Projet INTEG
Option Robotique
Ecole Centrale Nantes

Dernière mise à jour :*9 février 2025*

Juan Sandoval Andrea Cherubini

Table des matières

Introduction

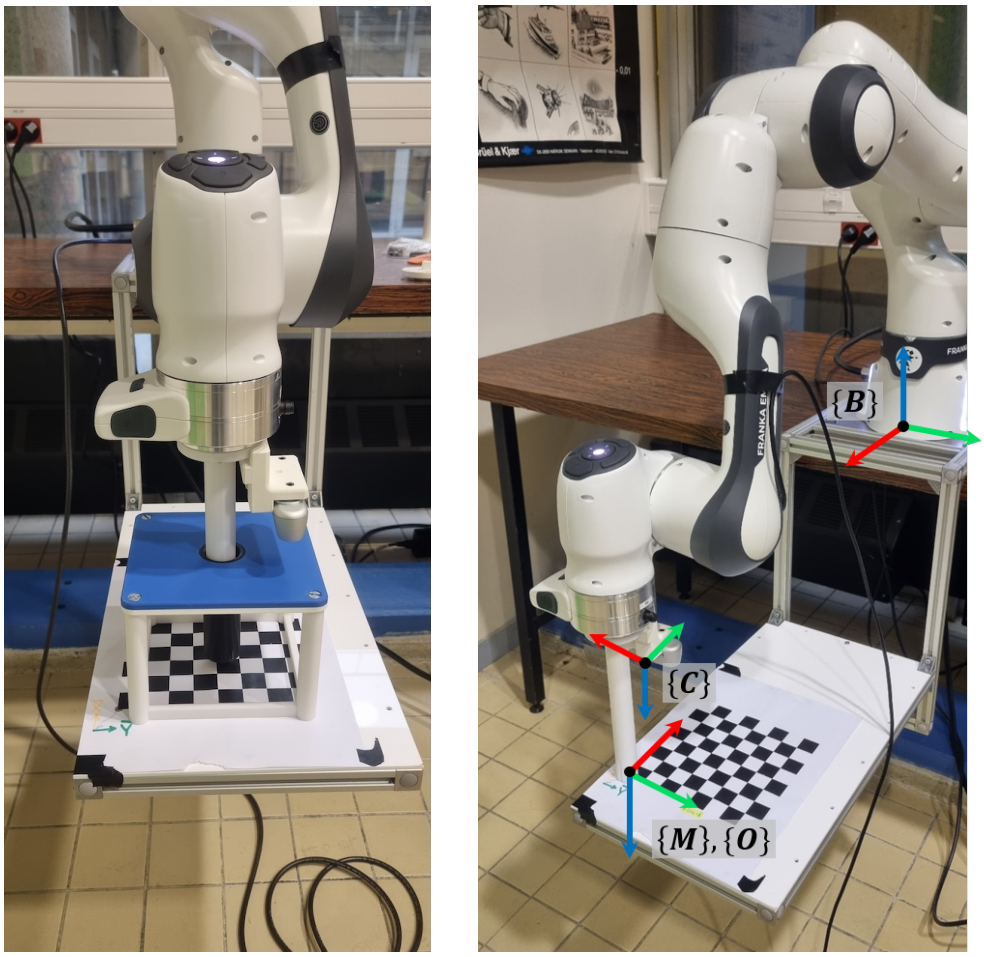
L'objectif de ce projet est de vous familiariser avec les notions de contrôle référencé capteur, pour un robot manipulateur. En particulier, il faudra réaliser une tâche d'insertion (*peg-in-hole*), en utilisant une loi de commande hybride (vision/force). Vous travaillerez avec le robot Franka Emika FR3, équipé d'une caméra Intel Realsense D435 sur son effecteur. Le robot est aussi équipé de capteurs de couple sur chacune de ses articulations, capteurs permettant de reconstruire le torseur dynamique (trois composants de force et trois composants de couple) à son outil.

