Travaux Pratiques

Automatique
Contrôle de la trajectoire d'un drone

1- Énoncé

Ce TP a pour objet le contrôle de la trajectoire d'un drone à voilure tournante. Pour cela, il met en œuvre un drone à voilure tournante et un système de capture de mouvements (MOtion CAPture) constitué d'un ensemble de caméras Qualisys Miqus ultra-rapides (1 Mpixels, 250 images/s). Ces dernières sont associées à un logiciel de traitement d'images qui restitue en temps réel la position du drone évoluant dans le champ visuel couvert par les caméras.

Le Crazyflie 2.1 contrôlé est représenté sur la figure 1.1. Il s'agit d'un quadrirotor produit par la société suédoise Bitcraze. Il est entrainé par quatre moteurs électriques à courant continu et peut être commandé depuis un *smartphone*, une tablette ou un PC. Il embarque différents capteurs tels que des accéléromètres, des gyromètres et des magnétomètres. Pratiquement, vous commanderez le Crazyflie depuis un PC avec une interface programmée sous Python 3.10 via une liaison radio 2.4GHz.

Fig. 1.1 – Le Crazyflie 2.1 de Bitcraze

1.1 Présentation du système étudié

1.1.1 Objectifs

Au cours de cette séance, vous devrez établir un modèle du drone Crazyflie et développer une loi de commande dans le but de contrôler sa trajectoire. Les résultats seront d'abord testés en simulation sur le modèle, puis validés par l'expérimentation.

1.1.2 Organisation de la séance

Le TP se réalise en binôme ou en trinôme. Le document réponse est à rendre complété en fin de séance. Les réponses aux questions doivent être impérativement justifiées dans les cadres prévus à cet effet. Par ailleurs, les résultats des simulations et des essais seront restitués dans le fichier TP_Drone_Document_Reponse.doc que vous commencerez par renommer TP_Drone_Brigade__nom1_nom2.doc

Ce TP est constitué de quatre parties. Il est conseillé de traiter les parties dans l'ordre proposé et d'organiser les 4 heures du TP avec le découpage suivant :

Partie	**Durée**
Lecture du sujet et prise de connaissance des documents ressources	30 min
Modélisation et synthèse d'une loi de commande pour le contrôle de la trajectoire	1 h 30 min
Implémentation et simulation de la loi de commande	1 h
Validation expérimentale	1 h

TAB. 1.1 – Organisation du travail pratique

1.1.3 Moyens mis à disposition

Le matériel suivant est mis à votre disposition pour ce TP :
- Arène de vol ;
- MOCAP constitué d'un réseau de caméras QUALISYS MiQus ;
- Drone Nano-Copter Bitcraze Crazyflie 2.1 ;
- Batteries LiPo 1S et leurs chargeurs spécifiques (veillez à ce que des batteries chargées soient toujours disponibles) ;
- Robot terrestre ;
- Dongle USB de communication radio 2.4GHz ;
- Routeur TPLink ;
- PC + logiciel de capture de mouvements QUALISYS ;
- PC + Matlab et Sisotool ;
- PC + Linux Ubuntu 22.04 + Pycharm CE + Python 3.10.

1.2 Modélisation du drone Crazyflie

1.2.1 Repères et forces

On définit :

— $R_E = (O, \overrightarrow{x_E}, \overrightarrow{y_E}, \overrightarrow{z_E})$: le repère de référence attaché à l'arène de vol, supposé inertiel et défini *infra* ;

— $R_b = (c.g., \overrightarrow{x_b}, \overrightarrow{y_b}, \overrightarrow{z_b})$: un repère orthonormé attaché au drone et défini *infra* ;

— R_v un repère ayant pour origine le centre de gravité du drone et dont les vecteurs qui le définissent ont à tout instant la même direction que ceux qui définissent R_E. Soit $\overrightarrow{x}_v$ colinéaire à $\overrightarrow{x}_E$, $\overrightarrow{y}_v$ colinéaire à $\overrightarrow{y}_E$ et $\overrightarrow{z}_v$ colinéaire à $\overrightarrow{z}_E$;

— R_n est le repère de navigation. Son origine coïncide avec le centre de gravité du drone, il est obtenu en opérant à une rotation d'angle ψ de R_v autour de l'axe $\overrightarrow{z}_v$.

