAIDI
MCA, UMMTO, Examinateur

M Amar HAMACH
MCB, UMMTO, Examinateur

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 avoir accepté de diriger ce mémoire.

Nous remercions monsieur Ouazar Yahia pour l'aide qui nous a apportée durant ce travail.

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 chères frères et sœurs*

A** toute la famille **LIMANI

***A** mon binôme*

***A** tous mes amis*

***A** tous ceux qui me sont chers*

Je dédie ce modeste travail

A *mes très chers parents*

A *mes chères frères et sœurs*

A *toute la famille* ***LARIBI***

A *mon binôme*

A *tous mes amis*

A *tous ceux qui me sont chers*

Liste des figures

Chapitre 1 : Généralités

Chapitre 2 : Etude du robot

Chapitre 3 : Actionneurs et Capteurs.

Chapitre 4 : modélisation de la structure du robot.

Chapitre 5 : vision par ordinateur.

Chapitre 6: La Commande.

Liste des tableaux

Chapitre 2 : Etude du robot

Chapitre 3 : Actionneurs et Capteurs.

Chapitre 4 : modélisation de la structure du robot.

Chapitre 5 : vision par ordinateur.

Chapitre 6: La Commande.

Résumé:

Ce projet porte sur la conception d'une commande gestuelle assistée par la vision, des servomoteurs qui contrôlent les différentes articulations d'un bras manipulateur à 4 DDL (degré de liberté).

Introduction:

Le mot robot fut utilisé pour la première fois en 1920, dans la pièce théâtrale de science fiction RUR (Rossum's universal robot) de KAREL CAPEK. Inventé par son frère JOSEF, le mot robot a été dérivé du mot tchèque « robota », qui signifie "travail-corvée".

Le robot moderne, qui est un dispositif mécatronique, n'est en faite que le fruit d'une évolution exponentielle dans les domaines de la mécanique, l'électronique et l'informatique. Cette évolution à ouvert un nouvel horizon à la fabrication de robot, en les rendant plus fiable, plus précis, et surtout avec des tailles miniaturisées facilitant leur exploitation dans différents domaines qui révolutionne la vie quotidienne de l'homme.

Le premier robot industriel nommé « UNIMATE », fut intégré aux lignes d'assemblage de GMC (General Motors Company) en 1961. Par la suit, on a connu une large utilisation de robots dans différents champs de travail et de recherche, cela est dû à leurs précision et leurs capacité de réaliser des taches complexes et répétitifs.

Pour des systèmes autonomes, être au courant de l'environnement qui l'entoure est très important pour prendre des décisions. Pour cela, avoir accès à des images de son environnement est cruciale pour extraire des donnés nécessaire sur ce dernier.

La carte de profondeur joue un grand rôle pour déterminer l'environnement qui entoure les systèmes, puisque elle fournit des informations tel que la distance de l'objet du système. Pour accéder à ces informations un seul moyen existe, c'est la stéréovision.

Difficultés:

-Complexité de la création de la carte de profondeur;

-la qualité de la carte de profondeur;

-le nombre de degré de liberté qui cause des difficultés pour avoir un bon modèle du bras manipulateur.

Introduction :

Dans ce chapitre, on va essayer de donner quelques définitions, pour rendre la lecture et la compréhension de ce mémoire plus limpide et facile. De nos jours, la définition qu'on peut attribuer à un robot industriel diffère d'un pays à autre.

1- Définition :

Au pays du soleil levant (japon), selon la **JIRA** (**Association Japonaise de Robotique Industrielle**) le mot robot prend une définition vague, en le qualifient comme suit: "Tout mécanisme permettant d'effectuer, en tout ou en partie, une tâche normalement réalisée par l'homme"**[1].**

Pour l'oncle Sam (USA), la **RIA (Robot Institute of America)** a une définition plus spécifique: "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"**[1]**.

Selon l'**ISO** (**International Standard Organization**) le robot signifie 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 **[2]**.

Pour les français, la **AFRI** (**Association Française de Robotique Industrielle**) donne une définition bien plus explicite que celle des trois derniers:" Un robot industriel est une machine formée de divers mécanismes comportant divers degrés de liberté, ayant souvent l'apparence d'un ou de plusieurs bras se terminant par un poignet capable de maintenir un outil, une pièce ou un instrument de contrôle. En particulier, son unité de contrôle doit contenir un système de mémorisation, et il peut parfois utiliser des accessoires sensitifs et adaptables qui tiennent compte de l'environnement et des circonstances. Ces machines, ayant un rôle pluridisciplinaire, sont généralement conçues pour effectuer des fonctions répétitives, mais sont adaptables à d'autres fonctions"**[1]**.

2- Les éléments constitutifs d'un robot:

D'une vue globale, on distingue deux grandes unités complémentaires dans un robot, l'unité d'information et l'unité d'opération. Ces dernières servent à assurer le bon fonctionnement du robot dans son environnement.

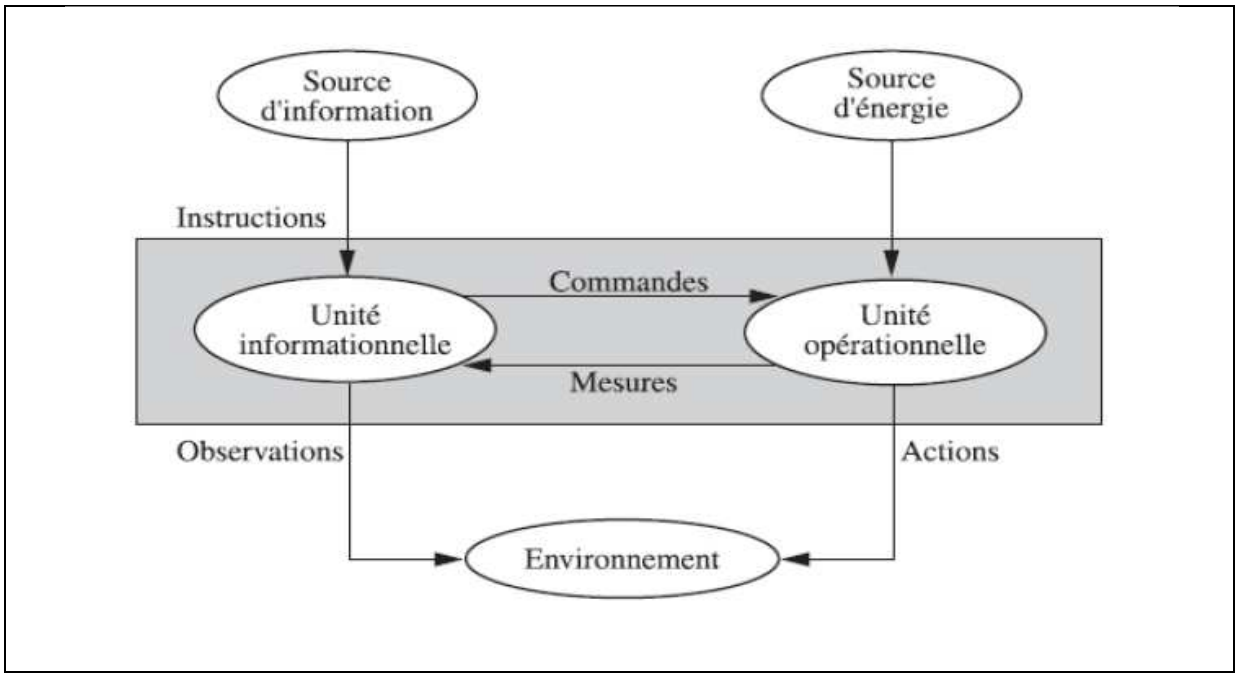


Figure n°1: Structure fonctionnel d'un robot.

2.1- L'unité d'information:

Cette unité a pour but de recevoir les informations sur l'environnement du robot et la tâche à accomplir, afin de concevoir une commande adéquate des différents actionneurs du robot, en vue de réaliser cette tâche.

2.2- L'unité d'opération:

Cette unité constitue la partie physique du robot, elle sert à mettre en œuvre la commande envoyée par l'unité d'information.

3- Vocabulaire du robot:

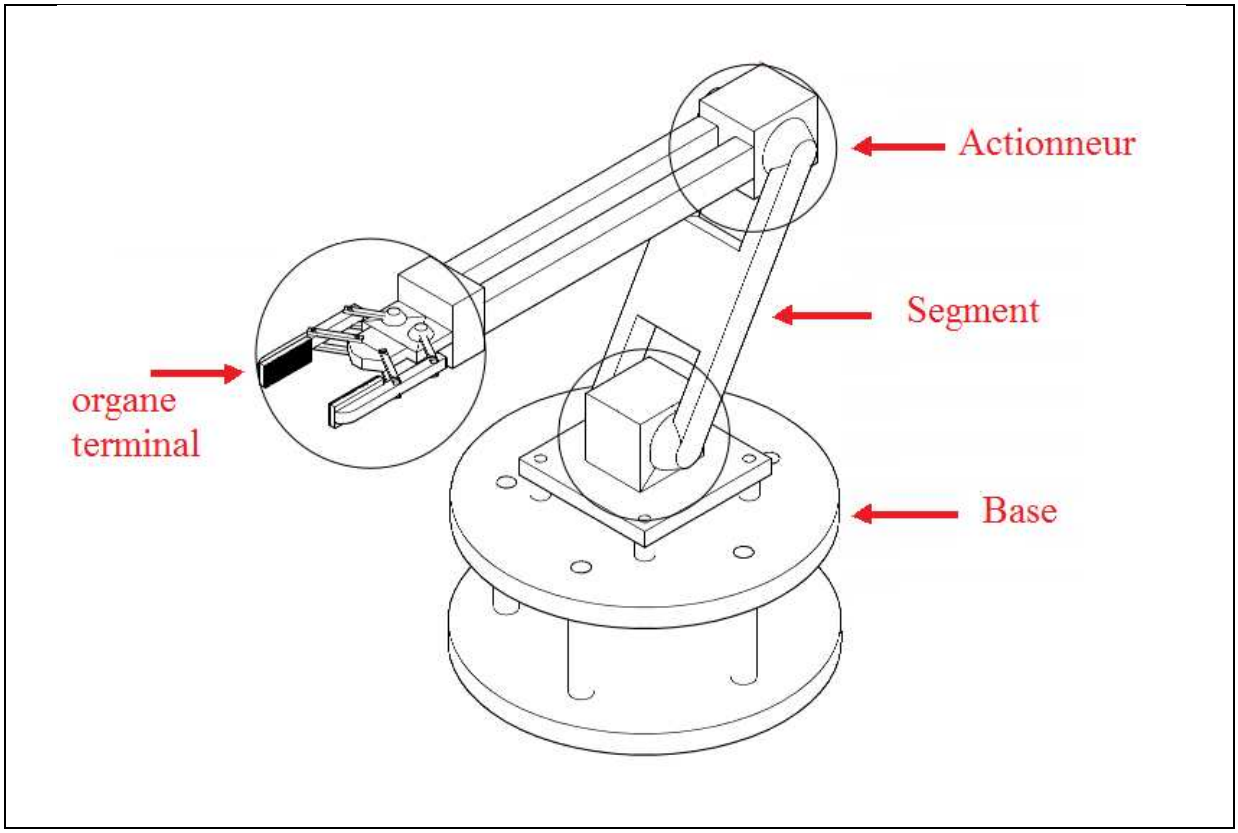


Figure n°2: vocabulaire du robot.

Comme le montre la figure ci dessus, le robot est constitué de plusieurs organes qui forment le vocabulaire du robot.

3.1- La base:

Pour la quasi totalité des robots manipulateurs, la base est fixée sur le milieu de travail.

3.2- Le système mécanique articulé (SMA):

Ayant une structure proche du bras humain, le système mécanique articulé (SMA), est essentiellement constitué d'un ensemble de solide appelé segment, qui sont supposés êtres en mouvement par rapport à la base. Ces derniers sont reliés entre eux par des articulations, qui définissent et limite le degré de liberté (le nombre de déplacements élémentaires indépendants autorisés par cette liaison) de chaque jonction de segment. Le SMA sert à amener l'organe terminal à la position désirée, selon des caractéristiques de vitesse et d'accélération données.

3.3- L'actionneur:

Les actionneurs d'un robot manipulateur servent à animer le système mécanique articulé, ils sont généralement fixés au niveau des articulations. Les actionneurs sont souvent des moteurs électriques à aimant permanent, à courant continu…*etc*. Combiné avec des transmissions. Pour les petits robots, la plupart du temps en utilise des servomoteurs ou des moteurs pas à pas. En ce qui concerne les robots manipulant de grandes charges, en remplace les moteur par des actionneurs hydrauliques ou pneumatique selon le besoin.

3.4- L'organe terminal:

On le qualifie par tout dispositif destiné à manipuler des objets ou a les transformer (dispositifs de serrage, dispositifs magnétiques, torche de soudage, pistolet de peinture…*etc.).* En d'autres termes, il s'agit d'une interface permettant au robot d'interagir avec son environnement. L'organe terminal peut être équipé de plusieurs dispositifs ayant des fonctionnalités différentes.

3.5- Les capteurs:

Ce sont les organes de perception du robot, ils servent d'intermédiaire entre le robot et sont environnement. En envoyant des informations sur l'état interne du robot, tel que la position et la vitesse des actionneurs, ou sur l'environnement (présence d'objet, mesure de distance…*etc*.).

4- Classification des robots:

Les robots sont classés d'un point de vue fonctionnel ou géométrique.

4.1- Classification fonctionnel:

Le nombre de classe varie d'un pays à un autre (on trouve 4 classes en France et 6 classes au japon). Selon la AFRI il existe 4 classes distinctes illustrées ci-dessous.

4.1.1. Classe (A) manipulateurs à commande manuelle ou télécommande:

Cette classe de robot, se sert d'une commande manuelle ou à distance qui nécessite la présence de l'homme **[1]**.

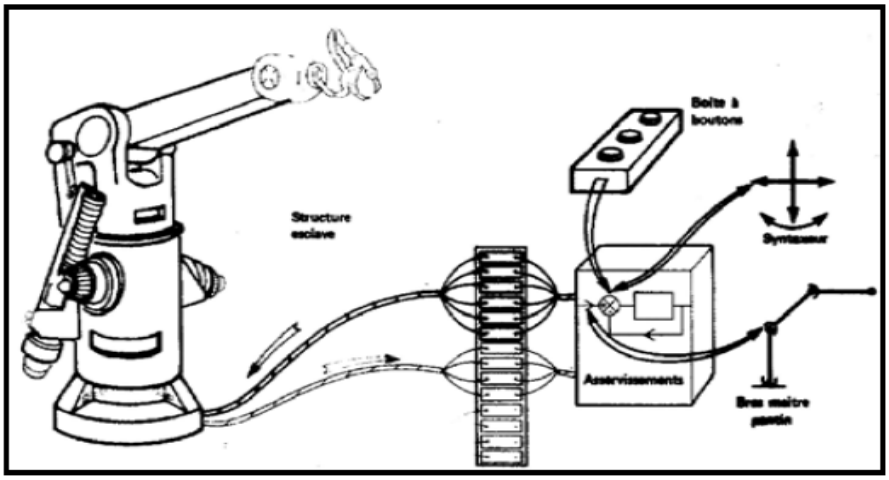


Figure n°3: robot télécommandé.

4.1.2. Classe (B) manipulateurs automatiques à cycles préréglés:

La commande de cette classe se fait par automate programmable **[1]**.

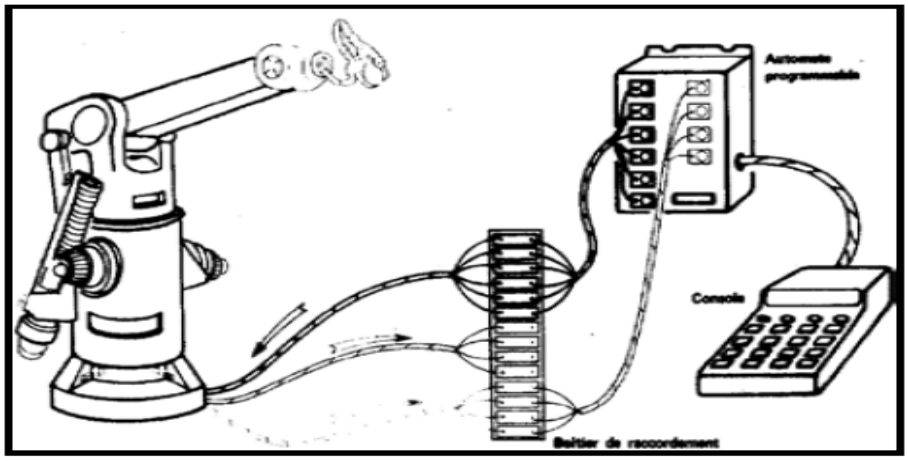


Figure n°4: robot préréglé.

4.1.3. Classe (C) robots programmables:

C'est la première génération de robot industriel. Cette classe de robot reproduis les mouvements ou le cycle qu'on lui à apprit ou programmer au préalable **[1]**.

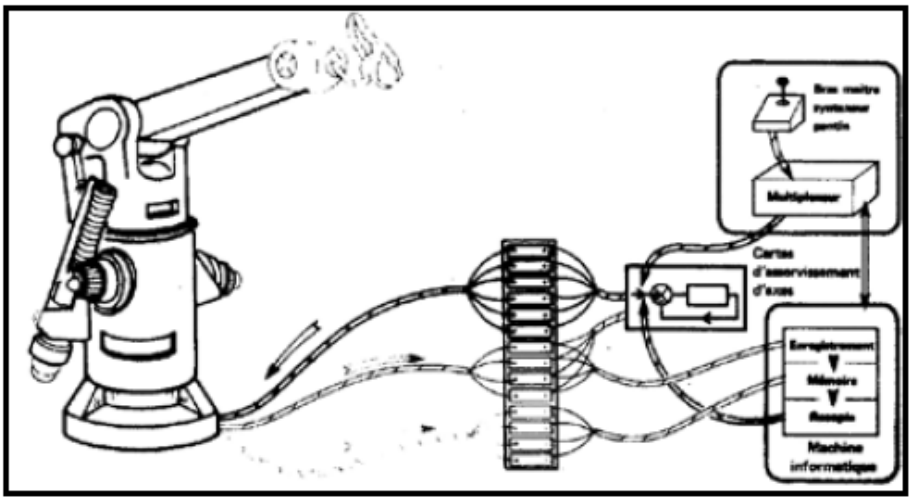


Figure n°5: robot programmé.

4.1.4. Classe (D) robots intelligents:

Cette classe de robot se distingue de la précédente par l'autonomie de ses mouvements. Ces robots sont équipés de plusieurs capteurs (capteur de proximité, capteur d'effort, système de vision…*etc.*) qui servent à acquérir des informations sur l'environnement et l'état interne du robot, les rendant ainsi autonome **[1]**.

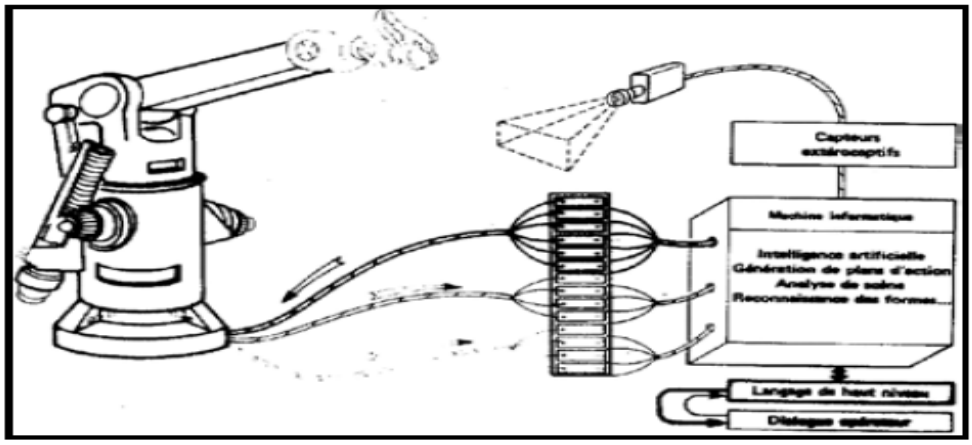


Figure n°6: robot intelligent.

4.2- Classification géométrique:

Les robots sont aussi classés selon leur configuration géométrique, en d'autres termes l'architecture de leur porteur. Les porteurs peuvent êtres réalisés avec un grand nombre de combinaison de translation ou de rotation, autrement dit prismatique (P) ou rotatif (R) **[1]**.

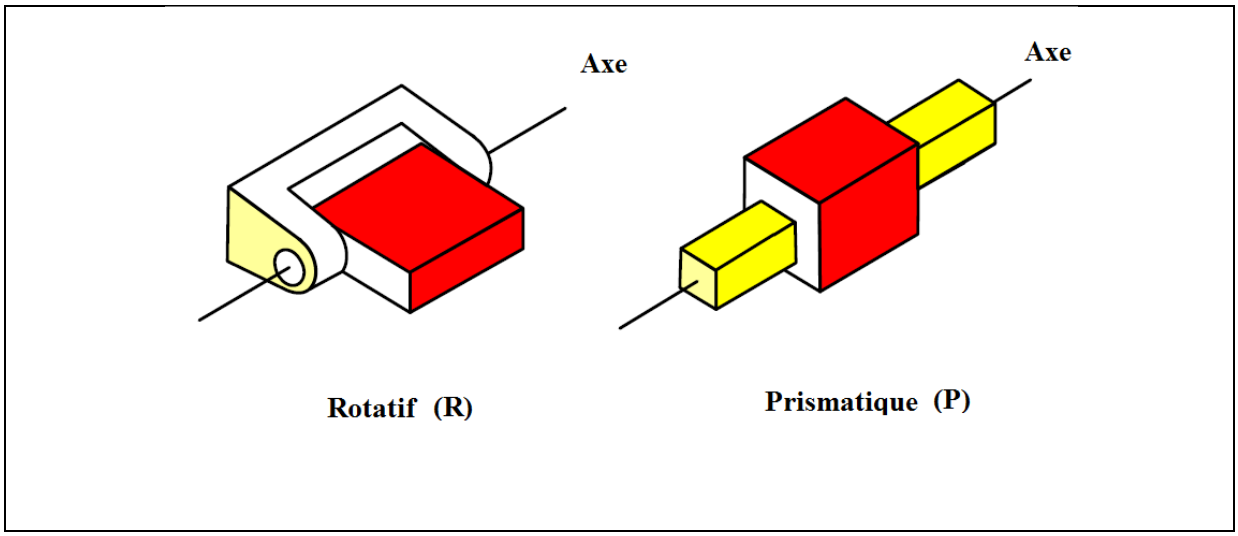


Figure n°7: liaison rotatif et prismatique.