La tâche à réaliser est montrée en Fig. 1a, et les repères utilisés dans le projet sont détaillés en Fig. 1b.

Le projet se compose des étapes suivantes :

1. vous allez développer un nœud sous ROS-1 pour récupérer les données de positionnement du robot ainsi que les images de la caméra ;
2. vous allez comanipuler le robot, afin d'acquérir une base de photos de la mire d'étalonnage et du trou d'insertion. Ces bases de données vous serviront plus tard pour les étapes d'étalonnage de la caméra et pour développer l'algorithme de détection du trou ;
3. vous étalonnerez la caméra afin d'obtenir ses paramètres intrinsèques et extrinsèques ;
4. vous estimerez la pose relative entre la caméra et l'outil ;
5. vous écrirez et validerez un algorithme de traitement d'images permettant de reconnaître le trou d'insertion dans les images acquises par la caméra ;
6. vous développerez une loi de commande hybride vision/force permettant au robot de réaliser la tâche d'insertion.

Ces étapes constitueront les paliers notés du projet. À chaque fois que vous atteignez un des paliers, appelez l'enseignant et montrez-lui le résultat obtenu, en expliquant comment vous avez procédé. Vous pouvez (et êtes fortement encouragé.es à) avancer en dehors des séances présentielles.



(a) (b)

FIGURE 1 – (a) Tâche à réaliser : insertion de l'outil dans le trou (*peg-in-hole*), indépendamment de la pose du trou dans l'espace de travail. (b) Repères base {B}, outil {O}, mire {M} et caméra {C}, avec convention RGB pour XYZ. A noter que, seulement pour cette configuration particulière, {M} et {O} coïncident.

1 Récupération des photos et des données du robot

Développer un nœud ROS-1 en Python pour souscrire à :

— la pose cartésienne de l'outil du robot, représentée par ${}^{B}\mathbf{T}_{O}$ (topic : `/franka_states_controller/O_T_EE`) ;
— les images de la caméra montée sur l'outil du robot (topic : `/camera/color/image_raw`) ;
— les touches du clavier ou une interface GUI pour déclencher deux actions.

Vous pouvez télécharger, compiler puis lancer un brouillon du nœud en exécutant les commandes suivantes à partir d'un nouveau terminal :

```
ros_reset
ros1ws
cd ros
git clone https://github.com/jssandovala/franka_integ_node.git
catkin_make
source devel/setup.bash
rosrun franka_integ_dev franka_integ_node.py
```

Deux fonctionnalités devront alors être développées dans le nœud `integ_vision_node` :

1. *Pour l'étalonnage de la caméra :*
   À chaque déclenchement de la première action (lorsqu'une touche désignée du clavier est enfoncée ou lorsqu'un bouton de la GUI est pressé), une photo au format `.jpg` est enregistrée simultanément avec la pose cartésienne de l'outil dans le repère {B} du robot, notée ${}^{B}\mathbf{T}_{O}$. Cette pose est sauvegardée sous forme d'une ligne dans un fichier texte (`.txt` ou `.csv`). **Attention**, il est essentiel que chaque paire photo/pose soit facilement identifiable.
2. *Pour l'algorithme de détection du trou :*
   Une deuxième fonctionnalité à ajouter permettra d'acquérir en continu le flux d'images capturées par la caméra. L'enregistrement pourra être déclenché par une autre touche du clavier ou par un bouton de la GUI. Contrairement à la première fonctionnalité, il ne sera pas nécessaire d'enregistrer la pose de l'outil.