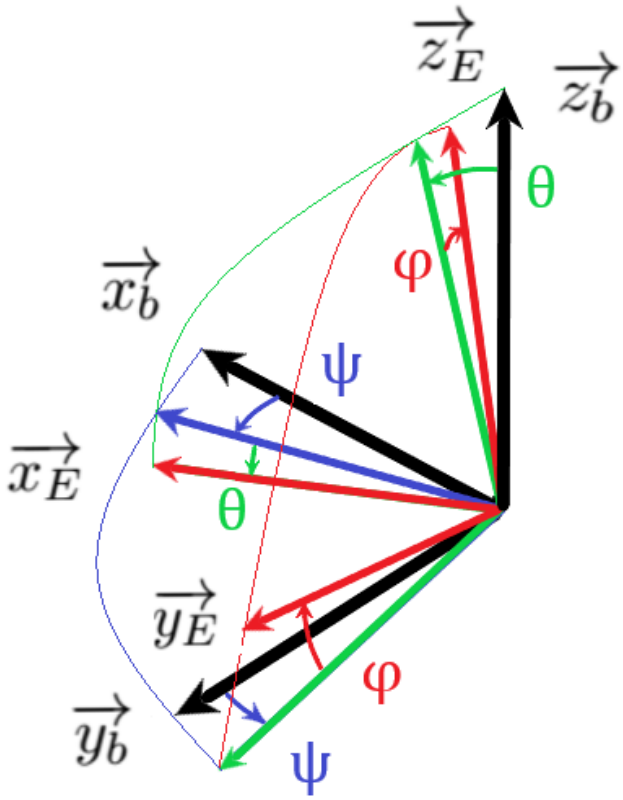

FIG. 1.2 – Angles

On note par ailleurs :
— x_C, y_C : les coordonnées du point à atteindre ;
— x_E, y_E les coordonnées du drone ;
— u, v les vitesses de translation longitudinale et latérale portées respectivement selon $\overrightarrow{x_b}$ et $\overrightarrow{y_b}$;
— ϕ, θ, ψ qui repèrent l'attitude du drone dans $(O, \overrightarrow{x_E}, \overrightarrow{y_E}, \overrightarrow{z_E})$ sont respectivement désignés gîte, assiette et cap, ils sont illustrés sur la figure 1.2 ;
— s désigne la variable de Laplace ;
— m désigne la masse du Crazyflie égale à 0.035 kg.

On détermine par la suite la matrice de passage $\mathbf{T}_{bE}$ du repère drone au repère terrestre.
Pour une rotation en gîte uniquement :

$$\begin{cases} \overrightarrow{x_E} = \overrightarrow{x_b} \\ \overrightarrow{y_E} = \cos(\phi)\overrightarrow{y_b} - \sin(\phi)\overrightarrow{z_b} \\ \overrightarrow{z_E} = \sin(\phi)\overrightarrow{y_b} + \cos(\phi)\overrightarrow{z_b} \end{cases} \rightarrow R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

Pour une rotation en assiette uniquement :

$$\begin{cases} \overrightarrow{x_E} = \cos(\theta)\overrightarrow{x_b} + \sin(\theta)\overrightarrow{z_b} \\ \overrightarrow{y_E} = \overrightarrow{y_b} \\ \overrightarrow{z_E} = -\sin(\theta)\overrightarrow{x_b} + \cos(\theta)\overrightarrow{z_b} \end{cases} \rightarrow R_y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

Pour une rotation en lacet uniquement :

$$\begin{cases} \overrightarrow{x_E} = \cos(\psi)\overrightarrow{x_b} - \sin(\psi)\overrightarrow{y_b} \\ \overrightarrow{y_E} = \sin(\psi)\overrightarrow{x_b} + \cos(\psi)\overrightarrow{y_b} \\ \overrightarrow{z_E} = \overrightarrow{z_b} \end{cases} \rightarrow R_z = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

En appliquant la convention $Z \rightarrow Y \rightarrow X$, on trouve la matrice de passage suivante :

$$\mathbf{T}_{bE} = R_z R_y R_x$$

$$\mathbf{T}_{bE} = \begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \cos(\phi)\cos(\psi) + \sin(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

Du fait de la faible vitesse de déplacement du drone, la trainée est négligée, seuls le poids $-mg\overrightarrow{z}_E$ et la force de portance $P\overrightarrow{z}_b$ développée par les hélices sont considérés.

Le PFD appliqué au drone :

$$\mathbf{F}^E = \mathbf{T}_{bE}\begin{pmatrix}0\\0\\P\end{pmatrix} + \begin{pmatrix}0\\0\\-mg\end{pmatrix}$$

L'expression des forces dans le repère de navigation :

$$\mathbf{F}^n = \underbrace{\begin{pmatrix}\cos(\psi) & \sin(\psi) & 0\\ -\sin(\psi) & \cos(\psi) & 0\\ 0 & 0 & 1\end{pmatrix}}_{\mathbf{R}(\psi)}\mathbf{T}_{bE}\begin{pmatrix}0\\0\\P\end{pmatrix} + \begin{pmatrix}0\\0\\-mg\end{pmatrix}$$

Où $\mathbf{R}(\psi)$ désigne la matrice de rotation d'un angle ψ autour de l'axe de lacet. Cette matrice est orthogonale et $\mathbf{R}(\psi) = {\mathbf{R_z}}^T = {\mathbf{R_z}}^{-1}$.

A l'équilibre, la portance équilibre le poids et $P = mg$.

Question *1* : *Montrez pour des petits angles ϕ et θ que les forces dans le repère de navigation ont pour expression :*

$$\mathbf{F}^n = \begin{pmatrix}mg\theta\\-mg\phi\\0\end{pmatrix} \tag{1.1}$$

1.2.2 Principe du guidage

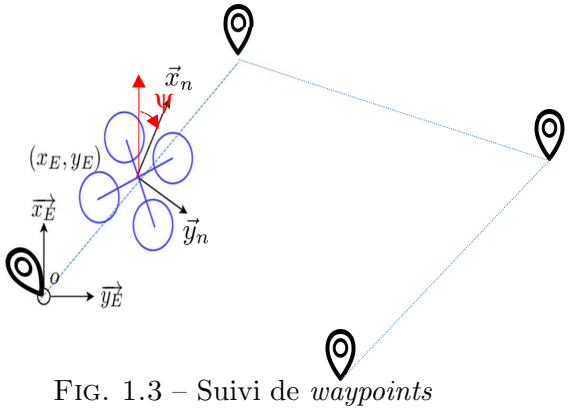

FIG. 1.3 – Suivi de *waypoints*

Calcul de l'erreur de position dans le repère terrestre :

$$\begin{pmatrix}\varepsilon_x\\\varepsilon_y\end{pmatrix}^E = \begin{pmatrix}x_C - x_E\\y_C - y_E\end{pmatrix}^E \tag{1.2}$$