4.2.1. Porteur cartésien (PPP):

Dans cette classe de robot on trouve une succession de trois liaisons prismatique. Cette structure et relativement peu utilisée, sauf dans quelques applications particulières, robots pratiques, robots de magasinage, par exemple

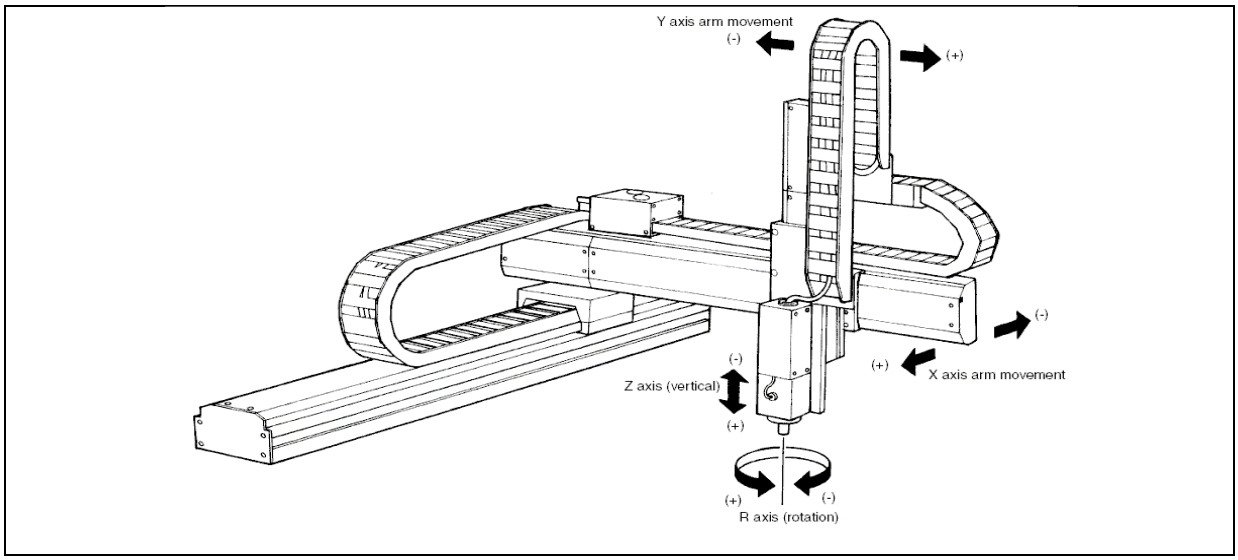


Figure n°8: Robot cartésien.

4.2.2. La structure cylindrique (RPP) ou (PRP) :

Cette structure associe une rotation et deux translations. Elle est pratiquement pas utilisée vu son volume de travail réduit.

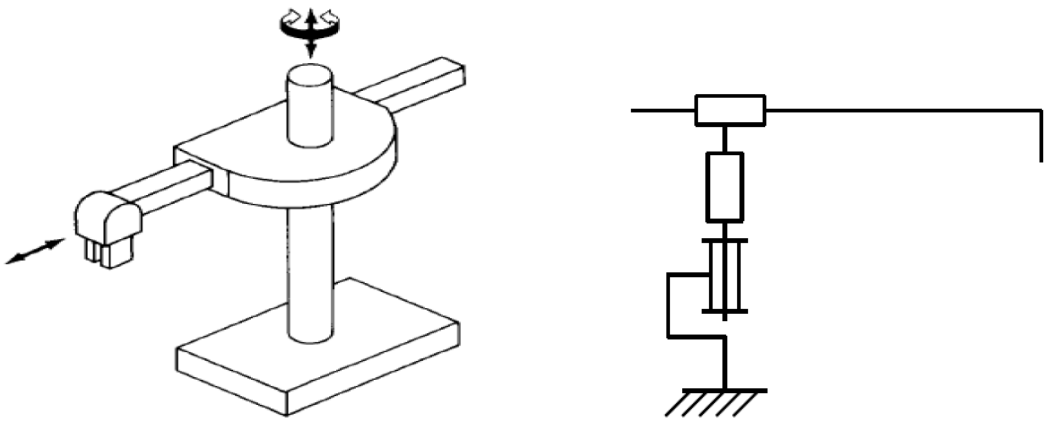


Figure n°9: Robot cylindrique.

4.2.3. La structure sphérique ou polaire à axe de rotation orthogonale :

Pour des raisons similaires que la classe précédente, cette classe est peut utilisée ou même quasiment abandonnée.

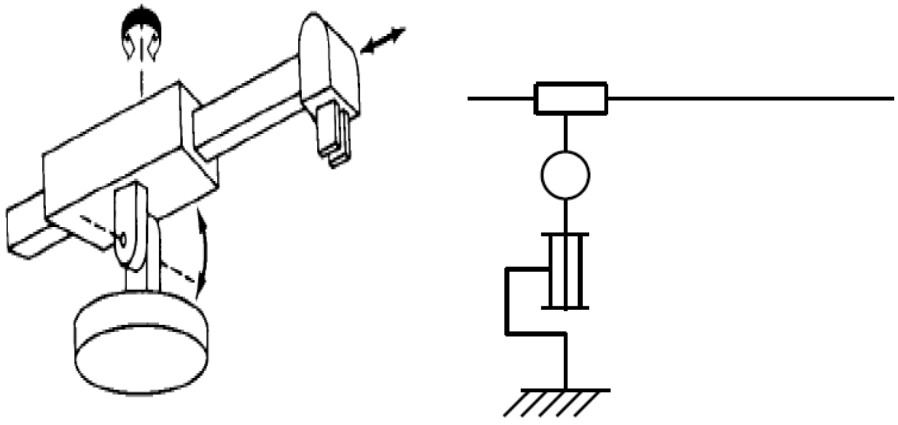


Figure n°10: Robot sphérique.

4.2.4. La structure SCARA (Selective Compliance Arm for Robotic Assembly):

Une des structures les plus utilisées dans le domaine de l'industrie, cette structure comporte des axes de rotation parallèles à la verticale, avec un ratio très favorable entre le volume de travail et l'encombrement.

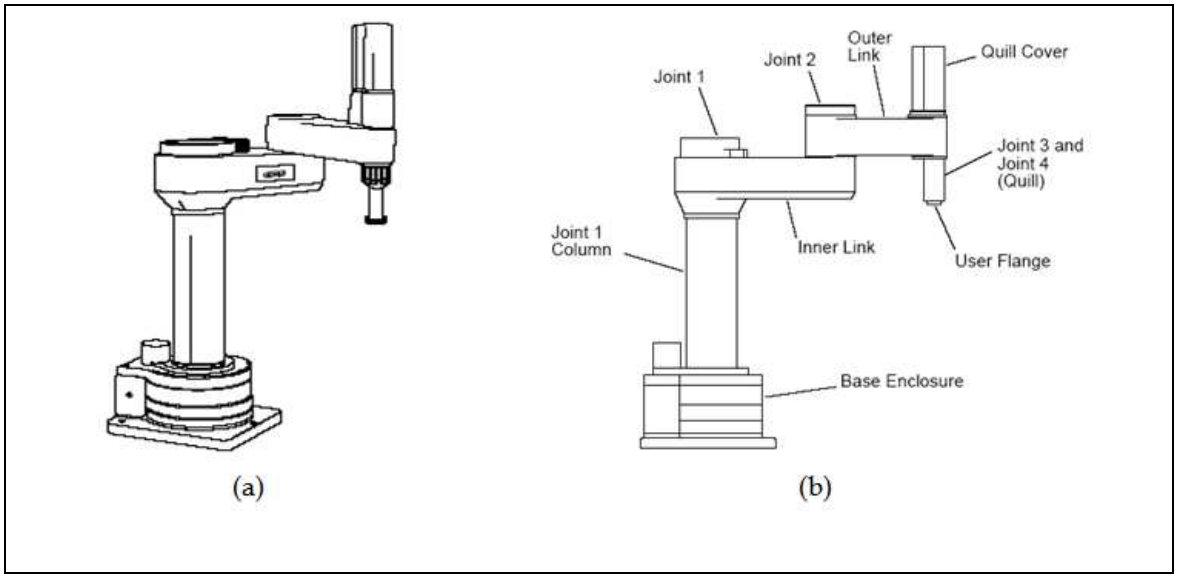


Figure n°11: robot SCARA.

4.2.5. La structure 3R (anthropomorphe) :

Reproduisant la structure d'un bras humain, cette structure allie force et précision en offrant un volume de travail plus grand que les classes précédentes.

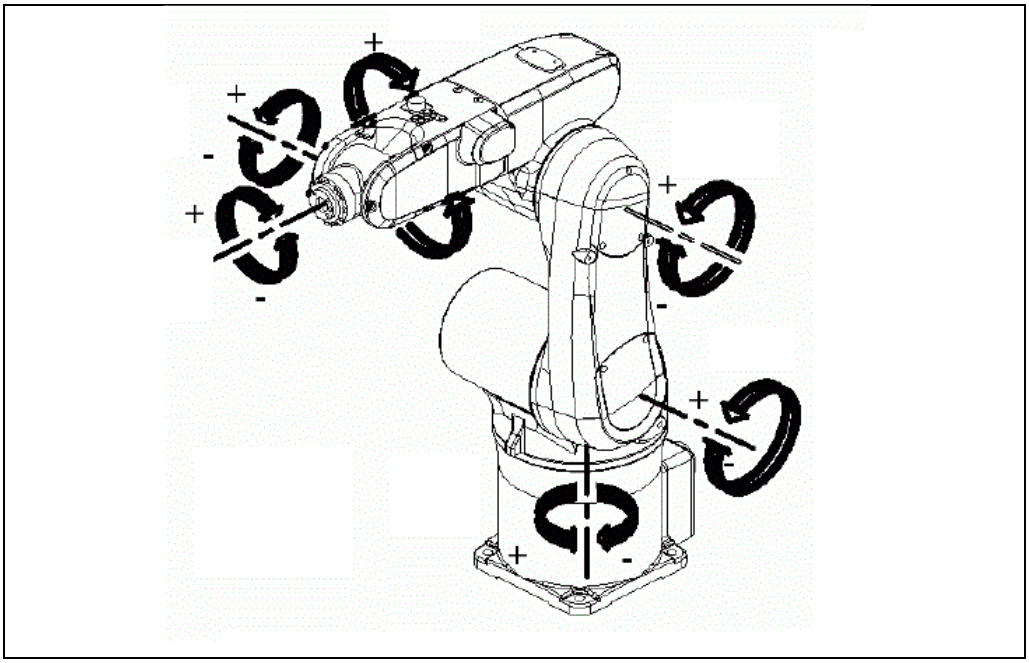


Figure n°12: robot anthropomorphe.

Conclusion:

Dans ce chapitre, on a vu une description générale sur le domaine de la robotique, en mettant l'accent sur les différents composants d'un robot, et le vocabulaire utilisé dans ce domaine. Une présentation détaillée du robot fait l'objet du chapitre suivant.

1- Le corps du robot:

Le bras manipulateur et le composent mécanique du robot. Dans ce projet, on a opté pour un bras particulièrement conçu pour l'usage dans les sphères d'éducation et de formation de robotique. La structure anthropomorphe du bras est constituée de 4 axes et une pince, donc 4 DDL (degré de liberté), qui sont actionnés par des servomoteurs à courant continu. De plus le robot est équipé de deux caméras CMOS.

Figure n°13: robot.

2- Le volume atteignable (volume de travail):

Le volume de travail d'un robot, est défini comme l'espace physique formé par un point de son organe terminale en mouvement suivant toutes les orientations possible. Cette espace est exprimée en unité volumique où la forme de son enveloppe peut être simple ou complexe, selon la forme et la géométrie du robot, plus précisément, la géométrie de ses articulations **[1]**.

Ce volume dépend :

- de la géométrie du robot;
- de la longueur des segments.

Le bras manipulateur utilisé dans ce projet est de classe anthropomorphe (poly articulé). Cette classe de robot est réputée d'avoir un grand volume de travail et complexe, vu qu'elle reproduit la structure d'un bras humain. Le volume de travail est habituellement représenté graphiquement par deux sections perpendiculaires, choisies en fonction de la classe du robot manipulateur, cette représentation étant préférable à une seule vue en perspective. Dans la figure ci-dessous, on a illustré le volume atteignable du bras manipulateur avec deux perspectives (vue de profile à gauche et vue de haut à droite).

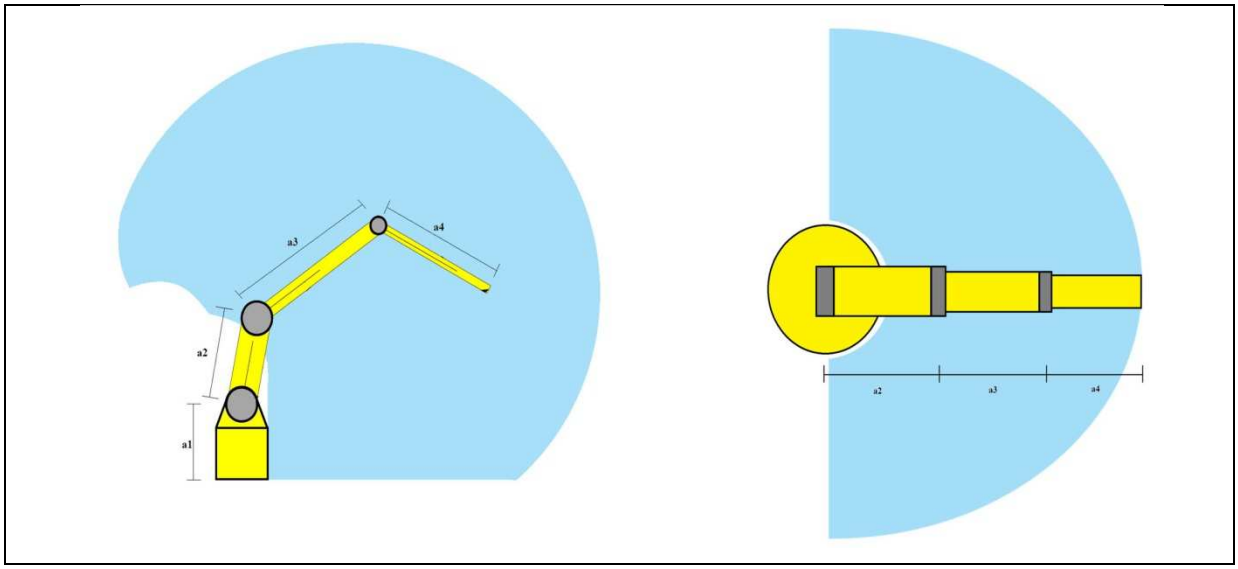


Figure n°14: volume de travail du robot.

3- Charge utile:

La charge utile est définit comme la charge que peut porter le robot sans dégrader la répétabilité et les performances du robot.

Reste à savoir que la charge utile est nettement inférieure à la charge maximale que peut porter le robot qui est directement liée aux actionneurs.

4- Précision et Répétabilité:

4.1- Précision:

La précision du robot manipulateur est définit par l'écart entre un point désiré (programmer) et une orientation dans l'espace cartésien (le point atteint) **[3]**.

4.2- Répétabilité:

La répétabilité d'un robot est l'erreur maximale de positionnement répété de l'outil en tout point de son espace de travail. En d'autres termes c'est sa capacité à revenir à la position initiale qu'on lui a attribué à charge constante. Autrement dit, si le robot porte une charge d'un point A vers un autre point B avec une charge d'un certain poids, et que cette charge change au retour à la position initiale (de B vers A), cette dernière ne sera pas la même.

En général, la répétabilité < 0.1 mm **[3]**.

5- Le système de commande :

Le système de commande du robot est composé d'un ordinateur et une carte de commande des servomoteurs nommé « Arduino » (voir annexe).

L'ordinateur muni d'un logiciel d'interfaçage, reçoit les informations envoyées par les deux caméras pour synthétiser une loi de commande en temps réel qui sera envoyé à la carte de commande par un port USB en vu de commander en PWM (pulse width modulation) les actionneurs du bras manipulateur.

6- Caractéristiques techniques:

Dans ce qui suit, on trouve toutes les principales caractéristiques du robot.

6.1- Caractéristique technique du bras manipulateur:

Axes	**3 axes de révolution**
Motorisation des axes	Servomoteurs RC à courant continu
Respectabilité en position	± 0.1 mm
Commande de l'effecteur	commande électrique (2 positions)
Poids	Ne dépasse pas 1Kg
Alimentation électrique pour chaque actionneur	6V/1.5A DC
Camera	Webcams CMOS Logitech 1.3 Méga pixel port USB
Matériaux	Aluminium

Tableau n°1: caractéristiques du bras manipulateur.

Les limites angulaires de chaque articulation sont illustrées dans la figure ci-dessous.

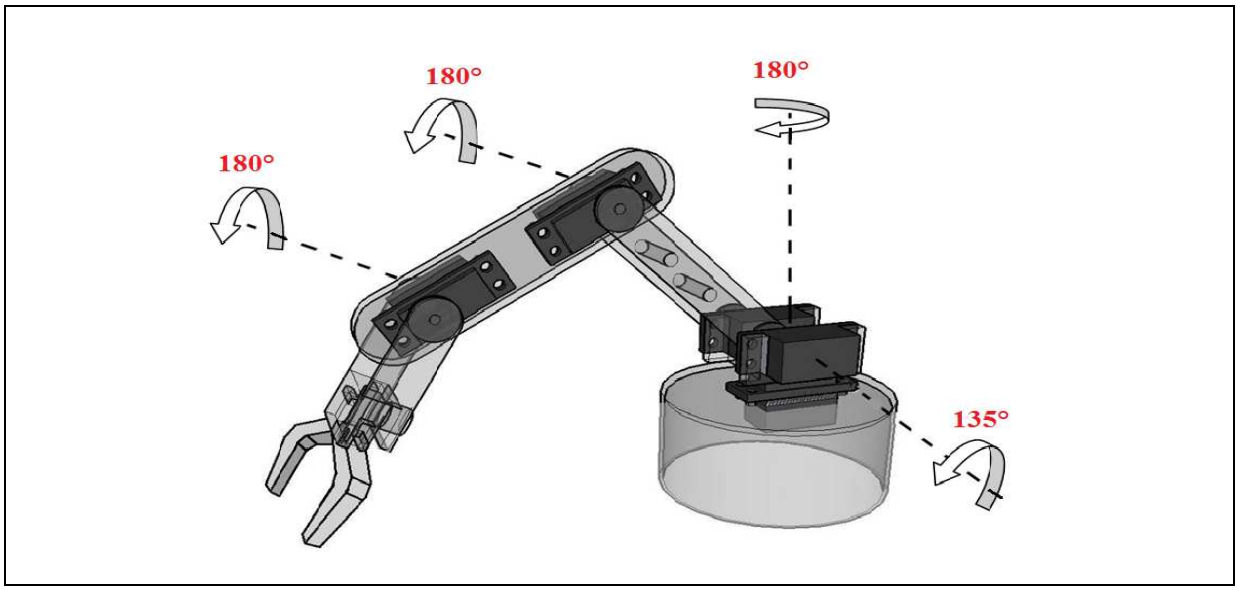


Figure n°15: limite angulaire des articulations.

6.2- Caractéristique technique du système de commande:

Le système de commande est composé de deux parties, l'ordinateur et une carte de commande Arduino.

6.2.1. Ordinateur:

Microprocesseur	Intel core i5 cpu 1.7 GHz
Type de l'OS	**64 bits**
RAM	**8 Go**
Carte graphique	NVIDIA GeForce 820M 2Go de RAM
Système d'exploitation	Linux ubuntu 14.04
Langage de programmation	C/C++/JAVA
E/S	Ports USB

Tableau n°2: caractéristiques de l'ordinateur.

6.2.2. La carte de commande Arduino:

Il s'agit d'une carte électronique programmable faite a base de microcontrôleur Atmel, elle est très utilisée dans le domaine de la robotique et de l'électronique vu la facilité à la programmer et son aptitude à contrôler les servomoteurs et les moteurs pas à pas.

Microcontrôleur	**Atmel ATmega2560**
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 dispose d'une sortie PWM)
Broches analogique d'entrées	16 (utilisables en broches E/S numériques)
Intensité maxi disponible par broche E/S (5V)	40 mA (200mA cumulé pour l'ensemble des broches E/S)
Intensité maxi disponible pour la sortie 3.3V	50 mA
Mémoire flash	256 KB dont **8 KB** sont utilisés par le bootloader
SRAM (mémoire volatile)	8KB
EEPROM	4KB
Fréquence d'horloge	16 MHz

***Tableau n°3: caractéristiques de la carte de commande* [4].**

7- Domaine d'utilisation du robot:

Le plus souvent, les robots sont utilisés pour des taches simples et répétitives. Pour la classe anthropomorphe, elle est utilisée fréquemment dans le domaine d'industrie pour divers taches tel que l'assemblage, le soudage et le tri, et cela à cause de ses qualités de précision et de flexibilité.

Pour notre robot, vu qu'il est commandé à distance (classe A), ce dernier peut être utilisé dans des environnements hostiles tel que le domaine de la recherche (laboratoires de chimie nucléaire).

De là, on peut dire que la tâche donnée au robot peut varier selon l'organe effecteur qu'on lui attribut.

8- Fonctionnement du robot:

Les articulations du robot sont configurées pour avoir une position initiale de la pince, où chaque servomoteur d'articulation est initialisé à un angle donné comme suit:

θ_1=90

θ_2=90

θ_3=90

θ_4=90

La configuration initiale du robot est illustrée sur la figure qui suit:

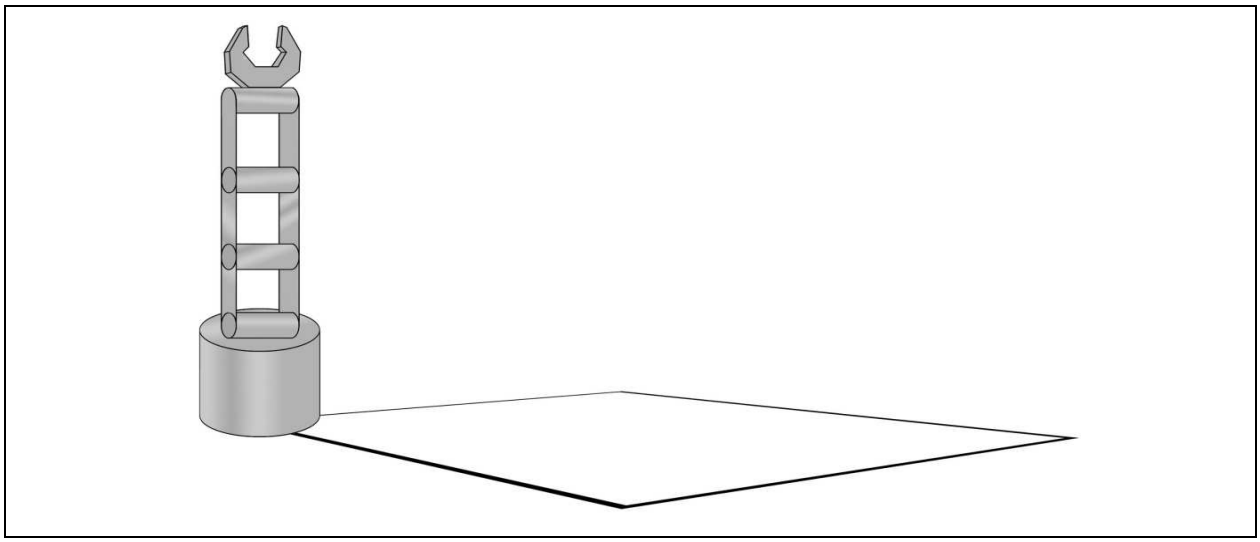


Figure n°16: configuration initiale du robot.

Les caméras vérifie la présence dans sont champ de vision d'un objet de couleur désiré bien différente de celle de l'arrière plan. Dans notre cas l'objet recherché c une main portant un gant. Autrement dit les cameras se focalise sur la couleur de l'objet, non sur sa forme.