1.1 Vérification en simulation

Afin de vérifier le fonctionnement du nœud développé, suivez ces étapes pour mettre en route un simulateur de la manip sur Gazebo :

- Seulement si vous travaillez dans une machine virtuelle, ouvrez un terminal et installez les paquets nécessaires en lançant :
  ```
  ./latest_patch.sh -u integ
  ```
- dans un nouveau terminal, copiez et compilez le package suivant dans votre workspace ROS :
  ```
  ros_reset
  ros1ws
  cd ros
  git clone https://github.com/jssandovala/franka_integ_sim.git
  echo "export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:$(rospack find
  franka_integ_sim)/models" | tee -a ~/.bashrc
  source ~/.bashrc
  ```
- lancez le simulateur :
  ```
  roslaunch franka_integ_sim cart_imp_integ_sim.launch
  ```
- Déplacez le robot et le trou à l'aide des marqueurs interactifs sur Rviz. Testez ensuite votre nœud pour les deux fonctionnalités demandées. Enfin, faites valider votre nœud par votre enseignant avant de passer à l'étape suivante.

2 Comanipulation pour acquérir les photos

Accompagné de votre enseignant, réalisez les acquisitions sur le robot réel, à savoir :

- les photos de la mire + les poses cartésiennes ${}^{B}\mathbf{T}_{O}$ associées ;
- une séquence continue de photos du trou, pendant que le robot s'en approche, et ce jusqu'à l'insertion de l'outil dans le trou.

Les étapes sont détaillées ci-dessous.

2.1 Photos de la mire

Pour contrôler le robot à l'aide de la vision, il est d'abord nécessaire d'étalonner le système robot/caméra. Autrement dit, il faut déterminer la pose géométrique de la caméra par rapport à la chaîne cinématique du robot. Plus précisément, l'objectif est d'identifier la pose de la caméra dans le repère de l'outil, notée ${}^{O}\mathbf{T}_{C}$.

Pour ce faire, vous devrez acquérir n photos d'une mire en damier, avec n compris entre 20 et 30 (chaque photo étant notée $\mathbf{I}_i$, $i = 1, \ldots, n$). L'uti-

lisation d'OpenCV en Python est recommandée, en vous appuyant sur des tutoriels tels que :

`https://learnopencv.com/camera-calibration-using-opencv/`
`https://nikatsanka.github.io/camera-calibration-using-opencv-and-python.html`

Ces ressources illustrent le type d'images nécessaires pour effectuer l'étalonnage.

Avant de procéder à l'étalonnage, il est nécessaire d'acquérir ces photos ainsi que, pour chacune d'elles, la pose correspondante de l'outil du robot, notée ${}^{B}\mathbf{T}_{O,i}$, avec $i = 1, \ldots, n$.

De plus, il faudra déterminer la pose de la mire dans le repère de la base du robot, notée ${}^{B}\mathbf{T}_{M}$, en positionnant l'outil sur la mire, comme illustré à la Fig. 1b.

Pour acquérir les photos et les poses, vous pourrez comanipuler le robot Franka Emika FR3 afin de le placer dans les configurations souhaitées, puis utiliser le code préparé en Sec. 1 pour réaliser les acquisitions.

L'acquisition pourra être réalisée depuis vos machines en suivant ces étapes :

- Fermez tous les terminaux actuellement ouverts ;
- Ouvrez un nouveau terminal et exécutez les commandes suivantes :
  `ros_reset`
  `ros_franka`
  `ros1ws`
- Lancez votre nœud afin d'être prêt à réaliser les enregistrements.

2.2 Mise en mode compensation gravitationnelle

En parallèle, pour comanipuler le robot, il devra être mis en mode compensation gravitationnelle. **Accompagné de votre enseignant**, procédez comme suit :

- Allumez le robot et son PC (**usr** : `franka`, **mdp** : `franka`) ;
- Ouvrez un navigateur et accédez à l'interface web du robot : `https://172.16.0.30` ;
- Déverrouillez les freins articulaires du moteur, puis passez le robot en mode "programmation" (les voyants de sa base doivent être blancs) ;
- Comanipulez le robot en appuyant légèrement sur les deux boutons situés sur son effecteur. Placez-le au centre de son espace de travail ;
- Ouvrez un terminal et exécutez la commande suivante pour placer le robot en position initiale :

