SCIENCES SORBONNE UNIVERSITÉ

Master SAR 1ère année

2024 - 2025

Rapport de Projet de Modélisation :

Conception, Modélisation et Commande d'un robot parallèle 3RRR

Auteurs :

Baptiste Braun-Delvoye
Julien Sagnes

Professeur :

Faiz Ben-Amar

16 mai 2025

Table des matières

Introduction 2
I Conception 2
I.1 Bras du robot 2
I.2 Effecteur 3
I.3 Moteur 3
I.4 Assemblage 4
I.5 Singularités 4
II Simulation 5
III Réalisation 6
Conclusion 6
Bibliographie 7

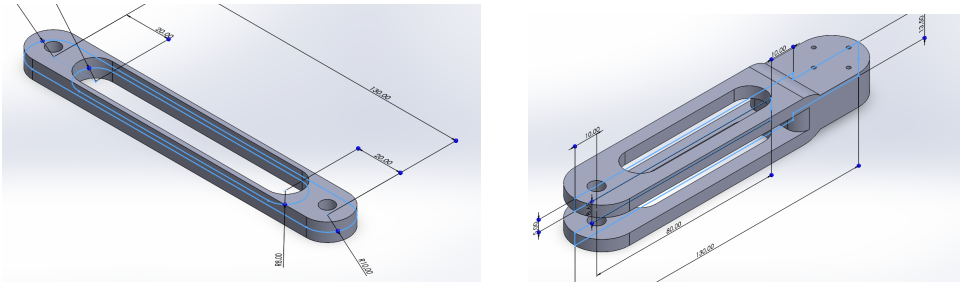
Introduction

I Conception

Pour la conception, nous utilisons le logiciel SolidWorks. Nous avons décidé de suivre le modèle du TP pour dessiner les différentes pièces, en s'assurant de garder des distances de l'ordre de la dizaine de centimètre ; nous nous assurons de ne pas concevoir un robot trop grand.

I.1 Bras du robot

Dans un premier temps, nous concevons les bras qui composent le robot pour décider de la taille générale de la maquette. Également, il est question de minimiser la quantité de matériaux utilisée pour les différentes pièces dans l'objectif de réduire le coût d'impression 3D. Nous opérons donc une optimisation topologique sur les différents bras en effectuant des allègements, comme nous pouvons le voir sur la figure I.1, qui détaille aussi les côtes des pièces en millimètres. Du reste, nous utilisons des congés pour adoucir les bords des pièces afin de rendre le robot plus harmonieux.



