CEG4553
Rapport de Projet

Mathieu Mallet
1976865
Dominic Bergeron
2064116

Rapport présenté à Dr. Payeur
Professeur de robotique

Université d'Ottawa
Le 1er décembre 2003

# Table des Matières

# Table des Illustrations

# Modèle du Robot

## *Modèles*

Le robot que nous avons utilisé est le Robix-6, monté dans la configuration du Chimiste.

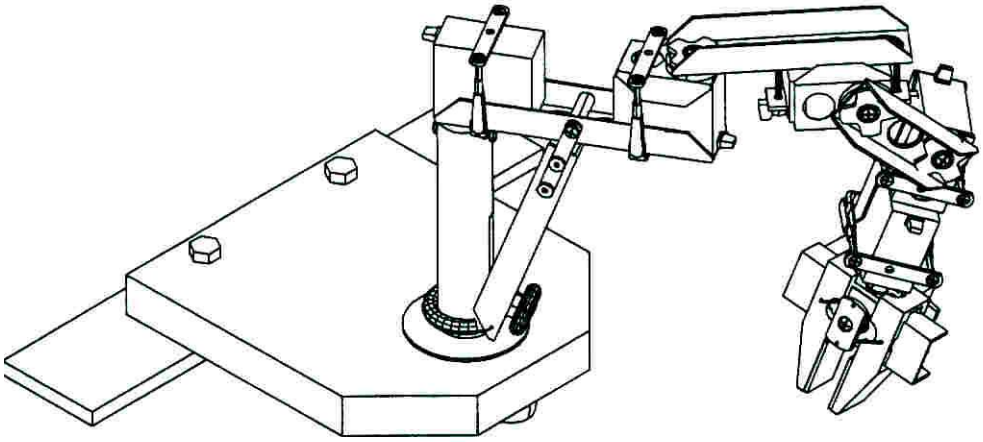

*Illustration 1: Robix-6 dans la configuration du Chimiste*

De ce robot, nous avons choisi de mesurer les paramètres suivants:

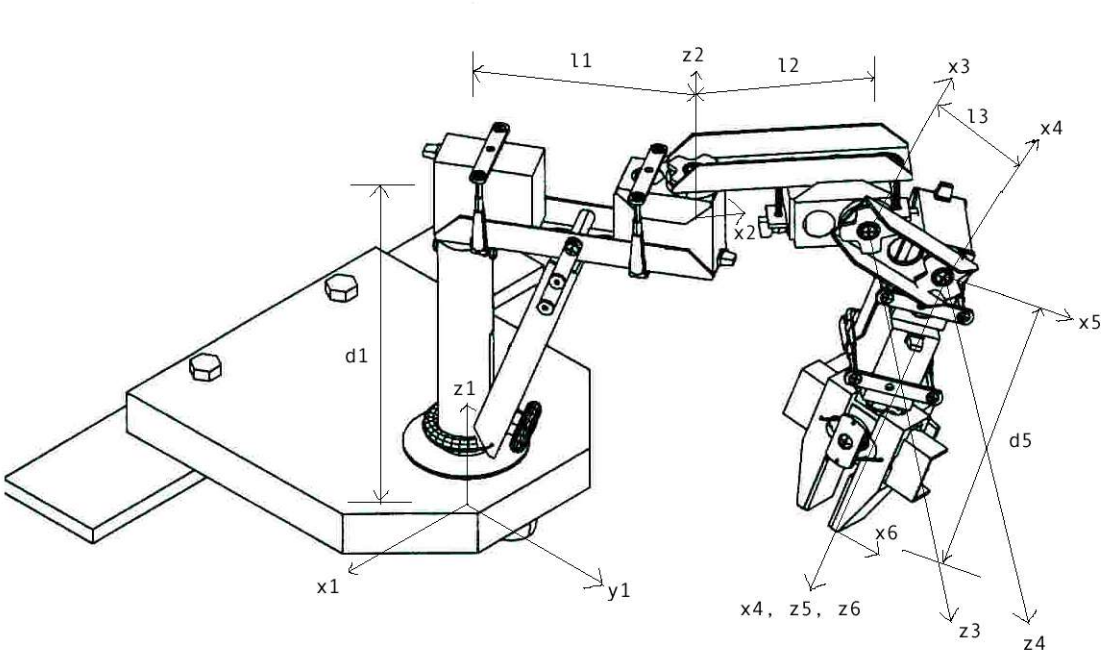


*Illustration 2: Paramètres Denavit-Hartenberg*

Nous avons choisi le référentiel ($x_1$, $y_1$, $z_1$) afin qu'un z de 0 représente une position exactement sur la ta surface de l'espace de travail. Nous avons aussi choisi le référentiel ($x_6$, $y_6$, $z_6$) afin que le bout de la pince corresponde au milieu du référentiel. Ceci a permis, plus tard, de simplement utiliser un z de 0 pour faire la cinématique inverse d'un objet placé sur la surface de travail.

Initialement, nous utilisions un modèle plus complexe, qui tenait compte des distances horizontales minimes entre les différents actuateurs (par exemple, entre le membre $l_2$ et le membre $l_3$). Après avoir rencontré quelque problèmes lors du calcul de la cinématique inverse, nous avons décidé d'utiliser ce modèle simplifié.