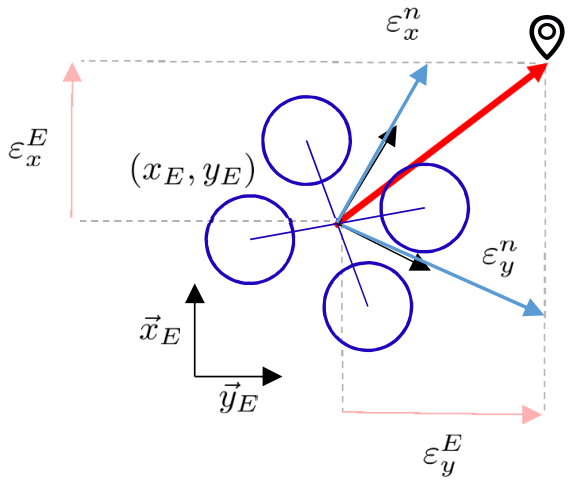

FIG. 1.4 – Erreur de position dans R_E et R_n

Calcul de l'erreur de position dans le repère de navigation :

$$\begin{pmatrix} \varepsilon_x \\ \varepsilon_y \end{pmatrix}^n = \mathbf{R}_{2\times 2}(\psi) \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \end{pmatrix}^E \tag{1.3}$$

Génération d'une force de poussée orientée vers le point à atteindre :

$$\mathbf{F}^n = \begin{pmatrix} mg\theta \\ -mg\phi \\ 0 \end{pmatrix} \tag{1.4}$$

1.3 Conception et réglage de la loi de commande

1.3.1 Fonctions de transfert du drone

Le PFD appliqué au drone :

$$\mathbf{F}^E = m \begin{pmatrix} \ddot{x}_E \\ \ddot{y}_E \end{pmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{pmatrix} mg\theta \\ -mg\phi \end{pmatrix} \tag{1.5}$$

Question 2 : *A cap constant, établissez la transformée de Laplace de ces équations, les conditions initiales étant toutes nulles.*

Question 3 : *En supposant le cap ψ nul et les angles de gîte et d'assiette parfaitement asservis à leur consigne. Établissez les fonctions de transfert $G(s) = \dfrac{X_E(s)}{\theta(s)}$ et $H(s) = \dfrac{Y_E(s)}{\phi(s)}$.*

1.3.2 Commande proportionnelle dérivée

On calcule les lois de commande en supposant le cap ψ nul et on suppose les angles de gîte et d'assiette parfaitement asservis à leur consigne. Ce sont ces consignes de gîte et d'assiette qui jouent le rôle d'entrée de commande des boucles d'asservissement de trajectoire du drone. Soient les lois de commandes et le schéma fonctionnel afférent :

$$\theta_{ref} = \theta = k_d \left(\underbrace{k_p(x_C - x_E)}_{\mu_C} - \mu \right) \tag{1.6}$$

$$\phi_{ref} = \phi = -k_d \left(\underbrace{k_p(y_C - y_E)}_{\nu_C} - \nu \right) \tag{1.7}$$

où (μ, ν) désignent les coordonnées du vecteur vitesse du drone dans le repère de navigation.

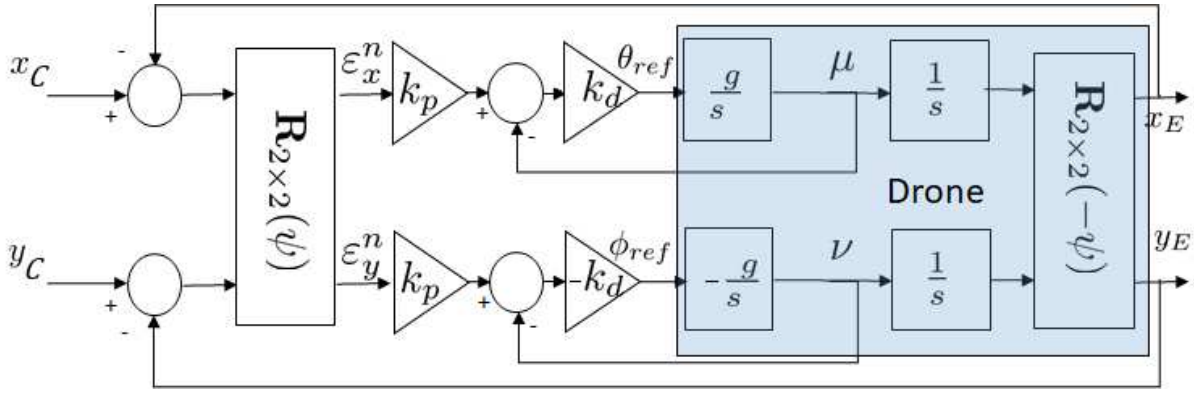

FIG. 1.5 – Lois de commande en tangage et roulis

Pratiquement, du fait de la géométrie du drone, les lois de commande sont identiques pour l'axe de tangage et de roulis. Vous adopterez donc pour la commande de l'axe de roulis une architecture et des gains identiques (au signe de k_d près) à ceux trouvés pour le contrôle de l'axe de tangage.

