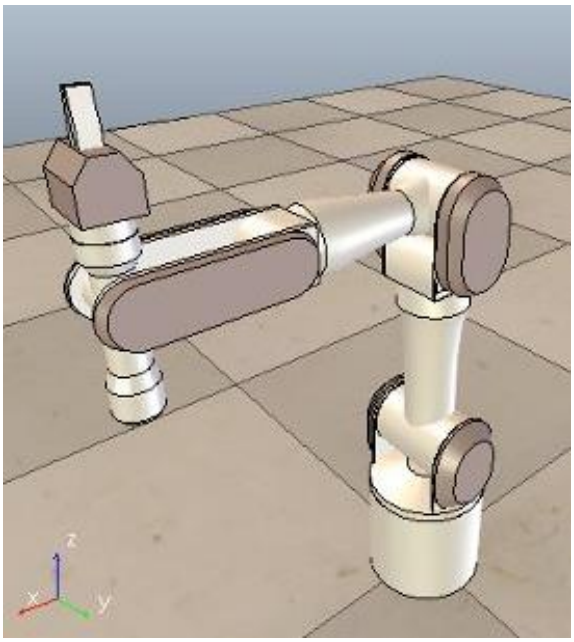
Apprentissage en ligne d'un bras manipulateur

Embedded Systems and Robotics – Mines de Nancy



Noémie LAGUELLE – Juin 2019

Dans le cadre du cours “Embedded Systems and Robotics” présenté par M. Hénaff, nous devions conduire un projet sur le thème de l’apprentissage de loi de commande par réseau de neurones en utilisant le logiciel de simulation V-REP et des scripts python. Dans cet optique, j’ai choisi de m’intéresser à un bras manipulateur et à son apprentissage en ligne afin qu’il puisse saisir des objets.

I. Objet d’étude

Le logiciel V-REP propose plusieurs modèles de bras robots et j’ai choisi celui dénommé « 7 DoF manipulator » qui avait l’avantage de présenter une pince à l’extrémité de son bras. D’autre part, afin de rendre son étude moins complexe, j’ai préféré bloquer 4 de ses articulations pour pouvoir le considérer comme un bras plan à 3 articulations. Les liaisons pivots encadrées en vert ci-dessous sont celles non bloquées :

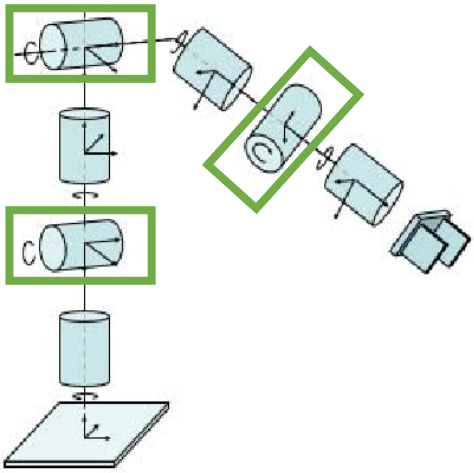
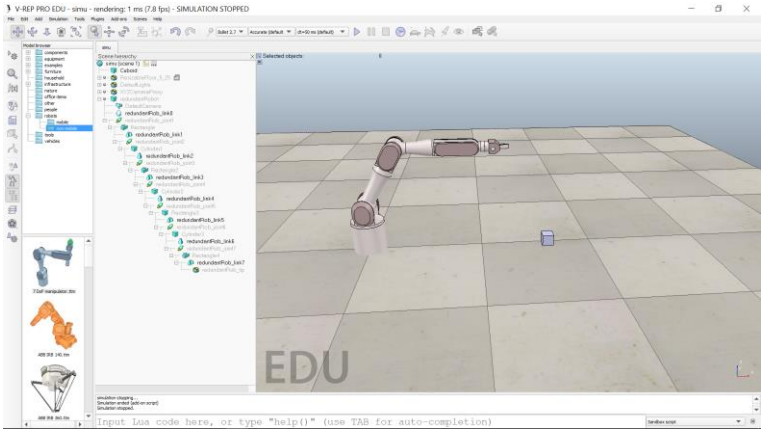


Fig. 1.structure of the manipulator

Source: Kinematics-based studies on a 7-DOF redundant manipulator – Liu, Tsai et Hsiao

Je m’intéresse à la scène suivante :