## *Paramètres de Denavit-Hartenberg*

Au laboratoire, nous avons mesuré les valeurs suivantes:

| ***I*** | ***$l_i$ (cm)*** | ***$d_i$ (cm)*** | ***$\alpha_i$ (radians)*** | ***$\theta_i$ (radians)*** |
|---|---|---|---|---|
| **1** | 9.8 | 15.7 | 0 | $\theta_1 + \pi/2$ |
| **2** | 9.6 | 0 | $\pi/2$ | $\theta_2$ |
| **3** | 6 | 0 | 0 | $\theta_3$ |
| **4** | 0 | 0 | $\pi/2$ | $\theta_4 + \pi/2$ |
| **5** | 0 | 11 | 0 | $\theta_5$ |

De ces valeurs, nous avons calculé les matrices A suivantes:

$$A_1 = \begin{bmatrix} -\sin(\theta_1) & -\cos(\theta_1) & 0 & -49/5*\sin(\theta_1) \\ \cos(\theta_1) & -\sin(\theta_1) & 0 & 49/5*\cos(\theta_1) \\ 0 & 0 & 1 & 157/10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) & 48/5*\cos(\theta_2) \\ \sin(\theta_2) & 0 & -\cos(\theta_2) & 48/5*\sin(\theta_2) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 6*\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 6*\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} -\sin(\theta_4) & 0 & \cos(\theta_4) & 0 \\ \cos(\theta_4) & 0 & \sin(\theta_4) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_5 = \begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5) & 0 & 0 \\ \sin(\theta_5) & \cos(\theta_5) & 0 & 0 \\ 0 & 0 & 1 & 11 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## *Cinématique Directe*

La cinématique directe du robot est calculée avec $Q_{BE} = A_1 A_2 A_3 A_4 A_5$ . Nous avons effectué ce calcul dans MATLAB en calculant les matrices A avec des variables symboliques et en multipliant toutes les matrices ensemble. La matrice résultante était:

$$Q_{PB} = \begin{matrix} -0.25\cos(\zeta) - 0.25\cos(\varepsilon) + 0.25\cos(\lambda) + 0.5\sin(\delta) - 0.5\sin(\gamma), \\ 0.25\sin(\varepsilon) - 0.25\sin(\zeta) - 0.25\sin(\eta) + 0.25\sin(\lambda) + 0.5\cos(\gamma) + 0.5\cos(\delta), \\ -0.5\sin(\alpha) - 0.5\sin(\rho), \\ -5.5\sin(\alpha) - 5.5\sin(\rho) - 3\sin(\theta_1+\theta_2+\theta_3) - 3\sin(\theta_1+\theta_2-\theta_3) - 9.6\sin(\theta_1+\theta_2) - 9.8\sin(\theta_1) \end{matrix}$$

$$\begin{matrix} 0.25\sin(\eta) + 0.25\sin(\lambda) - 0.25\sin(\varepsilon) - 0.25\sin(\zeta) + 0.5\cos(\gamma) - 0.5\cos(\delta), \\ -0.25\cos(\lambda) + 0.25\cos(\eta) + 0.25\cos(\zeta) - 0.25\cos(\varepsilon) + 0.5\sin(\delta) + 0.5\sin(\gamma), \\ 0.5\cos(\rho) + 0.5\cos(\alpha), \\ 5.5\cos(\rho) + 0.5\cos(\alpha) + 3\cos(\theta_1+\theta_2-\theta_3) + 3\cos(\theta_1+\theta_2+\theta_3) + 9.6\cos(\theta_1+\theta_2) + 9.8\cos(\theta_1) \end{matrix}$$

$$\begin{matrix} 0.5\cos(\theta_3+\theta_4-\theta_5) + 0.5\cos(\theta_3+\theta_4+\theta_5), \\ -0.5\sin(\theta_3+\theta_4+\theta_5) + 0.5\sin(\theta_3+\theta_4+\theta_5), \\ \sin(\theta_3+\theta_4), \\ 15.7 + 11\sin(\theta_3+\theta_4) + 6\sin(\theta_3) \end{matrix}$$

$$0,\ 0,\ 0,\ 1$$

*where :*

$\alpha = \theta_1 + \theta_2 + \theta_3 + \theta_4$
$\rho = \theta_1 + \theta_2 - \theta_3 - \theta_4$
$\gamma = \theta_1 + \theta_2 - \theta_5$
$\delta = \theta_1 + \theta_2 + \theta_5$
$\varepsilon = \theta_1 + \theta_2 + \theta_3 + \theta_4 + \theta_5$
$\zeta = \theta_1 + \theta_2 + \theta_3 + \theta_4 - \theta_5$
$\eta = \theta_1 + \theta_2 - \theta_3 - \theta_4 + \theta_5$
$\lambda = \theta_1 + \theta_2 - \theta_3 - \theta_4 - \theta_5$

En remplaçant les angles dans cette matrice et en la plaçant égale à la matrice QTRTL, on peut facilement trouver les équations pour x, y, z, θ, φ et ψ.

$$Q_{TRTL} = Q_{PB}|_{\theta_1,\theta_2,\theta_3,\theta_4,\theta_5}$$

$$\begin{bmatrix} cos\theta\cos\varphi & \cos\theta\sin\varphi\sin\ \psi - \sin\theta\cos\ \psi & \cos\theta\sin\varphi\cos\ \psi + \sin\theta\sin\ \psi & x_T \\ \sin\theta\cos\ \psi & \sin\theta\sin\varphi\sin\ \psi + \cos\theta\cos\ \psi & \sin\theta\sin\varphi\cos\ \psi - \cos\theta\sin\ \psi & y_T \\ -\sin\varphi & \cos\varphi\sin\ \psi & \cos\varphi\cos\ \psi & z_T \\ 0 & 0 & 0 & 1 \end{bmatrix} \ldots$$

$$= \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{bmatrix}$$

$$x_T = D$$
$$y_T = H$$
$$z_T = L$$

$$-sin\varphi = I$$
$$\varphi = asin\ (-I\ )$$

$$\frac{E}{A} = \frac{\sin\theta\cos\varphi}{\cos\theta\cos\varphi} = \frac{\sin\theta}{\cos\theta} = \tan\theta$$
$$\theta = atan2\left(\frac{E}{A}\right)$$

$$\frac{J}{K} = \frac{\cos\varphi\sin\ \psi}{\cos\varphi\cos\ \psi} = \frac{\sin\ \psi}{\cos\ \psi} = \tan\ \psi$$
$$\psi\ = atan2\left(\frac{J}{K}\right)$$

Ces équations ne sont toutefois pas utilisées dans notre projet.