Question 4 : *Dans le cas où $\psi = 0$, calculez les fonctions de transfert en boucle fermée $\frac{X_E(s)}{X_C(s)}$ et $\frac{Y_E(s)}{Y_C(s)}$ du drone muni de ces lois de commande et identifiez-les à des fonctions de transfert canoniques de systèmes du second ordre.*

Question 5 : *Réglez les coefficients k_d et k_p de sorte qu'en réponse à un échelon de position d'un mètre, le drone présente une réponse à temps de réponse minimal possédant les caractéristiques suivantes : pas d'erreur de position, autrement dit $x_e(\infty) = 1m$, $t_{r5\%} \approx 2.5s$ et $D_1\% = 0$.*

Question 6 : *Saisissez sous MATLAB les fonctions de transfert $G_1(s) = \frac{g}{s}$ et $G_2(s) = \frac{1}{s}$ telles que $G(s) = G_1(s)G_2(s)$. Lancez Sisotool (Cf. document ressources 2.1) et implémentez la loi de*

commande proportionnelle dérivée sur une architecture telle que celle représentée sur la figure 1.6 . Réglez les valeurs de k_p et k_d Visualisez la carte des pôles et vérifiez que le réglage désiré est réalisé. Concluez cette étude par l'observation de la réponse indicielle en boucle fermée pour laquelle vous relèverez les caractéristiques. Copiez le lieu des pôles et le chronogramme sur le document Word.

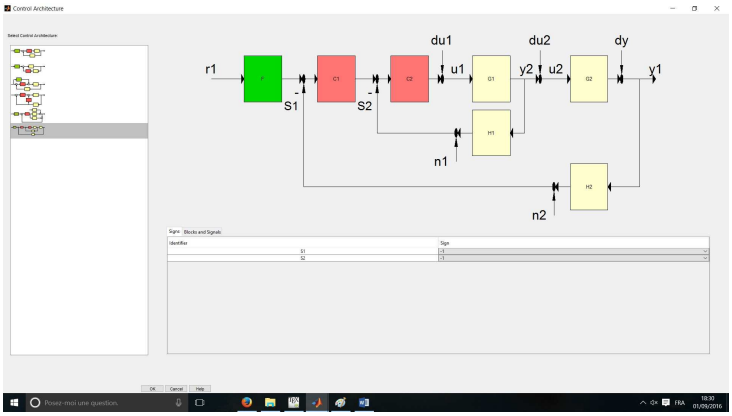

FIG. 1.6 – Structure de la boucle d'asservissement

1.4 Implémentation du correcteur proportionnel dérivé

Le correcteur est désormais dimensionné pour respecter le cahier des charges, et son implémentation peut commencer.

1.4.1 Mise en place

Sur le PC Linux, assurez-vous de bien avoir le dossier TP_trajectoire_circulaire associé à ce TP disponible sur le bureau (pour la suite de ce TP, on appellera ce dossier le répertoire de travail). Assurez-vous également qu'un *Dongle* de communication radio est bien branché sur l'un des ports USB du PC, et qu'une manette ou un joystick est branché.

Démarrez le logiciel PyCharm. A partir de PyCharm, ouvrez le projet matérialisé par le dossier TP_trajectoire_circulaire sur le bureau. Dans l'arborescence à gauche, vous avez alors accès aux différents fichiers Python qui le composent. Vous pouvez dès à présent ouvrir les fichiers suivants sur PyCharm :

- main.py
- simulation.py
- uav_control_law.py

1.4.2 Ressources supplémentaires

Des informations et des programmes supplémentaires sont fournis sous forme de NoteBooks Python interactifs (extension .ipynb) situés dans le répertoire de travail. Pour lire ces NoteBooks, démarrez le logiciel JupyterLab.

Cela provoque l'ouverture d'une fenêtre sur un navigateur web, dans laquelle il est possible de parcourir les fichiers de l'ordinateur pour ouvrir les NoteBooks suivants :

— Documentation.ipynb
— Flight_logs_dashboard.ipynb

Note : Assurez-vous de bien utiliser les fichiers situés dans le répertoire de travail, ce TP coexiste avec plusieurs autres projets qui ont chacun leurs propres fichiers "Documentation" et "Flight_logs_dashboard".

1.4.2.1 Documentation

Bien que les information nécessaires à la réalisation de ce TP soient présentées sur ce sujet, une documentation plus détaillée des drones et des programmes utilisés est disponible dans le Notebook Documentation.ipynb.

1.4.2.2 Visualisation des trajectoires

Chaque fois qu'un drone est mis en vol (en environnement réel comme en environnement de simulation), un fichier contenant les données de vol est créé (respectivement logs.csv et sim_logs.csv). Le NoteBook Flight_logs_dashboard.ipynb est un outil de visualisation de ces données de vol. Il contient des cellules de code Python qui lisent les données d'un fichier csv spécifié pour les afficher sur des graphiques.

Dans la dernière cellule de code de ce NoteBook, vous pouvez choisir de fichier à lire par le programme en modifiant la variable "csv_filename".

Note : Si vous voulez conserver les fichiers csv d'un essai à l'autre, pensez à les renommer, les fichiers appelés "logs.csv" ou "sim_logs.csv" seront effacés à chaque fois qu'un drone sera à nouveau mis en vol

1.4.3 Loi de commande

Les asservissements en roulis (roll), tangage (pitch), lacet (yaw) et en poussée des moteurs (thrust) sont gérés par une unique fonction control_law(), située dans le script Python uav_control_law.py. Cette fonction est utilisée à la fois pour contrôler le modèle du drone en simulation, et pour contrôler le drone réel.

