MU4RBI04 Introduction à la Robotique
Modèle et Commande d'un bras à 4ddl

...

Consignes pour le rapport

- Votre code doit être clair et bien commenté.
- Les fichiers du code Python sont à rendre dans un dossier. Le fichier principal devra être nommé *votre-nom-px100.py*.
- *Vous devez fournir un rapport format pdf contenant les réponses aux questions sous le nom votre-nom-tp.pdf. Vous êtes libre de mettre dans le rapport les résultats qui vous semblent les plus pertinents.*
- *L'ensemble doit être mis dans un dossier (zip ou rar) nommé votre-nom puis déposé dans le répertoire Compte-Rendu-TP sur Moodle.*
- *Vous pouvez rendre également une vidéo d'une durée de 1 à 2 mn.*

1 Objectif

L'objectif de ce TP est de comprendre, résoudre et d'implémenter les notions théoriques liés aux modèles géométriques-cinématiques directs et inverses d'un manipulateur sériel. On s'appuie sur un exemple relativement simple d'un bras manipulateur à 4ddl, muni en plus d'une pince à deux doigts. Le travail demandé est développer le code permettant de contrôler le manipulateur dans une tâche de prise-dépose d'un objet ou une série d'objets dont on connait la position cartésienne sur une table horizontale.

2 Introduction

On considère le bras sériel Pincher X100 représenté sur la figure (1) dont les paramètres de Denavit-Hartenberg sont donnés dans le tableau 1. La figure 3 rappelle cette convention et définit les repères de liaison, de la base et de l'effecteur du bras étudié.

Le tableau des paramètres DH est le suivant

i	1	2	3	4
d_i	L_1	0	0	0
θ_i	q_1	q_2	q_3	q_4
a_i	0	L_r	L_3	L_4
α_i	$-\pi/2$	0	0	0

TABLE 1 – Paramètres Denavit-Hartenberg du px100

avec $L_1 = 89.45$mm, $L_2 = 100$mm, $L_m = 35$mm, $L_r = \sqrt{L_2^2 + L_m^2} = 105.95$mm, $L_3 = 100$mm, L_4=109mm.

FIGURE 1 – Vue 3D du Pincher X100.