# Cinématique Inverse

## *Théorie*

Afin d'effectuer les calculs de cinématique inverse, nous avons décider d'utiliser la puissance de calcul symbolique de MATLAB. Nous avons pris cette décision car nous voulions obtenir une solution ne dépendant pas de paramètres Denavit-Hartenberg spécifiques.

Les deux outils de calcul symbolique que nous avons utilisé dans MATLAB sont `syms`, permettant de définir des variables symboliques, et `solve`, permettant de résoudre des équations symboliques. Tous les calculs symboliques de MATLAB font appel au noyau Maple de MATLAB. Pour résoudre des équations symboliques, Maple essaie premièrement de trouver des expressions simplifiées pour chacune des variables symboliques dans les équations initiales. Si il n'y parvient pas, alors Maple essaie de résoudre les équation numériquement.

Évidemment, un résultat en équation est désiré: ceci permet de conserver les équations trouvées et de les appliquer en remplaçant simplement les variables. Un résultat numérique, bien que juste, est très lent à recalculer lorsque les variables d'entrés changent.

.

## *Implémentation*

Comme mentionné au paravent, nous avons obtenu les matrices A symboliquement dans MATLAB. Nous les avons ensuite multipliées ensemble afin d'obtenir la matrice de transformation de la pince par rapport à la base du robot, $Q_{PB}$.

Au début, nous espérions pouvoir utiliser MATLAB pour obtenir les équations pour $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$ et $\theta_5$ en utilisant MATLAB à partir de la matrice de transformation $Q_{PB}$. Toutefois, puisqu'il n'existe pas de solution unique pour ces équations, MATLAB ne pouvait pas nous donner celles-ci. Nous avons essayé de résoudre directement les équations de $Q_{PB} = Q_{TRTL}$ et avons obtenu des résultats acceptables. Toutefois, MATLAB ne pouvait pas résoudre les équations lorsque l'angle de la pince n'était pas un multiple de 90 degrés... De plus, la résolution des équations prenait beaucoup trop de temps – entre 30 seconds lorsqu'une solution existait jusqu'à 5 minutes lorsqu'aucune solution ne pouvait être trouvée.

Afin d'accélérer le processus, nous avons utilisé un ensemble d'équation intermédiaire. Nous avons utilisé MATLAB pour résoudre QPB = QTRTL, mais en utilisant des variables symboliques dans QTRTL représentant x, y, z et θ (rotation de la pince). Ceci nous donnait 8 nouvelles équations: X, Y, Z, $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$ et $\theta_5$. Ces 8 équations sont nos équations intermédiaires de cinématique inverse: individuellement, elles sont inutiles car les variables utilisées dans ces équations, que l'ont appelle variables intermédiaires, n'ont aucune connections avec la réalité. Plus tard, lorsque l'on veut les valeurs de $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$ et $\theta_5$, on remplace X, Y et Z par les vraies positions et on demande à MATLAB de calculer les valeurs de $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$ et $\theta_5$. Les variables intermédiaires disparaissent alors et on obtient les angles désirés. Le programme `FindIKEqs.m` est utilisé pour calculer les équations intermédiaire alors que le programme `FindSolutions.m` est utilisé pour résoudre les équations pour une position (x, y, z) spécifique.

Toutefois, si la complexité des équation était trop élevée, MATLAB ne pouvait pas trouver de solution. Afin de solutionner nos équations, nous avons donc dû faire quelque simplifications. Premièrement, la position de la pince fut fixée à $\theta = ?, \varphi = 0, \psi = 180$. Ceci ne cause pas de problèmes car la configuration du robot ne permet pas vraiment le saisissement d'objets dans d'autre configurations que cela (à moins que l'objet soit placé plus haut que la surface de l'espace de travail). La deuxième simplification que nous avons effectué fut de fixer la rotation de la pince à 0 degrés. Ceci est dû aux limitations de MATLAB: lorsque l'angle n'était pas un multiple de 90 degrés, les équations ne se résolvaient pas. Cette contrainte ne causait pas de problèmes non plus, car la configuration du robot nous permet de calculer la rotation de la pince à partir des angles $\theta_1$, $\theta_2$ et de la rotation de l'objet à saisir.

## *Équations*

Afin d'obtenir les équations intermédiaires, nous avons placé:

$$Q_{TRTL} = Q_{PB}$$

$$\begin{bmatrix} 1 & 0 & 0 & X \\ 0 & -1 & 0 & Y \\ 0 & 0 & -1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{bmatrix}$$

Nous avons alors obtenu les équations suivantes:

$$\begin{array}{lll} 1 = A & 0 = E & 0 = I \\ 0 = B & -1 = F & 0 = J \\ 0 = C & 0 = G & -1 = K \\ X = D & Y = H & Z = L \end{array}$$

Évidemment, il est inutile de résoudre les dernières lignes des matrices puisque l'on aurait simplement 0 = 0 et 1 = 1. Les équations obtenues lorsque l'on résout ces équations en utilisant `solve` sont:

$$X = 3\cos(c_1 + c_2 - c_4) - 3\cos(c_1 + c_2 + c_4) - 9.6\sin(c_1 + c_2) - 9.8\sin(c_1)$$
$$Y = -3\sin(c_1 + c_2 + c_4) + 3\sin(c_1 + c_2 - c_4) + 9.6\sin(c_1 + c_2) - 9.8\cos(c_1)$$
$$Z = 4.7 - 6\cos(c_4)$$
$$\theta_1 = c_1$$
$$\theta_2 = c_2$$
$$\theta_3 = -0.5\pi - c_4$$
$$\theta_4 = c_4$$
$$\theta_5 = \arctan\left(\frac{\sqrt{\left(1 - \sin(c_1 + c_2)^2\right)}}{-\sin(c_1 + c_2)}\right)$$