Les correcteurs en lacet et en poussée des moteurs sont déjà présents dans cette fonction, mais les lois de commande en roulis et en tangage sont manquantes.

Question 7 : *Implémentez votre propre code dans la partie balisée "A remplir" de la fonction control_law() pour créer en Python le correcteur proportionnel dérivé en roulis et en tangage que vous avez dimensionné plus tôt.*

Note : Attention aux différences entre le repère de navigation du drone et le repère de l'arène. Vous pouvez utiliser la bibliothèque numpy pour avoir accès aux fonctions sin() et cos(), sachant que l'argument d'entrée de ces fonctions est exprimé en radians.

Une fois que la fonction control_law() est complète, vous pouvez tester votre loi de commande en simulation. Pour ce faire, dans le script simulation.py, cliquez sur la flèche verte associée à la ligne de code if __name__ == '__main__' :, puis cliquez sur "run simulation.py". Une interface graphique s'ouvre, avec plusieurs boutons et graphiques qui vous permettent de contrôler le modèle du drone sur la simulation et d'en observer son comportement. Il est alors possible avec le bouton Step de valider votre implémentation du correcteur sous Python en vérifiant que la réponse indicielle est bien cohérente avec celle obtenue sur *Sisotool*.

Note : Une fois que le bouton Step est activé, vous pouvez faire varier l'angle de lacet avec les boutons + et -. Pensez à faire varier cette valeur pour vérifier que le changement de repère que vous avez codé est bien valide

1.4.4 Signaux de consigne

La finalité de ce TP est d'assurer de manière automatique la surveillance d'une cible mobile au sol. En supposant qu'une caméra soit embarquée à l'avant du drone, l'objectif principal est alors de poursuivre la cible tout en la maintenant dans le champ de vision de la caméra.

Un deuxième objectif vient s'ajouter : on cherche à suivre la cible le plus longtemps possible. Ainsi, on portera attention à l'autonomie du drone, et donc à l'efficacité énergétique des manœuvres de poursuite.

Pour accomplir ce dernier objectif, on utilise la figure 1.7 pour se rendre compte que, de manière générale pour un drone à voilure tournante, le vol stationnaire est plus consommateur d'énergie que le vol à vitesse modérée et constante. On cherche alors à faire décrire au drone une trajectoire qui lui permette de suivre la cible tout en gardant une vitesse non-nulle dans le plan horizontal, et en évitant les changements brusques de direction.

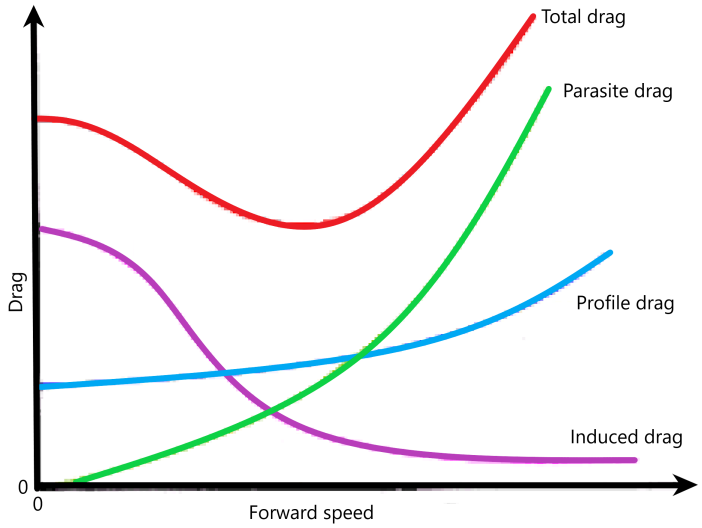

FIG. 1.7 – Traînée aérodynamique perçue par les rotors en fonction de la vitesse du drone

Une manœuvre adaptée à ce type d'objectif et couramment utilisée dans des opérations de surveillance est appelée Point Of Interest (POI). Elle consiste à suivre une trajectoire de type circulaire autour de la cible à surveiller, tout en pilotant le cap pour que le drone soit en permanence face à la cible.

On se propose d'automatiser cette manœuvre en décomposant son implémentation selon les trois étapes suivantes, qui correspondent respectivement aux questions 8, 9 et 10 :

1. On génère une trajectoire circulaire à cap constant autour d'un point fixe.
2. On s'approprie la gestion du cap en générant une trajectoire circulaire autour d'un point fixe, telle que l'axe de la supposée caméra du drone soit colinéaire à son vecteur vitesse (le drone "regarde où il va").

3. On implémente la manoeuvre Point Of Interest en générant une trajectoire circulaire autour d'une cible en mouvement dans l'arène, tout en pilotant le cap pour en assurer la surveillance (le drone "regarde la cible").

Question *8* : *Dans le script* uav_control_law.py*, remplacez les parties balisées "A remplir" de la fonction* circle() *par votre propre code pour générer une trajectoire de consigne circulaire respectant les paramètres suivants :*

- *Rayon :* $0.50m$.
- *Hauteur par rapport au sol :* $0.40m$.
- *Rythme d'évolution : 1 tour parcouru en* $2\pi s$.
- *Centre du cercle : centre de l'arène, soit* $(x_0, y_0) = (0, 0)$.
- *Cap constant* ($\psi = 0$).