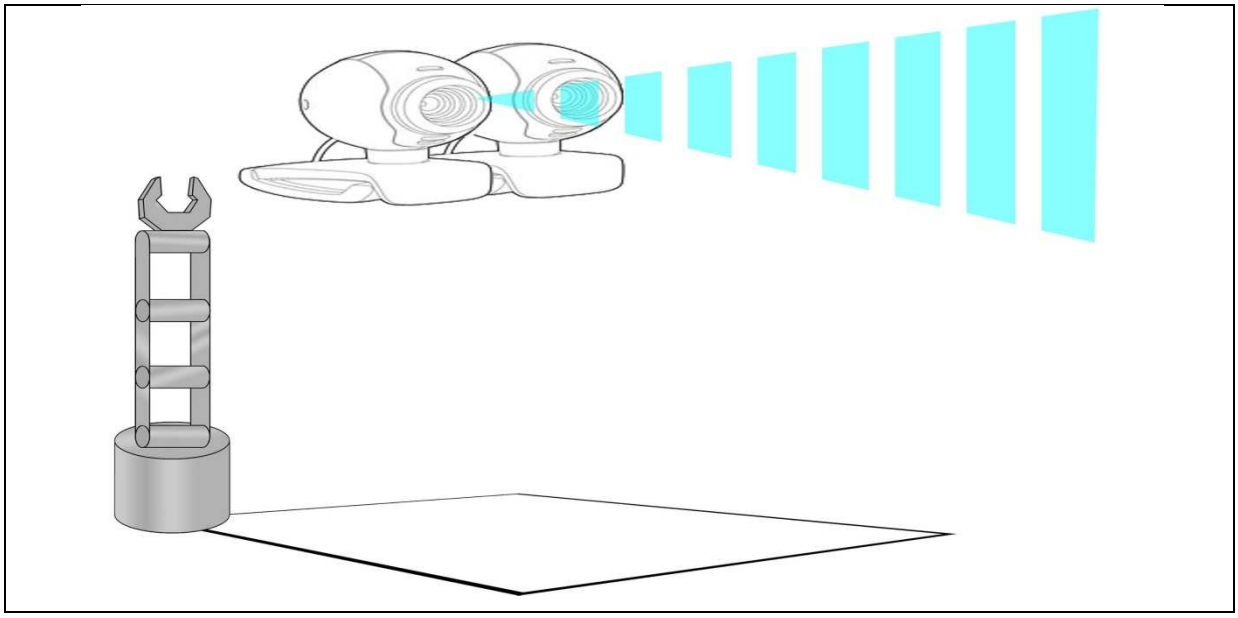


Figure n°17: recherche de l'objet.

Une fois que l'objet est présent dans le champ de vision des caméras, la stéréovision intervient pour extraire les coordonnées (x,y,z) du centre de gravité de la forme détectée.

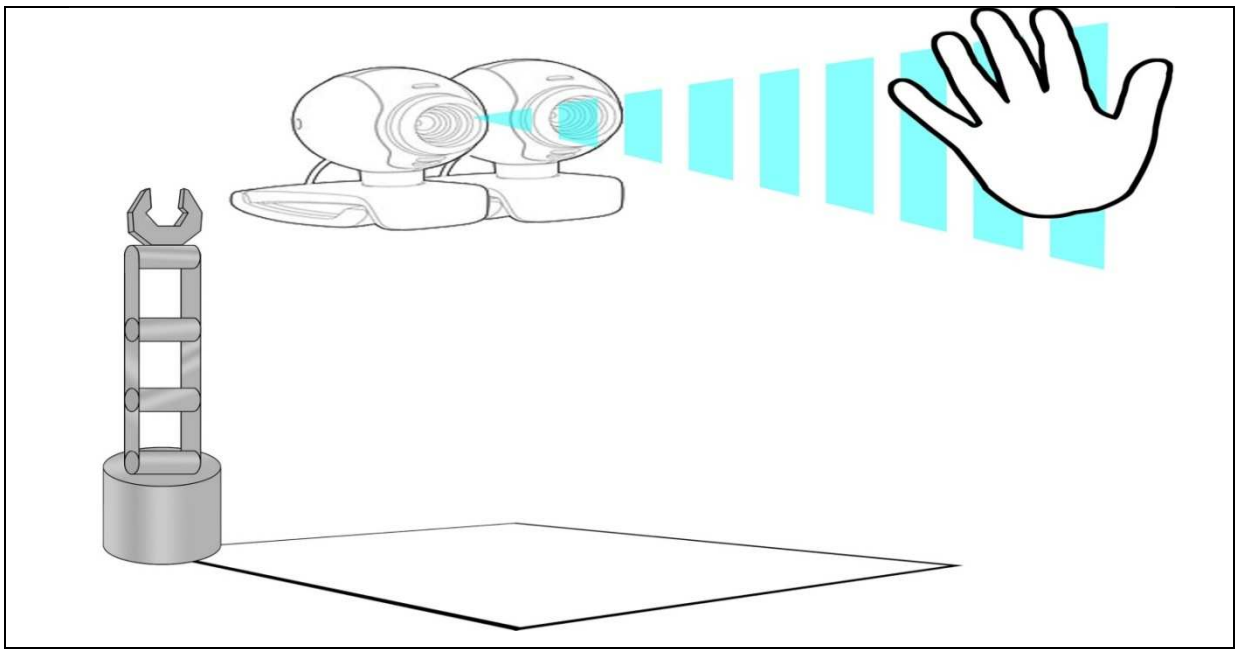


Figure n°18: objet détecté.

Les coordonnées de la forme détectée sont envoyées à la carte de commande via le câble USB, en passant par le programme d'interfaçage sur PC.

En utilisant le modèle géométrique inverse, la carte de commande va convertir les coordonnées cartésien (x,y,z) en coordonnées angulaire pour chaque actionneur afin de ramener le centre de la pince à la position désirée sur le volume de travail.

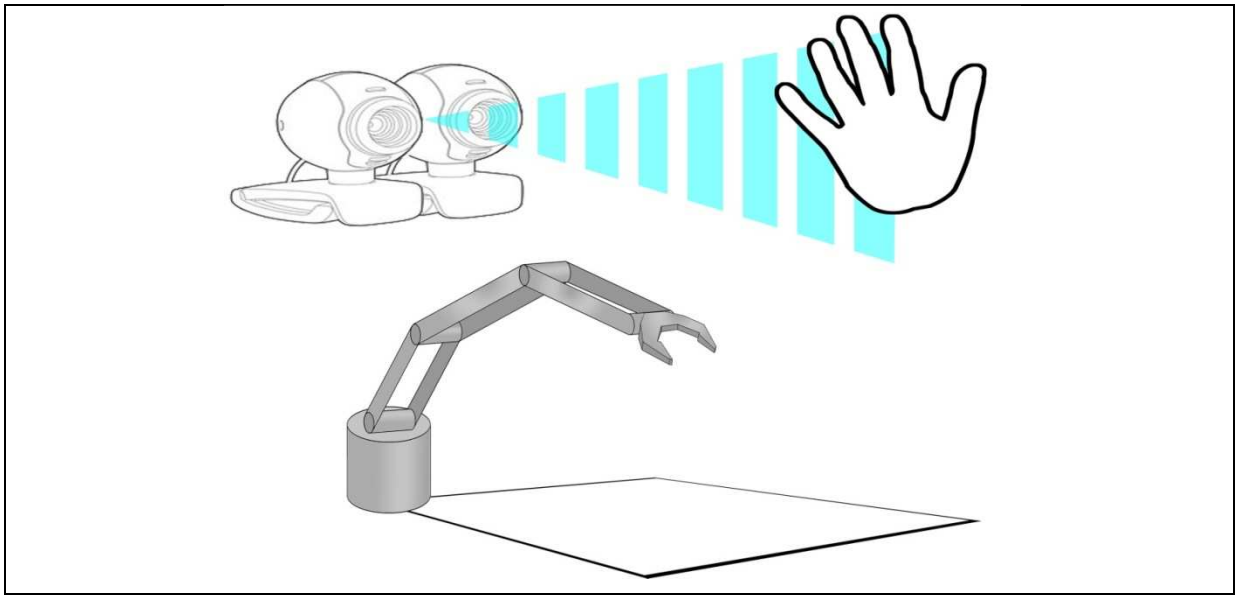


Figure n°19: cible atteinte.

Pour chaque position de la main, le bras manipulateur va essayer de joindre sa pince à la position reçue. Ainsi, le bras manipulateur va bouger au fur et à mesure que la main bouge. En d'autres termes, il va imiter les mouvements de la main.

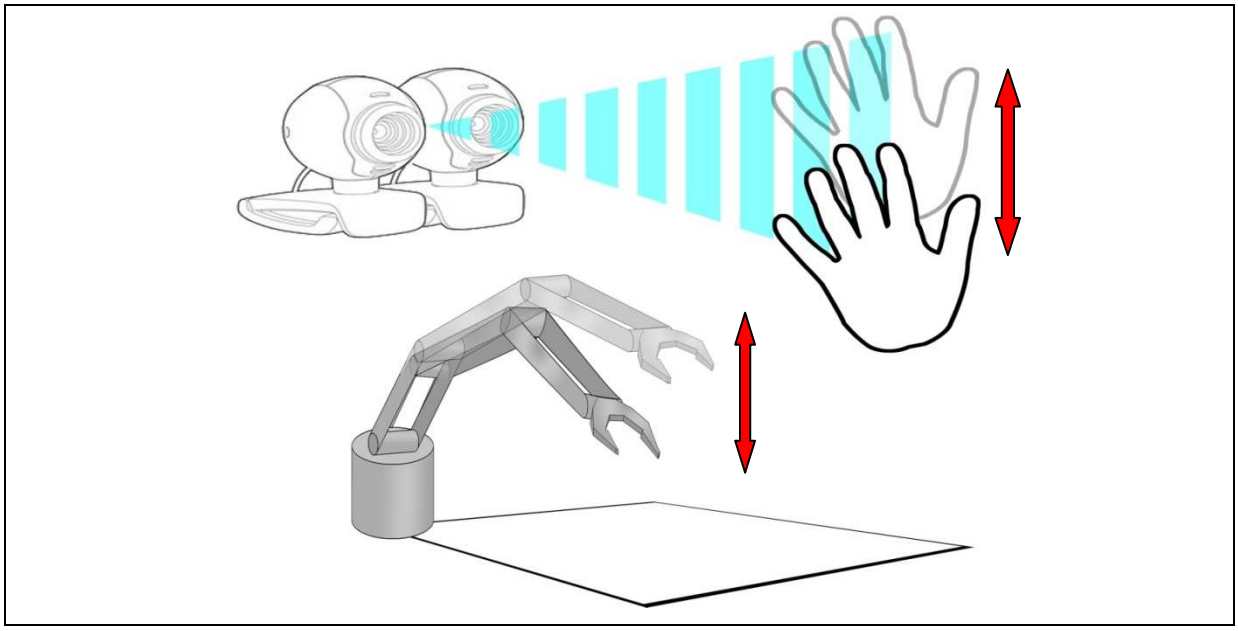


Figure n°20: imitation des mouvements.

Introduction:

Les derniers robots sont parvenus à effleurer le summum d'évolution, en intégrant une multitude de capteurs, ces robots peuvent imiter presque à la perfection tous les gestes d'un être humain. Tous ces mouvements sont réalisés à l'aide d'actionneurs. Dans ce chapitre on va expliciter les actionneurs et capteurs qui composent le robot.

1- Les actionneurs:

Comme un bras humain peut réaliser ses mouvements grâce à ses muscles, les robots de même, ont besoin d'actionneurs pour convertir les informations qui lui sont fournis via un éventuel pré-actionneur, en un travail utile.

En d'autres termes, l'actionneur est l'organe qui anime la structure mécanique en lui fournissant la force physique nécessaire pour exécuter un travail ordonné.

Il existe plusieurs familles d'actionneurs dont on peut citer:

- Les actionneurs électriques;
- Les actionneurs pneumatiques;
- Les actionneurs hydrauliques.

Le choix des actionneurs d'un robot dépend de la tâche à accomplir. Évidemment, pour pouvoir valider un choix, il faut connaître les performances que nous voulons atteindre.

Pour notre cas, on a choisit un bras manipulateur équipé de servomoteur RC pour cause de:

- Le couple fourni assouvit nos exigence;
- Le servomoteur tourne à la bonne vitesse pour notre robot;
- Le fil du signale (commande) à faible courant peut être raccordé directement à une sortie du microcontrôleur. Pas besoin de circuit d'interface.

1.1- Les servomoteurs:

1.1.1- Présentation et caractéristiques:

De manière semblable aux moteurs à courant continu, les servomoteurs disposent d'un axe de rotation qui est entravé par un système de bridage. Cela ne veut pas dire qu'il ne tourne pas, mais cela signifie qu'il ne peut pas tourner au delà d'une certaine limite. Les Servomoteurs RC (radio commandé) sont souvent utilisés dans beaucoup d'applications industrielles ou de robotique pratique vu leur taille compact **[5]**.

1.1.2- Les composants d'un servomoteur:

Un servomoteur est composé d'un moteur DC, d'une boite de vitesse, d'un capteur de retour de position (potentiomètre) ainsi qu'une électronique de commande (faisant office de "cerveau"). Le nom vient en fait du latin "Servus" qui signifie esclave. Le servomoteur peut être contrôlé par un signal externe à large impulsion de modulation (PWM) **[5]**.

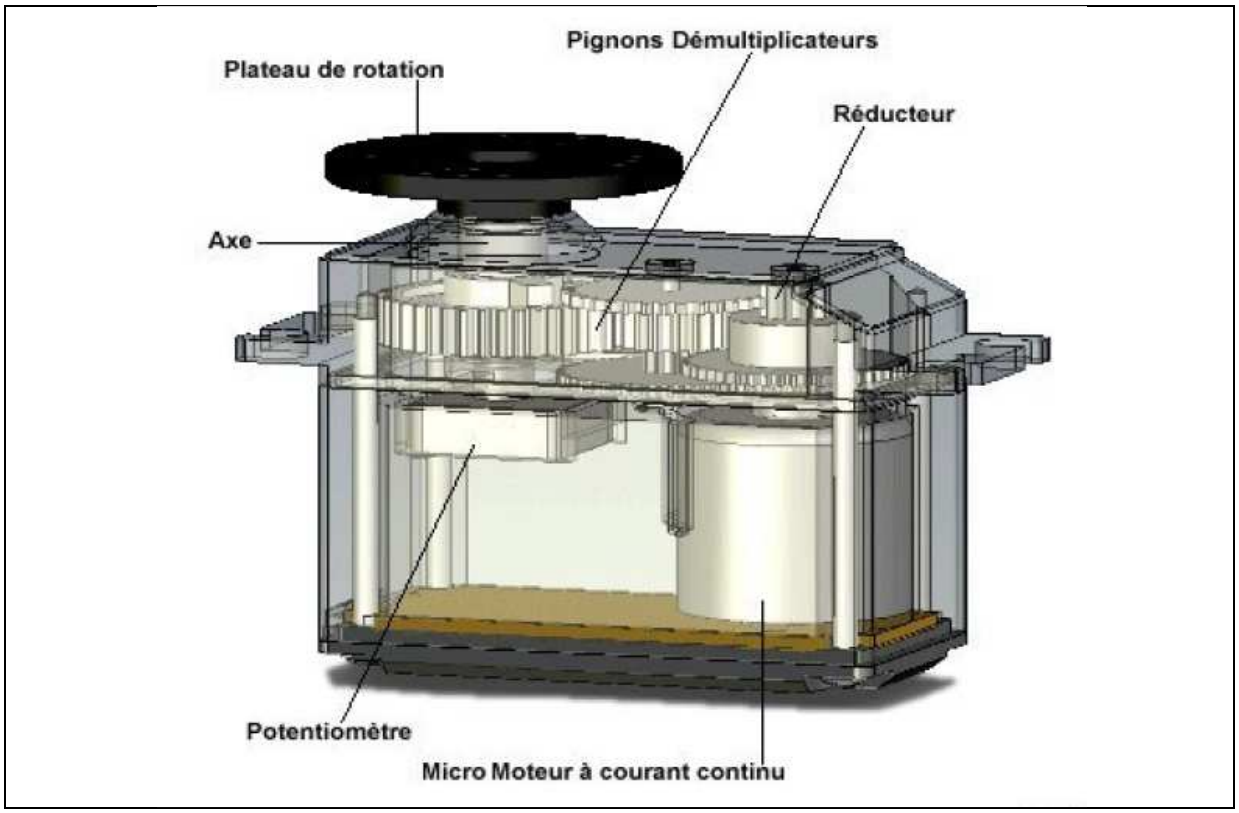


Figure n°21: composants d'un servomoteur.

1.1.3- La commande d'un servomoteur:

Si un signal de commande est envoyé au servomoteur, cela défini habituellement une position cible d'entrée pour l'électronique de commande du servomoteur. La consigne envoyée au servomoteur n'est autre qu'un signal électronique de type PWM (pulse width modulation). Il dispose cependant de deux caractéristiques indispensables pour que le servo puisse comprendre ce qu'on lui demande. À savoir : une fréquence fixe de valeur 50Hz et d'une durée d'état HAUT fixée à certaines limites **[5]**.

Le signal que nous allons devoir générer doit avoir une fréquence de 50 Hz. Autrement dit, le temps séparant deux fronts montants est de 20 ms.

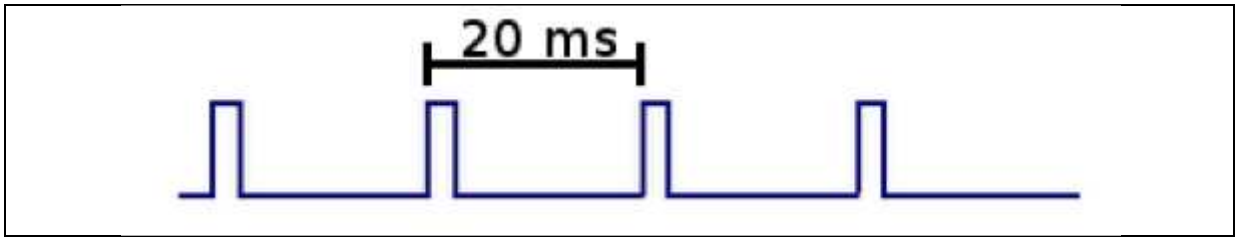


Figure n°22: période du signal d'entrée.

L'électronique de commande du servomoteur est constituée d'un capteur de position du bras qui n'est autre qu'un potentiomètre couplé à l'axe du moteur et d'une zone de commande. La mesure de la tension au point milieu du potentiomètre permet d'obtenir une tension image de l'angle d'orientation du bras. Cette position est comparée ensuite au signal de commande transmit au servomoteur par la zone commande.

La position du rotor dépend du signal de contrôle reçu par les servomoteurs RC. L'angle de rotation maximum du moteur peut changer en fonction du type de servomoteurs, la plupart des servomoteurs ont une limite de rotation de 180°, les servomoteurs à rotation constante se font rares.

Le signal de contrôle du servomoteur est une impulsion spécifique en rapport avec le signal modulé (PWM), en vue de déterminer la position du rotor. 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 **[5]**.

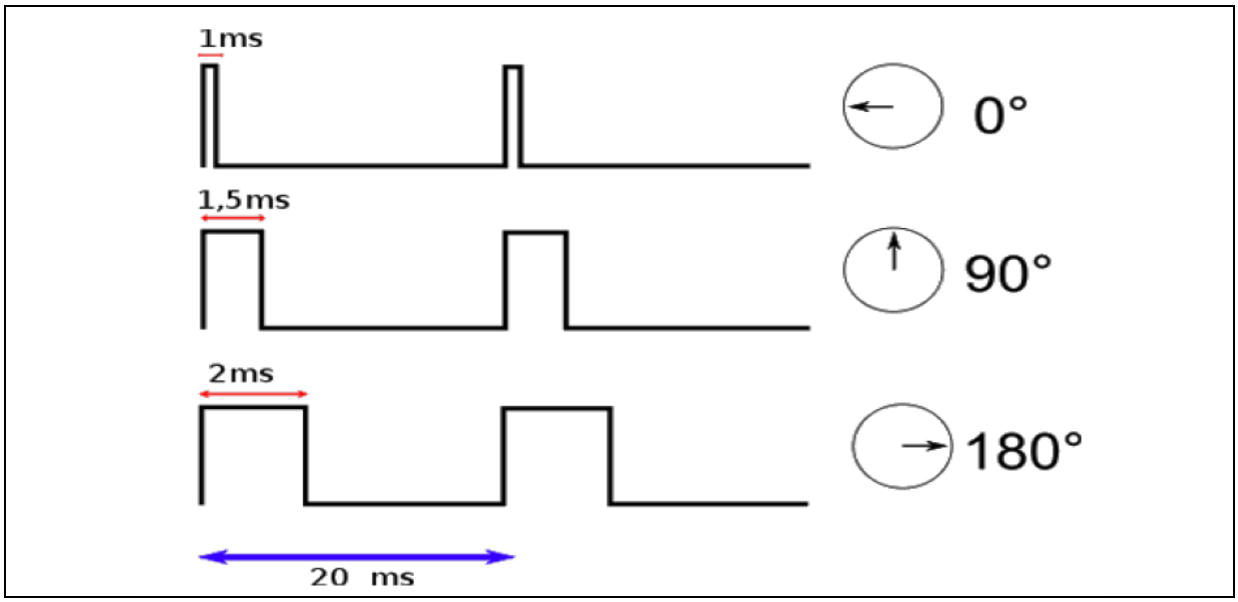


Figure n°23: signale de contrôle de servomoteur.

1.1.4- Variation de la vitesse:

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. Les moteurs à courent continu tournent à des vitesses trop élevée, donc ils perdent en précision contrairement aux servomoteurs.

Reste à citer qu'il existe deux types de servomoteur : ceux qui possèdent une électronique de commande de type analogique, qui sont les plus courants et les moins chers et ceux qui sont asservis par une électronique de commande numérique, très fiables et très performants, mais bien plus chers que les analogiques.

1.1.5- Connectique et mécanique:

- **Connectique:**

Les servomoteurs sont équipés de trois fils de connexion:

- **Rouge** : pour l'alimentation positive (4.5V à 6V en général);
- **Noir** ou **marron** : pour la masse (0V);
- **Orange**, **jaune**, **blanc**, … : entrée du signal de commande.

- **Mécanique:**

Sur un servomoteur on trouve plusieurs pignons (engrenage) reliés a l'axe du moteur à courent continu. Cet ensemble constitue le réducteur, qui sert à deux choses: d'une part il réduit la vitesse de rotation en sortie de l'axe du servomoteur (et non du moteur CC), d'autre part il permet d'augmenter le couple en sortie du servomoteur (et non en sortie du moteur CC). Plus le levier est long et plus le couple exercé sur l'axe est important. L'image qui suit illustre ce qu'on vient de dire **[5]**.

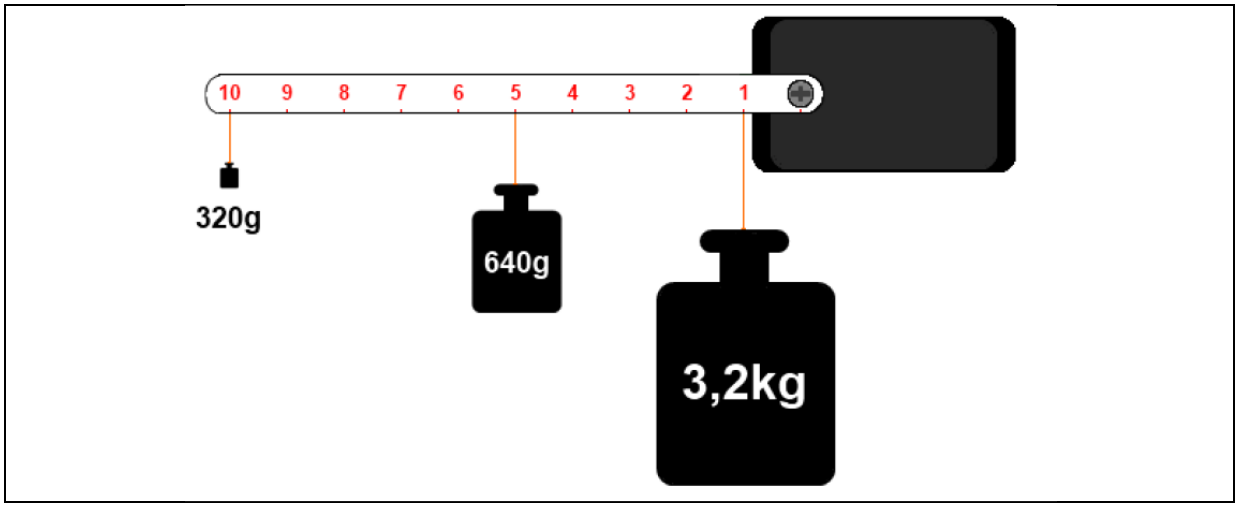


Figure n°24: couple d'un servomoteur.