Lorsque l'on remplace X, Y et Z dans ces équations et que l'on utilise `solve` pour trouver les valeurs de $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$ et $\theta_5$, on obtient plusieurs solutions – dans notre cas, on obtient 8 solutions possible. Si aucune solution n'est possible, alors `solve` retourne une matrice vide. La ou les solutions correctes sont alors les solutions pour lesquelles les angles ne sont pas a l'extérieur des angles maximums et minimum des actuateurs. Avant de pouvoir choisir la solution correcte, on doit donc savoir à quoi correspont un angle θ en pas d'actuateurs.

## *Conversion d'angles en pas*

Deux paramètres sont utilisés pour convertir un angle θ en pas d'actuateur: le facteur d'échelle et le décalage. Le facteur d'échelle défini combien de pas correspond à un degré. Le décalage est utilisé pour définir 0 degrés = position 0 du robot. Ceci est nécessaire car un pas de 0 pour l'actuateur ne correspond pas à la position 0 du robot (bras en plaine extension et aligné le long de l'axe $y_1$). Afin d'obtenir ces deux paramètres, nous avons bougé chacun des actuateurs à -1400 et 1400 et mesuré l'angle effectif. Après quelques simples calculs, nous avons obtenu les valeurs du facteur d'échelle et du décalage. Nous avons par la suite effectué des ajustement mineurs au décalage afin d'obtenir une position 0 correcte. Voici les valeurs que nous avons obtenu ainsi que les angles maximums et minimums atteignables:

| *Actuateur* | *Angle maximum* | *Angle minimum* | *Facteur d'échelle* | *Décalage* |
|---|---|---|---|---|
| **1** | 77 | -73 | -18.6666 | 37.3333 |
| **2** | 85 | -89 | -16.0919 | -32.1800 |
| **3** | 87 | -84 | -16.3742 | 24.5614 |
| **4** | 83 | -88 | 16.3742 | 40.9356 |
| **5** | 81 | -100 | -15.4500 | -145.0000 |

Évidemment, on n'as pas besoin d'effectuer de conversion d'angle pour l'actuateur #6: un pas de 1400 veut simplement dire 'pince fermée' et un pas de -1400 veut dire 'pince ouverte'. Un pas de 0 signifie que la pince est entrouverte. Si on avait à saisir des objet fragiles, on pourrait calculer un facteur de conversion 'largeur de l'objet → taux d'ouverture de la pince' afin de ne pas mettre trop de pression sur l'objet lorsque l'on le saisit.

## *Choix de la solution*

Comme mentionné précédemment, MATLAB nous donne 8 solutions possible lorsque l'on résout les équations de cinématiques inverse. Généralement, 4 de ces solutions contiennent des valeurs imaginaires et peuvent être éliminées immédiatement. Si d'autre solutions contiennent des valeurs imaginaires, celles-ci peuvent aussi être éliminée.

En théorie, n'importe quelle des solutions qui reste est valide. En pratique, toutefois, on ne peut pas choisir une solution dont les angles sont en dehors des angles maximums et minimums des actuateurs. On élimine alors chacune des solutions pour lesquelles les angles $\theta_1$, $\theta_2$, $\theta_3$ ou $\theta_4$ excédent les angles maximums ou minimums. On laisse toutefois une certaine marge d'erreur afin d'augmenter la taille de l'espace de travaille. À noter que l'angle $\theta_5$, qui correspond à l'angle de rotation de la pince, peut prendre n'importe quelle valeur: un angle de -135 degré est équivalent à un angle de 45 degré et peut donc être ramené à une valeur sur une plage de -90 à 90 degrés.

Une fois la solution choisie, les cinq pas d'actuateur trouvés sont placé dans une commande ROBIX qui va bouger les actuateurs 1 à 5 en même temps.

Un des problèmes que nous n'avions pas prévu mais qui est survenu pendant les test fut le problème des solutions multiples valides. Dans certains cas, plus d'une solution peut être trouvée et être valide. Ceci peut causer des problème lorsque l'on veut descendre la pince vers l'objet à saisir: les actuateurs #1 et #2 peuvent alors faire une grande rotation et la pince va pousser l'objet hors de la position prévue. Pour prévenir ceci, un paramètre fut ajouté à la fonction qui choisit la solution: l'angle de l'actuateur #1. Lorsque l'on utilise la cinématique inverse pour trouver la position du bras au dessus l'objet, on garde en mémoire la valeur de l'actuateur #1. Lorsque l'on trouve la position du bras pour saisir l'objet (lorsque le bras descend) et que l'on choisit une des solutions, l'ancienne valeur de l'actuateur #1 est conservée. Le programme qui choisit la solution, `PickSolution.m`, va alors choisir, parmi les solutions valides, la solution dont la valeur de l'actuateur #1 est la plus proche de la valeur que l'on avait sauvegardée.

## *Espace de travail*

À cause de la configuration de notre robot, des limitations des actuateurs et des limitations que nous lui avons imposé, l'espace de travail de notre robot, c'est-à-dire l'espace dans lequel notre robot peut saisir un objet, est très restreint. En effet, l'espace de travail que l'on obtient consiste en un demi-anneau.

