Travail Pratique #5

ELE8203 - Robotique

Robots mobiles à roues et asservissement visuel

Département de génie électrique
Polytechnique Montréal

Automne 2023
Travail à remettre après la séance 6 de TP (dates en dernière page)

Ce document couvre les séances de TP 5 et 6. Vous devrez rédiger **un seul rapport** pour ces deux séances (ce rapport compte pour un pourcentage plus important de la note finale que les autres rapports). Toutefois, merci de noter que pour respecter le calendrier académique, votre rapport est dû à une date stricte après la séance 6 fournie en dernière page. Il est donc fortement conseillé de terminer le travail et de rédiger la partie du rapport en lien avec la séance 5 avant le début de séance 6, afin de pouvoir vous consacrer aux questions supplémentaires lors de la séance 6 et pendant la dernière semaine, avec un environnement de simulation et de calcul déjà fonctionnel et maîtrisé.

Présentation de votre rapport : il est conseillé mais pas requis d'écrire votre rapport à l'ordinateur. Si vous l'écrivez à la main, il faudra scanner le document de façon à ce qu'il reste bien lisible. Dans tous les cas, soignez votre présentation, car des points pourront être retirés à un document mal présenté (voir le plan de cours). À la limite, un document illisible où pour lequel la correction demanderait un effort exagéré pourrait se voir attribuer la note 0. Vous devriez tenir compte des conseils généraux suivants.

- Page ou section de présentation : *Le nom et le matricule de chacun des membres de l'équipe (2 étudiants par équipe, sauf exception), le numéro du cours, le numéro de la section et le nom du laboratoire effectué sont correctement indiqués.*
- Qualité des figures et tableaux : *La présentation des figures et des tableaux est claire et soignée. Chaque figure ou tableau est accompagné d'un titre clair et sa présence est justifiée par une discussion y renvoyant dans le texte du rapport. Les différents éléments présentés sur une figure ou dans une image doivent être clairement différentiables.*
- Qualité des développements mathématiques : *Les développements mathématiques sont clairs et complets, simples à comprendre et à suivre.*
- Qualité du texte et des explications : *Le rapport est facile à lire et les idées sont clairement énoncées, dans un français de qualité.*
- Utilisation de variables et symboles : *La signification des variables ou des symboles utilisés ainsi que leurs unités respectives sont clairement indiquées dans le rapport.*
- Professionnalisme : *Le rapport comprend une introduction ainsi qu'une conclusion résumant les objectifs à atteindre et commentant leur réalisation. Dans l'ensemble le rapport fait preuve d'une démarche professionnelle.*

1 Objectifs

Ce TP met en application les notions introduites dans les notes de cours en rapport avec les robots mobiles à roues et l'asservissement visuel. Les objectifs poursuivis sont

- de pouvoir choisir et implémenter des contrôleurs permettant à différents robots mobiles (omnidirectionnels ou non holonomes) d'exécuter des trajectoires désirées ;
- de se familiariser avec une stratégie de planification de trajectoire, la descente du gradient d'un champ de potentiel, afin de permettre à un robot de se déplacer vers une destination en évitant les obstacles ;
- d'être exposé à l'intégration des mesures de capteurs extéroceptifs (lidar, caméra, en simulation et dans des conditions de quasi-absence de bruit et d'outliers) dans la planification et la commande du mouvement d'un robot ;
- de se familiariser un peu plus avec l'utilisation d'outils courants de simulation et de conception pour la robotique (`CoppeliaSim`, `Matlab`).

Ce TP demande de programmer des scripts dans `CoppeliaSim`, dans le langage Lua. Il sera nécessaire de vous reporter à la documentation de l'API.

2 Commande des robots mobiles à roues

2.1 Un robot omnidirectionnel

Nous nous intéressons tout d'abord à la commande d'un robot omnidirectionnel, le youBot, qui était commercialisé jusqu'en 2017 par la compagnie allemande Kuka. Ce robot mobile montré sur la figure 1 est un robot omnidirectionnel possédant quatre roues mecanum avec $\gamma = \pm 45°$. Il est aussi équipé d'un manipulateur à cinq degrés de liberté, dont nous n'aurons pas l'usage dans ce TP.