`roslaunch franka_integ_controllers move_to_start.launch \`
`robot_ip:=172.16.0.30 load_gripper:=false` ;

— Lancez le mode de commande en impédance ainsi que le nœud de la caméra :
`roslaunch franka_integ_controllers cart_imp_integ_follow.launch \ robot_ip:=172.16.0.30 load_gripper:=false` ;

— Réduisez à zéro les coefficients de raideur dans la fenêtre *rqt_ gui* afin d'annuler la loi d'impédance et de n'activer que la compensation gravitationnelle ;

— Positionnez l'outil sur la mire de sorte à faire coincidir les repères {O} et {M} (voir Fig. 1b). Récupérez ensuite la pose ${}^{B}\mathbf{T}_{O} = {}^{B}\mathbf{T}_{M}$;

— Retirez l'outil (cylindre) de l'effecteur du robot ;

— Déplacez le robot aux poses souhaitées et procédez aux enregistrements des photos / poses depuis votre machine.

2.3 Acquisition des photos du trou

Positionnez la pièce trouée sur la plateforme, au-dessus de la mire. La position précise n'a pas d'importance, tant que le trou reste accessible par l'outil.

En manipulant le robot, vous allez maintenant acquérir une séquence continue de photos (à la cadence de la caméra) capturant l'approche du robot vers le trou, jusqu'à l'insertion complète de l'outil. Un exemple de séquence est disponible ici :
`https://youtu.be/wJG0y-qmHKU`

2.4 Récuperation des données

Une fois l'opération terminée, vous pouvez récupérer (par exemple sur une clé USB) tous ces fichiers, que vous pourrez traiter sans le robot. Il s'agit des fichiers suivants :

— n images $\mathbf{I}_i$, $i = 1, \ldots, n$ de la mire (format .jpg), sous différents points de vue ;

— n poses de l'outil dans le repère base {B}, dans un fichier texte cart_-poses.txt ; ces valeurs définissent les n transformations homogènes ${}^{B}\mathbf{T}_{O,i}$, $i = 1, \ldots, n$;

— la pose de la mire dans le repère base, relevée en mettant le robot dans la configuration de fig. 1b, pour que les repères mire et outil coïncident ; on notera cette pose ${}^{B}\mathbf{T}_{M}$;

— une séquence continue (à la cadence de la caméra) de m photos du trou, pendant que le robot s'en approche, et ce jusqu'à l'insertion. On

les notera $\mathbf{I}_{t,j}$, $j = 1, \ldots, m$ (format .jpg).

3 Étalonnage de la caméra

A ce stade, vous pouvez réaliser l'étalonnage du système robot/camera. Pour cette étape, vous n'aurez besoin ni du robot ni de ROS, mais uniquement d'un ordinateur avec python et opencv.

Afin de réaliser l'étalonnage de la caméra à partir des n images, vous pourrez vous inspirer d'un des tutoriels indiqués en Sec. 2.1. Les sorties de votre algorithme seront : les paramètres intrinsèques de la caméra : f_x, f_y, c_x et c_y et les paramètre extrinsèques, soit la transformation homogène ${}^{C}\mathbf{T}_{M}$ caractérisant la pose de la mire dans le repère caméra, pour chaque photo.

Questions :

Q1 Rentrez, dans le programme, la position cartesienne de chaque point de la mire dans le repère mire (carrés de coté 2 cm).

Q2 Pour réaliser l'étalonnage, il est nécessaire que les points de la mire soient toujours dans le même ordre (pour qu'ils soient - dans chaque image - associés à leur position cartésienne dans le repère mire). Est-ce le cas ? Sinon, deux possibilités :

— soit vous supprimez de la base de données les photos où l'ordre n'est pas respecté

— soit vous adaptez votre programme pour que les points soient toujours dans le même ordre (par exemple, en demandant à l'opérateur de cliquer dessus si nécessaire).