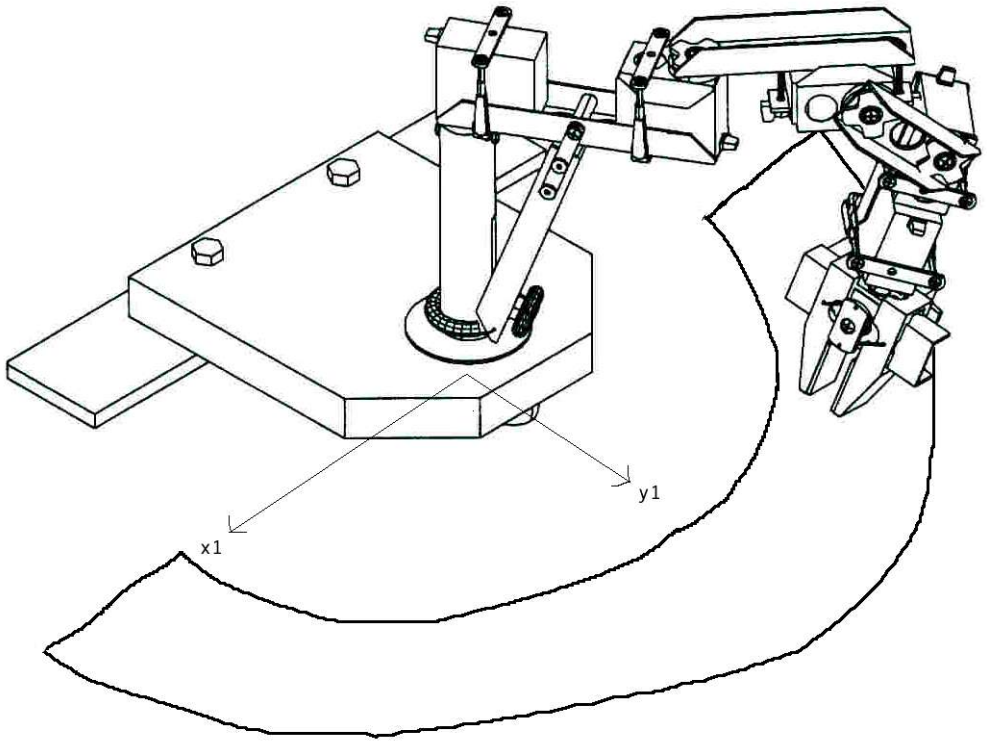


*Illustration 3: Espace de travail*

# Traitement d'image

Le traitement d'images se fait en plusieurs parties. Nous avons tout d'abord une phase de calibration, utile afin d'aligner le référentiel repère de notre espace de travail. Puis, une fois l'alignement effectué, nous pouvons procéder avec la capture de l'image.

L'image est par la suite traitée grâce à un programme (fonction) dans MATLAB. Le traitement de l'image débute par le filtrage de l'image. Puis, l'image est segmentée afin d'isoler le référentiel. Ce référentiel est ensuite analysé pour déterminer la position de son point de base et ses dimensions en pixels.

Ensuite, une autre fonction est appelée afin de trouver l'objet. Pour se faire, une segmentation est faite afin d'éliminer la partie ombragée de l'image. Puis, on tente de retrouver l'objet. Une fois l'objet retrouvé, son centre et orientation sont aussi déterminés.

Les données acquises permettront au bras manipulateur d'aller rejoindre l'objet.

## *Calibration*

La calibration de l'image est une étape essentielle chaque fois que la caméra et notre plan de travail est déplacé. Cette étape, plutôt simple, consiste tout simplement à exécuter le logiciel `i_view32.exe` (responsable pour acquérir l'image) sans paramètres. Ceci nous permet de sélectionner la source d'acquisition (dans notre cas la caméra Logitech) et d'aligner la caméra et notre plan de façon à ce que le référentiel repère se retrouve au coin bas-gauche de notre image.

Une fois cette étape terminée, nous pouvons procédé à l'utilisation de la caméra pour l'acquisition et le traitement d'images.

## *Capture de l'image*

La capture d'image se fait de façon sophistiquée puisque le tout est automatisé et ne requiert aucune intervention de l'utilisateur. La fonction utilisée pour capturée l'image est *captureimage.m*. En gros, cette fonction fait appel au logiciel `i_view32.exe` avec paramètres afin de capturer automatiquement l'image de grandeur 640x480 pixels et de la sauvegarder en *grayscale* et format *tiff* à la location `c:\myfile.tif`.

## *Filtrage*

L'image capturée est par la suite filtrée grâce à un filtre de moyennage 3x3 (moving average filter). Ce filtre permet de mettre notre image plus lisse (smooth) car chaque pixel devient la moyenne des pixels voisins. Grâce à ce filtre, nous avons pu éliminer les bruits de distorsions qui ressemblaient à du interlacing. En plus, ce filtre agit comme filtre passe-bas (mais passe-bas non agressif), ce qui veut dire que les hautes fréquences sont coupées, ce qui a comme effet d'agrandir la transition du noir au blanc au contour des objets.