1.2- Les servomoteurs utilisés:

1.2.1- Futaba S3003:

Les servomoteurs RC utilisés pour la base et la pince, sont des "Futaba S3003" qui fournissent un couple moyen, assez pour orienter la base du bras manipulateur et pour arriver à soulever l'objet à prendre.

Les dimensions du servomoteur sont détaillées dans la figure ci-dessous.

Figure n°25: Les démentions du servomoteur Futaba S3003 **[6].**

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

Système de contrôle	Contrôle par largeur d'impulsion
Vitesse de fonctionnement à 4.8 V	0.23s/60°
Vitesse de fonctionnement à 6 V	0.19s/60°
Couple à 4.8V	3.2kg.cm
Couple à 6V	4.1kg.cm
Type de moteur	3-pole
Type de roulement	roulement à bills
Type d'engrenage	Plastic
Poids	37.2g

Tableau n °4: caractéristiques techniques du servomoteur Futaba S3010 **[6].**

1.2.2- Futaba S3306:

Ce servomoteur est utilisé pour l'épaule du bras manipulateur pour soulever sa structure mécanique, et ca à cause du couple important qui fourni.

Figure n°26: dimensions du servomoteur Futaba S3306 **[7].**

Ses caractéristiques techniques sont synthétisées sur le tableau qui suit:

Système de contrôle	Contrôle par largeur d'impulsion
Vitesse de fonctionnement à 4.8 V	0.20 sec / 60°
Vitesse de fonctionnement à 6 V	0.16 sec / 60°
Couple à 4.8V	19.23 kg-cm
Couple à 6V	23.98 kg-cm
Type de moteur	3-pole
Type de roulement	Double roulement à bills
Type d'engrenage	Plastic
Poids	72 g

Tableau n°5: caractéristiques techniques du servomoteur Futaba S3306 **[7].**

1.2.3- Towerpro MG995:

Le servomoteur utilisé pour le coude est un Towerpro MG995. Avec un couple assez élevé qui vient en aide pour le servomoteur précédent. Ses dimensions sont montrées sur la figure qui suit:

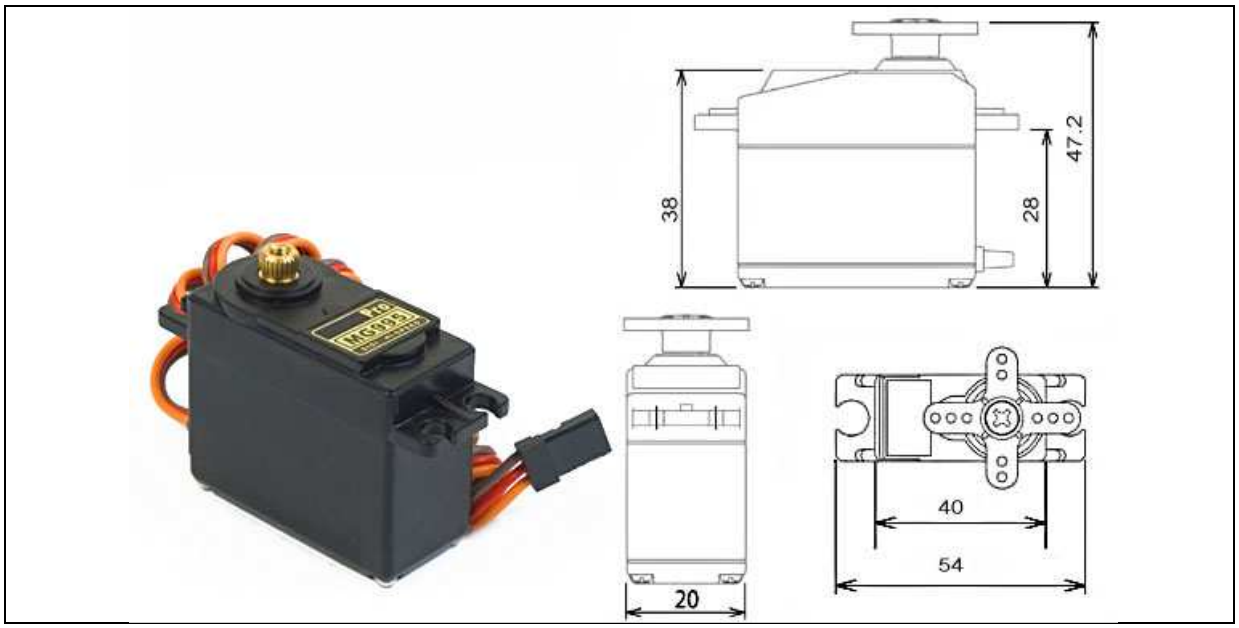


***Figure n°27: dimensions du servomoteur Towerpro MG995* [8].**

Ses caractéristiques sont synthétisées dans le tableau suivant :

Système de contrôle	Contrôle par largeur d'impulsion
Vitesse de fonctionnement à 4.8 V	0.20 sec / 60°
Couple à 4.8V	10.00 kg-cm
Type de moteur	Coreless
Type de roulement	Double roulement à bills
Type d'engrenage	métallique
Poids	55.0 g

***Tableau n°6: caractéristiques du servomoteur Tower pro MG995* [8].**

2- Les capteurs:

Afin de réaliser la commande gestuelle, des capteurs de vision sont nécessaire pour capter les mouvements de la main. Pour cela on a opté pour des webcaméras.

- **Les webcaméras:**

Une webcam est une caméra conçue pour être utilisée comme un périphérique d'ordinateur, cette dernière produit une vidéo qui n'est pas d'une haute qualité, mais qui peut être transmise en direct à travers d'un réseau, typiquement Internet.

Les webcams possèdent un capteur, qui peut être CCD (charge-coupled device), ou CMOS (complementarity metal-oxide-semiconductor). 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 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.

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

Figure n°28: Webcam Logitech c160

Introduction:

La commande d'un robot fait appel aux calcules de certains modèles mathématiques, tels que :

- les modèles de transformation entre l'espace opérationnel (où est définie l'organe terminal) et l'espace articulaire (dans lequel est définie la configuration du robot). On distingue trois classes de modèles :

- les modèles géométriques direct et inverse, qui expriment la situation de l'organe terminal en fonction des variables articulaires de la structure mécanique et inversement.

- les modèles cinématiques direct et inverse qui expriment la vitesse de l'organe terminal en fonction des vitesses articulaires et inversement.

- les modèles dynamiques définissant les équations du mouvement du robot, qui permettent d'établir les relations entre les couples ou forces exercés par les actionneurs et les positions, vitesses et accélérations des articulations.

Pour notre cas on va s'intéresser au modèle géométrique uniquement. Mais avant de commencer à le traiter, on va essayer de présenter quelques outils géométrique et mathématique qui vont nous êtres utiles pour modéliser les mouvements des corps solides dans l'espace.

1- Matrices de transformation:

Une transformation rigide est le résultat d'un mouvement rigide, amenant un solide d'une situation initiale à une situation finale. En transformant ses coordonnées de leur position initiale vers leur position finale.

La représentation de la matrice de transformation homogène noté ${}^{i}_{j}\boldsymbol{T}$ est basée sur les coordonnés homogènes (représentation d'un point et d'une direction).

Pour une matrice de transformation complète ${}^{i}_{j}\boldsymbol{T}$ d'un repère $\boldsymbol{R_j}$ ayant subi des rotations et des translations par rapport à un repère $\boldsymbol{R_i}$, il est possible d'utiliser une écriture générale. Cette écriture prend en compte les différents vecteurs (***s, n, a***) de la matrice de passage **[9]**.

$$ {}^{i}_{j}T = [\, {}^{i}_{j}s\ ,\ {}^{i}_{j}n\ ,\ {}^{i}_{j}a\ ,\ {}^{i}_{j}P\,] = \begin{bmatrix} s_x & n_x & a_x & p_x \\ s_y & n_y & a_y & p_y \\ s_z & n_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

La matrice ${}^{i}_{j}T$ est le résultat d'un vecteur de translation ${}^{i}_{j}P$ (3x1) et une matrice (3x3) de rotation formée par les vecteurs (***s, n, a***) soit:

$$ {}^{i}_{j}T = [\, {}^{i}_{j}s\ ,\ {}^{i}_{j}n\ ,\ {}^{i}_{j}a\ ,\ {}^{i}_{j}P\,] = \begin{bmatrix} & {}^{i}_{j}A & & {}^{i}_{j}P \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Où ${}^{i}_{j}s$, ${}^{i}_{j}n$ et ${}^{i}_{j}a$ de la matrice ${}^{i}_{j}T$ désignent respectivement les vecteurs unitaire suivant les axes X_j, Y_j et Z_j du repère R_j exprimés dans le repère R_i et ${}^{i}_{j}P$ est l'origine de repère R_j exprimés dans le repère R_i.

Les éléments de la matrice ${}^{i}_{j}A$ sont appelés **cosinus directeurs,** car ils représentent les coordonnées des trois vecteurs de la base R_j exprimés dans R_i.

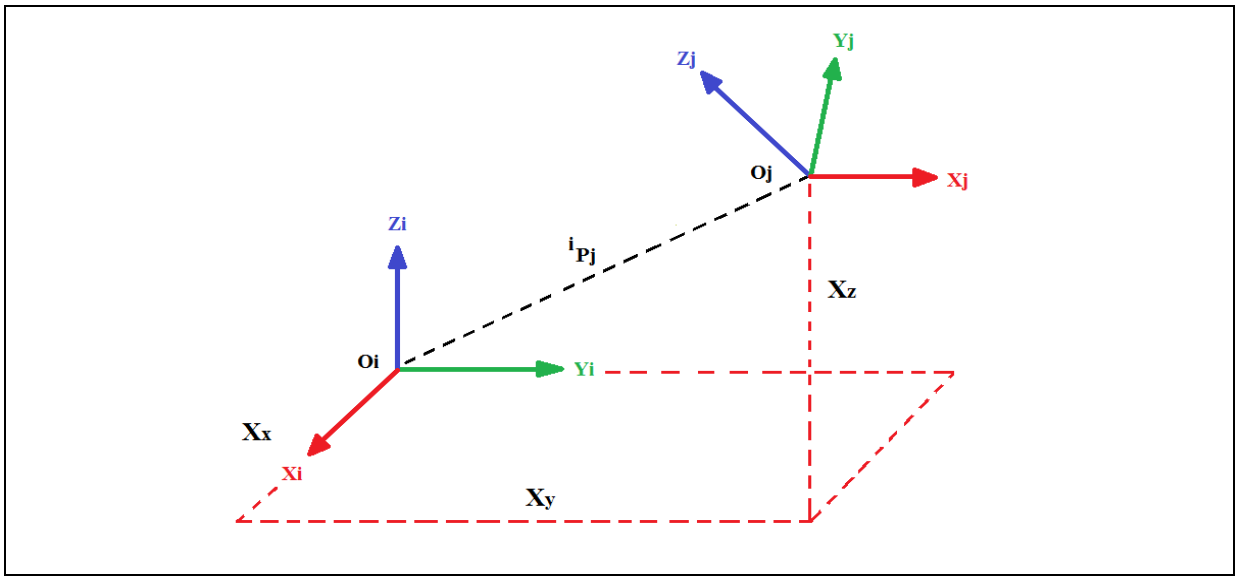
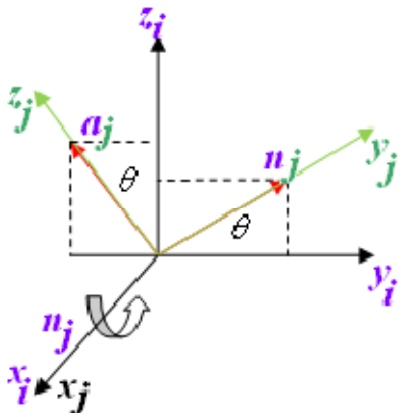


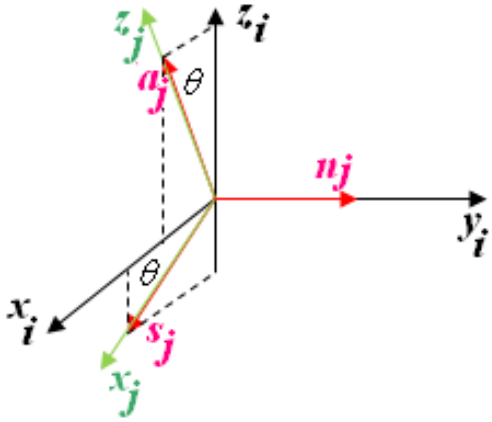
Figure n°29: transformation d'un repère.

1.1- Matrice de transformation à rotation pure:

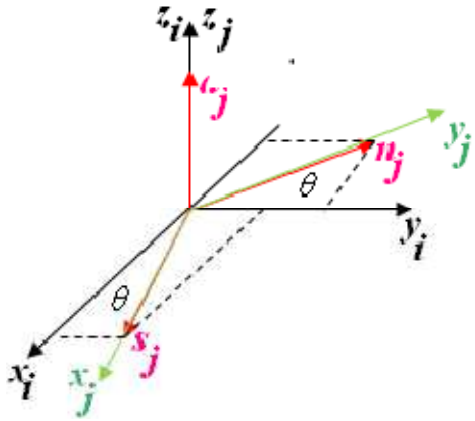
Les matrices de transformation à rotation pure sont formées par les vecteurs (s, n, a) et le vecteur P_i étant nulle. Soient Rot(x, θ), Rot(y, θ) et Rot(z, θ) des matrices de rotations autour des axes X, Y et Z successivement **[9]**.



$$Rot(x,\theta)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos(\theta) & -\sin(\theta) & 0\\ 0 & \sin(\theta) & \cos(\theta) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$



$$Rot(y,\theta) = \begin{bmatrix}cos(\theta) & 0 & sin(\theta) & 0\\ 0 & 1 & 0 & 0\\ -sin(\theta) & 0 & cos(\theta) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$



$$Rot(z,\theta)=\begin{bmatrix}cos(\theta) & -sin(\theta) & 0 & 0\\ sin(\theta) & cos(\theta) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

1.2- Matrice de transformation à translation pure:

Soit Trans(a, b, c) une transformation a, b, et c qui désigne respectivement une translation le long des axes x, y, et z.

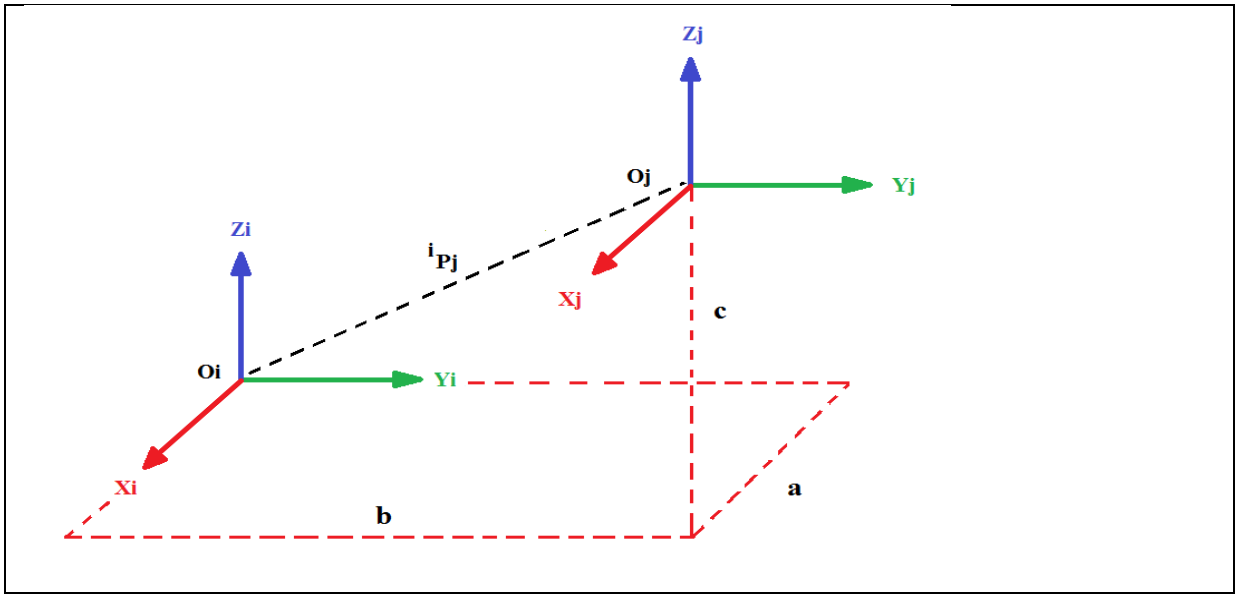


Figure n°30: translation d'un repère.

$$\text{Trans(a, b, c)}=\begin{bmatrix}1 & 0 & 0 & a\\ 0 & 1 & 0 & b\\ 0 & 0 & 1 & c\\ 0 & 0 & 0 & 1\end{bmatrix}$$

2- Modélisation géométrique:

2.1- Description géométrique:

Définir les différentes tâches d'un robot nécessite le positionnement de l'organe terminal par rapport à un repère de référence. Le modèle géométrique donne la relation entre les variables opérationnelles, **X**, représentant la position/orientation de l'organe terminal et les variables articulaires $\boldsymbol{\theta_i}$ qui représentent les translations et les rotations des liaisons articulaires du robot. Du moment que les liens de la SMA sont modélisés comme des corps solides, leurs propriétés de déplacement prennent une place essentielle dans la robotique:

- Les informations proprioceptives (issues du SMA) sont généralement définies dans des repères liés aux différents solides du robot.

- La position à atteindre est souvent définie dans un repère lié à la base du robot.

2.2- Modèle géométrique directe (MGD):

Le modèle géométrique directe est réduit à trouver une matrice de transformation homogène qui relie le repère de l'organe terminale à celui de la base du robot. Etant données les positions articulaires (distance et angle pour une articulation rotoïde respective), trouver l'attitude de l'organe terminal par rapport à la base. La modélisation des robots de façon systématique et automatique exige une méthode adéquate pour la description de leur morphologie. Plusieurs méthodes et notations ont été proposées, la plus répandue est celle de Denavit-Hardenberg.

2.2.1- Methode de Denavit-Hardenberg:

L'avantage d'utilisation de cette méthode est son universalité algorithmique. Cette méthode est 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.

Une structure ouverte simple est composée de n+1 corps notés C_0,…., C_n et de n articulations. Le corps C_0 désigne la base du robot et le corps C_n le corps qui porte l'organe terminal. L'articulation I connecte le corps C_i au corps C_{i-1} :

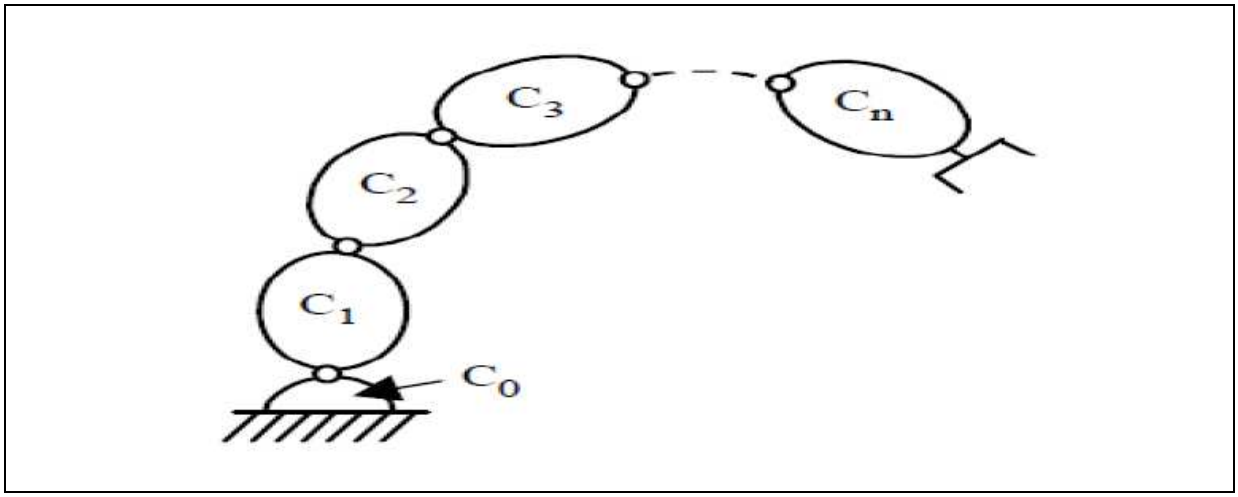


Figure n°31: Robot à structure ouverte simple **[9].**

2.2.2- Les paramètres de Denavit-Hartenberg :

Le passage du repère R_{i-1} au repère R_i s'exprime en fonction des quatre paramètres géométriques suivants **[10]**:

$\boldsymbol{\alpha_i}$: Angle entre les axes Z_{i-1} et Z_i correspondant à une rotation autour de X_i

$\boldsymbol{d_i}$: Distance entre X_{i-1} et X_i le long de Z_i ;

$\boldsymbol{\theta_i}$: Angle entre les axes X_{i-1} et X_i correspondant à une rotation autour de Z_i ;

$\boldsymbol{a_i}$: Distance entre Z_{i-1} et Z_i le long de X_i .

Ces paramètres sont illustrés dans la figure ci-dessous:

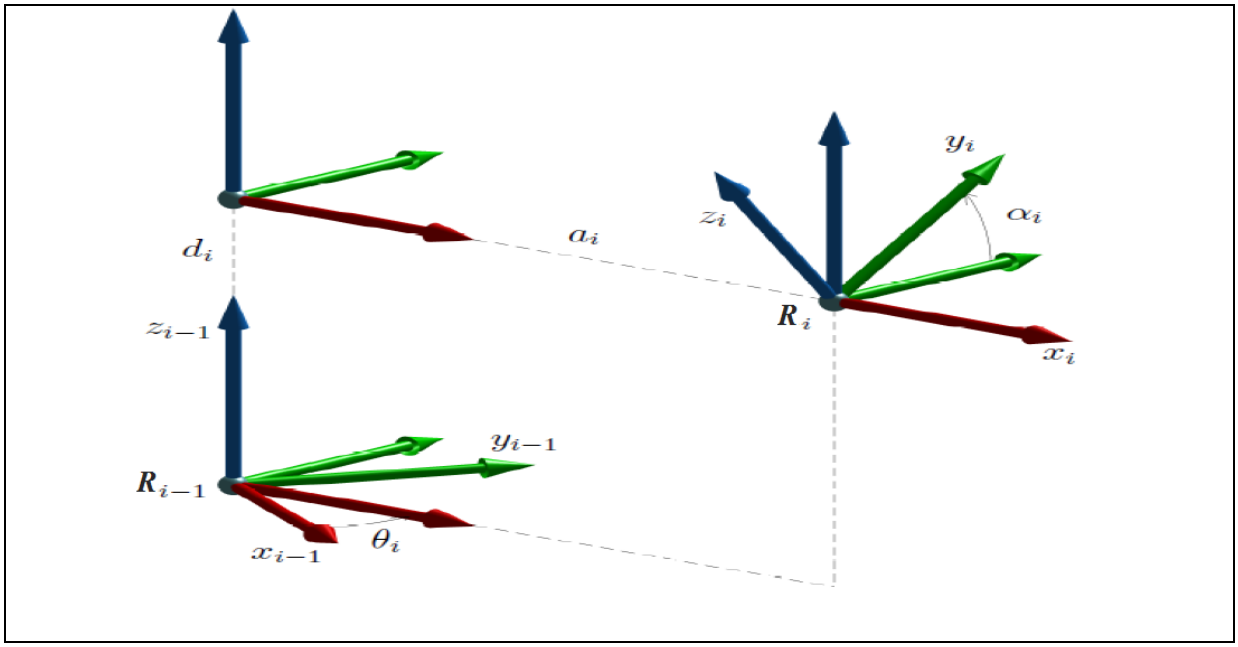


Figure n°32: Paramètres géométriques.