Q3 Est-ce que les paramètres intrinsèques et extrinsèques obtenus vous paraissent cohérents avec la géométrie du système robot/caméra ? Commentez.

Q4 Vous pouvez vérifier votre résultat en projetant les points (ou un seul point, par exemple le premier) du repère mire vers l'image, puis en comparant cette projection avec la position effective dans l'image.

Q5 Les paramètres intrinsèques de la caméra (étalonnée par le constructeur) sont (en pixels) : $c_x = 324.9$, $c_y = 245.8$, $f_x = 607.9$, et $f_y = 607.8$. La résolution de la caméra étant 640×480, vérifiez que ses angles d'ouverture horizontal et vertical, sont bien $55.52°$ et $43.09°$.

4 Pose de la caméra par rapport à l'outil

A ce stade, vous pouvez identifier la transformation homogène ${}^{O}\mathbf{T}_{C}$ caractérisant la pose de la caméra dans le repère outil. On notera que – à tout

moment – la relation suivante est valide :

$$ {}^{O}\mathbf{T}_{C} = {}^{O}\,\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C}. \tag{1} $$

Dans cette relation :

- ${}^{B}\mathbf{T}_{O}$, pose de l'outil dans le repère de la base, a été enregistrée pour chaque image (voir Sec. 2.1) : ${}^{B}\mathbf{T}_{O,i}$, $i = 1, \ldots, n$.
- ${}^{B}\mathbf{T}_{M}$, pose constante de la mire dans le repère de la base, a aussi été enregistrée comme indiqué en Sec. 2.1.
- ${}^{M}\mathbf{T}_{C}$, pose de la camera dans le repère de la mire (paramètres extrinsèques), est estimée par votre programme d'étalonnage (Sec. 3) pour chaque image : ${}^{M}\mathbf{T}_{C,i}$, $i = 1, \ldots, n$.

On note $t_{x,y,z} = t_{x,y,z}\left({}^{1}\mathbf{T}_{2}\right)$ les composantes de translation correspondantes à la transformation ${}^{1}\mathbf{T}_{2}$, et (en utilisant la répresentation angle/axis) $\theta\mathbf{u}_{x,y,z} = \theta\mathbf{u}_{x,y,z}\left({}^{1}\mathbf{T}_{2}\right)$ les composantes de rotation correspondantes à ${}^{1}\mathbf{T}_{2}$.

Avec cette notation, la relation (1) peut être écrite pour obtenir – pour chacun des six composant de la pose – un système de n équations à 1 inconnue (inconnues : ${}^{O}t_{x,C}$, ${}^{O}t_{y,C}$, …, ${}^{O}\theta u_{z,C}$:

$$ \begin{cases} {}^{O}t_{x,C} = t_{x}\left({}^{O,1}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,1}\right) \\ \ldots \\ {}^{O}t_{x,C} = t_{x}\left({}^{O,n}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,n}\right) \end{cases} \tag{2} $$

$$ \begin{cases} {}^{O}t_{y,C} = t_{y}\left({}^{O,1}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,1}\right) \\ \ldots \\ {}^{O}t_{y,C} = t_{y}\left({}^{O,n}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,n}\right) \end{cases} \tag{3} $$

$$ \ldots \tag{4} $$

$$ \begin{cases} {}^{O}\theta u_{z,C} = \theta u_{z}\left({}^{O,1}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,1}\right) \\ \ldots \\ {}^{O}\theta u_{z,C} = \theta u_{z}\left({}^{O,n}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,n}\right). \end{cases} \tag{5} $$

Questions :