Le filtre utilisé est illustré ci bas:

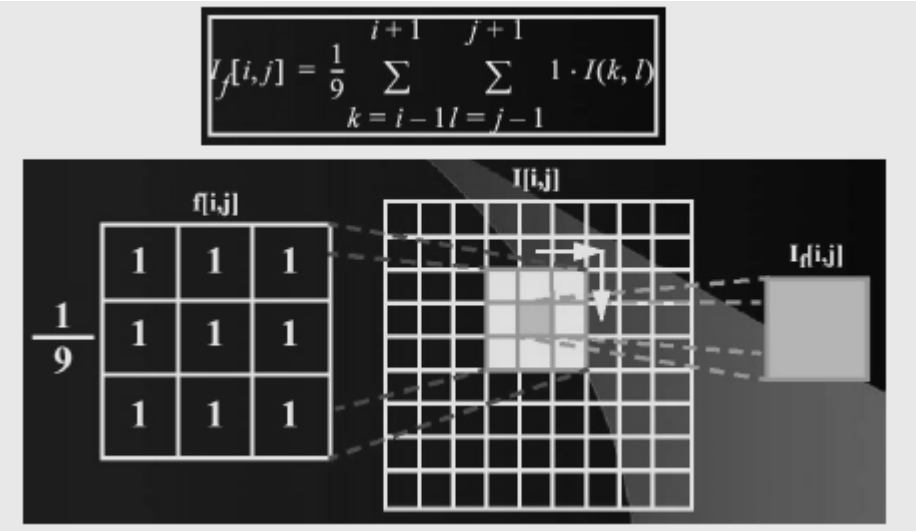


*Illustration 4: Filtre de moyennage 3x3*

Dans le domaine fréquentiel, le filtre a le contour suivant:

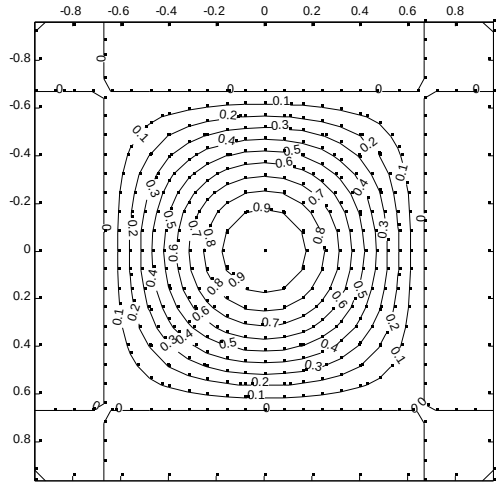

*Illustration 5: Réponse fréquentielle du filtre - contour*

Dans le domaine fréquentiel, le filtre a la perspective suivante:

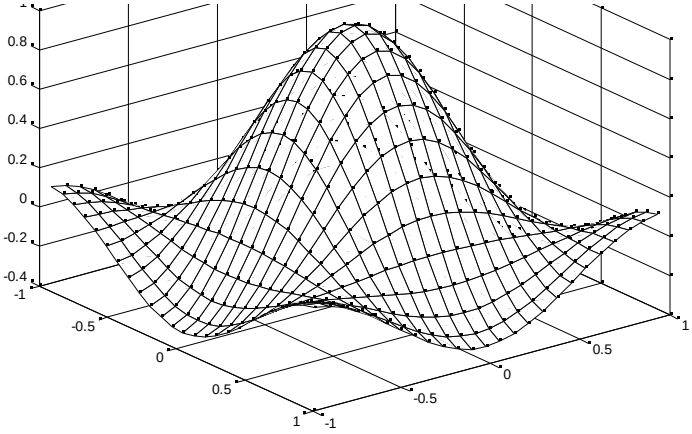

*Illustration 6: Réponse fréquentielle du filtre - perspective*

Voici une partie de l'image capturée par la caméra avant et après filtrage par le filtre par moyennage 3x3:

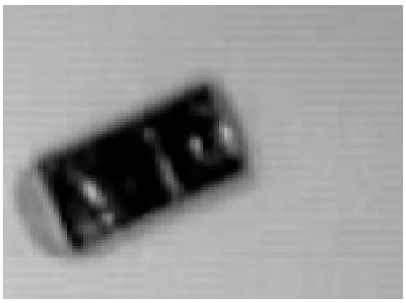

*Illustration 7: Image avant filtrage*

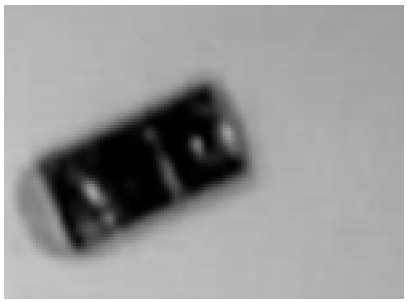

*Illustration 8: Image après filtrage*

## *Segmentation et seuillage*

La segmentation et le seuillage sont tous deux utilisées afin d'isoler les objets pertinents. En premier lieu, les 150 premiers pixels verticaux, du haut vers le bas, de l'image capturée sont éliminés afin de couper la partie de l'image qui inclue notre robot manipulateur. La majeure partie de l'ombre causée par les tablettes situées au-dessus de notre plan est aussi éliminée ici. L'image résultante est de grandeur 640x330 pixels.

Cette image sera donc à la fois utilisée dans l'isolation du référentiel repère et de l'objet à ramasser.