Vérifiez votre code à l'aide du programme de simulation (bouton "Circle").

Question *9* : *Dans le script* uav_control_law.py*, remplacez les parties balisées "A remplir" de la fonction* circle_with_tangent_x_axis() *par votre propre code pour générer une trajectoire de consigne circulaire ayant les mêmes paramètres qu'à la question précédente, mais avec un cap variable pour que l'axe x du repère de navigation du drone soit colinéaire à son vecteur vitesse.*

Plusieurs méthodes permettent de générer la consigne de cap, chacune ayant ses propres avantages et inconvénients. Proposez des solutions et implémentez celle qui vous paraît la plus adaptée à vos besoins et contraintes (précision souhaitée, complexité des calculs, temps de développement de la solution, temps de séance restant, etc...).

Note : Un copier-coller de votre travail depuis la fonction circle() *peut vous faire gagner du temps pour générer les coordonnées x et y de consigne.*

Vérifiez votre code à l'aide du programme de simulation (bouton "Circle with tangent x axis").

Question *10* : *Dans le script* uav_control_law.py*, remplacez les parties balisées "A remplir" de la fonction* point_of_interest() *par votre propre code pour générer une trajectoire qui réalise une manoeuvre de Point Of Interest autour d'un robot terrestre, avec les paramètres suivants :*

- *Rayon :* $0.50m$.
- *Hauteur par rapport au sol :* $0.40m$.
- *Rythme d'évolution : 1 tour parcouru en* $2\pi s$.
- *Centre du cercle : position du robot, réactualisée à chaque nouveau pas de temps.*
- *Cap variable pour que l'axe x du repère de navigation du drone pointe vers le robot terrestre.*

Note : Un copier-coller de votre travail depuis la fonction circle_with_tangent_x_axis *peut vous faire gagner du temps s'il est correctement réadapté aux nouveaux objectifs.*

Vérifiez votre code à l'aide du programme de simulation (bouton "Point Of Interest").

1.5 Vérification expérimentale

Une fois que votre loi de commande et vos signaux de consigne sont validés par la simulation, vous pouvez vous munir d'un drone Crazyflie et du robot terrestre identifié par l'étiquette "Chasseur" pour mettre à l'épreuve votre code dans des conditions expérimentales.

Le robot terrestre est programmé pour suivre automatiquement une trajectoire dessinée au sol par détection optique. Un marqueur infra-rouge similaire à celui du Crazyflie permet au logiciel de Motion Capture de repérer le robot dans l'arène de vol.

1.5.1 Préparation du vol

1.5.1.1 Système de Motion Capture

Vérifiez tout d'abord que le PC qui gère les caméras est bien connecté en filaire au PC Linux via le routeur TPLink, puis branchez les caméras. Sur le PC relié aux caméras, exécutez le programme Qualisys Track Manager (QTM) via le raccourci situé sur le bureau. Dans la fenêtre de lancement de QTM, double-cliquez sur le projet TP_trajectoire_circulaire (la calibration du système MOCAP et le réglage de tous les paramètres sur QTM ont été effectués au préalable pour ce TP).

Une fois que la fenêtre est ouverte, cliquez sur Fichier/Nouveau pour démarrer une session sur QTM. Dès que les images des caméras sont affichées à l'écran en mode Marker, les coordonnées des marqueurs infra-rouge repérés sont émises sur le réseau, et sont donc accessibles par le PC sous Linux. Vous pouvez également passer en vue 3D pour obtenir une représentation de la scène une fois que les données 2D des caméras sont fusionnées entre elles.

1.5.1.2 Arène de vol

Installez la zone d'évolution du robot terrestre dans l'arène de vol et déterminez la position initiale du drone, comme illustré sur la figure 1.8

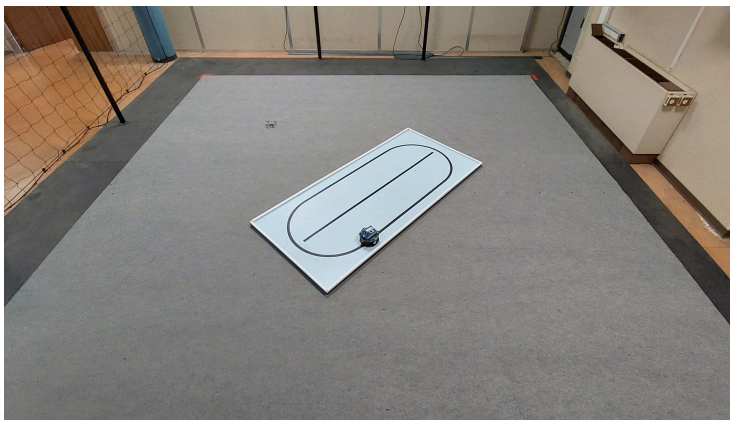

FIG. 1.8 – Zone d'évolution du robot terrestre placée dans l'arène de vol

Attention : les axes du repère de navigation du Crazyflie doivent être colinéaires aux axes du repère de l'arène avant le décollage. Le drone doit donc être initialement placé face au mur Est de l'arène. Cf.documentation (1.4.2).

1.5.2 Exécution du programme de contrôle

Dans le script main.py, cliquez sur la flèche verte associée à la ligne if __name__ == '__main__' :, puis sélectionnez "run main.py" pour lancer l'exécution du programme. Une interface graphique apparaît.