Q1 Écrivez un de ces systèmes sous la forme $\mathbf{Ax} = \mathbf{b}$, pour montrer que leur solution aux moindres carrés $\mathbf{x} = \mathbf{A}^{\top}\left(\mathbf{A}\mathbf{A}^{\top}\right)^{-1}\mathbf{b}$, n'est rien d'autre qu'une moyenne :

$$ \begin{array}{l} {}^{O}t_{x,C} = \sum_{i=1}^{n} t_{x}\left({}^{O,i}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,i}\right)/n \\ {}^{O}t_{y,C} = \sum_{i=1}^{n} t_{y}\left({}^{O,i}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,i}\right)/n \\ {}^{O}t_{z,C} = \sum_{i=1}^{n} t_{z}\left({}^{O,i}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,i}\right)/n \\ {}^{O}\theta u_{x,C} = \sum_{i=1}^{n} \theta u_{x}\left({}^{O,i}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,i}\right)/n \\ {}^{O}\theta u_{y,C} = \sum_{i=1}^{n} \theta u_{y}\left({}^{O,i}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,i}\right)/n \\ {}^{O}\theta u_{z,C} = \sum_{i=1}^{n} \theta u_{z}\left({}^{O,i}\mathbf{T}_{B}\ {}^{B}\mathbf{T}_{M}\ {}^{M}\mathbf{T}_{C,i}\right)/n. \end{array} \tag{6} $$

Q2 Écrivez un programme python permettant de résoudre (6), afin de déterminer la pose de la caméra dans le repère outil, ${}^{O}\mathbf{T}_{C}$. Quelques ressources utiles :

`https://stackoverflow.com/questions/61360556/opencv-camera-calibration-rotation-vector-to-matrix`

`https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.html`

`https://alexsm.com/homogeneous-transforms/.`

Q3 Pour vérifier la qualité de votre résultat, vous pouvez vérifier la projection de points de la mire vers l'image (cette fois-ci, "en passant" par la chaîne robotique) ainsi que l'écart type des solutions obtenues avec (6).

Q4 Est-ce que la solution trouvée vous parait cohérente avec la géométrie du système robot/caméra (fig. 1b) ? Vos chers professeurs ont trouvé :

$$ {}^{O}\mathbf{T}_{C} = \begin{bmatrix} -0.016 & 0.999 & 0.017 & 0.045 \\ -0.999 & -0.016 & 0.018 & 0.029 \\ 0.018 & -0.017 & 0.999 & -0.134 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7} $$

5 Reconnaissance du trou dans les images

L'objectif de cette partie est de développer un algorithme de traitement d'images permettant de localiser la position du trou et d'estimer sa surface dans l'image. A cette fin, vous traiterez les images du trou acquises en Sec. 2.3, en utilisant python et opencv. A chaque itération, votre algorithme aura en entrée une photo $\mathbf{I}_i$ et sortira les coordonnées images du barycentre du trou ainsi que sa surface : u, v, a (trois scalaires, exprimés en pixel, voir Fig. 2). Vous pourrez vous appuyer sur la détection effectuée à l'itération (sur l'image) précédente, en considérant que la position actuelle du trou sera proche de la précédente.

Si on considère le trou comme un blob noir entouré d'un blob bleu, vous pourrez probablement le détecter en utilisant les notions/opérations suivantes :

— conversion du format RGB vers le format HSV,
— histogramme d'image,
— égalisation d'histogramme,
— binarisation avec le seuil d'Otsu,
— opérations morphologiques (erosion/dilatation),
— calcul de moments.