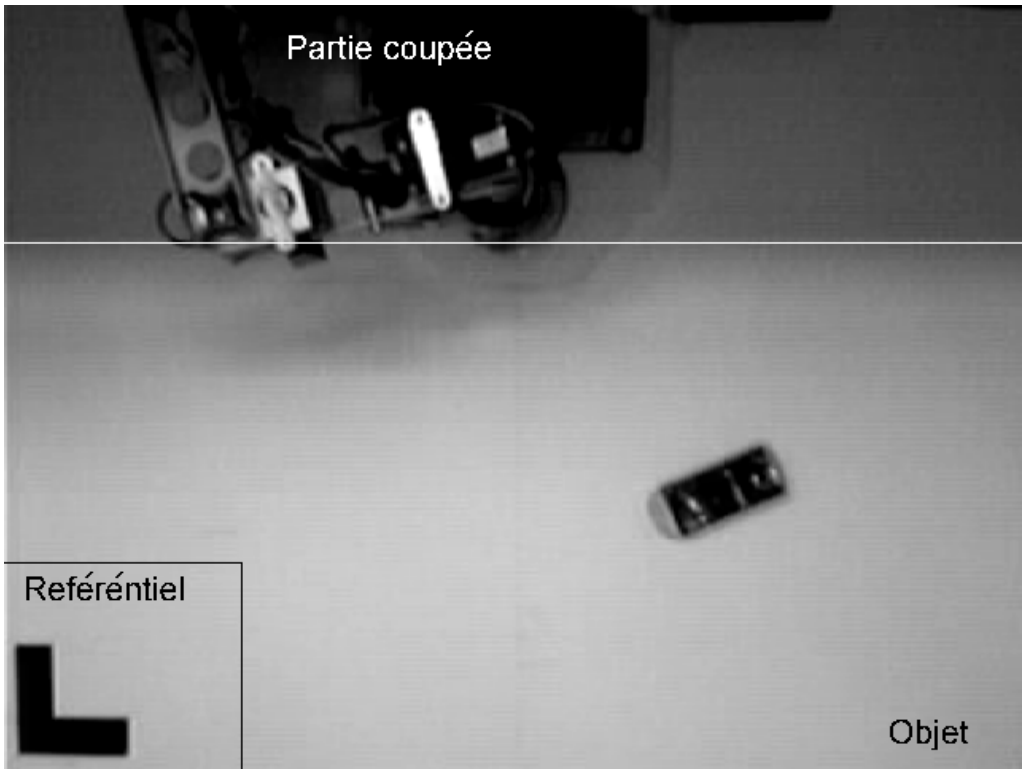


*Illustration 9: Segmentation de l'image*

## Référentiel repère

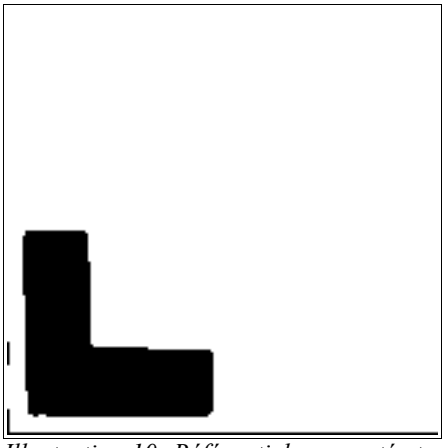

*Illustration 10: Référentiel segmenté et seuillé*

L'image de grandeur 640x330 pixels est à nouveau divisée verticalement en deux. Puis la grandeur horizontale est réduite à la hauteur verticale résultant. L'image résultante, contenant le référentiel est une image de 165x165 pixels.

Un seuillage est effectué à cette image puisque qu'elle est difficile à traiter due aux diverses intensités de noir et blanc. L'image est donc approximé et tout pixel ayant une couleur plus petite de 0.55 devient 0 (noir) alors que les pixels ayant une valeur plus grande ou égale à 0.55 prennent une valeur de 1 (blanc).

Ce seuillage a donc comme effet de rendre tout pixel de notre image soit noir ou blanc et la valeur centre de 0.55 fut choisie afin de rendre la surface du référentiel repère la plus visible possible tout en éliminant tout autre déchet autour de l'image.

## Objet

L'image de grandeur 640x330 pixels est tout d'abord modifié. Les 20 premiers pixels verticaux à partir du haut sont remplis en blanc.

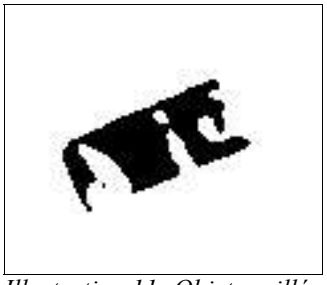

*Illustration 11: Objet seuillé*

Puis, réciproquement, un seuillage est effectué à cette image puisque qu'elle est difficile à traiter due aux diverses intensités de noir et blanc. Afin de trouver la position de l'objet, l'image est donc approximé et tout pixel ayant une couleur plus petite de 0.10 devient 0 (noir) alors que les pixels ayant une valeur plus grande ou égale à 0.10 prennent une valeur de 1 (blanc). L'image de droite démontre un problème encontré avec cette technique: si l'objet reflète trop de lumière, des 'trous' blancs apparaissent dans l'image de l'objet. Ceci peut être évité en utilisant un objet qui ne réflète pas la lumière.

Un seuillage différent est utilisé pour trouver la rotation de l'objet. Dans ce cas, tout pixel ayant une couleur plus petite que 0.25 devient 0 (noir) alors que les pixels ayant une valeur plus grande ou égale à 0.25 prennent une valeur de 1 (blanc).

## *Estimation du facteur d'échelle*

L'estimation du facteur d'échelle se fait grâce à notre référentiel repère. La segmentation et seuillage effectuée, nous avons maintenant le référentiel repère bien isolé. Le référentiel a été mesuré précisément grâce à une règle et a en réalité une grandeur de 5.3cm et une hauteur de 5.2cm.

Maintenant, tout ce qu'il nous faut est d'identifier la grandeur et hauteur de notre référentiel en pixels. Pour se faire, nous avons écrit quelques lignes de code qui trouve la rangée de l'image contenant le plus de pixels noirs et réciproquement, la colonne contenant le plus de pixels noirs. Le nombre de pixel de la rangée et colonne la plus grande est compté et un facteur d'échelle en cm/pixels est trouvé pour la grandeur (horizontale) et la hauteur (verticale) de l'image.

C'est grâce à ces facteurs que nous allons pouvoir convertir toute mesure pixels directement en centimètres par une simple multiplication.

## *Estimation de la position et l'orientation*

### Référentiel repère

Nous savons déjà la position réelle de la base de notre référentiel repère puisque nous l'avons précisément mesurée. Sa position est de (23.6 cm, 20.55 cm) par rapport au référentiel de base du robot. Vu d'en haut, le référentiel se situe à gauche et en bas du centre du robot.

