Séance 1
Modélisation et commande géométrique d'un robot sériel nR
– Application au 3R plan –

1 Considérations préliminaires

Où on vous explique un peu ce que vous devez faire par la suite.

1.1 Objet de la séance de TP

Ce TP a pour objet le développement d'outils de calcul des modèles géométrique direct et inverse pour un robot 3R plan. Ces outils prennent la forme de fonctions Python, qui permettent de calculer les transformations entres les coordonnées articulaires (*les angles aux articulations*) et opérationelles (*la position et l'orientation de l'outil terminal*).

Hormis le modèle géométrique inverse pour lequel il est impossible de trouver une solution analytique générale pour tout robot, les fonctions développées ont une portée plus large que le simple 3R plan et peuvent être utilisées pour tout robot sériel plan à n degrés de liberté de type rotoïde [1]. En 2nde partie de TP, ces fonctions sont utilisées pour mettre en œuvre des techniques d'interpolation géométrique simples mais dont les concepts associés sont fondamentaux en Robotique.

1.2 Modèle géométrique direct

Le modèle géométrique direct (**MGD**) permet de connaître la pose (position et orientation) du repère $\mathcal{R}_{OT}$ associé à l'organe terminal d'un robot et exprimé dans un repère de référence $\mathcal{R}_0$ en fonction des paramètres géométriques constants et variables (degrés de liberté) du système.

Dans le cas d'un système constitué de solides rigides, cette pose peut être calculée à partir des transformations homogènes successives permettant de passer du repère associé à l'organe terminal à des repères intermédiaires et finalement au repère de référence.

Un choix judicieux du repérage intermédiaire facilite grandement la description des calculs à mettre en œuvre afin d'obtenir le MGD. Nous proposons ici l'utilisation d'une convention dédiée au robots nR plans et donc plus simple d'interprétation, comme il a été vu en TD 1.2.

1.3 Convention de repérage

Etant donnés deux corps successifs $i-1$ et i et la liaison rotoïde i entre ces deux corps, les repères intermédiaires d'intérêts retenus pour ce TP sont décrits par la figure 1. Par convention, l'axe de la liaison i est porté par l'axe $\vec{z}_{i-1}$ et le centre O_i du repère $\mathcal{R}_i$ est placé à l'extrémité du corps i.

La transformation homogène permettant d'exprimer les coordonnées d'un point P dans le repère $\mathcal{R}_{i-1}$ à partir de ses coordonnées dans le repère $\mathcal{R}'_i$ (*la transformation* $\mathcal{R}_{i-1} \rightarrow \mathcal{R}'_i$) est une rotation pure d'axe $\vec{z}_{i-1}$ et d'angle q_i. Cette première transformation permet de tenir compte du degré de liberté en rotation associé à la $i$ème liaison. Elle s'écrit :

$$ {}^{\mathcal{R}_{i-1}}H_{\mathcal{R}'_i} = \begin{bmatrix} R_{z_{i-1}}(q_i) & \mathcal{O}_{3\times 1} \\ \mathcal{O}_{1\times 3} & 1 \end{bmatrix} \tag{1} $$

où $R_z(q)$ est la matrice de rotation 3×3 correspondant à une rotation d'angle q autour du vecteur $\vec{z}$. $\mathcal{O}_{3\times 1}$ indique une colonne (ou $\mathcal{O}_{1\times 3}$ respectivement une ligne) de zéros. Donc la matrice H est de dimensions 4×4.

La transformation homogène permettant d'exprimer les coordonnées d'un point P dans le repère $\mathcal{R}'_i$ à partir de ses coordonnées dans le repère $\mathcal{R}_i$ est une translation pure (*la transformation* $\mathcal{R}'_i \rightarrow \mathcal{R}_i$). Cette seconde transformation permet de tenir compte de la translation $\overrightarrow{O_{i-1}O_i}$ qui correspond donc à la longueur du segment l_i. Elle s'écrit :

$$ {}^{\mathcal{R}'_i}H_{\mathcal{R}_i} = \begin{bmatrix} \mathcal{I}_{3\times 3} & {}^{\mathcal{R}'_i}\overrightarrow{O_{i-1}O_i} \\ \mathcal{O}_{1\times 3} & 1 \end{bmatrix}. \tag{2} $$

1. L'extension au cas prismatique serait une formalité mais, dans les séances de TP suivantes, seuls de robots à liaisons rotoïdes pourront être construits.