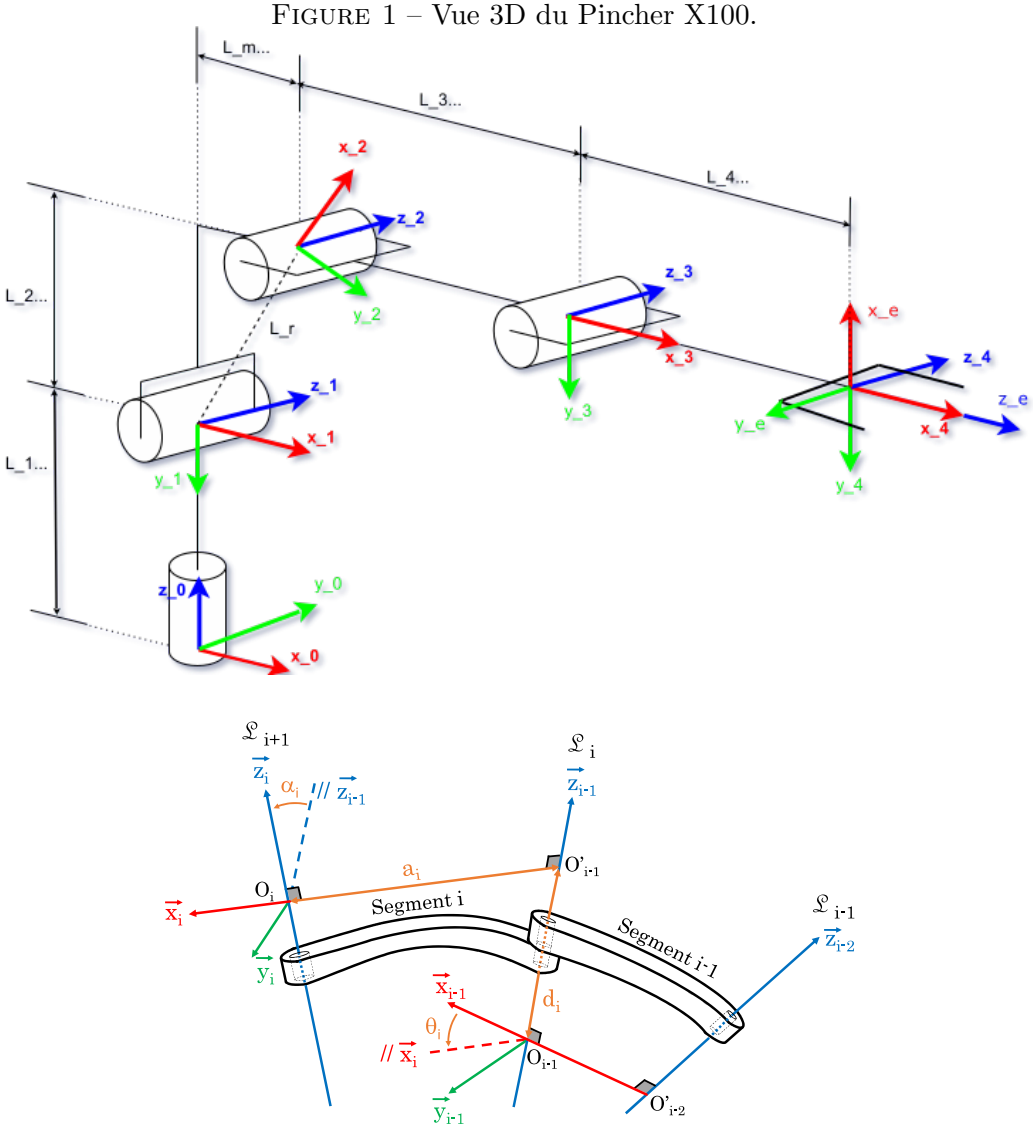

FIGURE 2 – Repères de la convention DH et Rappel de la convention.

