TD : Modèles géométriques

Introduction

Ce TD contient deux fichiers : `simulation.py` qui est une interface permettant d'exécuter différentes parties du code facilement et `control.py` dans lequel vous allez écrire la majeure partie de votre code.

Vous allez devoir implémenter certains éléments spécifiques aux robots dans les classes `RTRobot` et `RRRRobot`.

Dans tous les cas, il est nécessaire de définir la cible avec l'option `--target` et le mode de contrôle parmi les suivants :

- `--joint-space` : La cible est donnée dans l'espace articulaire du robot
- `--jacobian-inverse` : La cible est donnée dans l'espace opérationnel, la méthode utilisant l'inverse de la jacobienne est utilisée pour déterminer quels positions utiliser.
- `--jacobian-transposed` : La cible est donnée dans l'espace opérationnel, la méthode utilisant la transposée de la jacobienne est utilisée pour déterminer quels positions utiliser.
- `--analytical-mgi` : La cible est donnée dans l'espace opérationnel, les positions d'outil à utiliser sont calculées à partir d'un modèle analytique.

Attention, la plupart des fonctions à implémenter ne sont pas trivial et il est donc important que vous écriviez vos propres tests pour vous assurez de votre implémentation plutôt que de lancer à chaque fois la simulation.

Robot RT

Vous pouvez choisir le robot `RT` avec l'option `--robot rt`, le code de son modèle est à écrire dans la classe `RTRobot`

Ce robot comporte un joint rotoïde suivi d'un joint prismatique. Son espace opérationnel est de dimension 2 car l'outil peut être déplacer dans un plan, et pas dans un espace.

La première étape est d'obtenir un MGD satisfaisant. Pour ce faire, implémenter la méthode `computeMGD` en extrayant les données géométriques depuis le modèle `URDF` qui est fournit.

Vous pouvez tester votre implémentation en spécifiant la configuration cible dans l'espace articulaire : `./simulation --robot rt --target Q0 Q1 --jointSpace`

Si votre méthode est correctement implémentée, la croix noire dénotant la position devrait systématiquement apparaître au bout de l'outil (bleu-ciel).

Ensuite vous pouvez implémenter la méthode `computeMGI` et tester son fonctionnement en spécifiant différentes positions cibles.

Afin d'utiliser les méthodes se basant sur la jacobienne, il faudra implémenter `computeJacobian`, mais aussi les fonctions `searchJacInv` et `searchJacTransposed`. Vous pourrez vous aider de la fonction `minimize` de `scipy` pour `searchJacTransposed` [1].

1. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html`

Robot RRR

Vous pouvez choisir le robot `RRR` avec l'option `--robot rrr`, le code de son modèle est à écrire dans la classe `RRRRobot`

Ce robot possède trois articulations rotoïdes et l'objectif est de contrôler la position de l'outil dans l'espace.

Écrivez les trois fonctions `computeMGD`, `computeMGI` et `computeJacobian` et testez les comme pour le robot `RT`.

Le contenu des fonctions `searchJacInv` et `searchJacTransposed` devrait être compatible avec les deux robots.

Évaluation

Les documents seront à déposer sur Moodle dans une archive nommée : `nom_prenom.tar.gz` L'exécution de `tar -xzf nom_prenom.tar.gz` ne devra pas afficher d'erreur.

L'archive devra contenir un dossier `nom_prenom` dans lequel vous mettrez :

— Votre fichier `control.py`
— D'autres fichiers python si votre `control.py` si besoin
— Si vous le jugez pertinent, vous pouvez ajouter un fichier `README` expliquant le travail effectué et les problèmes rencontrés.