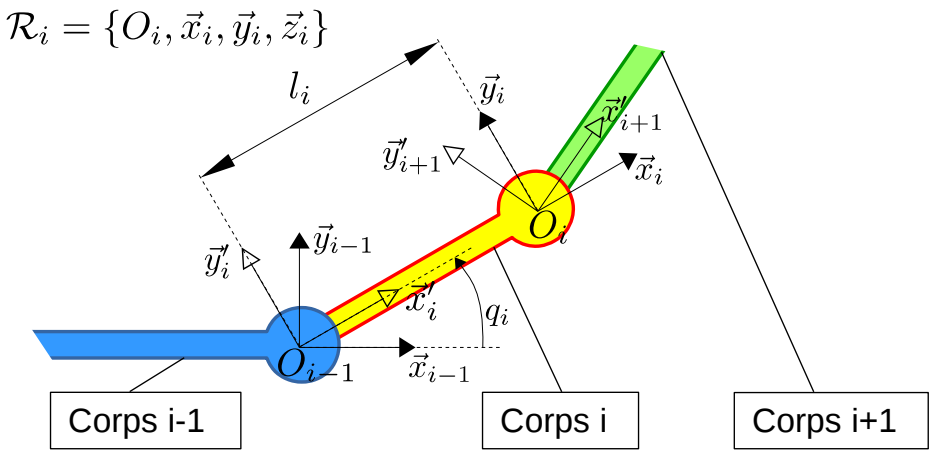

FIGURE 1 – Repères intermédiaires associés au corps et à la liaison i pour la convention retenue.

Finalement, la transformation homogène ${}^{i-1}H_i$ s'écrit comme la multiplication des deux matrices précédentes (*la transformation* $\mathcal{R}_{i-1} \to \mathcal{R}'_i \to \mathcal{R}_i$) :

$$ {}^{\mathcal{R}_{i-1}}H_{\mathcal{R}_i} = {}^{\mathcal{R}_{i-1}}H_{\mathcal{R}'_i} \times {}^{\mathcal{R}'_i}H_{\mathcal{R}_i}. \tag{3}$$

2 Modélisation

2.1 Préparation

L'ensemble des TPs s'appuient sur le code fourni au sein du fichier `robot3rPlan` qui utilise la programmation objet orientée avec Python. Ce fichier contient tous les éléments nécessaires pour modéliser un robot 3R plan, l'affichage avec Pygame, mais également communiquer avec la maquette du robot réel que vous utiliserez dans les séances suivantes.

Voici une explication rapide des trois classes :

— La classe Robot permet de créer un robot en associant les paramètres articulaires initiales et la taille des segments du robot. Dans un second temps votre travail sera de remplir cette classe qui doit contenir toutes les fonctions qui permettront de modéliser le robot.

— La classe Univers permet de simuler un robot plan à n articulations (notamment un robot 3R) avec visualisation Pygame, en offrant des modes de contrôle articulaires et cartésiens, ainsi qu'une interface optionnelle avec des moteurs réels via une connexion série. Elle gère l'affichage, les interactions et le pilotage en temps réel.

— La classe Motor permet d'initialiser et contrôler des moteurs Dynamixel AX-12 pour un robot articulé, en assurant la conversion entre positions en degrés et valeurs brutes des moteurs, la lecture de l'état des moteurs, et l'envoi de commandes de position.

1. Ouvrir le fichier principale `main.py`, vous devez apercevoir une instance de la classe robot (création d'un robot avec tout les paramètres nécessaire de la classe Robot) et une instance univers (de même avec la classe Univers). C'est en appelant ces classes que vous pouvez modifier les paramètres des objets. Par exemple, pour un robot vous devez indiquer la taille des segments et les paramètres articulaire d'origine.

   Pour pouvoir afficher le robot nous devons connaître les positions de ses articulations, d'où l'importance de calculer le MGD.

2.2 Construction du modèle géométrique directe

Fonction à compléter dans la classe Robot.