(a) Bras effecteur (b) Bras moteur

FIGURE I.1 – Allègement des bras du robot

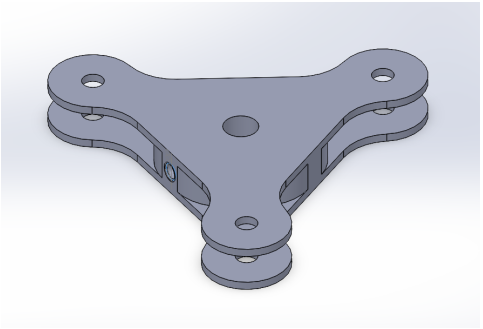


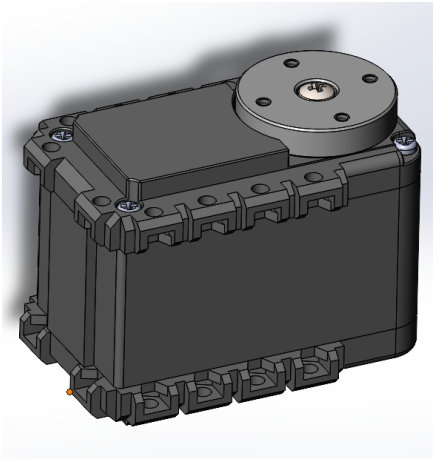
FIGURE I.2 – Effecteur

I.2 Effecteur

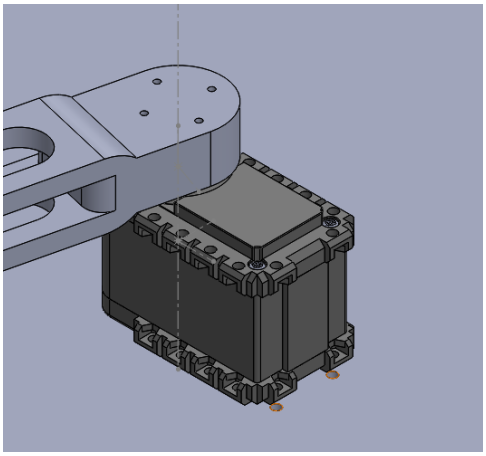
Pour l'effecteur, nous optons pour une forme triangulaire, dont les côtés sont agencés de telle sorte qu'ils puissent auceuillir les boulons. En effet, les trous que nous faisons sur les pièces (liaison pivot entre les bras et l'effecteur) ont pour but d'acceuillir des boulons M6 (diamètre du trou 6.3 mm), auquel on rajoute un jeu de 0.15 mm. Dans un but de maximisation de l'espace de travail, nous effectuons des extrusions sur le bord de la pièce pour éviter les blocages et assurer une rotation libre, comme nous pouvons le voir sur la figure I.2. Nous ajoutons également un trou au centre pour acceuillir un stylo que l'on serre avec une vis M5.

I.3 Moteur

Quant aux moteurs, nous avons importé le cerveau moteur de type AX12 de Dynamixel que nous fixerons au bras grâce à quatre vis M2, comme nous pouvons le voir sur la figure I.3.

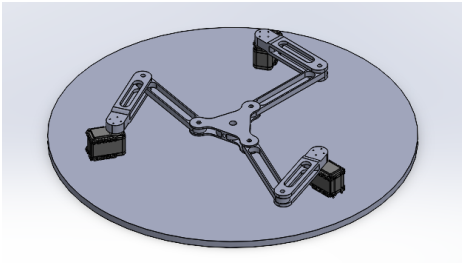


(a) Moteur assemblé

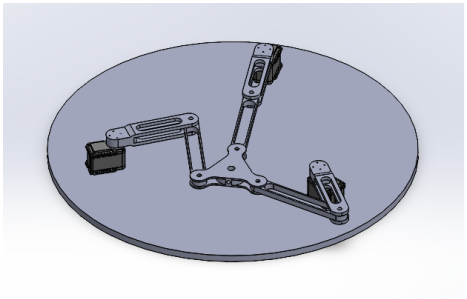


(b) Moteur accroché

FIGURE I.3 – Moteur AX12



(a) Assemblage du robot



(b) Assemblage après mouvement

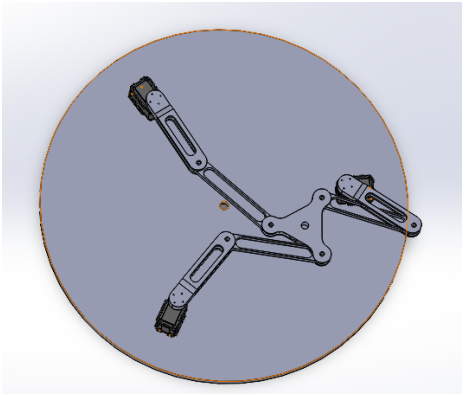
FIGURE I.4 – Assemblage sur SolidWorks

I.4 Assemblage

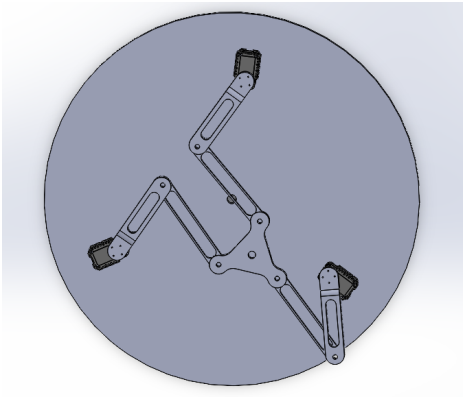
Enfin, nous effectuons l'assemblage du robot en utilisant des contraintes standards de coaxialité et de coïncidence pour les vis, et des liaisons pivots entre les bras et l'effecteur. Nous accrochons ensuite les moteurs à une base fixe pour que ces derniers restent immobiles, comme illustré sur la figure I.4.