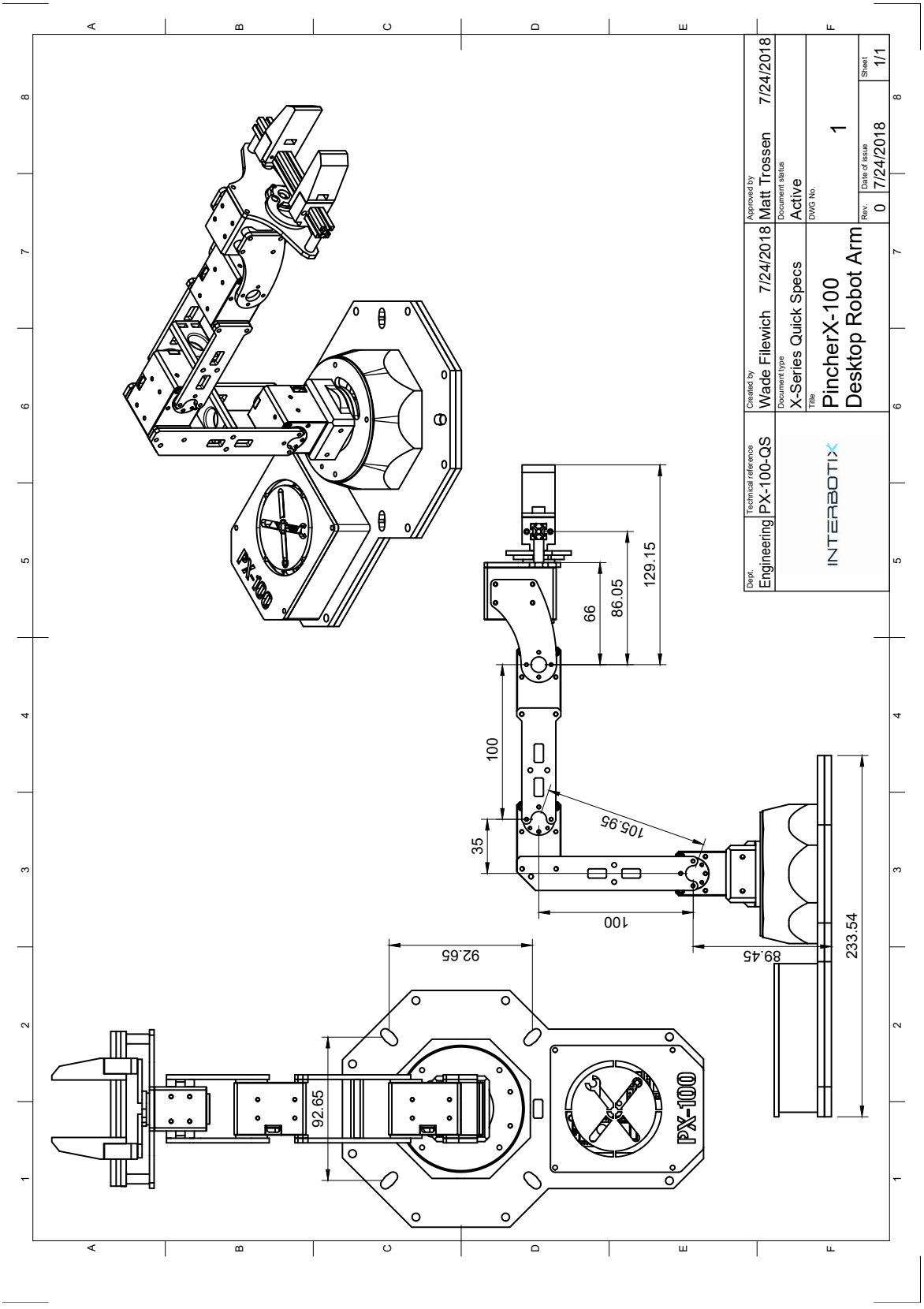

FIGURE 3 – Mise en plan du robot PincherX-100.

3 Modèle Géométrique Direct

On choisit comme vecteur de paramètres opérationnels (de tâche), la position du repère de l'effecteur et l'orientation de l'effecteur par rapport à un plan horizontal. Soit $\mathbf{x} = [p_x, p_y, p_z, \psi]^T$ avec (p_x, p_y, p_z) les coordonnées du vecteur $\boldsymbol{O_0O_4}$ dans $\mathcal{R}_0$ et $\psi = (\boldsymbol{x}_1, \boldsymbol{x}_4)$ mesuré autour de $\boldsymbol{z}_1$. En utilisant la méthode des transformations homogènes, on obtient la transformation de l'effecteur par rapport à la base

$$ {}^0T_4 = \begin{bmatrix} C_1C_{234} & -C_1S_{234} & -S_1 & C_1(L_rC_2 + L_3C_{23} + L_4C_{234}) \\ S_1C_{234} & -S_1S_{234} & C_1 & S_1(L_rC_2 + L_3C_{23} + L_4C_{234}) \\ -S_{234} & -C_{234} & 0 & -(L_rS_2 + L_3S_{23} + L_4S_{234}) + L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

et le modèle géométrique suivant

$$ \begin{cases} p_x = (L_r\cos(q_2) + L_3\cos(q_2+q_3) + L_4\cos(q_2+q_3+q_4))\cos(q_1) \\ p_y = (L_r\cos(q_2) + L_3\cos(q_2+q_3) + L_4\cos(q_2+q_3+q_4))\sin(q_1) \\ p_z = L_1 - L_r\sin(q_2) - L_3\sin(q_2+q_3) - L_4\sin(q_2+q_3+q_4) \\ \psi = q_2+q_3+q_4 \end{cases} \tag{1} $$

4 Modèle Géométrique Inverse

Après résolution de ces équations algébriques, les relations inverses peuvent se déduire pas à pas.

D'abord pour la première liaison : à partir des deux premières équations, on peut déduire q_1 directement des deux premières équations

$$ q_1 = \arctan2(p_y, p_x) \qquad \text{ou} \qquad q_1 = \arctan2(p_y, p_x) \pm \pi $$

Connaissant q_1, on peut déduire q_2

$$ q_2(q_1) = \arctan2\left(BW - \epsilon A\sqrt{A^2+B^2-W^2}, AW + \epsilon B\sqrt{A^2+B^2-W^2}\right). $$

avec

$$ A = 2L_rU, \;\; B = 2L_rV, \;\; W = U^2+V^2+L_r^2-L_3^2, \;\; \epsilon = \{-1, 1\} $$