2. Expliquer comment trouver une matrice de rotation d'un angle en rotation autours de l'axe z.
3. Expliquer comment se forme une matrice de transformation homogène permettant de décrire le changement de position et d'orientation d'un point ou d'un repère dans l'espace. Ces transformations peuvent être obtenues de manière itérative en partant du repère de référence $\mathcal{R}_0$ et en utilisant les propriétés de composition des transformations homogènes vues en cours et étudiées dans le TD1.

4. En déduire, deux fonctions Python permettant d'utiliser les fonctions précédentes respectivement `matriceRotZ` et `transformationHomogene`.

5. D'après les relations établies précédemment, expliquez comment calculer le modèle géométrique directe avec la méthode des transformations homogènes.

6. Établir la fonction Python `MGD` qui permet de mettre à jour les positions des articulations *self.pos* sans ne rien renvoyer.
7. Tester votre fonction `MGD` dans trois configurations articulaires simples du modèle où la solution peut se calculer facilement de manière manuelle.

2.3 Construction et affichage d'un robot 3R plan

Dans le dossier où se situe les fichiers téléchargés précédemment, ouvrir le fichier `main.py`.

1. Commencez par créer une instance de la classe robot avec comme configuration articulaire $\left[\frac{\pi}{2}\ 0\ 0\right]^T$ et ayant des segments respectivement de longueurs $0.5m$, $0.3m$ et $0.2m$.
2. Créer ensuite une instance de la classe Univers avec comme argument votre robot.
3. Utiliser la fonction `run` de la classe Univers pour lancer la simulation. Dans cette fonction le `MGD` et la fonction d'affichage `plotRobot` également. Il est vivement conseillé de bien comprendre cette partie du code.

2.4 Construction du modèle géométrique inverse

Fonction à compléter dans la classe Robot.

1. Dans le cas particulier du robot 3R plan, inversez le MGD pour trouver le `MGI` qui retourne les configurations du robot correspondant à une position cartésienne x, y et à une orientation θ dans le plan.

2. Implémentez vos résultats sur python dans la fonction `MGI`.

3. Utiliser maintenant la fonction afin de déterminer les configurations articulaires nécessaires pour atteindre une cible opérationnelle triviale ou non triviale. Vérifier alors que l'utilisation du modèle direct conduit bien à atteindre la cible souhaitée. *Existe-t-il une seule solution ?*
4. Utiliser le modèle inverse pour déterminer les configurations articulaires permettant d'atteindre la position $[1.0\ 1.0]^T$ et l'orientation $\frac{3\pi}{2}$. Tester ensuite la configuration obtenue en utilisant le modèle direct. Que peut-il être dire de la solution obtenue avec le modèle inverse ?

3 Simulation : Génération de trajectoires par interpolation

En pratique, lorsque l'application robotique nécessite que vous placiez l'organe terminal du bras en une position opérationnelle particulière (pour récupérer une pièce, par exemple), vous devez générer une trajectoire pour chacun des moteurs du robot, i.e. une suite de configurations articulaires que le robot doit atteindre dans le temps afin de suivre une trajectoire prédéfinie avec l'extrémité du robot. Cette suite de configuration est déterminée par interpolation, effectuée soit dans l'espace articulaire, soit dans l'espace opérationnel.

Fonction `simulateVirtualRobot` à compléter dans la classe Univers.
Selon le paramètre mode passer en argument de la fonction, il devra être possible de choisir les différents types de comportement du robot.

3.1 Interpolation articulaire

1. Étant données une configuration articulaire de départ quelconque et une cible opérationnelle (position et orientation dans le plan) que vous choisissez, écrire dans le script principal les instructions permettant de rejoindre la cible en effectuant une interpolation sur N positions articulaires intermédiaires dans l'espace articulaire.
   Proposer une critère simple permettant de choisir entre les 2 solutions A et B du modèle géométrique inverse.
2. Quels sont les avantages et inconvénients inhérents à cette manière d'interpoler ?

3.2 Interpolation opérationnelle

1. Étant données une configuration articulaire de départ quelconque et une cible opérationnelle (position et orientation dans le plan) que vous choisissez, écrire dans le script principal les instructions permettant de rejoindre la cible en effectuant une interpolation sur 100 configurations opérationnelle intermédiaires dans l'espace opérationnel.
2. Quels sont les avantages et inconvénients inhérents à cette manière d'interpoler ?