1.5.2.1 Réglage des paramètres de vol

Utilisez la documentation pour régler les paramètres de vol sur cette interface graphique. Une fois que le drone et le robot (si utilisé) sont correctement paramétrés, installés et mis sous tension, cliquez sur le bouton "Valider" pour lancer la procédure de connexion au drone.

1.5.2.2 Connexion au drone

Une phase d'initialisation est d'abord lancée. Elle permet d'établir les connexions au drone et au logiciel de Motion Capture, d'identifier le drone et le robot dans l'arène, et de détecter et configurer la manette ou le joystick branché au PC. Attendez bien la fin de cette phase d'initialisation avant d'appuyer sur les boutons de la manette ou du joystick. Restez attentifs aux informations renvoyées par le programme lors de cette phase, et n'envisagez pas de faire décoller le drone si un avertissement — Warning — apparaît dans la console.

La fin de la phase d'initialisation est repérée par un tableau récapitulatif de la connexion au drone, Cf. documentation (1.4.2), qui contient :

- Les coordonnées du drone trouvé. Vérifiez alors qu'elles soient bien cohérentes avec la position réelle du drone dans l'arène ;
- Le niveau de batterie du drone. Le décollage sera bloqué si le niveau de batterie est inférieur à 20% ;
- La mention "Flight enabled" ou "Flight disabled", qui informe l'utilisateur que le décollage du drone est soit autorisé, soit bloqué. Si le décollage est bloqué, cherchez des avertissements — Warning — pour en connaître la cause, Cf. documentation (1.4.2).

Une fois que la phase d'initialisation est terminée, vous pouvez vous munir de la manette ou du joystick pour faire décoller le drone. L'affectation des touches est présentée dans le tableau 1.2 et dans la documentation.

Notes :

- *La touche Echap du clavier est un arrêt d'urgence. Elle permet de stopper les moteurs du drone et de le déconnecter ;*
- *Une fois l'atterrissage terminé, les commandes sont bloquées. Il est alors nécessaire de redémarrer le programme pour faire décoller le drone à nouveau ;*
- *L'arrêt du programme s'effectue en deux étapes. La première est de couper les moteurs avec la touche correspondante de la manette ou du joystick, et la deuxième étape est d'appuyer sur la touche Echap du clavier. Si cette procédure n'est pas respectée, des reliquats de programme peuvent continuer d'être exécutés en arrière plan.*

Question *11* : *Vérifiez vos trois fonctions de contrôle automatique du drone dans les conditions expérimentales. Votre fonction Point Of Interest remplit-elle ses objectifs ? Votre loi de commande est-elle satisfaisante ? Quelles sont les pistes d'amélioration et quelles solutions proposeriez-vous ?*

Note : pas de réponse standard attendue, donnez votre avis et proposez vos idées.

Fonction	**Joystick**	**Manette mode 1**	**Manette mode 2**
Décollage / Atterrissage	2	A	A
Fonction "Circle"	11	X	X
Fonction "Circle with tangent x axis"	9	Y	Y
Fonction "Point Of Interest"	7	B	B
Fonction "Standby" (vol stationnaire)	3	LX	LX
Contrôle manuel du drone	4	RX	RX
Déplacement selon x (repère de l'arène)	Manche vers l'avant/arrière	Stick droit vers l'avant/arrière	Stick gauche vers l'avant/arrière
Déplacement selon y (repère de l'arène)	Manche vers la gauche/droite	Stick droit vers la gauche/droite	Stick gauche vers la gauche/droite
Déplacement selon z (repère de l'arène)	Manette "Thrust"	Stick gauche vers l'avant/arrière	Stick droit vers l'avant/arrière
Lacet	5 / 6	Stick gauche vers la gauche/droite	Stick droit vers la gauche/droite
Coupure des moteurs	1 (gâchette)	Start	Start

TAB. 1.2 – Affectation des touches

Travaux Pratiques

Automatique
Contrôle de la trajectoire d'un drone

2- Document ressource

2.1 Utilisation de sisotool

La saisie des polynômes est nécessaire à celle des fonction de transfert :

Un polynôme de degré n : $P(x) = a_n x^n + \ldots + a_2 x^2 + a_1 x + a0$ est défini sous MATLAB comme une matrice de dimension $1 \times n$ sous la forme suivante :

$$P = [an \ldots a3\ a2\ a1\ a0]$$

La fonction de transfert qui caractérise un système est un objet défini de la façon suivante : *system=tf(num,den)* où *num* et *den* sont les polynômes apparaissant au numérateur et au dénominateur.

Sisotool (single input single output tool) est un programme qui permet la synthèse de lois de commande et l'évaluation des performances des systèmes asservis. Lorsque l'on souhaite mettre en œuvre une correction pour un système asservi décrit par une ou plusieurs fonctions de transfert.