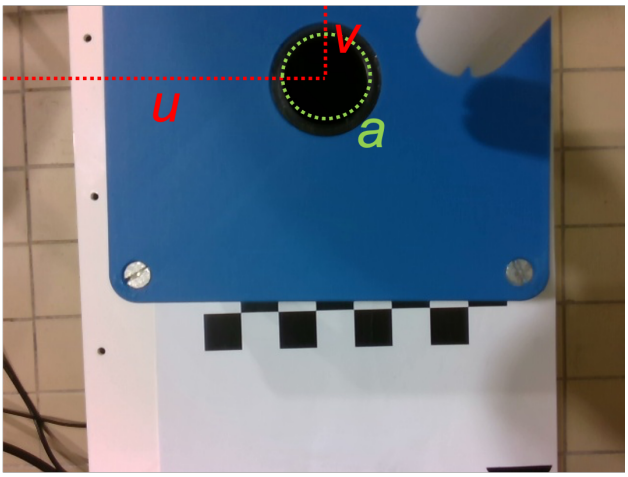


FIGURE 2 – Image du trou acquise par la caméra ; on note u, v les coordonnées images du barycentre du trou, et a sa surface.

Vous pouvez traire inspiration de ce site : `https://learnopencv.com/find-center-of-blob-centroid-using-opencv-cpp-python/`

6 Commande pour réaliser l'insertion

6.1 Loi de commande hybride vision/force

L'objectif de cette partie est de développer une loi de commande permettant au robot d'insérer l'outil dans le trou. Cette loi de commande sera validée en deux étapes :

1. en simulation (aussi bien du robot que des images) dans Gazebo,
2. puis sur le robot réel.

Dans l'hypothèse – réaliste – que la plaque du trou est horizontale (parallèle au sol), et que l'outil démarre toujours de la position verticale illustrée en Fig. 1b) on viendra contrôler uniquement les translations.

La loi d'impédance utilisée sur le robot est décrite par la consigne en couple suivante :

$$\tau_d = \mathbf{J}^\top \left[\mathbf{K}^B\mathbf{v}_O - \mathbf{D}\dot{\mathbf{X}} \right] + \mathbf{C}\left(\mathbf{q}, \dot{\mathbf{q}}\right) + \mathbf{g}\left(\mathbf{q}\right) \tag{8}$$

où ${}^B\mathbf{v}_O$ est la consigne, en vitesse, du déplacement du robot. Vous devrez utiliser cette variable pour déplacer le robot en fonction de la détection du

trou. Utilisez le topic `/cart_vel_des` pour communiquer votre consigne ${}^{B}\mathbf{v}_O$ au robot (en simulation et aussi en réel).

On souhaite

$$\dot{\mathbf{s}} = \mathbf{L}_s^C \mathbf{v}_C = -\Lambda \mathbf{e} \tag{9}$$

avec uniquement les trois vitesses de translation. Donc :

$${}^{C}\mathbf{v}_C = -\mathbf{L}_s^{-1} \Lambda \mathbf{e} \tag{10}$$

qui peut se ré-écrire dans le repère base comme :

$${}^{B}\mathbf{v}_C = {}^{B}\mathbf{R}_C^C \mathbf{v}_C \tag{11}$$

La vitesse correspondante de l'outil sera :

$${}^{B}\mathbf{v}_O = {}^{B}\mathbf{v}_C \tag{12}$$

6.2 Démarche d'insertion

Utilisez le noeud `integ_visual_servoing_node.py` présent dans le package `franka_integ_node` comme base pour préparer la commande du robot. Intégrez à ce noeud vos algorithmes de détection du trou et la publication des consignes ${}^{B}\mathbf{v}_O$ à travers le topic précedemment mentionné.

Procédez à l'insertion du trou en prévoyant dans votre code les étapes suivantes :

1. Contrôlez le robot pour qu'il suive les mouvements du trou dans le plan horizontal (seulement en XY), jusqu'à ce qu'il soit correctement aligné avec celui-ci ;
2. une fois l'alignement effectué, déclenchez le mouvement selon l'axe Z (vertical) pour insérer l'outil dans le trou ;
3. arrêtez automatiquement le mouvement d'approche du robot si la force de contact dépasse 5 Nm selon l'axe Z de l'outil. Cette force extérieure peut être mésurée à travers le topic :
   `/franka_states_controllers/F_ext`