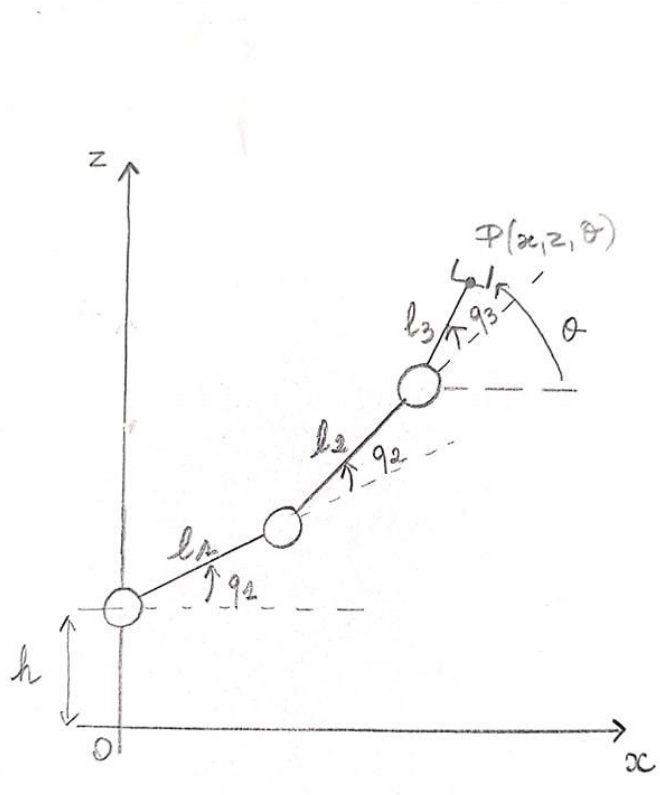
Mon robot est placé à l'origine et je souhaite que l'extrémité de son bras touche l'objet placé en un point connu et non pas le saisisse dans un premier temps.

Dans cet optique, j'ai repris la structure vue en cours pour l'apprentissage en ligne du robot Pioneer et l'ai adaptée à mon robot. Je me place dans un plan différent de celui du Pioneer, ici ce sont les coordonnées x, z et θ dans le plan (0,x,z) de l'extrémité du bras qui m'intéressent. Cependant de la même manière, lorsqu'on exécutera run.py, deux coordonnées de position et une d'orientation nous seront demandées et en sortie du réseau de neurones nous n'obtiendrons non pas 2 mais 3 vitesses angulaires relatives aux 3 articulations laissées libres.

II. Détermination du gradient

1) Etude géométrique du bras

On se place dans le plan (O,x,z) puisque les trois liaisons ont pour axe de rotation l'axe y :



Avec ce schéma, on peut déterminer le <u>modèle géométrique direct</u> de ce manipulateur plan 3 axes :

$$x = l_1 \cos(q_1) + (l_1 + l_2) \cos(q_1 + q_2) + (l_1 + l_2 + l_3) \cos(q_1 + q_2 + q_3)$$
$$z = l_1 \sin(q_1) + (l_1 + l_2) \sin(q_1 + q_2) + (l_1 + l_2 + l_3) \sin(q_1 + q_2 + q_3)$$

Pour la suite, il sera nécessaire de connaitre les expressions de q2 et q3 en fonction de x,y et θ ainsi, comme $\theta = q_1 + q_2 + q_3$:

$$x = l_1 \cos(\theta - q_2 - q_3) + (l_1 + l_2) \cos(\theta - q_3) + (l_1 + l_2 + l_3) \cos(\theta)$$

$$z = l_1 \sin(\theta - q_2 - q_3) + (l_1 + l_2) \sin(\theta - q_3) + (l_1 + l_2 + l_3) \sin(\theta)$$

D'où $(x - l_3 \cos(\theta))^2 + (z - l_3 \sin(\theta))^2 = {l_1}^2 + {l_2}^2 + 2 * l_1 * l_2 * \cos(q_2)$

Ainsi $\boxed{q_2 = cos^{-1}(((x - l_3 \cos(\theta))^2 + (z - l_3 \sin(\theta))^2 - {l_1}^2 - {l_2}^2)/\ 2 * l_1 * l_2)}$

Ceci permet ensuite de déterminer q3 :

$$q_3 = cos^{-1}([(l_1 \cos(\theta - q_2) + l_2 \cos(\theta))(x - l_3 \cos(\theta)) + (z - l_3 \sin(\theta))(l_1 \sin(\theta - q_2) + l_2 \sin(\theta))]/({l_1}^2 + {l_2}^2 + 2 l_1 l_2 (\cos(\theta) + \sin(\theta))))$$

2. Passage aux vitesses

La détermination du modèle géométrique direct permet de relier la vitesse de rotation des articulations à la position du point P :

$$\dot{x} = -l_1 \dot{q1} \sin(q_1) - (l_1 + l_2)(\dot{q_1} + \dot{q_2}) \sin(q_1 + q_2) - (l_1 + l_2 + l_3)(\dot{q_1} + \dot{q_2} + \dot{q_3}) \sin(q_1 + q_2 + q_3)$$

$$\dot{z} = l_1 \dot{q1} \cos(q_1) + (l_1 + l_2)(\dot{q_1} + \dot{q_2}) \cos(q_1 + q_2) + (l_1 + l_2 + l_3)(\dot{q_1} + \dot{q_2} + \dot{q_3}) \cos(q_1 + q_2 + q_3)$$

$$\dot{\theta} = \dot{q1} + \dot{q_2} + \dot{q_3}$$

L'avantage est que pour le robot Pioneer, deux coordonnées de position et une d'orientation sont demandés en entrée lorsque l'on exécute run.py, on peut donc reprendre son expression du gradient en modifiant les dérivées partielles de $\dot{x}, \dot{z}\ et\ \dot{\theta}$ par rapport à $\dot{q}_i$ et en s'affranchissant du θ_s. C'est ici qu'interviennent les expressions de q2 et q3 en fonction x,z et θ à un instant t. Vous pouvez retrouver l'expression de notre gradient dans le fichier « online_trainer.py ».