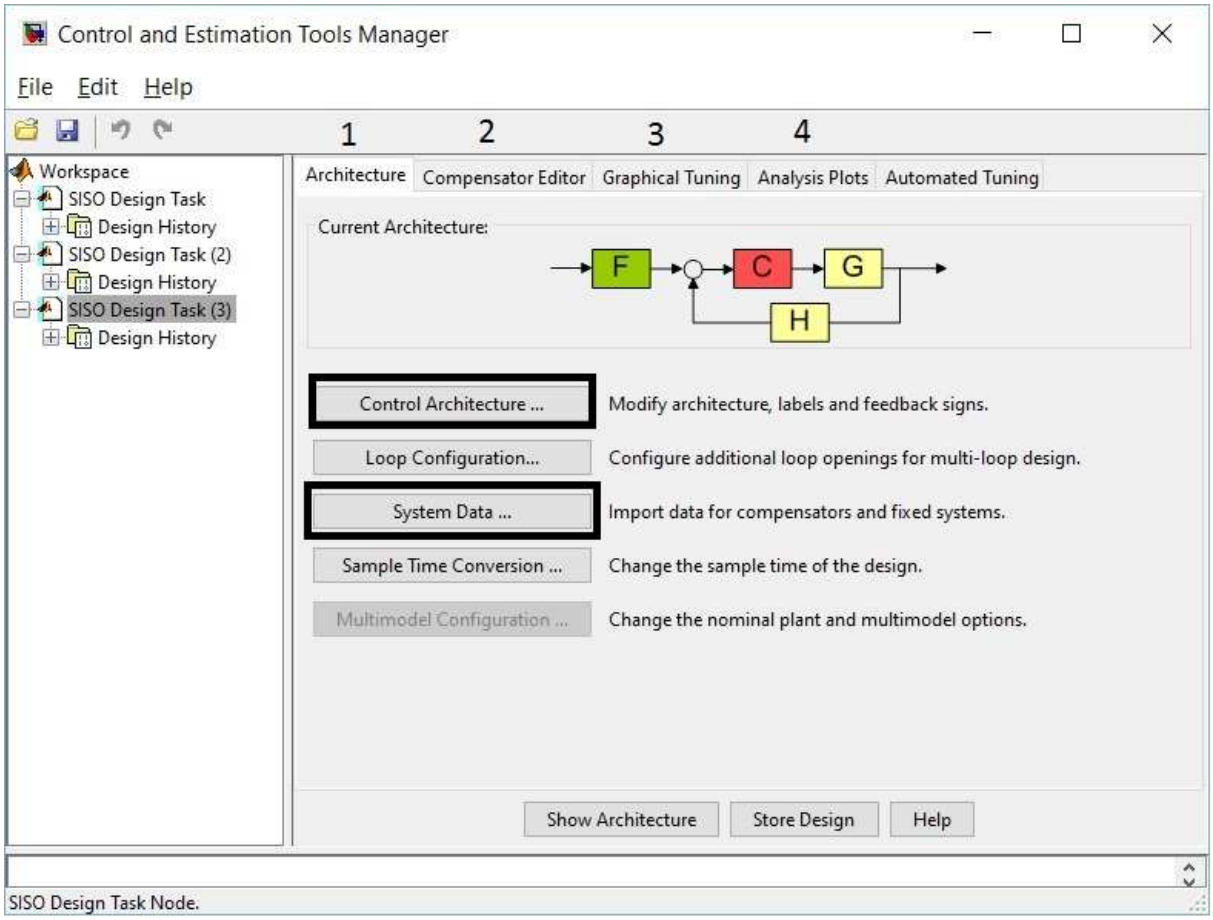

FIG. 2.1 – L'environnement de sisotool

Depuis le Command Window sous MATLAB, tapez et exécutez *sisotool*, une fenêtre s'ouvre. La conception de la loi de commande et l'évaluation des performances du système ainsi corrigé s'opèrent en ouvrant successivement les onglets de la gauche vers la droite.

1. L'onglet *Architecture* permet de définir la structure de la boucle à contrôler et les différentes fonctions de transfert.
   — bouton *Control Architecture* : vérifiez que la structure sélectionnée est celle désirée, relevez sur le schéma le nom des variables utilisées pour définir les fonctions de transfert et les signaux.
   — bouton *System Data* : allouez les fonctions de transfert dans les champs prévus à cet effet de sorte que la structure de la boucle proposée par *sisotool* soit cohérente avec vos notations. A ce stade, le correcteur $C = 1$. Le bouton *Browse* permet d'importer dans *sisotool* les fonctions de transfert déclarées dans *MATLAB*.
2. L'onglet *Compensator Editor* : C représente le gain statique du correcteur. A ce stade, laissez $C = 1$.
   Notez qu'il est possible de synthétiser des correcteurs plus sophistiqués en ajoutant des pôles et des zéros.
3. onglet *Graphical Tuning* : dans ce menu, vous choisissez le type de graphique utilisé pour la synthèse du correcteur, nous utiliserons le lieu de Black-Nichols en boucle ouverte et le diagramme de Bode en boucle fermée.
   — *plot1* : choisissez *open-loop* $\rightsquigarrow$ *nichols*
   — *plot2* : choisissez *closed loop* $\rightsquigarrow$ *closed-loop Bode*
   Cliquez ensuite sur le bouton *Show design plot*. Pour les différents graphiques, cliquez droit $\rightsquigarrow$ *grid*.
4. onglet Analysis Plot : pour afficher une réponse indicielle correspondant au réglage effectué :
   — sélectionnez *Plot type : step*
   — cochez *closed loop r to y*
   — bouton *Analysis plot*

Pour le réglage du correcteur, sous l'onglet *Graphical tuning*, translatez le lieu de Black-Nichols jusqu'à par exemple atteindre la marge de phase désirée. Il faut ensuite lire dans l'onglet *Compensator editor* la valeur de C ayant permis ce réglage.