Le but est d'implémenter un contrôleur permettant au robot de suivre la trajectoire montré sur la figure 1. Un "dummy" appelé `ref_point` se déplace sur le chemin et indique le point à traquer par le centre du repère du chassis du robot, dénoté $\{\mathtt{b}\}$ ici et `youBot_frame` dans la simulation, qui a été déjà été défini dans la scène de départ fournie. Ce repère suit les conventions utilisées en cours : $\boldsymbol{x}_{\mathtt{b}}$ vers l'avant du robot, $\boldsymbol{y}_{\mathtt{b}}$ vers la gauche. De plus, lorsque le robot suit la trajectoire il doit aussi maintenir son cap (défini par la direction de $\boldsymbol{x}_{\mathtt{b}}$) aligné avec une balise nommée `balise_direction` dans la scène. Autrement dit, l'angle de gisement ("bearing angle") de cet balise par rapport au robot doit rester proche de 0.

Vous utiliserez les dimensions du robot prédéfinies dans le script déjà associé au `youBot` dans la scène.

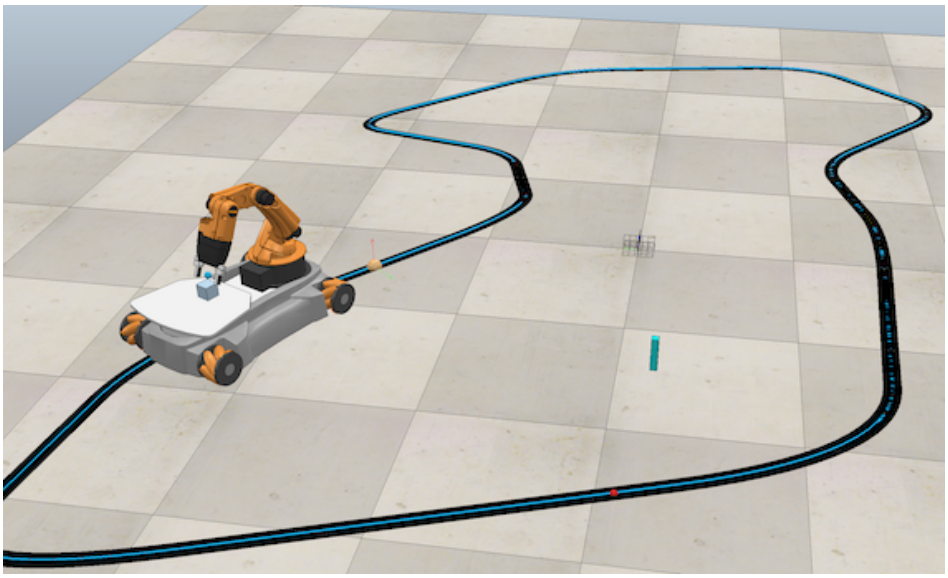

FIGURE 1 – Robot omnidirectionnel youBot et trajectoire à suivre. La balise définissant la direction vers laquelle le robot doit pointer est visible en bleu cyan.

1. [1 point] Dans la scène `CoppeliaSim` fournie, implémentez la fonction `setWheelVelocities` qui prend 3 arguments v_x, v_y, ω, représentant les vitesses désirées dans le repère {`b`} du robot, et ajuste la vitesse des roues afin de satisfaire ces consignes de vitesses. Expliquez votre démarche.
2. [2 points] Concevez et implémentez un contrôleur pour le youBot dans la fonction `motionController`, qui effectue la tâche de suivi de trajectoire demandée. Expliquez votre démarche de conception. Incluez un terme de feedforward dans votre commande de suivi du point de référence sur la trajectoire. Il n'y a pas de spécifications de performance stricte à rencontrer, mais vous chercherez à obtenir une erreur de positionnement d'au plus quelques cm et une erreur de direction d'au plus quelques degrés en régime permanent. Un terme intégral dans votre contrôleur n'est pas requis. Vérifiez que les vitesses des roues restent dans des limites physiquement raisonnables le long de la trajectoire (par exemple en ajoutant un graphe indiquant les vitesses angulaires des roues).

   N.B.1 : Dans tout le TP, vos solutions doivent fonctionner indépendamment de la position des balises, de la trajectoire, etc. Autrement dit, l'instructeur pourra repositionner les éléments spécifiant la trajectoire dans votre scène (dans une limite raisonnable) et votre contrôleur devra encore fonctionner correctement.