et où

$$ U = p_xC_1 + p_yS_1 - L_4C_\psi, \;\; V = L_1 - p_z - L_4S_\psi, $$

puis q_3 en fonction q_1 et q_2

$$ q_3(q_1, q_2) = \arctan2\left(-US_2 + VC_2, UC_2 + VS_2 - L_r\right) $$

Enfin, l'angle q_4 peut être obtenu

$$ q_4 = \psi - q_2 - q_3 $$

5 Modèle cinématique et redondance

Pour beaucoup d'objets, la saisie peut être réalisée de plusieurs façons, par exemple l'orientation de la pince ψ peut avoir plusieurs valeurs possibles. L'idée ici est de ne plus spécifier cette orientation et de ne pas l'inclure dans le vecteur des paramètres opérationnels. On considère donc maintenant seulement la position de l'effecteur dans le vecteur des paramètres opérationnels $\mathbf{x} = [p_x, p_y, p_z]^T$. Le modèle géométrique devient donc

$$\begin{cases} p_x = (L_r\cos(q_2) + L_3\cos(q_2+q_3) + L_4\cos(q_2+q_3+q_4))\cos(q_1) \\ p_y = (L_r\cos(q_2) + L_3\cos(q_2+q_3) + L_4\cos(q_2+q_3+q_4))\sin(q_1) \\ p_z = L_1 - L_r\sin(q_2) - L_3\sin(q_2+q_3) - L_4\sin(q_2+q_3+q_4) \end{cases} \quad (2)$$

Le problème est maintenant redondant, c'est à dire qu'il existe une infinité de solutions possibles au modèle géométrique inverse.

Pour inverser ce modèle, on va chercher la meilleure configuration $\mathbf{q}$ qui satisfait la tâche $\mathbf{x}$ et qui optimise un fonction cout. On choisit une fonction qui mesure l'éloignement aux butées articulaires ou plus simplement le rapprochement à la configuration nominale $\mathbf{q}_{\text{nom}}$. La configuration nominale choisie est celle donnée sur la figure 2 où les variables d'angles actionneurs sont nulles.

Pour inverse le modèle, on utilise un algorithme itératif basé sur la solution de moindre carrée (pseudo-inverse) et la projection dans le noyau de la Jacobienne.

On rappelle les modèles cinématique $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$ ou des petits déplacements $\delta\mathbf{x} = \mathbf{J}\delta\mathbf{q}$, où $\mathbf{J}$ est la Jacobienne de dimension (3,4)

$$\mathbf{J} = \begin{pmatrix} -(L_rC_2 + L_3C_{23} + L_4C_{234})S_1 & (-L_rS_2 - L_3S_{23} - L_4S_{234})C_1 & (-L_3S_{23} - L_4S_{234})C_1 & -L_4S_{234}C_1 \\ (L_rC_2 + L_3C_{23} + L_4C_{234})C_1 & (-L_rS_2 - L_3S_{23} - L_4S_{234})S_1 & (-L_3S_{23} - L_4S_{234})S_1 & -L_4S_{234}S_1 \\ 0 & -L_rC_2 - L_3C_{23} - L_4C_{234} & -L_3C_{23} - L_4C_{234} & -L_4C_{234} \end{pmatrix}$$

La solution particulière dite de moindre carré (qui minimise $\delta\mathbf{q}^T\delta\mathbf{q}$) est

$$\delta\mathbf{q}_p = \mathbf{J}^{\#}\delta\mathbf{x} \quad (3)$$

où $\mathbf{J}^{\#}$ est la pseudo-inverse (ou de Moore-Penrose) de $\mathbf{J}$.

La solution générale est la suivante

$$\delta\mathbf{q} = \mathbf{J}^{\#}\delta\mathbf{x} - (\mathbf{I}_4 - \mathbf{J}^{\#}\mathbf{J})\nabla_{\mathbf{q}}\mathbf{\Phi} \quad (4)$$

Le second terme permet de contrôler une tâche secondaire en minimisant une fonction $\mathbf{\Phi}$ et n'influe pas sur la tâche princiaple.

Comme dit précédemment, on cherche à s'approcher au mieux de la configuration nominale $\mathbf{q}_{\text{nom}}$, d'où la fonction $\mathbf{\Phi} = \frac{1}{2}(\mathbf{q} - \mathbf{q}_{\text{nom}})^2$ qui a son minimum en $\mathbf{q} = \mathbf{q}_{\text{nom}}$. La matrice $(\mathbf{I}_4 - \mathbf{J}^{\#}\mathbf{J})$ est appelée projecteur dans le noyau de la Jacobienne car le second terme dans l'espace articulaire ne produit pas de mouvement opérationnel (on peut vérifier facilement que $\mathbf{J}(\mathbf{I}_4 - \mathbf{J}^{\#}\mathbf{J}) = \mathbf{0}$) et donc ne perturbe pas la tâche principale.