La matrice de transformation définissant le repère R_i dans le repère R_{i-1} est donnée par 4 transformations élémentaires **[10]**:

- Rotation autour de z d'un angle θ_i;
- Translation le long de z d'une longueur d_i;
- Translation le long de x d'une longueur a_i;
- Rotation autour de x d'angleα_i.

$$DH_{i-1,i} = R_{(z_{i-1},\theta_i)}\, T_{(z_{i-1},d_i)}\, T_{(x_i,a_i)}\, R_{(x_i,\alpha_i)}$$

Ces transformations sont illustrées ci-dessous.

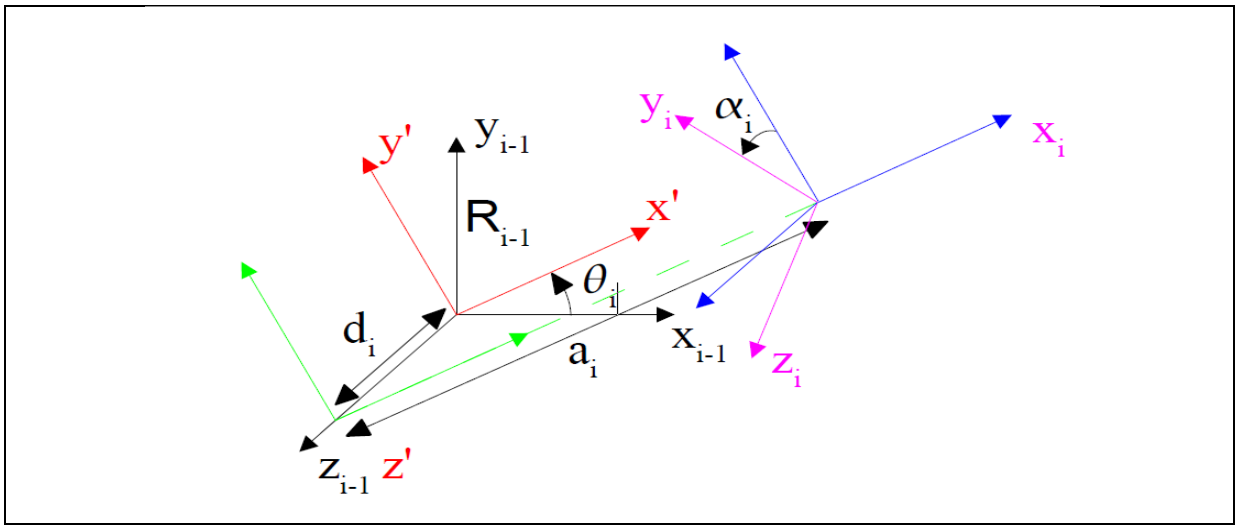


***Figure n°33: transformations élémentaires* [10].**

$$DH_{i-1,i} = \begin{bmatrix} cos\,(\theta) & -sin\,(\theta) & 0 & 0 \\ sin\,(\theta) & cos\,(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\,(\theta) & -\sin\,(\theta) & 0 \\ 0 & \sin\,(\theta) & \cos\,(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pour fixer les repères sur les différentes liaisons articulaires on suit les deux contraintes fixées par Denavit-Hartenberg **[10]**:

- DH1 : l'axe X_i de R_i est perpendiculaire à l'axe Z_{i-1} de R_{i-1} ;
- DH2 : l'axe X_i coupe l'axe Z_{i-1} .

Ces deux contraintes nous emmène à avoir le schéma de la représentation géométrique et le placement des repères de notre robot suivant:

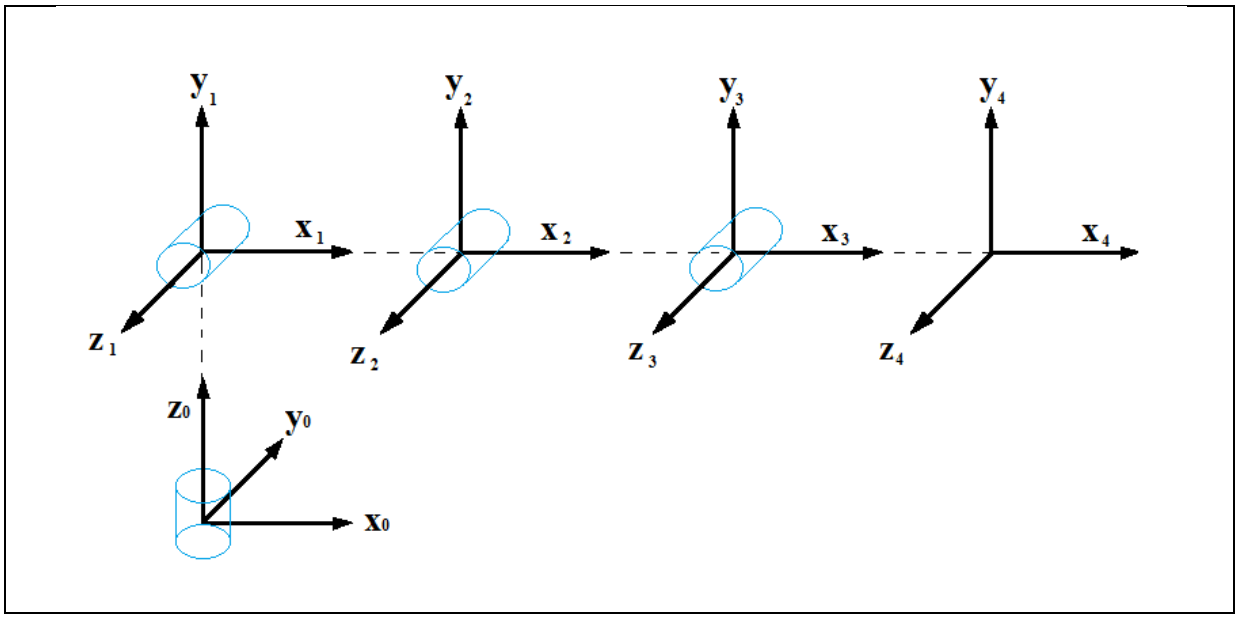


Figure n°34: placement des repères du robot.

2.2.3- Calcul du MGD:

Le tableau suivant représente tout les paramètres géométriques du robot :

Corps/paramètres	$\boldsymbol{\theta_i}$	$\boldsymbol{d_i}$	$\boldsymbol{\alpha_i}$	$\boldsymbol{a_i}$
1	$\boldsymbol{\theta_1}$	$\boldsymbol{d_1}$	-90	0
2	θ_2	0	0	a2
3	θ_3	0	0	a3
4	θ_4	0	0	a4

Tableau n°7 : les paramètres géométriques du robot

On remplaçant ces paramètres dans la matrice homogène $\boldsymbol{DH_{i-1,i}}$ on aura 4 matrices distinctes:

$$T_{0,1}=\begin{bmatrix} c\theta_1 & 0 & -s\theta_1 & 0 \\ s\theta_1 & 0 & c\theta_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{1,2}=\begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_2c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & a_2s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{2,3}=\begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & a_3c\theta_3 \\ s\theta_3 & c\theta_3 & 0 & a_3s\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{3,4}=\begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & a_4c\theta_4 \\ s\theta_4 & c\theta_4 & 0 & a_4s\theta_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On multipliant ces 4 matrices on obtient le modèle géométrique directe de notre robot, qui est le suivant:

$$T_{0,4}=\begin{bmatrix} c_{234}c\theta_1 & -s_{234}c\theta_1 & -s\theta_1 & c\theta_1(a_3c_{23}+a_2c\theta_2+a_4c_{234}) \\ c_{234}s\theta_1 & -s_{234}s\theta_1 & c\theta_1 & s\theta_1(a_3c_{23}+a_2c\theta_2+a_4c_{234}) \\ -s_{234} & -c_{234} & 0 & d_1-a_3s_{23}-a_2s\theta_2-a_4s_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Les notations étant généralement lourdes, on adopte un certain nombre de conventions soit:

$c\theta_i$=cos(θ_i);

$s\theta_i$=sin(θ_i);

c_{234}=cos($\theta_2+\theta_3+\theta_4$);

s_{234}=sin($\theta_2+\theta_3+\theta_4$).

2.3- Modèle géométrique inverse (MGI):

2.3.1- méthodes utilisées pour résoudre le MGI :

Ayant les coordonnées à atteindre par l'organe terminale, quelles est la configuration (variables articulaires $\boldsymbol{\theta_i}$) adéquate des différentes liaisons articulaires de la SMA? Voila la problématique du modèle géométrique inverse. La détermination des différentes variables

articulaires revient à résoudre des équations algébriques non linéaire couplées, cependant, il n'existe pas une méthode standard pour le résoudre. Contrairement au MGD qui est unique, il existe plusieurs MGI pour un robot (plusieurs configurations articulaire pour une position d'organe terminale donnée), ce qui est montré dans la figure ci-dessous pour un bras manipulateur planaire 2R.

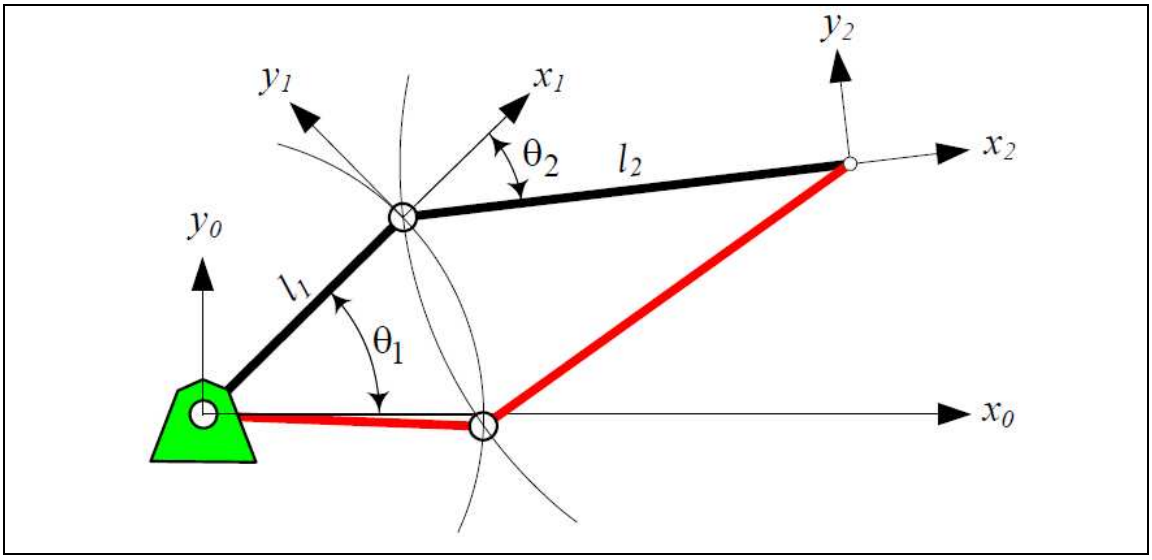


Figure n°35: multitude de solutions pour un bras manipulateur 2R.

Il existe principalement trois méthodes utilisées pour résoudre le MGI:

- **Méthode de Paul** : Elle traite séparément chaque cas particulier et convient pour la plupart des robots industriels **[11]**.

- **Méthode de Pieper** : Elle permet de résoudre le problème pour les robots à six degrés de liberté possédant trois articulations rotoïdes, d'axes concourants ou trois articulations prismatiques **[11]**.

- **Méthode générale de Raghavan et Roth** : on donnera la solution générale des robots à six articulations à partir d'un polynôme de degrés au plus égal à 16 **[11]**.

2.3.2 - Calcul du MGI:

Pour notre robot qui est un bras manipulateur à 4ddl, on a opté pour une méthode géométrique qui est plus simple, et qui consiste à une projection sur les différents plans du repère fixé sur la base du robot. Pour rendre plus simple le calcule du MGI, on a procédé à un découplage qui consiste à traiter la première variable articulaire liée à la base seule, et les trois restantes seules.

- **Première variable articulaire:**

Comme le montre la figure ci-dessous, on faisant une simple projection sur le plan (*OXY*) on obtient facilement la première variable articulaire $\boldsymbol{\theta_1 = atan2(o_y, o_x)}$

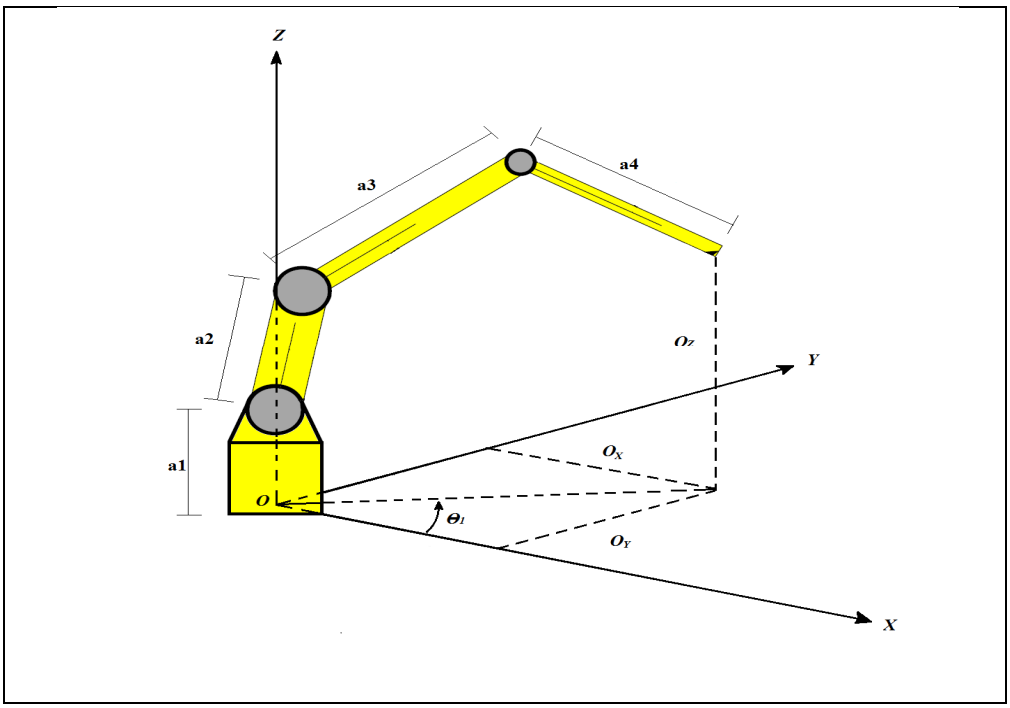


Figure n°36: projection sur le palan (OXY)

- **Les trois variables restantes:**

On faisant un découplage, le traitement du problème sur un plan nous rendra la tache plus facile **[12]**.

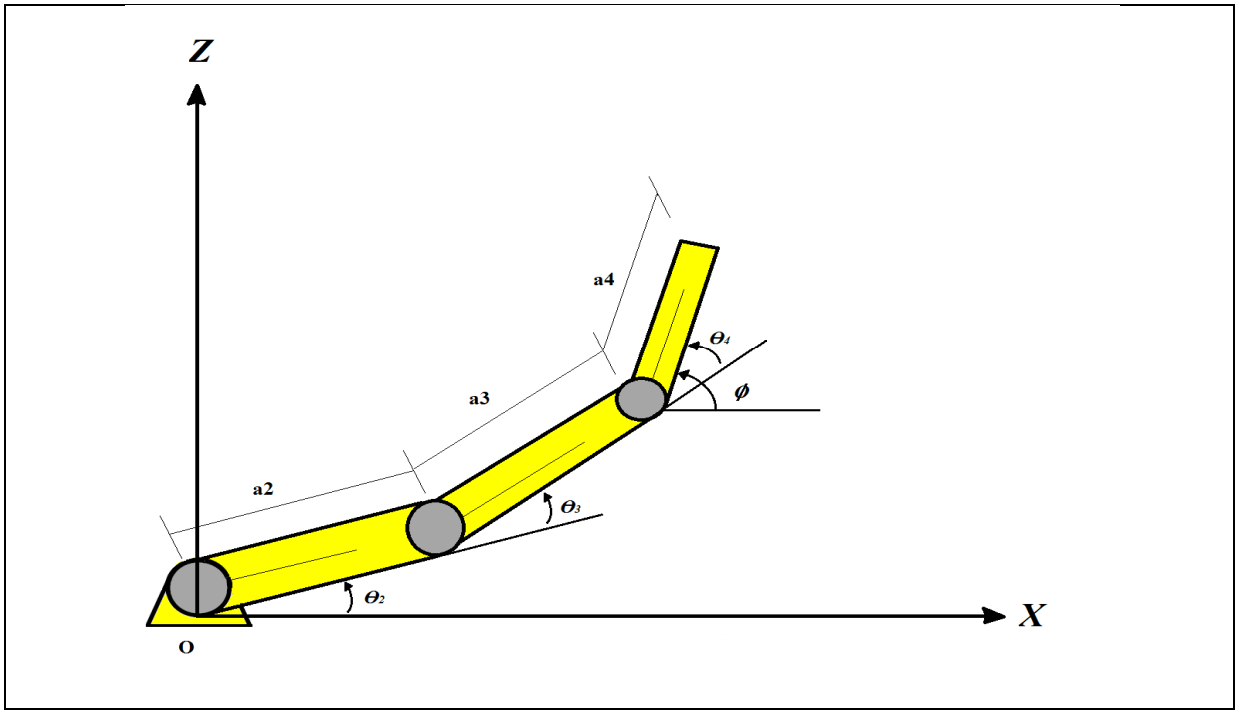


Figure n°37: projection sur le plan (OXZ)

Par simple projection sur les axes OX et OZ on obtient les deux équations suivantes:

- ✓ $O_x = a_2 c\theta_2 + a_3 c_{23} + a_4 c_{234}$
- ✓ $O_z = a_1 + a_2 s\theta_2 + a_3 s_{23} + a_4 s_{234}$

On pose $\emptyset = \boldsymbol{\theta_2 + \theta_3 + \theta_4}$

Où $\emptyset$ c'est l'angle que fait l'organe terminal avec l'horizontale (l'axe des X) comme c'est illustré dans la figure ci-dessus. Par la suite on aura:

$$O_x = a_2 c\theta_2 + a_3 c_{23} + a_4 c_{\emptyset}$$

$$O_z = a_1 + a_2 s\theta_2 + a_3 s_{23} + a_4 s_{\emptyset}$$

$$\begin{cases} W_x = O_x - a_4 c_{\emptyset} = a_2 c\theta_2 + a_3 c_{23} \\ W_z = O_z - a_4 s_{\emptyset} - a_1 = a_2 s\theta_2 + a_3 s_{23} \end{cases}$$

On sommant les deux équations au carré et on utilisant la formule trigonométrique suivante:

cos (a-b) = cos (a) cos (b) + sin (a) sin (b)

On obtient la formule suivante :

$$W_x{}^2 + W_z{}^2 = \mathrm{a}_2{}^2 + \mathrm{a}_3{}^2 + 2\mathrm{a}_2\mathrm{a}_3\, c\theta_3$$

On tire: $c\theta_3 = \frac{W_x{}^2 + W_z{}^2 - \mathrm{a}_2{}^2 - \mathrm{a}_3{}^2}{2\mathrm{a}_2\mathrm{a}_3} = A$

De là on a la troisième variable articulaire :

$$\boldsymbol{\theta_3 = \mathbf{acos}\,(A)}$$

De même, on utilisant la formule trigonométrique suivante:

Cos (a+b)=cos (a) cos (b) – sin (a) sin (b)

On aura:

$$\begin{cases} W_x = (\mathrm{a}_2 + \mathrm{a}_3 c\theta_3)c\theta_2 - \mathrm{a}_3 s\theta_2 s\theta_3 \\ W_z = (\mathrm{a}_2 + \mathrm{a}_3 c\theta_3)s\theta_2 - \mathrm{a}_3 c\theta_2 s\theta_3 \end{cases}$$

$$s\theta_2 = \frac{(\mathrm{a}_2 + \mathrm{a}_3 c\theta_3)\, W_z - \mathrm{a}_3 s\theta_3 W_x}{W_x{}^2 + W_z{}^2} \qquad c\theta_2 = \frac{(\mathrm{a}_2 + \mathrm{a}_3 c\theta_3)\, W_x + \mathrm{a}_3 s\theta_3 W_z}{W_x{}^2 + W_z{}^2}$$

De là :

$$\boldsymbol{\theta_2 = atan2(s\theta_2, c\theta_2)}$$

On imposant l'angle ∅ que fait l'organe terminal avec l'axe des X, on pourra facilement tirer la variable articulaire :

$$\boldsymbol{\theta_4 = \emptyset - \theta_3 - \theta_2}$$

Conclusion:

Modéliser un robot, revient à modéliser sont comportement sous forme mathématique. D'une manière générale, on recherche toujours le modèle le plus simple qui permet d'expliquer, de manière satisfaisante, le comportement du processus dans son domaine d'application.

Introduction:

"De même qu'un trou n'est défini que par ce qui l'entoure, un robot n'a de sens que dans un environnement qu'il modifie" (P.Coiffet). Un robot doit posséder des capacités de perception et de mouvement nécessaires pour qu'il puisse exécuter ses tâches dans son environnement. L'enjeu de la vision par ordinateur est de permettre à un ordinateur de "voir" et de prendre conscience de ce qui l'entoure, en récupérant des informations visuelles par l'intermédiaire d'un système d'acquisition d'image puis les exploiter.

1- Définition:

La vision par ordinateur est la transformation de données acquises par l'intermédiaire d'un système d'acquisition d'image en une commande ou une autre représentation. C'est une science qui développe les bases théoriques et algorithmiques, afin d'extraire et d'analyser 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 (existence d'une pièce), la reconstruction en trois dimensions d'une scène ou un objet, la position et l'orientation ou la mesure de toute propriété spatiale d'un objet (la distance entre deux de ses points distingués ou le diamètre de la section circulaire…*etc*.).

Dans notre projet, le traitement désiré est d'extraire la position d'un objet en se basant sur la stéréovision et la segmentation d'image. Pour ce faire, nous avons opté pour une bibliothèque en C/C++ nommée OpenCV (voir annexe).

2- Stéréovision:

La stéréovision est l'un des sujets clés de la recherche en vision par ordinateur actuellement. Elle s'est considérablement développée dans le domaine de la robotique où une reconnaissance spatiale de l'environnement entourant le robot est nécessaire au déplacement.

La stéréovision par ordinateur est la reproduction artificielle de la perception naturellement présente chez l'homme, elle regroupe des techniques d'extraction d'informations relatives au relief à partir de deux images, qui seront récupérer par un algorithme de stéréo-dense afin de former une carte de disparité.

Il existe trois étapes cruciales qui rentrent en jeu pour calculer la distance ou les coordonnées d'un objet de la caméra:

- La calibration du capteur stéréoscopique (stéréo-calibration);
- La mise en correspondance stéréo dense;
- la triangulation.

2.1- Capteur stéréoscopique actif:

En combinant une caméra avec une source de lumière, un faisceau laser éclaire une partie de la scène qui correspond à l'intersection du plan lumineux et de la scène afin d'extraire les coordonnées tridimensionnelles de points sur la surface d'un objet.