   N.B.2 : Il est généralement plus simple et plus naturel de travailler en coordonnées dans le repère du robot. Lisez bien la documentation de la fonction `sim.getObjectPosition`.
3. [1 point] Complétez votre scène en ajoutant un graphe montrant la norme de l'erreur de position au cours du temps, ainsi qu'un second graphe montrant l'erreur de direction au cours du temps (en degrés).
4. [1 point] (bonus, cette question vous demande d'interfacer `CoppeliaSim` et `Matlab`) En partant du code fourni, utilisez `Matlab` pour enregistrer dans un tableau les erreurs de position (erreur en x et y dans le repère du robot) au cours du temps et calculez la recine de l'erreur quadratique moyenne obtenue avec votre contrôleur

$$\epsilon = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{p}_i - \boldsymbol{p}_{d,i}\|^2},$$

   où i indique l'échantillon numéro i et N le nombre total d'échantillons. Une valeur suggérée pour N est par exemple 10^5. Il faudra typiquement sous-échantillonner le long de la trajectoire (ne pas enregistrer toutes les valeurs reçues) afin de limiter la taille du tableau tout en couvrant au moins un tour entier de la trajectoire. Tracez sur deux graphes dans Matlab les erreurs en x et y au cours du temps et incluez ces graphes dans votre rapport.

2.2 Un robot à entraînement différentiel

On considère maintenant un robot à entraînement différentiel, le Pioneer 3-DX, réalisant un modèle cinématique de monocycle et montré sur la figure 2. Le but est d'implémenter un contrôleur de suivi de trajectoire cartésienne, ainsi qu'un contrôleur de commande point-à-point.

FIGURE 2 – Robot à entraînement différentiel Pioneer 3-DX et 3 points destinations intermédiaires.

2.2.1 Suivi d'une trajectoire cartésienne

Comme à la section 2.1, nous voulons d'abord implémenter un contrôleur permettant au robot de suivre une trajectoire quelconque, spécifiée de la même façon par un dummy se déplaçant sur un chemin. Il n'y a pas de spécification ici sur le cap θ du robot, le but est uniquement de suivre une trajectoire cartésienne.

5. [1 point] Dans la scène `CoppeliaSim` fournie, implémentez la fonction `setWheelVelocities` qui prend 2 arguments v_x et ω représentant les vitesses désirées dans le repère {b} du robot, et ajuste la vitesse des roues afin de satisfaire ces consignes de vitesses. Ce repère {b} est défini comme dans les notes de cours par le repère `Pioneer_frame`, avec son centre B au milieu de l'axe des roues, $\boldsymbol{x}_{\mathsf{b}}$ vers l'avant et $\boldsymbol{y}_{\mathsf{b}}$ vers la gauche. Les dimensions du robot sont fournies dans le script. Expliquez votre démarche.
6. [2 points] Concevez et implémentez un contrôleur pour le robot qui effectue la tâche de suivi de trajectoire demandée, en utilisant la méthode de commande de la vitesse d'un point P attaché au repère {b} (section 12.2.1 des notes de cours). Ajoutez à votre scène un graphe montrant la norme de l'erreur de position au cours du temps entre *le point choisi* P et le point de référence, ainsi qu'un graphe montrant la vitesse des roues (en rpm). Expliquez dans votre rapport comment vous avez généré le graphe d'erreur de position. Assurez-vous que les vitesses des roues restent dans des plages physiquement raisonnables le long de la trajectoire. La précision anticipée est de l'ordre de quelques centimètres.
7. [1 point] Commentez sur la performance de votre contrôleur, et sur les effets observés en changeant ses paramètres, notamment la position du point P, le gain, le comportement sur certaines parties spécifiques de la trajectoire, etc. Vous pouvez, si vous le souhaitez, tester votre contrôleur sur des trajectoires que vous aurez conçues vous-même. Pour cela, la vidéo disponible ici offre un tutoriel sur comment créer une trajectoire de référence dans `CoppeliaSim`.

2.2.2 Régulation de position

Finalement, nous considérons un scénario où le robot doit visiter des points de référence successifs. De nouveau, il n'y a pas de spécification sur le cap du robot. Vous devrez pour cela implémenter le contrôleur de la section 12.2.3 des notes de cours, qui régule simplement la position du robot vers un point fixe de référence, sans planification de trajectoire.

8. [3 points] Dans la scène `CoppeliaSim` fournie, implémentez un contrôleur qui permet au robot de visiter les 3 points de référence à tour de rôle, dans l'ordre 1, 2, 3, 1, 2, 3, 1, etc. Le mouvement devra se poursuivre indéfiniment. Ajoutez un graphe montrant la norme de l'erreur de position au cours du temps (distance à la destination courante). Expliquez votre démarche et commentez sur le comportement de votre contrôleur.

