Rapport de Projet de Robotique

Groupe n°7 : Étudiant 1, Étudiant 2, Étudiant 3, Étudiant 4

Date : June 4, 2025

École CentraleSupélec

Résumé :

Ce projet de robotique a exploré la manipulation de robots terrestres et de drones dans un environnement simulé, avec un objectif final d'implémentation dans une volière ROS2.

Contents

Introduction

Ce rapport présente le projet de robotique mené sur une période de deux mois. L'objectif principal était de concevoir un scénario permettant de manipuler des robots terrestres et des drones dans un environnement simulé. Ce projet s'inscrit dans le cadre de l'étude des systèmes multi-agents et de leur coordination dans des situations complexes.

Description du Projet

Conception du Scénario

Le scénario développé dans ce projet implique la manipulation de robots terrestres et de drones dans un environnement simulé. L'objectif est de coordonner ces robots pour accomplir des missions complexes, telles que la formation en groupe, l'évitement d'obstacles, et la descente contrôlée des drones. Les robots sont initialement positionnés selon des configurations spécifiques pour faciliter leur coordination.

Ce scénario de contrôle des robots provient d'une mise en scène qui s'inspire des noms des robots utilisés. Le voici:

Il était une fois, après une discussion houleuse pendant le dîner, le général Burger et le soldat Waffle qui se sont déclarés la guerre pour savoir lequel d'entre eux vaincra Monsieur Moutarde. Le général Burger est attendu pour une réunion très importante au Pincher pour discuter de la suite de la guerre. Cependant, ses positions ont été divulguées et le soldat Waffle va tenter de le tuer... Le principal souci est d'avoir un moyen de transport infaillible tout en se défendant contre les délinquants.

Cette mise en scène, bien que fictive, montre que l'utilisation de robot terrestre et de drone prend de plus en plus d'importance dans la façon dont les grandes nations font la guerre. En effet, le 1er juin 2025, l'Ukraine a mené une attaque coordonnée sur quatre bases aériennes russes en lançant 117 drones FPV, illustrant l'importance croissante des drones dans les conflits modernes. Ces systèmes, peu coûteux mais efficaces, permettent de mener des frappes ciblées à distance avec un risque humain réduit. Cette opération témoigne d'une évolution tactique vers une guerre automatisée mêlant reconnaissance, frappe et saturation ennemie [2].

Ce scénario utilise deux équipes de robots qui intéragissent entre eux et évoluent dans un environnement proche, il faudra donc que la gestion des collisions soient efficaces afin d'éviter tous contacts qui tout comme dans le cas d'une bataille, ferait échouer la mission.

Robots Utilisés et Leur Dynamique

Les robots utilisés dans ce projet incluent :

- **DCA** : Robot terrestre avec dynamique de type *singleIntegrator3D*.
- **Burger** : Véhicule terrestre avec dynamique de type *unicycle*.
- **Waffle** : Robot terrestre avec dynamique de type *unicycle*.
- **RMTT** : Drone avec dynamique de type *singleIntegrator3D*.
- **DCA_SI** : Robot terrestre immobile ou avec dynamique de type *singleIntegrator3D*.

Dynamiques des Robots et Défis de Contrôle

Les dynamiques des robots utilisées dans la simulation sont variées et adaptées à leurs rôles spécifiques. Voici les dynamiques principales :

- **Unicycle Dynamics** :

$$\dot{x} = v\cos(\theta),$$
$$\dot{y} = v\sin(\theta),$$
$$\dot{\theta} = \omega,$$

  où v est la vitesse linéaire, ω est la vitesse angulaire, et θ est l'orientation. Cette dynamique est utilisée pour les robots terrestres tels que *Burger* et *Waffle*. Les défis incluent :

  - **Mouvements réduits** : La dynamique ne permet des mouvements de translations que dans la direction de l'avant du robot, ce qui rend certains mouvements impossibles.
  - **Coordination** : Cette dynamique plus simple rend l'évitement de collision plus compliqué puisque certains mouvements demandent plus de temps à être effectués pour contourner.

- **Single Integrator Dynamics (3D)** :

$$\dot{x} = v_x,$$
$$\dot{y} = v_y,$$
$$\dot{z} = v_z,$$

  où v_x, v_y, et v_z sont les composantes de la vitesse dans les directions x, y, et z. Cette dynamique est utilisée pour les drones tels que *RMTT* et *DCA_SI*. Les défis incluent :

  - **Oscillations** : Implémenter des trajectoires oscillantes pour les drones, cette trajectoire plus compliquée demande de faire une implémentation plus approfondies en simulation.
  - **Stabilité** : Maintenir une stabilité dans les mouvements en 3D, avec des correcteurs plus complexes afin d'optimiser le comportement.