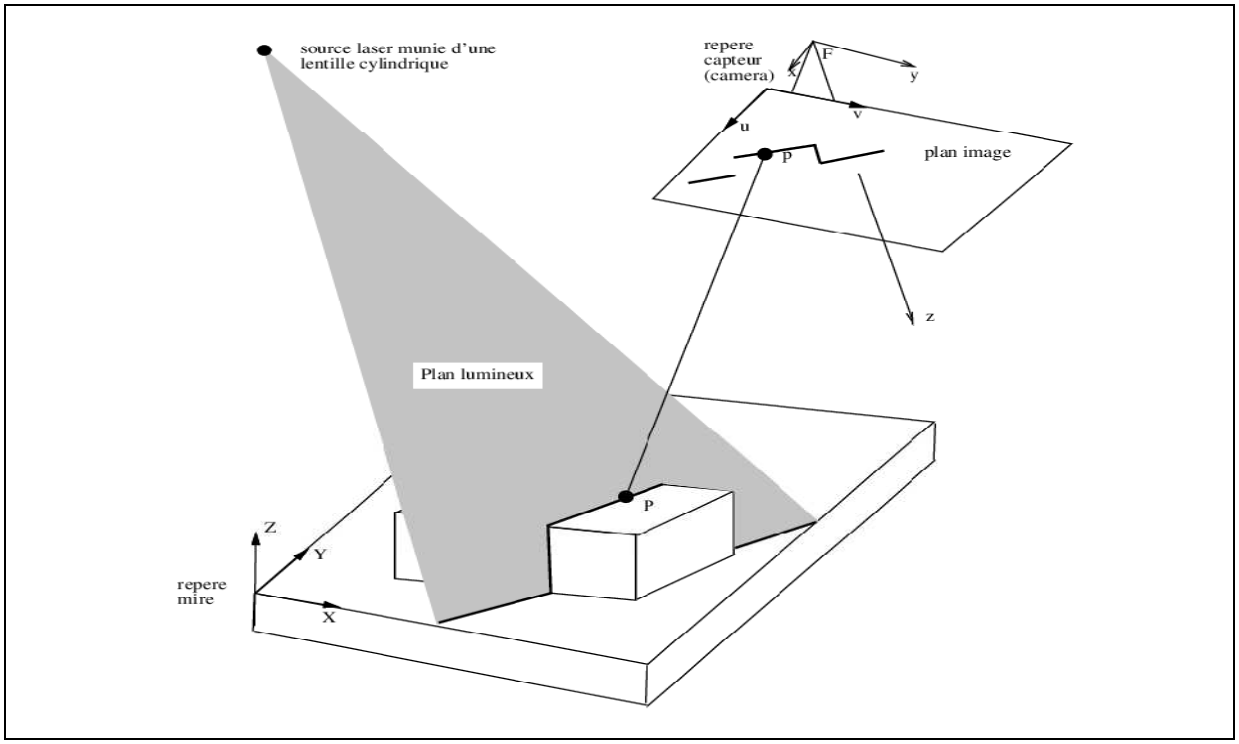


Figure n°38: capteur stéréoscopique actif.

2.2- Capteur stéréoscopique passif:

Les capteurs passifs sont constitués de deux caméras alignés horizontalement et séparés par une distance (baseline), afin qu'ils aient leurs axes optiques parallèles pour récupérer deux projections de la même scène mais de perspective différentes.

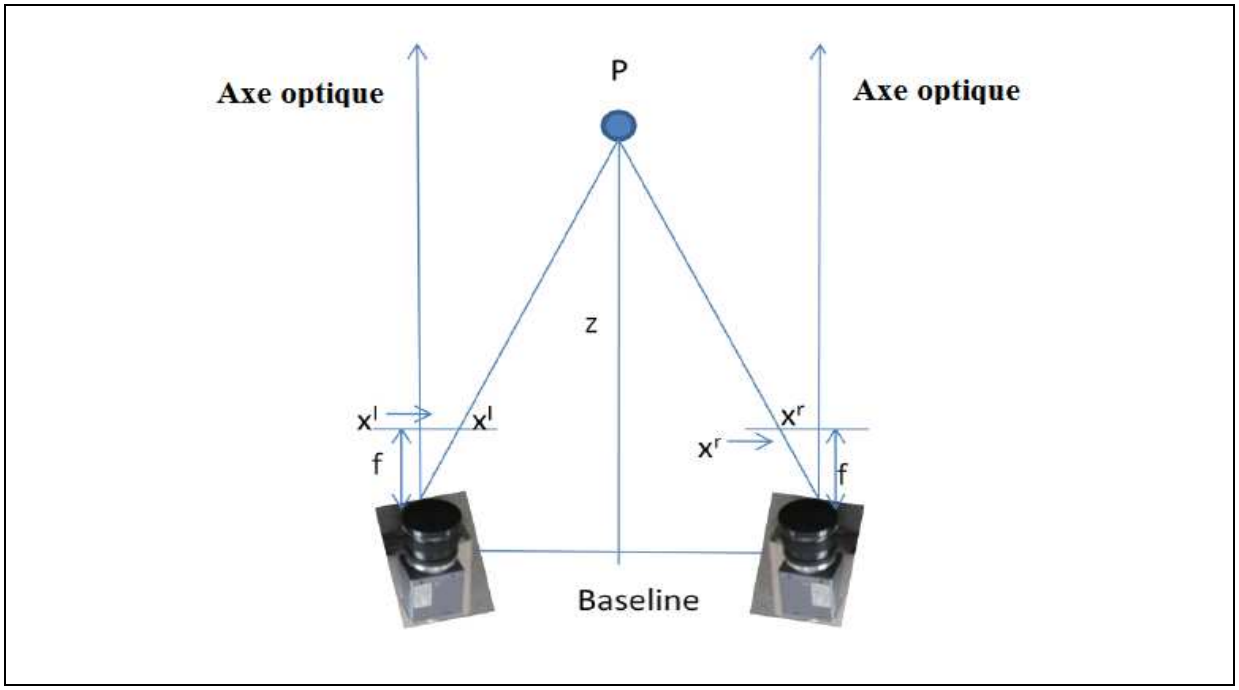


Figure n°39: capteur stéréoscopique passif.

2.3 - Modèle de caméra:

La plupart des méthodes de calibration utilisent le modèle du sténopé pour décrire le modèle d'une caméra. C'est une approche qui permet de modéliser la plupart des capteurs et qui simplifie l'estimation de ses paramètres. Dans ce simple modèle la lumière émie par le volume de travail ou un objet distant de la caméra, passe par le sténopé pour être projeté sur le plan du capteur. Le principe de ce modèle est illustré dans la figure suivante **[13]**.

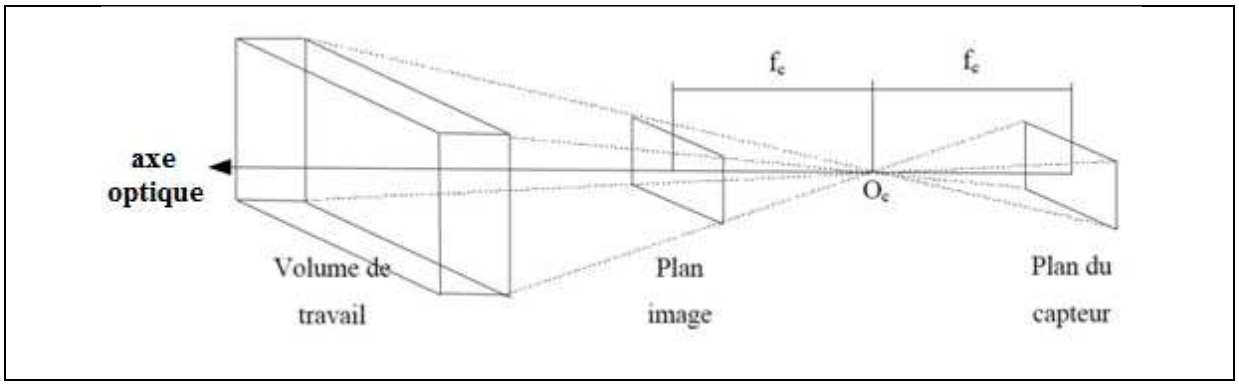


Figure n°40: modèle sténopé d'une caméra.

Dans le cas d'une caméra, les droites de vue sont issues d'un point unique : le centre optique de la caméra. Le plan image d'une caméra se trouve à la distance (en mm) $\boldsymbol{f_c}$ du centre optique$\boldsymbol{o_c}$, $\boldsymbol{f_c}$ étant la distance focale de l'objectif de la caméra utilisée.

Dans le plan du capteur, les coordonnées sont exprimées en pixel, tandis que dans les autres systèmes l'unité principale est le millimètre.

2.3.1- Modèle intrinsèque de la caméra:

Le modèle intrinsèque de la caméra permettra d'effectuer la relation entre un point 3D et son projeté 2D observable dans le plan du capteur. Les coordonnées du point 3D doivent être exprimées dans le repère du capteur, et les coordonnées 2D du point projeté seront exprimées dans le repère image, en pixel.

Les paramètres principaux du modèle intrinsèque sont la longueur focale " $\boldsymbol{f_c}$ " et le centre de l'image "$\boldsymbol{c}$".

En vérité, le centre de l'image ne se trouve pas toujours sur l'axe optique, pour cela, deux nouveaux paramètres $\boldsymbol{c_x}\ et\ \boldsymbol{c_y}$ sont introduits pour exprimer un éventuel déplacement du centre de l'image de l'axe optique **[13]**.

Pour une caméra de qualité moyenne, les pixels ont une forme rectangulaire au lieu de carré, ce qui nécessite l'introduction de deux focales $\boldsymbol{f_x}\ et\ \boldsymbol{f_y}$ qui sont les facteurs d'agrandissement de l'image. la longueur focale $\boldsymbol{f_x}$ est le produit de la longueur focale physique $\boldsymbol{f_c}$ et la taille du pixel $\boldsymbol{s_x}$ sur le plan de l'image, ce qui donne $\boldsymbol{f_x = f_c . s_x}$ .et de même pour $\boldsymbol{f_y}$.

Ces quatre paramètres sont réunis dans une matrice homogène nommée matrice intrinsèque:

$$\boldsymbol{A = \begin{vmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{vmatrix}}$$

2.3.2- Modèle extrinsèque de la caméra:

Le modèle extrinsèque de la caméra englobe les paramètres de positionnement de la caméra dans l'espace selon trois translations et trois rotations. Généralement nous prenons la caméra gauche comme référence et nous calculons la translation et rotation sur les trois axes X, Y et Z de la caméra droite par rapport à la référence (caméra gauche).

Soit le repère du monde, ayant comme origine o_w et le repère de la caméra placé sur le centre optique de la caméra o_c comme montré dans la figure suivante:

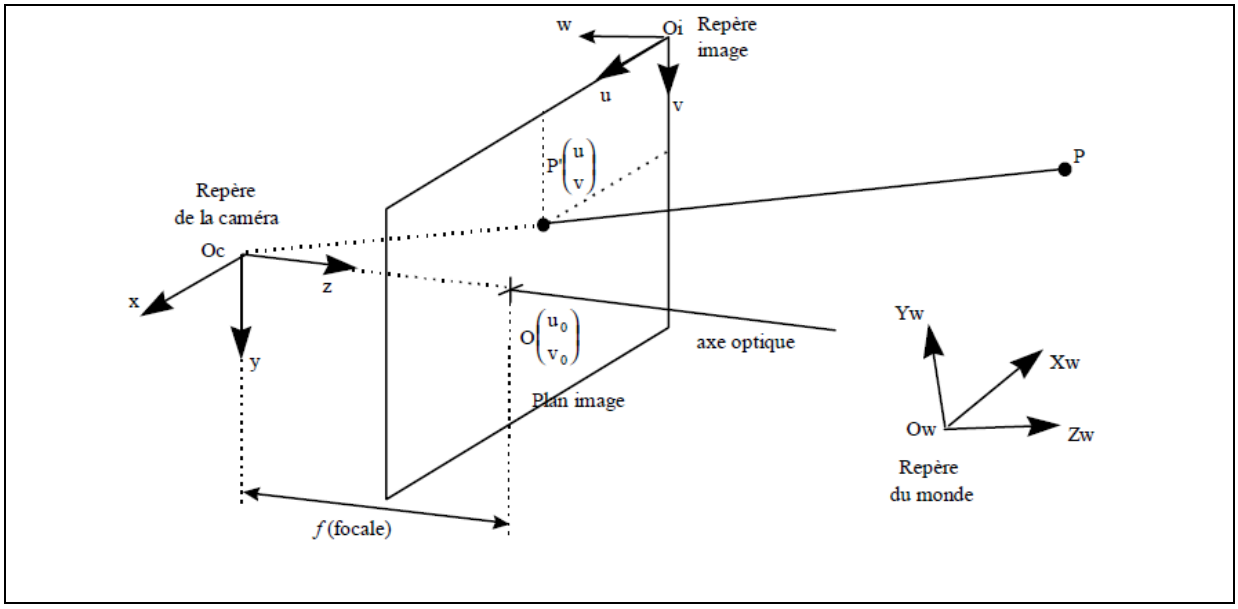


Figure n°41: représentation des repères utilisés.

La relation géométrique qui existe entre ces deux repères permet de définir les paramètres extrinsèques du modèle, afin de déterminent la position et l'orientation du repère de la caméra par rapport au repère du monde. La position du point focal o_c par rapport à o_w est donnée par le vecteur translation **T** de la forme:

$$\mathbf{T} = \begin{vmatrix} T_x \\ T_y \\ T_z \end{vmatrix}$$

L'orientation des axes du repère de la caméra par rapport au repère du monde est donnée par la matrice de rotation **R**, cette dernière est obtenue par un simple produit matriciel de trois matrices de rotation autour des axes X, Y et Z.

Ces deux matrices sont regroupées dans une matrice générale nommée matrice extrinsèque de la caméra qui est la suivante:

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

Les deux matrices **E** et **A** forment le modèle géométrique de la caméra.

2.3.3- Les distorsions:

En théorie, il est possible de définir une lentille parfaite qui ne cause pas de distorsion, en pratique par contre, il n'existe pas de lentille parfaites. Il existe deux types de distorsion essentielle à prendre en compte, les distorsions radiale et tangentielle.

La distorsion radiale est due à la forme imparfaite de la lentille, ce qui cause une distorsion de la localisation des pixels à la bordure du plan de l'image. Cette distorsion est exprimée par les trois termesk_1, k_2 *et* k_3.

La deuxième distorsion est la distorsion tangentielle, cette dernière est due au défaut de construction, où la lentille n'est pas fixée d'une manière parallèle exacte au plan de l'image. Cette distorsion est exprimée par deux paramètres p_1 *et* p_2.

Ces Cinque paramètres sont regroupés dans un vecteur nommé: vecteur de distorsion

$$\mathbf{D} = \begin{vmatrix} k_1 \\ k_2 \\ k_3 \\ p_1 \\ p_2 \end{vmatrix}$$

2.4- Calibration du capteur stéréoscopique:

Le calibrage des caméras est une étape essentielle pour l'extraction d'informations géométriques (distances, surfaces, volumes) précises à partir d'images stéréoscopiques. La stéréo-calibration consiste à trouver une relation géométrique entre les deux caméras dans l'espace, et de faire la correspondance entre les points de la scène observée et les points dans le plan du capteur.

Le processus standard de calibration consiste à prendre une image d'un objet dont la structure est parfaitement connue, le plus souvent l'objet à utiliser est un échiquier dont le

nombre de lignes et colonnes est connu. La mise en correspondance des cases de l'échiquier et de leur projection respective dans le plan image permet de déterminées les paramètres de la caméra cités auparavant. Un simple algorithme de détection de coins peut reconnaître les intersections noire et blanche des cases de l'échiquier.

2.5- Géométrie épipolaire et rectification:

La géométrie épipolaire combine deux modèle sténopé, elle 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 OL et le point X. Par contre, dans la vue de droite, que nous appellerons l'image de recherche, il est vu comme une ligne appelée ligne épi-polaire. Ce qui signifie que pour chaque pixel de l'image source, correspond une ligne épipolaire dans l'image de recherche **[13]**.

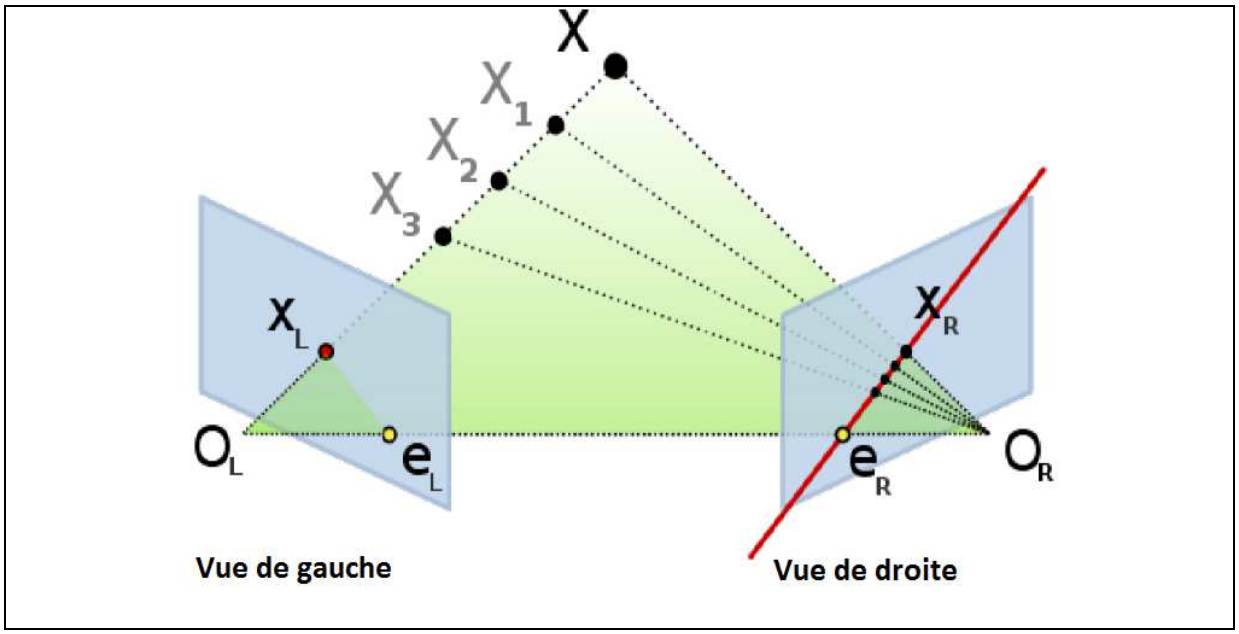


Figure n°42: géométrie épipolaire.

Pour trouver les lignes épipolaires, deux matrice essentielles sont nécessaires, la matrice E qui est la matrice extrinsèque de la caméra et la matrice F qui est la matrice E augmenté par les matrices intrinsèque des deux caméras.

Une fois ces deux matrices trouvées, on peut procéder à la rectification. Le but de cette dernière, est de projeter le plan image des deux caméras (d'une manière mathématique) afin qu'il soit sur le même plan où les lignes épi-polaires soient colinéaires. La figure ci-dessous montre le processus de la rectification.

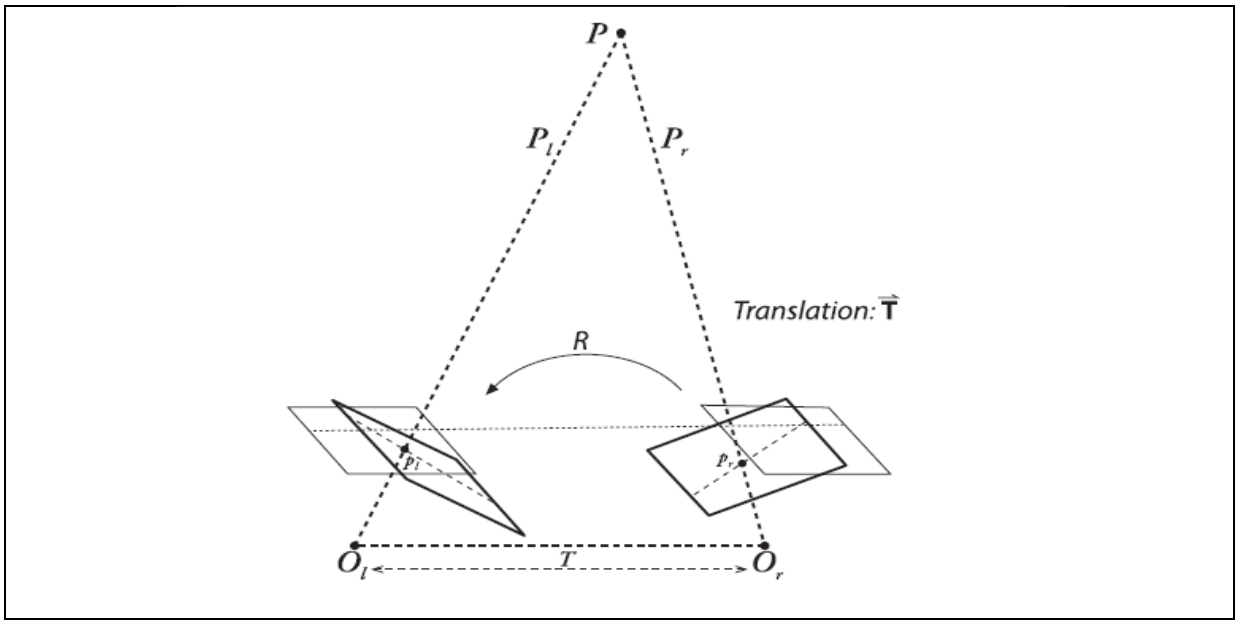


Figure n°43: processus de rectification **[13].**

Les résultats de ce processus sont huit paramètres, représentés par quatre pour chaque caméra. En effet, pour chaque caméra on a un vecteur de distorsion, une matrice de rotation et les matrices rectifié et non rectifié de la caméra. Ainsi, on peut créer une cartographie pour définir où interpoler les pixels de l'image originale afin de créer une image rectifiée.

2.6- La mise en correspondance stéréo dense:

Pendant la mise en correspondance, les images rectifiées des deux caméras sont utilisées pour faire correspondre les points d'une image à l'autre. Pour déterminer la distance d'un point de la caméra, une disparité doit être trouvée, qui représente le changement de localisation d'un point dans l'image gauche par rapport à l'image de droite.

Il existe plusieurs algorithmes de stéréo dense dont on peut citer:

- Graph Cut;

- Believe Propagation;

- programmation dynamique;

- Block Matching;

- Semi Global Block Matching.

2.6.1- Block matching (BM):

Cette méthode est basée sur l'idée d'utiliser la somme des distances absolus des blocks de pixels (sum of absolute difference SAD) pour trouver les points de correspondance entres les deux images rectifiées. Les étapes de cet algorithme sont énumérées comme suit:

- Normaliser les deux images afin qu'elles aient la même luminosité;

- Chercher la correspondance le long des lignes épipolaires;

- Eliminer la mauvaise correspondance.

Dans le bute d'avoir une bonne mise en correspondance, pour chaque pixel de l'image gauche on cherche la ligne correspondante dans l'image de droit. Pour un pixel de coordonnées (x_0, y_0) dans l'image de gauche, son correspondant dans l'image de droite doit être sur la même ligne, sur la même coordonnée ou à gauche de cette dernière. La figure suivante illustre ce qu'on vient d'expliquer.

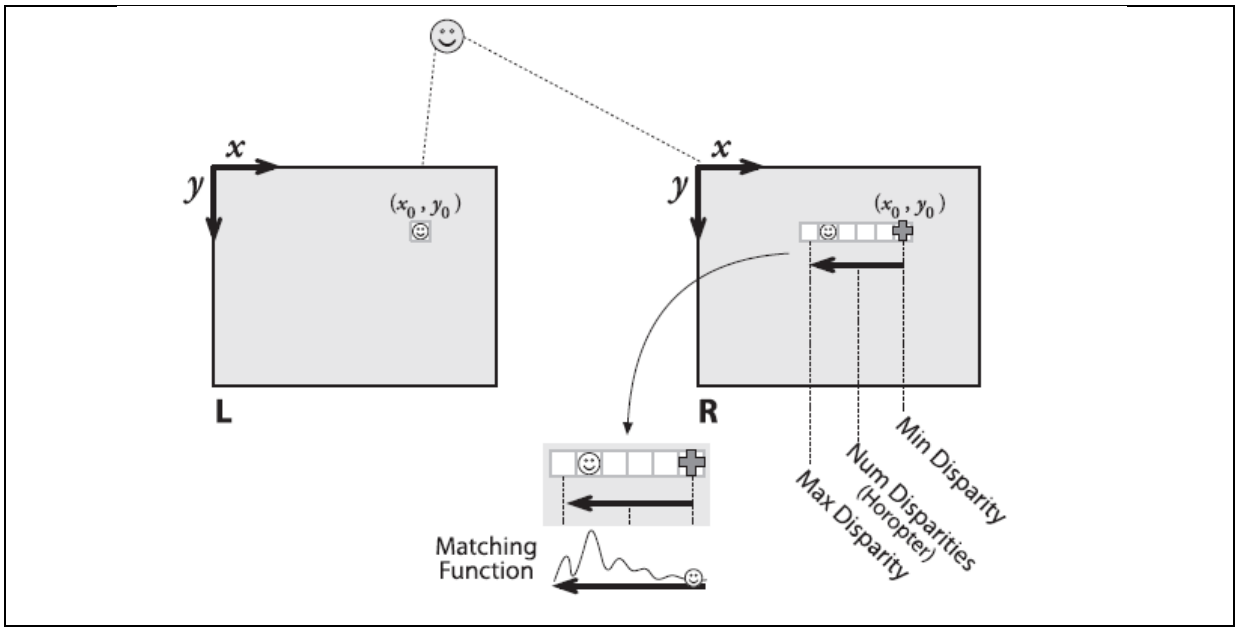


Figure n°44: principe de disparité.