3 Champs de potentiel pour la planification de trajectoire

Dans cette section, vous devez implémenter un algorithme utilisant les champs de potentiels pour guider le robot non holonome de la section 2.2 jusqu'à un point destination (appelé `Destination_point` dans la scène CoppeliaSim), tout en évitant les obstacles sur son chemin. La scène de départ est présentée sur la figure 3. Le robot est muni d'un lidar, qui permet d'obtenir une liste de coordonnées de points détectés $P_i^{\mathsf{r}} = [x_i, y_i]^T$ sur les obstacles autour du robot, exprimées dans le repère du lidar. Cependant le lidar a été positionné de telle façon à ce que son repère coïncide exactement avec le repère `Pioneer_frame` du robot, ainsi vous pouvez considérer ces coordonnées comme étant exprimées dans le repère {r} du robot.

9. [2 points] Implémentez et testez dans une scène sans obstacle un planificateur/contrôleur utilisant une force attractive vers la destination pour attirer le robot, en vous basant sur le potentiel hybride quadratique/conique vu en cours. Vous devez implémenter un algorithme de descente de gradient pour spécifier à chaque itération un nouveau point de référence dans la direction appropriée pour le contrôleur cinématique de la section 2.2.1. Réglez vos paramètres pour obtenir des vitesses commandées raisonnables. Discutez votre démarche dans votre rapport et remettez votre scène `CoppeliaSim` illustrant vos résultats.
10. [3 points] Ajoutez maintenant des forces répulsives qui éloignent le robot des obstacles. Chaque point P_i détecté par le lidar exerce une telle force répulsive sur le point commandé par le contrôleur de la section 2.2.1 (il n'est pas demandé d'ajouter des forces répulsives pour d'autres points du châssis du robot). Sommez toutes les forces agissant sur ce point et implémentez un algorithme de descente de gradient, en ajustant vos paramètres pour

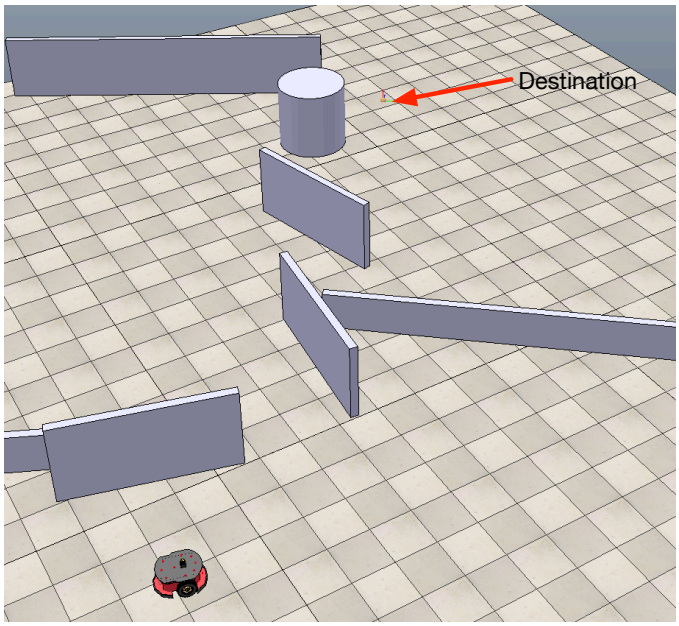

FIGURE 3 – Problème de planification de trajectoire.

permettre au robot de se rendre à sa destination sans toucher d'obstacle et à une vitesse raisonnable. Un exemple de simulation est montré ici. Discutez votre démarche dans votre rapport et remettez votre scène CoppeliaSim illustrant vos résultats.

4 Asservissement visuel basé sur les images (IBVS)

Dans cette section, vous devez compléter la simulation d'un scénario d'asservissement visuel par la méthode IBVS sous Matlab. Un fichier `sim_IBVS.m` vous est fourni, qui utilise quelques fonctions de base de la `Machine Vision Toolbox` pour Matlab. Cette toolbox, disponible ici, est normalement déjà installée sur les machines du laboratoire. Si vous l'installez sur votre machine personnelle, il est conseillé d'installer d'abord la `Robotics Toolbox`. Les questions préliminaires ci-dessous contiennent des rappels utiles pour développer votre simulation.