L'algorithme itératif peut donc s'écrire

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \alpha \mathbf{J}^{\#}(\mathbf{q}_k)(\mathbf{x}_{\text{but}} - \mathbf{x}_{\text{courant}}) + \beta(\mathbf{I}_4 - \mathbf{J}^{\#}(\mathbf{q}_k)\mathbf{J}(\mathbf{q}_k))(\mathbf{q}_{\text{nom}} - \mathbf{q}_k) \quad (5)$$

α, β des constantes scalaires positives à régler. Ils doivent être en général relativement petits pour des raisons de stabilité et de validité locale de la Jacobienne. $\mathbf{q}_k$ est la valeur courante (mesurée de préférence si disponible, sinon on prend la valeur calculée à l'instant précédent). $\mathbf{q}_{k+1}$ est la nouvelle valeur du vecteur des variables articulaires à envoyer comme consigne aux servo-moteurs.

6 Matériel et logiciel

On utilise Python3 comme langage de programmation et l'environnement ROS (qui doit être transparent pour vous et ce TP), le tout peut fonctionner aussi bien sous Linux ou Windows-WSL. Le docker expliqué sur cette page (lien) permet d'installer tous les paquets (ROS, Rviz, pilotes ...) nécessaires à la commande de ce robot.

Fonctions disponibles :

- `arm = InterbotixManipulatorXS("px100", "arm", "gripper")`
- `arm.set_joint_positions(joint_positions)`
- `arm.gripper.open(delay=)`
- `arm.gripper.close(delay=)`
- `arm.gripper.set_pressure(int)`
- `arm.go_to_home_pose()`
- `arm.go_to_sleep_pose()`
- `arm.capture_joint_positions(arm)`

Précautions et sécurité du matériel :

- Ne pas changer la vitesse des moteurs, car ils peuvent être endommagés si des impacts violents sur les butées articulaires ou avec l'environnement se produisent.
- Ne pas manipuler le robot s'il est sous tension, là encore on peut casser les engrenages de transmission des servos-moteurs.
- On peut vérifier la communication avec le robot sous Rviz. Entrer le nom du robot px100 puis faire update, ensuite vous pouvez tester les configurations *sleep* et *home*.
- A la fin de la séance, remettre le robot et son alimentation dans la boite prévu à cet effet. Ne pas déconnecter l'USB sur le robot.

7 Travail demandé

1. Donner en radians les valeurs des angles q_i pour la configuration de la Figure 2.
2. L'envoi d'un vecteur de commande $q_a = [0, 0, 0, 0]$ aux actionneurs, conduit le manipulateur à la configuration de la Figure 2. Ecrire deux fonctions qui permettent de convertir le vecteur q (qu'on peut appeler aussi q_{DH} par exemple) en q_a (espace actionneurs) et inversement.
3. Ecrire la fonction qui permet de donner la matrice de transformation homogène de l'effecteur en fonction du vecteur des paramètres articulaires $[q_1, q_2, q_3, q_4]$ (qDH).
4. Ecrire la fonction qui permet de convertir un vecteur de paramètre opérationnel $[p_x, p_y, p_z, \psi]$ en un vecteur de paramètre articulaire $[q_1, q_2, q_3, q_4]$ (qDH).
   - Calculer toutes les solutions,
   - Comment choisir la meilleure solution,
   - Traiter le cas des poses inatteignables et d'inexistence de solutions.
5. Sur la base de ce modèle, réaliser une prise-dépose d'un objet.

6. Généraliser cette tâche à un ensemble d'objets à déplacer.
7. Ecrire la fonction qui permet de calculer la Jacobienne du manipulateur.
8. Pour une pose donnée atteinte par le MGI, faire varier la configuration du robot sans que la position de l'effecteur bouge, en utilisant l'algorithme itératif de l'équation (5).
9. Sur la base de ce modèle, réaliser des prise-déposes d'objets.