La recherche de la correspondance se fait de droite vers la gauche, en commençant par "***min disparity",*** en passant par un nombre donné de disparité (distance en pixel à partir du pixel initial à localiser pour la correspondance).

Le résultat de la mise en correspondance est une image, où chaque pixel est la disparité trouvée à partir de l'image de gauche et de droite. Cette image et nommée "carte de disparité".

Cette dernière est une image 2D où chaque pixel a une couleur en niveau de gris qui représente sa distance par rapport à la caméra. La couleur des pixels proche de la caméra tend vers le blanc, et celle des pixels loin de la caméra tend vers le noir, tandis que les pixels entre les deux sont représentés par une échelle de gris.

Figure n°45: mise en correspondance stéréo dense.

2.6.2- Semi global block matching (SGBM):

Cette méthode établit une fonction de cout d'énergie, en suivant un chemin unidimensionnel dans différentes directions sur l'image. Il suffit de suivre 8 à 16 chemins pour couvrir toute l'image, pour chaque chemin le minimum de cout est calculé. L'algorithme SGBM minimise l'énergie pour trouver les valeurs parallaxe des pixels qui mènent à créer une carte de disparité. Les étapes de cet algorithme sont **[14]**:

- Calcule du cout;
- Agrégation du cout;
- Calcule de disparité;
- Le raffinement de disparité.

Pour avoir une carte de disparité optimale, l'algorithme d'SGBM qui est implémenté dans la bibliothèque OpenCV, nous donne la possibilité de travailler avec des paramètres réglables:

- 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.

2.7- Triangulation:

Apres avoir effectué les trois premières étapes de la stéréo vision (stéréo-calibration, mise en correspondance stéréo dense), 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, à savoir : les centres des rétines des deux caméras (C1, C2), la longueur focale (F) , les plans images (R1,R2), et les deux points (P1, P2) qui sont la projection du point P dans les deux images, V1 et V2 sont les distances horizontales entre P1 et C1, P2 et C2 respectivement et b la distance en mm qui sépare C1 de C2 comme le montre la figure ci-dessous.

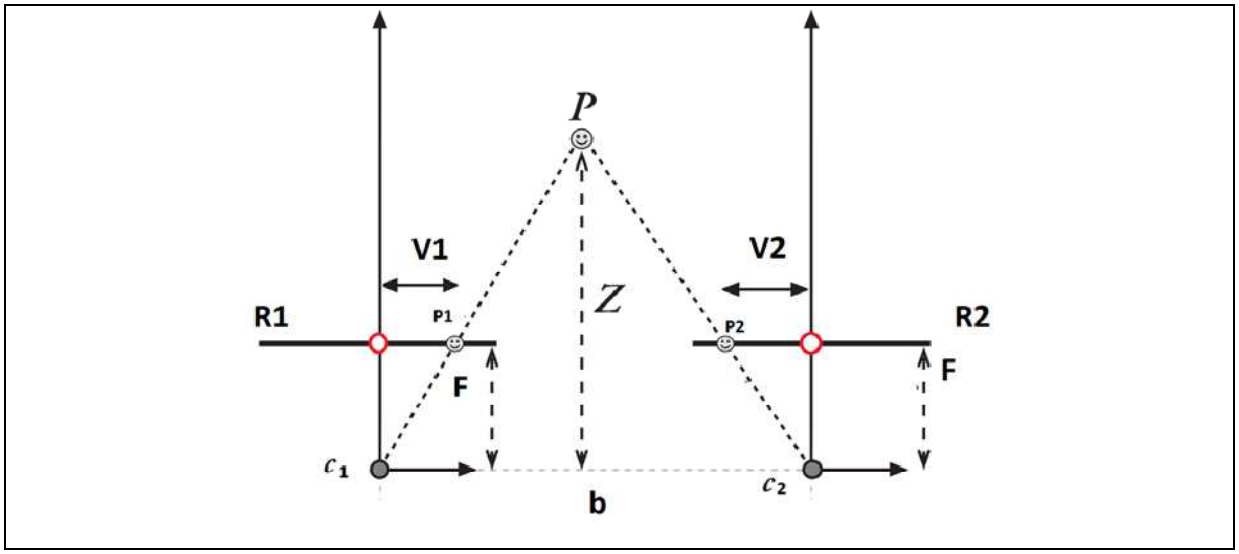


Figure n°46: principe de triangulation.

La disparité (D) des points P_1 et P_2 (les projections du point p) 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 calcule 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:

$$Z = \frac{b \cdot F}{D} \qquad X = \frac{V_1 \cdot Z}{F} \qquad Y = \frac{y_1 \cdot Z}{F}$$

Où y_1 est la position de P_1 selon l'axe des Y dans l'image gauche **[13]**.

Conclusion:

Dans ce chapitre on a expliqué le principe de la vision par ordinateur, ainsi que les différentes étapes nécessaires pour calibrer une paire stéréoscopique. Générer les paramètres cités auparavant et les combiner avec la robotique, nécessite certaines méthodes qu'on verra dans le prochain chapitre.

Introduction:

De manière générale, commander un robot consiste à déplacer son organe terminal pour qu'il atteigne une position désirée. Dans notre projet, nous avons commandé le robot en lui envoyant les coordonnées (x , y , z) du centre de la main qu'on a extraites à l'aide de la stéréovision, afin qu'il simule les mouvements de la main en temps réel (en même temps que les coordonnées de la main changent, la position de l'organe terminale change).

Ce chapitre va énumérer d'une manière explicite toutes les étapes utilisées pendant la pratique, en passant par le programme qu'on a utilisé pour la stéréovision et celui utilisé pour la commande des différents servomoteurs du robot et l'interfaçage entre les deux.

1- Robot/système de commande:

Le plan du projet se présente comme suit:

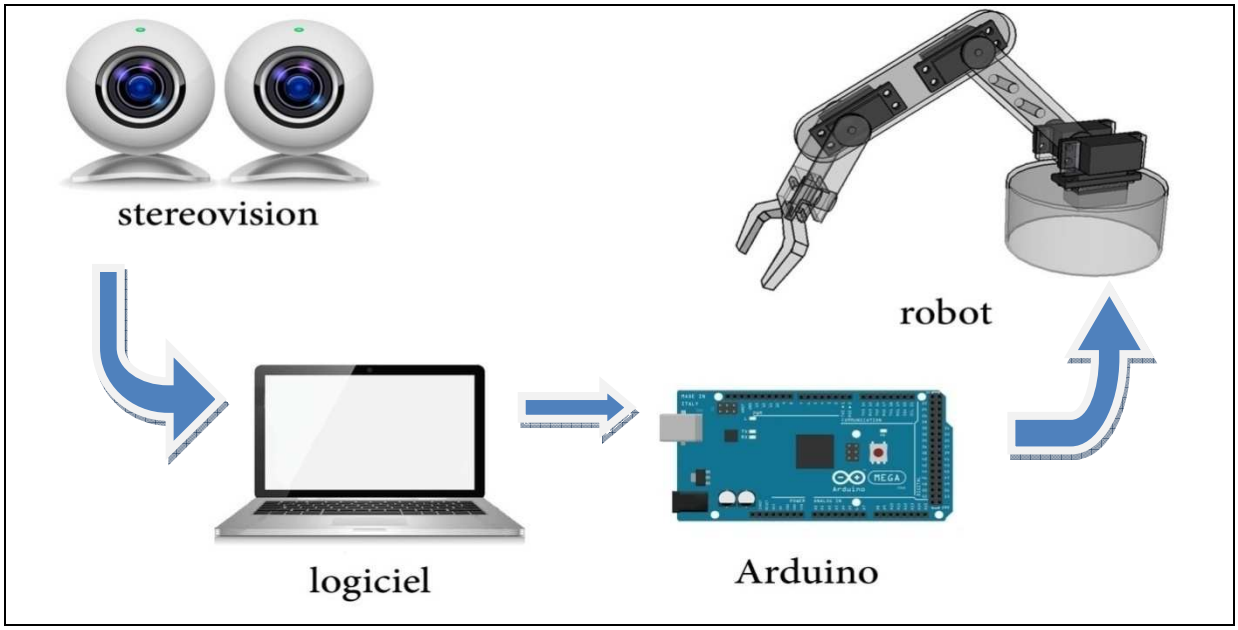


Figure n°47: plan du projet

2- Logiciel:

Le logiciel utilisé dans ce projet est composé de deux parties essentielles, la partie vision et la partie commande. Dans ce qui suit on va expliquer d'une manière détaillée les différents sous programmes appropriés à ces deux parties essentielles.

2.1- Partie vision:

La première tâche à faire est d'ajuster manuellement les deux caméras afin qu'elles aient les axes optiques parallèles. Pour ce qui en suit, les sous programmes sont implémentés et expliqués dans l'ordre cité dans le chapitre précédent.

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

2.1.1- Premier sous programme:

Le premier sous programme a pour but la capture de paires d'image gauches et droite simultanément. Les deux caméras ont la même scène à capturer mais de perspectives différentes. La mire utilisée pour la calibration est un damier de 10x7 carreaux. Les caméras prennent plusieurs captures en niveau de gris du damier avec différents positions (rotation et translation).

Pour ce faire on utilise les fonctions suivantes:

- *cvCreateCameraCapture()*: cette fonction prend une capture d'image à partir de la webcam.
- *cvtColor()*: cette fonction convertie l'image prise en niveau de gris.
- *cvSaveImage()*: cette fonction sert à enregistrer les images prises.

Le résultat est le suivant:

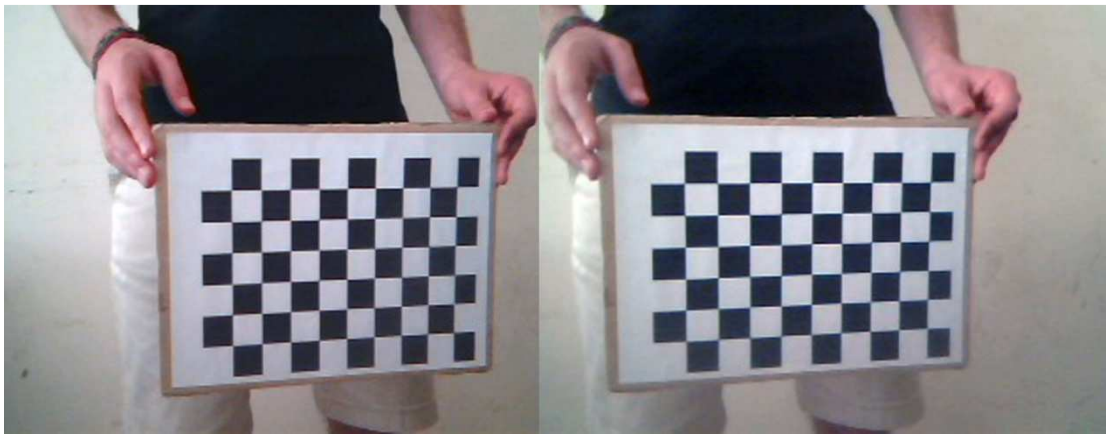


Figure n°48: Captures simultanées des deux images gauche et droite.

2.1.2- Deuxième sous programme:

Ce programme sert à calibrer et rectifier les deux caméras en utilisant les paires de captures d'image prises par le premier sous programme. Le résultat de ce programme est un fichier XML contenant les différentes matrices citées dans le chapitre précédant. Un extrait du fichier XML est illustré ci-dessous:


```
<?xml version="1.0"?>
<opencv_storage>
- <l_cameraMatrix type_id="opencv-matrix">
    <rows>3</rows>
    <cols>3</cols>
    <dt>d</dt>
    <data> 3.6029637830440447e+02 0. 1.6560713615257629e+02 0. 3.5396055075333567e+02 1.1946918417934494e+02 0. 0. 1.</data>
  </l_cameraMatrix>
- <r_cameraMatrix type_id="opencv-matrix">
    <rows>3</rows>
    <cols>3</cols>
    <dt>d</dt>
    <data> 3.6765556980097034e+02 0. 1.5539128845925831e+02 0. 3.6096617121862448e+02 1.1503410090125652e+02 0. 0. 1.</data>
  </r_cameraMatrix>
- <l_distCoeffs type_id="opencv-matrix">
    <rows>1</rows>
    <cols>5</cols>
    <dt>d</dt>
    <data> 1.1615652684460717e-01 -2.2883776852967394e-01 8.4765382301580435e-04 -1.5328178899545358e-03 -8.5810617125927136e-01</data>
  </l_distCoeffs>
- <r_distCoeffs type_id="opencv-matrix">
    <rows>1</rows>
    <cols>5</cols>
    <dt>d</dt>
    <data> 1.1893986339326329e-01 -3.2990245809621288e-01 -9.2852922320470319e-04 7.9126637416503315e-04 -3.4641439081959463e-01</data>
  </r_distCoeffs>
- <R type_id="opencv-matrix">
    <rows>3</rows>
    <cols>3</cols>
    <dt>d</dt>
    <data> 9.9975514875151694e-01 -1.0492247432845696e-02 -1.9482178744695711e-02 1.0471849643300939e-02 9.9994450886172004e-01
      -1.1487220471568507e-03 1.9493150332371326e-02 9.4442633459108113e-04 9.9980954443784853e-01</data>
  </R>
- <T type_id="opencv-matrix">
    <rows>3</rows>
    <cols>1</cols>
    <dt>d</dt>
    <data> -5.1270843483207045e+01 2.1339274513134066e+00 2.6026643826888405e+00</data>
```


Figure n°49: contenu du fichier XML.

Le programme lit d'abord la taille du damier et les paires d'images gauches et droites, ensuite il détecte les coins du damier 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. Les coins du damier trouvé sont juste approximatifs, pour cela on utilise les sous pixels des coins détectés pour trouver la localisation exacte des coins. Les fonctions utilisées pour ce fait sont:

- *cvFindChessboardCorners()*: cette fonction sert à détecter les coins du damier.
- *cvFindCornerSubPix()*: cette fonction détecte les sous-pixels des coins détectés.

Le résultat de ce processus est illustré ci-dessous:

Figure n°50: Détection de coins du damier.

En tenant compte de la liste des points trouvés, le programme appelle la fonction *cvStereoCalibrate ()* pour calibrer les deux caméras, le résultat de cette dernière action nous donne la matrice intrinsèque de la caméra A, la matrice extrinsèque de la caméra E, le vecteur de distorsion D, ainsi que la matrice essentielle F. Ces résultats sont sauvegardés dans un fichier nommé « *calib.xml* ».

Afin de rectifier les images, le programme utilise plusieurs fonctions. Il commence par annuler les distorsions qui déforment les points d'origine en utilisant la fonction *cvUndistortPoints ()*. Pour évaluer la précision du calibrage, le programme doit vérifier si un point d'image de gauche est sur la même ligne épipolaire du même point de l'image droite, pour se faire, le programme utilise 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 pour les deux images individuellement en utilisant la fonction d'OpenCV nommée *cvStereoRectify()*. La fonction *cvRemap()* est utilisée pour ajuster les pixels de l'image originale afin de produire l'image rectifiée. Le résultat est montré par la figure ci-dessous, où nous pouvons voir que les deux images originales rectifiées.

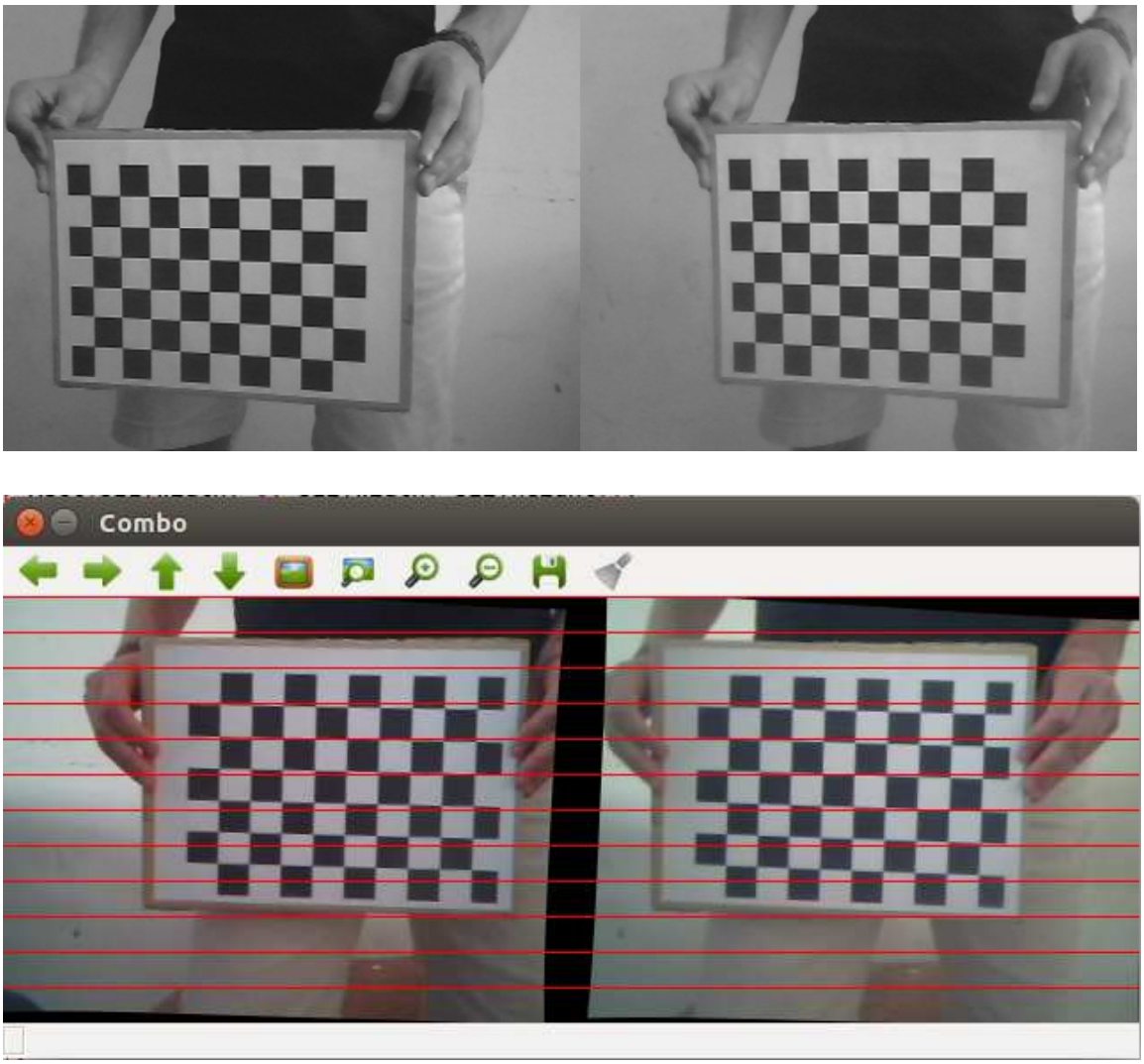


Figure n°51: Images avant et après rectification.

2.1.3- Troisième sous programme:

Apres avoir fait la calibration et rectification, on utilise l'algorithme SGBM disponible dans la bibliothèque openCV qui utilise les images rectifiées calculées auparavant. On commence par la déclaration de l'objet SGBM comme suit: *StereoSGBM sgbm;*

Afin de faciliter le réglage et de ne pas changer le code à chaque fois, on a crée une fenêtre où sont regroupés tout les paramètres à l'aide de la fonction *cvCreateTrackbar().*

La carte de disparité et calculée 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.

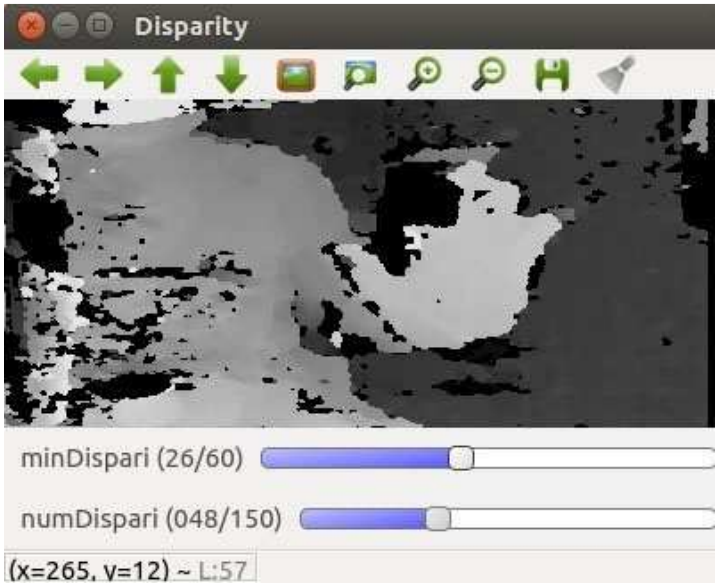


Figure n°52: Carte de disparité.

2.1.4- Quatrième sous programme:

Cette partie du programme sert à détecter l'objet qui est la main dans notre cas. Pour commencer, le programme prend des captures d'images en mode BGR (blue green red) et les convertir en mode HSV (hue saturation value) à l'aide de la fonction *cvtColor()*, après ca, on procède à une segmentation en jouant sur les paramètre regroupés dans une fenêtre crée pour ce fait.

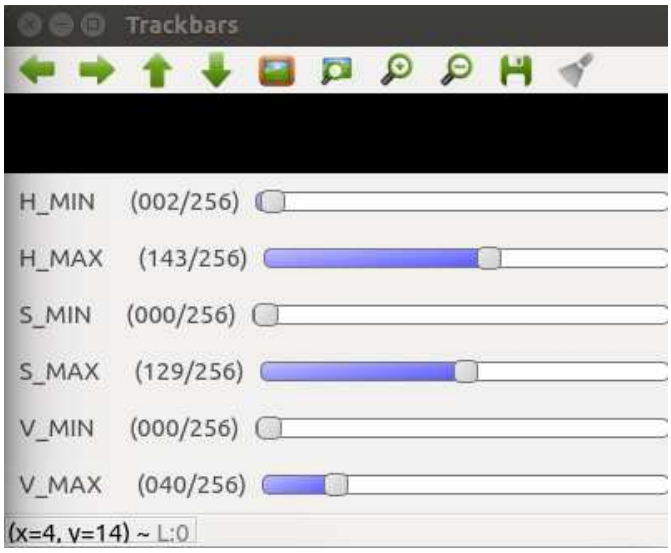


Figure n°53: Trackbare HSV

2.1.5- Triangulation:

Apres avoir détecter l'objet dans les deux caméras, les coordonnées (x, y, z) de sont centre de gravité sont calculées par la triangulation ensuit transmises en temps réel à la carte de commande Arduino.