11. [½ point] Soit un repère $\{\mathsf{c}\}$ en mouvement par rapport à un repère $\{\mathsf{s}\}$. On connaît la représentation $\boldsymbol{\mathcal{V}}^{\mathsf{c}}_{\mathsf{c/s}}$ du torseur cinématique du mouvement de c par rapport à s. Rappelez l'équation différentielle liant la pose $T^{\mathsf{s}}_{\mathsf{c}}$ à $\boldsymbol{\mathcal{V}}^{\mathsf{c}}_{\mathsf{c/s}}$. Si $T^{\mathsf{s}}_{\mathsf{c}}(t)$ est connue à un instant t, et que $\boldsymbol{\mathcal{V}}^{\mathsf{c}}_{\mathsf{c/s}}$ est considéré constant sur l'intervalle $[t, t+dt]$, expliquez comment calculer le plus précisément possible la pose $T^{\mathsf{s}}_{\mathsf{c}}(t+dt)$.
12. [½ point] On considère le repère caméra de la figure 14.1 des notes de cours. Si on connait les coordonnées en pixels (u, v) d'un point sur le plan image, ainsi que les paramètres intrinsèques de la caméra, expliquez comment calculer les coordonnées (x, y) de ce point (coordonnées standard dans le repère caméra, sur le plan image) à partir de (u, v) par une simple opération matricielle.
13. [2 points] En partant du code `Matlab` fourni, implémentez la loi de commande IBVS. Testez votre implémentation pour une pose de départ de la caméra par rapport au repère s égale à $T^{\mathsf{s}}_{\mathsf{c}} = I_4$ et une pose finale T_d où la caméra est encore au centre O du repère s, mais son attitude est donnée par la matrice de rotation

$$R^{\mathsf{s}}_{\mathsf{c}} = R_{\boldsymbol{z},\psi} R_{\boldsymbol{y},\theta} R_{\boldsymbol{x},\phi}, \text{ avec } \psi = -20°, \theta = 10°, \phi = 5°.$$

Notez que la connaissance complète de T_d n'est en fait pas nécessaire. À la place, il suffit que l'on vous donne comme dans le fichier l'image pour la caméra dans la pose finale de 4 points immobiles situés initialement sur le plan $z = 1$. La loi de commande IBVS n'a besoin que de cette information pour définir sa pose désirée.

Réglez votre contrôleur de manière à obtenir une convergence vers la pose désirée en quelques secondes, en utilisant les images des 4 points obtenues au cours du temps. Tracez sur 3 graphes l'évolution temporelle des quantités suivantes :

— Les coordonnées x, y, z dans `s` du centre de la caméra au cours du temps (1 graphe avec ces 3 courbes).
— Les erreurs sur les angles ϕ, θ, ψ, en degrés, au cours du temps (1 graphe avec ces 3 courbes).
— L'erreur image totale normalisée $\|\boldsymbol{e}(t)\|/\|\boldsymbol{e}(0)\|$, avec

$$\boldsymbol{e} = \boldsymbol{p}_d - \boldsymbol{p},$$

comme dans les notes de cours, c'est-à-dire que $\boldsymbol{p}_d = [x_{1,d}, y_{1,d}, \ldots, x_{4,d}, y_{4,d}]^T$ représente les coordonnées image des points désirées, tandis que $\boldsymbol{p}(t)$ représente les coordonnées image de la projection des points à l'instant t.

Expliquez brièvement votre démarche.

14. [1 point] Commentez sur le comportement observé du contrôleur, ses qualités comme ses défauts. Vous pouvez aussi tester le comportement avec des poses finales différentes supplémentaires et commenter sur ce que vous avez pu observer.

5 Travail à remettre

Votre rapport final sera à remettre après la séance du TP6 selon les dates ci-dessous, sur Gradescope. Il doit décrire brièvement votre démarche pour chacune des questions ci-dessus. Vous pouvez appuyer votre rapport par des captures d'écran et figures : dans ce cas, chaque figure doit être discutée dans le texte et sa présence justifiée. De plus, vous devrez fournir vos fichiers `CoppeliaSim` et `Matlab` dans l'espace prévu à cet effet dans Moodle. Vos scènes `CoppeliaSim` doivent être directement executables par votre instructeur et doivent montrer les graphes demandés. Vous êtes libres aussi d'ajouter des éléments pertinents appuyant vos explications, en les mentionnant dans votre rapport.

Dates de remise de votre rapport et fichiers :
— section 2 : **mercredi** 6 décembre à 23h55
— section 3 : **vendredi** 8 décembre à 23h55
— section 1 : **mercredi** 13 décembre à 23h55