Étude du Conundrum de Chaumette

Objectif de la séance

En asservissement visuel 2D, la tâche est exprimée dans l'image. Cette spécificité qui donne tout son attrait à ce type de commande peut engendrer des comportements 3D non optimaux voire irréalisables par l'effecteur du robot. L'exemple qui illustre le plus ce propos est le « retrait / avance » de l'effecteur quand la tâche consiste à réaliser une rotation pure dans l'image. L'objectif de la séance est de mettre en évidence ce problème et d'implémenter une loi de commande qui permet de le corriger[1].

Environnement de travail

Le développement se fera sous python et fera appel aux fonctions de la toolbox de P. Corke[2]. La documentation de cette toolbox est pour le moment très parcellaire car la toolbox est encore à ses premières versions.

Nous allons considérer un bras robotique anthropomorphe à 6 degrés de liberté. Une caméra est montée sur l'effecteur du robot. Le repère de la caméra est confondu avec celui de l'organe terminal du robot.

Une cible plane constituée de 4 points est utilisée pour réaliser l'asservissement visuel. Cette cible est parallèle au plan image lorsque le robot est dans sa position initiale.

Les axes du robot simulé sont asservis en vitesse. La sortie de la loi de commande par vision sera donc un vecteur de consignes de vitesses articulaires.

Travail à effectuer

Lecture de l'article

1. Lire attentivement l'article qui vous est fourni.

2. Expliquer pourquoi la matrice d'interaction (équation numéro 2), bien que différente de celle que nous avons démontré pendant la séance de cours, est correcte.

Asservissement visuel 2D classique

[1] P.I. Corke and S.A. Hutchinson. A new partionned Approach to Image-Based Visual Servo Control. IEEE Transactions on Robotics and Automation. Vol. 17(4). Pages 507-515. 2001.
[2] P.I. Corke : https://github.com/petercorke/robotics-toolbox-python/wiki

1. Dans le fichier « main_etu.py ». Implémenter l'asservissement visuel 2D classique. Pour cela il faudra :

    a. Lire attentivement le code écrit dans asservissement visuel.
    b. Compléter le code qui vous est demandé
    c. Ne pas oublier de renseigner la fonction matrix_interaction se trouvant dans « lab_tools_etu.py »

2. Le code comprend une série de de jeux de positions images désirées en pixels. Afin de tester le bon fonctionnement de votre code, procéder à l'essai des 3 premiers jeux de données. Commenter la forme de la trajectoire des points dans l'image.

3. La suite des jeux contient des positions images désirées qui correspondent à une rotation pure de l'effecteur.

4. Commenter le comportement lorsque la rotation est de 180 degés.

Asservissement visuel 2D « partitionné »

Pour corriger le comportement observé plus haut, une loi de commande partitionnée a été proposée par Corke et al. Cette loi de commande consiste à modifier les primitives visuelles qui permettent de contrôler la translation et la rotation autour de l'axe optique.

Le vecteur de commande $\begin{pmatrix} {}^{c}\widetilde{V}_{c/o}^{C} \\ {}^{c}\widetilde{\Omega}_{c/o} \end{pmatrix}$ s'écrit de la façon suivante :

$$\begin{pmatrix} \widetilde{\omega}_z \\ \widetilde{v}_z \end{pmatrix} = \begin{pmatrix} \gamma_{\omega_z} & 0 \\ 0 & \gamma_{T_z} \end{pmatrix} \begin{pmatrix} \theta^* - \theta \\ \sigma^* - \sigma \end{pmatrix}$$

Et

$$\begin{pmatrix} \widetilde{v}_x \\ \widetilde{v}_y \\ \widetilde{\omega}_x \\ \widetilde{\omega}_y \end{pmatrix} = \gamma L_{s_{xy}}^{+} \left((s^* - s) - L_{s_z} \begin{pmatrix} \widetilde{v}_z \\ \widetilde{\omega}_y \end{pmatrix} \right)$$

où :

θ est un angle défini entre deux points de l'image et l'axe des abscisses.

θ^* est l'angle désiré.

σ est la racine carré de la surface du polygone formé par les points de l'image.

σ^* est la valeur désirée.

L_{s_z} est constituée des colonnes 3 et 6 de la matrice d'interaction.

$L_{s_{xy}}$ est constituée des colonnes 1, 2, 4 et 5 de la matrice d'interaction.

$\gamma, \gamma_{\omega_z} \text{et } \gamma_{T_z}$ sont des gains.

1. Implémenter cette loi de commande dans le fichier « main_etu_part.py » Le calcul de l'angle ainsi que celui de la racine carrée de la surface sont déjà codés.

2. Tester votre loi de commande pour une rotation de 90 et puis 180 degrés. Commenter.