2.2- Interfaçage logiciel/Arduino:

Cette partie a pour but de lier la partie vision et la partie commande, cette liaison se figure par l'envoi de la position calculée de l'objet vers la carte de commande Arduino. Pour ce faire on a utilisé la bibliothèque *libSerial* qui nous permet d'accéder aux différents paramètres des ports séries telle que la vitesse de transmission, la taille des caractères et le contrôle du flux sortant et entrant. Reste à mentionner que la care de commande Arduino a un logiciel d'interfaçage, donc c'est ce dernier qui reçoit la position de l'objet afin de l'envoyer à la carte.

Pour l'implémentation de la bibliothèque *libSerial*, nous l'avons importé et nous avons définit le port relié à la carte Arduino comme suit:

```
#include SerialStream.h

#include iostream

#define PORT "/dev/ttyUSB3"
```

SerialStream ardu;

using namespace std;

using namespace LibSerial;

Par la suite on synchronise le logiciel et la carte Arduino sur la même vitesse de transmission afin qu'il n'ait pas d'ambigüité.

ardu.SetBaudRate(SerialStreamBuf::BAUD_115200);

2.3- Partie commande:

Cette partie comme son nom l'indique, sert à commander le bras manipulateur en recevant les coordonnées cartésiennes x, y, z de la main, et les convertir en coordonnées angulaires θ_i. L'organe essentiel de cette partie est la carte de commande Arduino muni de son logiciel d'interfaçage. Pour passer des coordonnées cartésiennes en coordonnées articulaires, on a implémenté le modèle géométrique inverse calculée précédemment dans le chapitre trois.

Pour mettre en communication l'ordinateur et la carte de commande, on a utilisé la fonction Serial.begin(115200) de la bibliothèque serial d'Arduino. Cette fonction démarre la liaison en la réglant à une vitesse de 115200 bits par seconde ou bauds.

Pour recevoir les données envoyées par l'ordinateur, on a utilisé la fonction Serial1.read(). Cette dernière sert à accéder caractère par caractère aux données reçues, ce type de fonctionnement est appelé FIFO (First In First Out, premier arrivé, premier traité). Si jamais rien n'est à lire, la fonction renverra -1 pour le signaler.

Pour utiliser les servomoteurs avec Arduino, il va nous falloir générer un signal PWM. Pour ce faire, Arduino est équipé d'une classe nommée servo dédiée à ce fait. Pour travailler avec cette classe il faut d'abord inclure la bibliothèque #include <Servo.h> au début du code, ensuite on pourra déclarer les servomoteurs on leurs attribuant un nom et un pin sur lequel ils sont branché en utilisant la syntaxe suivante:

Servo monServo;

monServo.attach(N° de pin);

Pour actionner les servomoteurs on utilise la fonction servo.write(position), cette fonction envoi au servomoteur l'angle sur lequel il doit se mettre.

Reste à mentionner que les servomoteurs ne renvoient pas la position atteinte à la carte de commande, d'une manière plus explicite, si on fait bouger l'axe du servomoteur manuellement, on ne peut pas lire l'angle de sortie.

Le montage général de la carte de commande avec les servomoteurs du bras manipulateur est illustré par la figure ci-dessous:

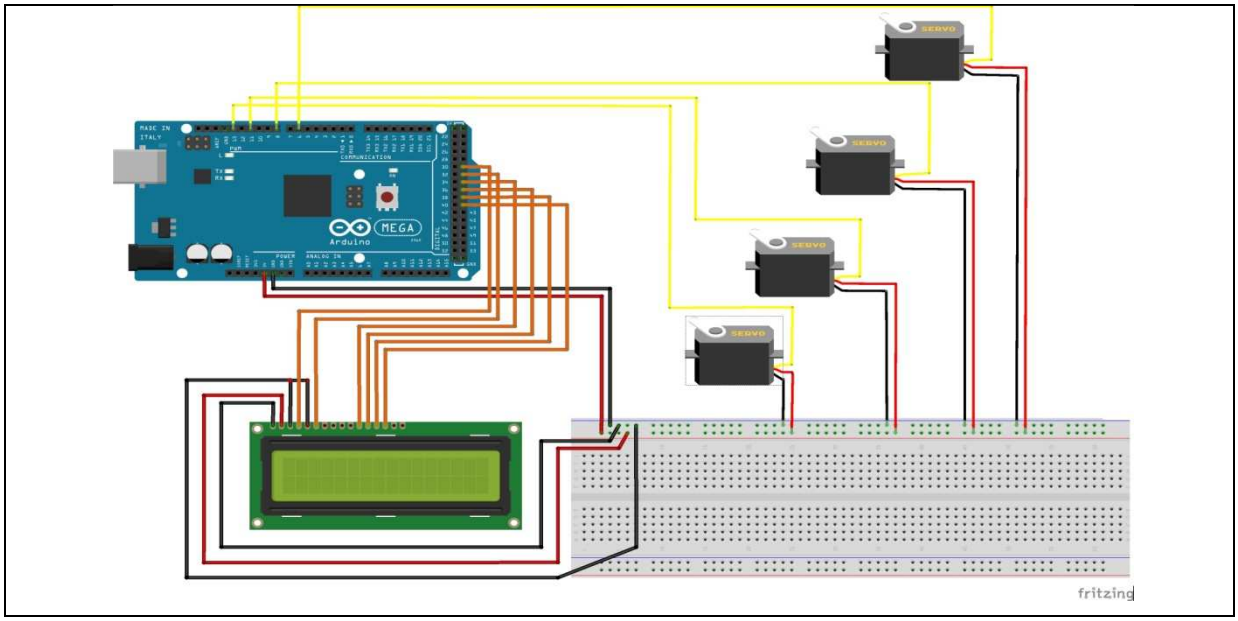
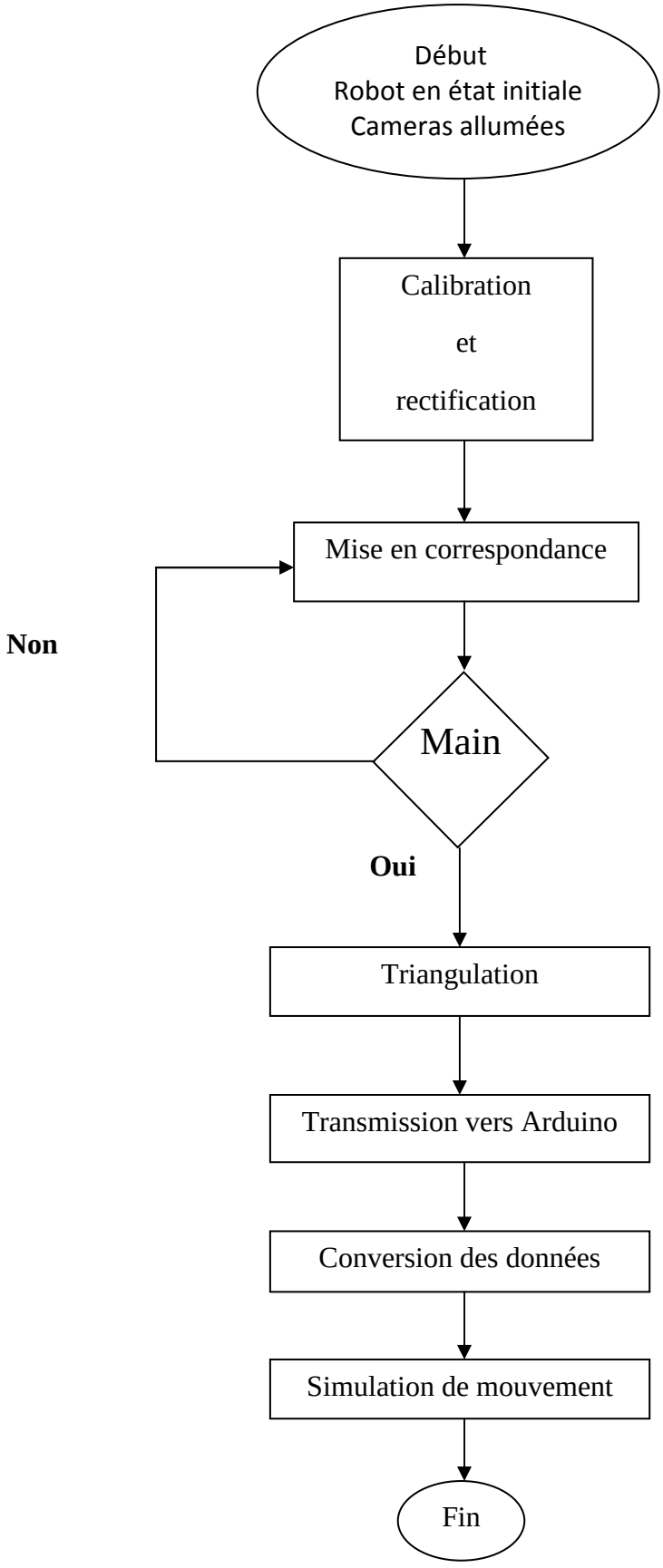


Figure n°54: schéma de montage.

Sur la figure ci-dessus, les servomoteurs représentent les actionneurs du bras manipulateur. On a ajouté un afficheur LCD à la carte de commande pour afficher les coordonnées x, y, z envoyées par l'ordinateur. La fonction utilisée pour le commander est LiquidCrystal lcd(), mais avant de l'utiliser il faut d'abord implémenter la bibliothèque #include "LiquidCrystal.h" au début du code.

3- Organigramme:

L'organigramme suivant résume tout notre projet, en montrant les étapes effectuées durant le travail en passant de la stéréovision jusqu'à la simulation de mouvement par le robot.


Début
Robot en état initiale
Cameras allumées
Calibration
et
rectification
Mise en correspondance
Non
Main
Oui
Triangulation
Transmission vers Arduino
Conversion des données
Simulation de mouvement
Fin

Conclusion générale:

Ce mémoire est le résultat d'un travail qui porte sur la réalisation d'une commande gestuelle assistée par la vision d'un bras manipulateur à structure ouverte à 4 DDL.

La réalisation de ce projet nous a était une expérience fructueuse et très bénéfique, du moment que ca englobe plusieurs domaines tel que l'informatique et la robotique.

Dans le premier chapitre, nous avons présenté des généralités sur la robotique pour rendre plus facile la lecture du deuxième chapitre, qui porte sur une étude détaillée du robot utilisé dans ce projet. Les caractéristique des différents actionneurs et capteurs utilisés dans ce projet ont fait l'objet du troisième chapitre. Dans le quatrième chapitre on a présentée la modélisation robotique où on a élaboré les modèles géométriques directe et inverse du robot en utilisant les conventions de Denavit - Hartenberg. Pour réaliser la commande du robot, on a expliqué les points majeurs de la vision par ordinateur dans le chapitre Cinque avant de l'intégrer dans notre système dans le chapitre six.

Durant ce projet on a eu affaire à des contraintes logiciels pendant la pratique, en effet, la non stabilité de la stéréovision à influencé sur la précision du robot et la souplesse de ses mouvements.

Dans le prolongement de ce projet plusieurs études sont envisageables:

- Remplacer l’approche de la stéréovision passive par une stéréovision active;

- Utilisation de la reconnaissance de doigts pour la commande de l'organe terminale;

- Utiliser des servomoteurs a retour de position;

- Intégrer une commande par neuro-flou;

Annexes n°1 :

1- Arduino:

Arduino est un circuit imprimé sur lequel se trouve un microcontrolleur généralement un Atmel AVR, qui peut être programmé pour analyser et produire de signaux électriques, de manière à effectuer des tâches très diverses comme le contrôle de matériels domestique (éclairage, chauffage…) et le pilotage de robot.

C'est une plateforme basée sur une interface entrée/sortie simple, Chaque module Arduino possède au moins un régulateur linéaire 5 V et un oscillateur à quartz 16 MHz. Le microcontrôleur est préprogrammé avec un bootloader de façon à ce qu'un programmateur externe ne soit pas nécessaire **[5]**.

1.1.Vue d'ensemble:

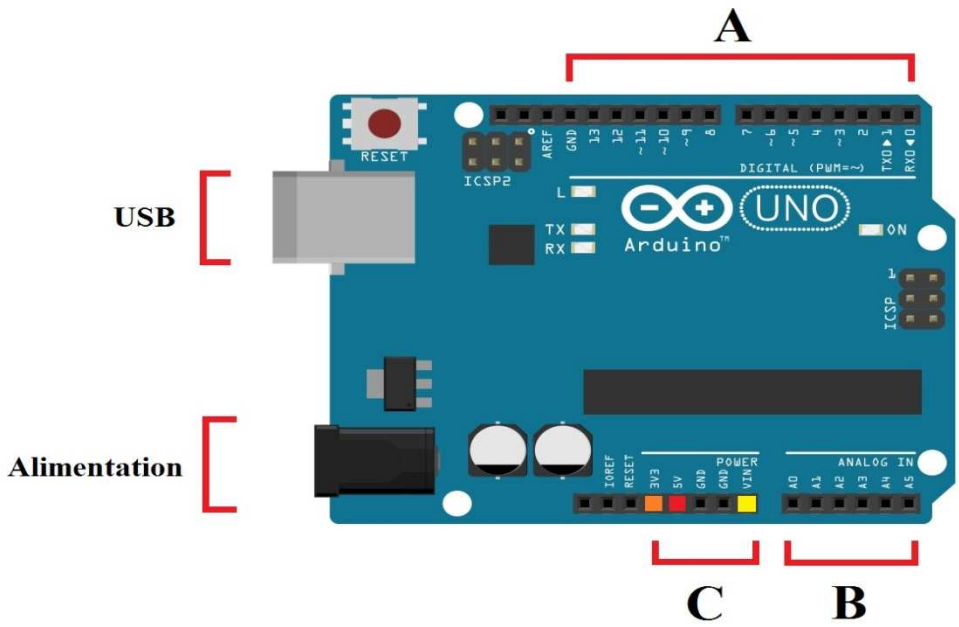


Figure n°56: vue d'ensemble d'une carte Arduino.

Les différentes versions des Arduino fonctionnent sous le même principe général:

-A: ce sont les pattes dites digitales (0,1) ou "tout ou rien", elles offrent en sortie 5V et acceptent en entrée 5V.

-fonction digitalWrite() et digitalRead().

-B: ce sont les pattes dites analogiques, valeur entre 0V et 5V

-fonction analogWrite() et analogRead().

-C: les différentes pattes d'alimentation:

-Rouge: sortie 5v (+);

-Orange: sortie 3,3V (+);

-Noire: les masses (-);

-Jaune: entrée reliée à l'alimentation (7V-12V).

-USB: sert à brancher le module avec le Pc.

-Alimentation: sert à brancher le module avec une alimentation externe.

1.2. Programmation:

Le logiciel de programmation des modules Arduino est une application Java, servant d'éditeur de code et de compilateur, et qui peut transférer le programme au travers de la liaison série (RS-232, bluetooth ou USB selon le module). Il est également possible de se passer de l'interface Arduino, et de compiler et uploader les programmes via l'interface en ligne de commande **[5]**.

Avec Arduino, nous devons utiliser un code minimal lorsqu'on crée un programme, Ce code permet de diviser le programme en deux parties.

```
1  //fonction d'initialisation de la carte
2  void setup()
3  {
4      //contenu de l'initialisation
5  }
6
7  //fonction principale, elle se répète (s'exécute) à l'infini
8  void loop()
9  {
10     //contenu de votre programme
11 }
```

Figure n°57: code minimale.

Dans le code se trouvent deux fonctions, les fonctions sont en fait des portions de code.

- **Setup():**

On appelle cette fonction "fonction d'initialisation", elle est appelée une seule fois lorsque le programme commence. Elle a pour fonction d'initialiser les variables, indiquer les modes des broches, déclarer les bibliothèques.

- **Loop():**

C'est dans cette fonction qu'on écrit le contenu du programme, cette fonction est appelé en permanence, elle se répète en boucle infinie.

- **Bibliothèque sevo:**

Cette bibliothèque permet à une carte Arduino de contrôler des servomoteurs RC, elle supporte jusqu'à 48 servomoteurs. Ses principales fonctions sont:

• attach()

• write()

• writeMicroseconds()

• read()

• attached()

• detach()

- **Bibliothèque LiquidCrystal:**

Cette bibliothèque permet à une carte Arduino de contrôler un écran LCD (liquid crystal display). Ses principales fonctions sont:

• lcd.begin()

• lcd.write()

Annexes n°2 :

2- Arduino Mega 2560:

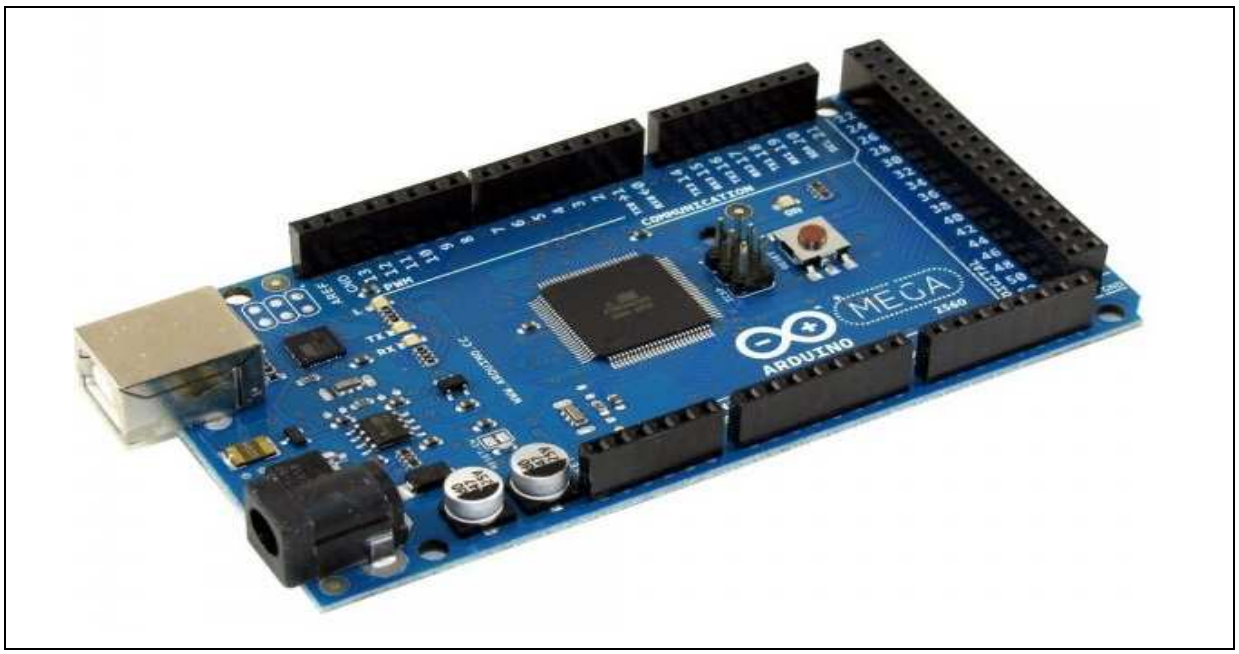


Figure n°58: carte de commande Aduino Mega 2560.

La carte Arduino Mega 2560 est une carte à microcontrôleur basée sur un ATmega2560, Cette carte dispose de :

- 54 broches numériques d'entrées/sorties (dont 14 peuvent être utilisées en sorties PWM (largeur d'impulsion modulée));
- 16 entrées analogiques (qui peuvent également être utilisées en broches entrées/sorties numériques);
- 4 UART (port série matériel);
- un quartz 16Mhz;
- une connexion USB;
- un connecteur d'alimentation jack;
- un connecteur ICSP (programmation "in-circuit");
- un bouton de réinitialisation (reset).

Annexe n°3 :

3- Communication:

La carte Arduino Mega2560 dispose de toute une série de facilités pour communiquer avec un ordinateur, une autre carte Arduino, ou avec d'autres microcontrôleurs.

L'ATmega2560 dispose de quatre UART (Universal Asynchronous Receiver Transmitter ou émetteur-récepteur asynchrone universel) pour communication série de niveau TTL (5V) et qui est disponible sur les broches 0 (RX) et 1 (TX). Un circuit intégré ATmega8U2 sur la carte assure la connexion entre cette communication série de l'un des ports série de l'ATmega 2560 vers le port USB de l'ordinateur qui apparaît comme un port COM virtuel pour les logiciels de l'ordinateur. Le code utilisé pour programmer l'ATmega8U2 utilise le driver standard USB COM, et aucun autre driver externe n'est nécessaire.

Le logiciel Arduino inclut une fenêtre terminal série sur l'ordinateur qui permet d'envoyer des textes simples vers la carte Arduino. Les LED RX et TX sur la carte, clignote lorsque les données sont transmises via le circuit intégré ATmega8U2 utilisé en convertisseur USB-vers-série.

Annexe n°4 :

4- OpenCV:

OpenCV(Open Source Computer Vision) est une bibliothèque proposant un ensemble de plus de 2500 algorithmes de vision par ordinateur, accessibles au travers d'API pour les langages C, C++, et Python. Elle est distribuée sous une licence BSD (libre) pour les plate-formes Windows, GNU/Linux, Android et MacOS.

Initialement écrite en C il y a 10 ans par des chercheurs de la société Intel, OpenCV est la bibliothèque de référence pour la vision par ordinateur, aussi bien dans le monde de la recherche que celui de l'industrie.

Afin de mieux présenter son étendue et ce qu'elle permet de faire, on va voir les principaux modules accessibles au travers de son API C:

-core: les fonctionnalités de base.

Cette bibliothèque permet de manipuler les structures de base, réaliser des opérations sur des matrices, dessiner sur des images, sauvegarder et charger des données dans des fichiers XML…

-imgproc: traitement d'image.

Les fonctions et structures de ce module ont servent aux transformations d'images, au filtrage, à la détection de contours, de points d'intérêt…

-features2d: descripteur.

Ce module concerne principalement l'extraction de descripteurs selon deux approches courantes (SURF et StarDetector).

-objdetect: détection d'objets.

Cette bibliothèque permet de faire de la reconnaissance d'objets dans une image au moyen de l'algorithme Adaboost (Viola & Jones, 2001).

-**video:** traitement de flux vidéo.

Ces fonctions servent à segmenter et suivre les objets en mouvement dans une vidéo.

-**highgui:** entrées-sorties et interface utilisateur.

OpenCV intègre sa propre bibliothèque haut-niveau pour ouvrir, enregistrer et afficher des images et des flux vidéo.

-**calib3d:** calibration, estimation de pose et stéréovision.

Ce module contient des fonctions permettant de reconstruire une scène en 3D à partir d'images acquises avec plusieurs caméras simultanément **[15]**.

Références bibliographiques:

[1] http://docslide.fr/documents/introduction-a-la-robotique-55b344a0ae1fc.html.

[2] https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:fr. "2012 ISO".

[3] http://icube-avr.unistra.fr/fr/img_auth.php/a/a4/Cours_rob_intro.pdf

[4] http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.MaterielMega2560. July 01, 2013

[5]http://www.centralmedia.fr/download/Premiers_pas_en_informatique_embarquee.pdf. 01 juin 2014.

[6] http://store.easyrobotics.fr/servomoteur/12-servomoteur-futaba-s3003.html

[7]http://www.rcteam.fr/mega-servos/231-futaba-mega-servo-s3306-24kg-016s-nylon-4513886012327.html

[8] http://www.mr-rcworld.co.uk/index.php?productID=1027

[9] http://fsi.univ-tlemcen.dz/cours/Cours_modelisationdes_robots.pdf. 2012.

[10] https://www.ensta-bretagne.fr/jaulin/mastersds_cours_robot_boimond.pdf

[11] http://www.memoireonline.com/07/08/1351/m_programmation-robots-industriels-application-robot-manipulateur-algerie7.html. 2007.

[12] http://web.iitd.ac.in/~saha/ethiopia/13lec.pdf. July 15, 2009.

[13] **Gary Bradski** and **Adrian Kaehler** "Learning OpenCV" O'REILLY. 2008.

[14] http://www.Jofcis.com. 2013.

[15] https://openclassrooms.com/courses/introduction-a-la-vision-par-ordinateur/avant-de-commencer-8. 5 décembre 2013.