I.5 Singularités

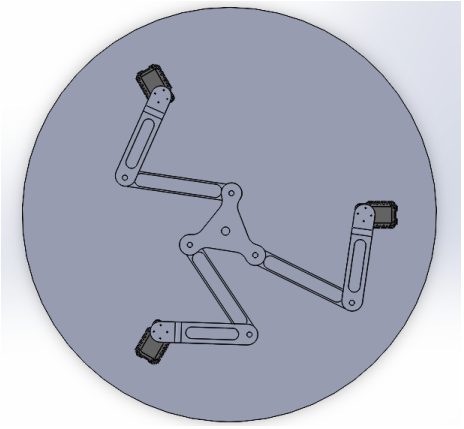
On propose 3 captures d'écrans des 3 différentes singularités sur la figure I.5.



(a) Singularité série



(b) Singularité parallèle (droites parallèles)



(c) Singularité parallèle (droites concourantes)

FIGURE I.5 – Singularite série et parallèles

II Simulation

Afin de simuler le robot, nous avions accès à des codes en MatLab. Ces codes permettent de calculer le modèle géométrique inverse de façon analytique et numérique, ainsi qu'une fonction permettant de tracer la configuration de l'appareil.

Notre première étape fut de transformer ces codes sous Python afin de les utiliser dans notre simulation. Pour celle-ci, nous avons utilisé la bibliothèque PyGame, qui permet d'afficher des éléments graphiques avec de l'animation. Par simplicité, nous avons créé une classe Robot3RRR pour tous les calculs nécessaire à la simulation.

Notre simulation permet de controller l'effecteur via les touches de notre clavier, mais aussi de dessiner des formes prédéfinis :

— un carré

— un cercle

— un polynôme quelconque

— un trèfle

Notre méthode *interpolate_path* prend les points de la figure (par exemple les quatres points du carré) et fait une interpolation entre ces points afin que le robot suive la bonne courbe.

Lors du calcul de la position, nous prenons en compte les singularités possibles du robot. Si nous sommes dans le cas de la singularité série, le robot se bloque dans la direction limité uniquement. Pour les singularités parallèles nous calculons la valeur de l'angle de rotation de l'effecteur permettant d'avoir le déterminant de la matrice A le plus grand entre ± 1 degrés de l'angle actuel.

Nous avions initialement voulu calculé pour chaque pas de temps la position de l'effecteur donnant le déterminant le plus grand, mais cela prenait un temps considérable en calcul et nous avons donc préférer opter pour une option plus rapide.

TODO - ajouter schéma + image + prendre en vidéo

Pour tracer les différentes figures, nous avons une fonction pour chacune d'elles permettant de crée les points que l'effecteur devra passer par. Ensuite nous utilisons une fonction d'interpolation de point qui va construire la ligne que le robot devra suivre entre 2 points

TODO - expliquer comment fonctionne la méthode interpolate_path (chaque fonction "trace" crée les points, et la fonction interpolate_path crée des points entre chaque pour interpoler la trajectoire avec n_steps points)

Afin d'éviter les singularités parallèles nous modifions la rotation de l'effecteur. Pour faire cela, nous déterminons la matrice A de l'énoncé, et nous calculons son déterminant. De par la limitation de la simulation, les coudes des 3 bras seront toujours dans le même "sens", ainsi nous évitons naturellement le cas de singularité parallèle où les avant-bras sont parallèles. Nous devons donc uniquement regardé si le déterminant de la matrice est proche de 0. Dans tous les cas nous recherchons à éloigner le plus possible le robot d'une configuration de droites concourantes en évitant les collisions.

TODO - Bien expliquer la façon dont on optimise l'orientation de θ_e avec la méthode (optimize_orientation). L'idée étant de s'éloigner le plus possible des configurations de droites concourantes, en évitant les collisions (avec la méthode check_collision et segment_distance).

III Réalisation

Conclusion

Bibliographie

[1] O. Hamdoun, « Inverse Kinematic Modeling of 3RRR Parallel Robot », *Congrès français de mécanique*, Congrès Français de Mécanique, A. F. de Mécanique, éd., août 2015. (visité le 20/04/2025).