III. Adaptation des scripts

1) « online_trainer.py »

Pour ce fichier, outre la modification du gradient, j'ai aussi modifié les alphas en fonction de l'amplitude de notre bras.

2) « vrep_manipulator_simulation.py »

J'ai d'autre part modifié le fichier « vrep_pioneer_simulation.py » en « vrep_manipulator_simulation.py » où je donne les valeurs des différentes longueurs rencontrées précédemment. Il est à noter que l'étude géométrique faite précédemment considère que le première articulation est située à l'origine or elle est située à une hauteur h, il est donc nécessaire de soustraire cette hauteur à position[1] (coordonnée z du tip à l'instant t) dans le calcul du gradient.

D'autre part, une différence avec le robot Pioneer est qu'ici, set_position et get_position concernent l'élément « redundantRob_tip » et non le robot lui-même. En effet, je souhaite que ce soit la pince qui soit en (x,z, θ) et non le centre du manipulateur qui n'est pas du tout situé à l'extrémité du bras. Une difficulté à laquelle je me suis heurtée à été de savoir quel objet choisir puisqu'à l'extrémité du bras il y a plusieurs composants dont on peut récupérer les coordonnées et dans un premier temps je n'avais pas commenté la ligne self.set_position(self.initial_position) ce qui entrainait une désolidarisation du composant choisi avec le reste du robot (cela fonctionnait très bien avec le Pioneer puisqu'on changeait la position du centre du robot).

On remarque aussi que notre θ correspond à l'angle beta donné par V-REP à 90° degrés près, il est donc nécessaire de soustraire 90 par le θ calculé dans la fonction set_position afin d'avoir l'orientation voulue.

IV. Modification du robot dans V-REP

Parallèlement à la modification des différents scripts, il a fallu aussi adapter le robot fourni par le logiciel.

En effet, initialement, ce robot utilisait la fonction de cinématique inverse afin qui, lorsqu'on déplaçait une sphère de manipulation, calculait les angles de chaque articulation afin que le bras se trouve dans la bonne configuration pour que sa pince coïncide avec la boule. J'ai donc désactivé cette fonction, passé chaque *joints* en mode « torque/force » avec « motor enabled » coché, « control loop enabled » décoché et « lock motor when target velocity is zero » coché. Cette *target velocity* je la mets à 0 initialement pour toutes les articulations puis à chaque nouvelle commande pour les *joints* 1-4-5-7 que je souhaite fixe dans le script « online_trainer.py ».

V. Résultat et commentaires

Avec le fichier vidéo capture_video.avi, vous pouvez voir un résultat obtenu en rentrant la commande « 0.8 0.1 0 ». La pince touche l'objet et semble vouloir s'en rapprocher néanmoins on finit par avoir un problème de domaine pour la fonction *acos* utilisée pour déterminer q3.

Ce problème n'arrive pas toujours et je ne comprends pas pourquoi pour une même configuration j'ai parfois une erreur et parfois non.

Si maintenant vous vous intéressez à capture_video2.avi vous pouvez voir que le robot semble diverger alors que pour cette même commande, il peut tout aussi bien rester près de l'objet. Là encore, j'ai une erreur de domaine pour *acos*.

VI. Difficultés

J'ai eu du mal à prendre en main V-REP même si le fait de faire préalablement des tutoriels m'a aidée à comprendre quelques fonctionnalités basiques mais essentielles notamment la notion de *joint* qui est omniprésente dans ce projet. Néanmoins, la compréhension de la commande du robot sous python a nécessité du temps et savoir quelles fonctionnalités activer et lesquelles désactiver sur V-REP n'était pas évident.

La compréhension de l'apprentissage en ligne et des scripts donnés a aussi été laborieuse cependant je discerne maintenant un peu mieux le fonctionnement d'un tel système.

J'ai rencontré de nombreux problèmes tout au long de ce projet qui m'ont permis aussi d'en apprendre plus puisqu'à chaque fois il a fallu les résoudre petit à petit. Mon script est encore loin d'être parfait, j'ai essayé de jouer sur les signes pour régler les problèmes persistants cependant je pense que certains points sont à revoir.

Dernier point, je n'ai pas réussi à trouver comment actionner la pince ce qui est dommage pour un robot qui est censé manipuler des objets mais mon objectif était avant tout de toucher l'objet puisque cela montre le bon positionnement de la pince.