Afin de trouver la position de base du référentiel repère en pixel, nous allons utilisé la rangée et la colonne la plus large du référentielle, tous deux identifiés lors du seuillage. Le premier pixel horizontal de la rangée la plus large et le dernier pixel vertical de la colonne la plus large serviront à former la position du référentiel repère.

Étant donné que l'image traitée était de grandeur 165x165 pixels, la position trouvée est par la suite ajustée pour l'image de 640x330 pixels puisque c'est l'image que nous utilisons pour travailler avec notre objet. Pour se faire, nous n'avons qu'à ajouter 165 pixels à la position verticale trouvée et à ajouter 475 pixels verticaux.

L'orientation du référentiel est déjà connu et est 0 degrés puisque nous avons ajusté l'appareil pour que le référentiel soit également droit durant la phase de calibration.

## Objet

Grâce au seuillage effectué pour l'objet, la seule partie noire de notre image de grandeur 640x330 pixels est l'objet photographié. Ainsi, afin d'estimer la position en pixels du centre de l'objet, il suffit de trouver recherché à partir du haut de l'image la première rangée de l'objet appelé Y1. Réciproquement, cette fois à partir du bas, nous trouverons la dernière rangée de l'objet appelé Y2. Puis, de la même façon, la première colonne de gauche et la première de droite sont trouvées et identifiés comme X1 et X2 respectivement.

Grâce à ces quatre données, la position centre de l'objet devient donc facile à trouver. La position du centre est donc [X1 + round((X2 - X1)/2), Y1 + round((Y2 - Y1)/2)].

Grâce à la position du centre de l'objet, nous pouvons alors trouver la rotation de cet objet. Pour ce faire, une recherche autour du point central est effectué afin de trouver le pixel blanc le plus près du centre de l'objet. L'arc tangente est alors effectué entre ce point et le point central et l'angle de rotation est alors trouvé.

La position de l'objet par rapport à la base du robot peut être simplement calculée en soustrayant la position du référentiel de la position de l'objet.

# Démonstration

### *Déroulement*

Notre programme en démonstration fonctionne remarquablement bien. Sans compter la phase de calibration, presque tout est automatique. Le programme prends une photo du plan de travail, traite l'image, identifie la position de l'objet par rapport au référentiel et détermine également l'orientation de cet objet. Grâce à la position du référentiel par rapport au à base du robot, la position de l'objet par rapport à la base du robot est également déterminée.

Avec les données recueillies, le programme calcule une trajectoire en trois points. Le premier point (non calculé) sert à positionner le robot dans un mode rebot (ou parking) à l'extérieur de notre espace de travail. Puis, pour le deuxième point, les angles pour chaque moteurs sont déterminées pour amener le bras au-dessus de l'objet (avec la pince pointant vers le bas et ouverte). Puis finalement, la position finale est calculée pour amener le bras vers l'objet et la pince est refermée. Pour chacune des positions calculées, un ensemble de solutions est trouvé. La première solution valide pour chaque angle est la solution utilisée. Une solution est valide si elle ne comporte aucune composante complexe et si l'angle convertie en pas se retrouve entre -1500 et 1500 (une marge de 100 fut ajoutée de chaque coté pour rendre la solution plus flexible).

Une trajectoire miroir est par la suite calculée afin de déplacer l'objet vers le coté opposé de l'espace de travail où il se trouve.

### *Contrôle de degrés de liberté*

Chacun de nos 6 degrés de libertés est pleinement exploités. Aucune restriction n'a été établie à ce niveau là. Aucun moteur n'est en position fixe. Par contre nous avons restreint les angles « tangage » et « lacet » de la matrice $Q_{TRTL}$ ce qui a comme effet de restreindre le bras manipulateur à ramasser l'objet vers le haut.

### *Résultats*

Les résultats de la démonstration sont très impressionnants. Pour des positions valides dans notre espace de travail, le bras manipulateur réussis à ramasser l'objet avec une précision assez bonne. Bien que la pince ne soit pas complètement centrée vers le centre de l'objet, elle réussis à le ramasser et à le déposer sans problèmes. Les raisons pour laquelle l'objet n'est pas retrouvé avec pleine précision sera discuté dans la discussion.

## Discussion

Les résultats obtenus lors de l'expérience se révèlent très satisfaisant. Néanmoins, une erreur est présente puisque parfois, le robot manipulateur n'aligne pas le centre de son effecteur directement avec le centre de l'objet. Il arrive que l'effecteur s'aligne de 1 à 3 centimètres plus loin que la cible.

Puisque les paramètres Denavit-Hartenberg furent précisément mesurés à plusieurs reprise, que le calcul de la cinématique inverse est précis puisqu'il est effectué par MATLAB et que la conversion d'angles à pas fut testé et ajusté avec précision, il semble évident que la source d'erreur se trouve au niveau du traitement d'image.

Lors du seuillage, une façon grossière d'identifier les pixels noirs et les pixels blancs est utilisée. Si il y a moindrement de la réflexion présente sur l'objet à ramasser ou si quelques déchets sont toujours présents sur l'image après le filtrage et seuillage, alors le centre et l'orientation de l'objet ne sera pas exact. C'est ici qu'on retrouve le seul manque majeur de précision, et qui, parfois, vient nuire à la performance de notre traitement d'image dépendant de l'intensité lumineuse de notre plan de travail. En général cette méthode fonctionne tout de même assez bien et s'est pourquoi nous l'avons utilisé. Toutefois, dans des applications où une précision constante devient très importante (chirurgie), un seuillage plus avancé est définitivement requis.

# Documentation

## *Exemple d'image utilisé*

## *Page de Soumission*

## *Code Source*