- **Stationary Dynamics** :

$$\dot{x} = 0,$$
$$\dot{y} = 0,$$
$$\dot{z} = 0,$$

  Cette dynamique est utilisée pour les robots immobiles tels que *DCA_SI* dans certaines phases de la simulation. Leur implémentation est notamment due à des contraintes de codes, en effet une flotte avec des robots de même types ne peut pas contenir qu'un seul robot. Il faut donc créer des robots immobiles pour y remédier. Les défis dans l'implémentation de ces robots incluent :

  - **Interaction avec les robots mobiles** : Assurer que les robots immobiles n'interfèrent pas avec les trajectoires des robots mobiles.

Ces dynamiques sont intégrées dans la simulation avec des lois de commande spécifiques pour répondre aux défis mentionnés. Cela permet de tester et valider les comportements des robots dans un environnement simulé avant leur déploiement réel.

Choix des Leaders-Followers et Stratégies de Commande

La stratégie *Leader-Follower* a été adoptée pour coordonner les robots. Dans cette approche, un robot leader guide les autres robots (followers) vers un objectif commun. Par exemple, le robot *DCA* agit comme leader, tandis que les autres robots suivent sa trajectoire en ajustant leur position et orientation.

Phases de la Simulation et Rôles des Robots

La simulation est divisée en plusieurs phases distinctes, chacune ayant des objectifs spécifiques et impliquant des rôles particuliers pour les robots :

- **Phase de Formation** :
  - Objectif : Les robots ajustent leur position pour maintenir une formation prédéfinie.
  - Rôles :
    - **DCA** : Agit comme leader et guide les autres robots vers la formation cible.
    - **Burger** : Suivent le leader en ajustant leur position relative.
    - **Waffle** : Maintient une position fixe à l'opposé de la formation pour équilibrer la structure.
    - **RMTT** : Drone qui surveille la formation depuis une altitude élevée.
    - **DCA_SI** : Suiveur immobile qui reste en position pour éviter les interférences.
- **Phase Forward** :
  - Objectif : Les robots avancent en ligne droite vers un objectif défini.
  - Rôles :
    - **DCA** : Continue de guider la flotte en tant que leader.
    - **Burger** : Maintiennent leur position relative tout en avançant.
    - **Waffle** : Suit le leader tout en restant à l'opposé de la formation.
    - **RMTT** : Drone qui surveille les mouvements de la flotte.
    - **DCA_SI** : Suiveur immobile qui reste en position.
- **Phase de Danse** :
  - Objectif : Les robots effectuent une danse circulaire autour du robot *Waffle*.
  - Rôles :
    - **DCA** : Participe à la danse en suivant une trajectoire circulaire.
    - **Burger** : Effectuent des mouvements synchronisés autour du *Waffle*.
    - **Waffle** : Sert de centre de la danse.
    - **RMTT** : Drone qui surveille la danse depuis une altitude élevée.
    - **DCA_SI** : Reste immobile pour éviter les interférences.
- **Phase d'Oscillation des Drones** :
  - Objectif : Les drones oscillent au-dessus du *Waffle* pour simuler des trajectoires complexes.
  - Rôles :
    - **RMTT** : Effectue des oscillations en altitude pour surveiller la flotte.

* **Waffle** : Sert de point de référence pour les oscillations.
* **DCA**, **Burger**, **DCA_SI** : Maintiennent leur position pour éviter les interférences.

Ces phases permettent de tester les capacités de coordination, de navigation et de contrôle des robots dans des scénarios variés.

Lois de Commande

Les lois de commande utilisées dans ce projet incluent :

- **Formation** : Les robots ajustent leur position pour maintenir une formation spécifique. La loi de commande est donnée par :

$$\mathbf{u}_i = -k_p(\mathbf{x}_i - \mathbf{x}_{\text{ref}}),$$

où $\mathbf{x}_i$ est la position du robot i, $\mathbf{x}_{\text{ref}}$ est la position de référence, et k_p est un gain proportionnel.

- **Suivi de Trajectoire** : Les robots suivent une trajectoire prédéfinie en ajustant leur vitesse et orientation. La loi de commande est donnée par :

$$\mathbf{u}_i = -k_p(\mathbf{x}_i - \mathbf{x}_{\text{target}}) + k_d\dot{\mathbf{x}}_i,$$

où $\mathbf{x}_i$ est la position actuelle du robot i, $\mathbf{x}_{\text{target}}$ est la position cible, k_p est un gain proportionnel, et k_d est un gain dérivé pour stabiliser le mouvement.

- **Évitement de Collision** : Les robots détectent et évitent les obstacles en ajustant leur trajectoire. La loi de commande est donnée par :

$$\mathbf{u}_i = -k_p\nabla\phi(\mathbf{x}_i),$$

où $\phi(\mathbf{x}_i)$ est une fonction de potentiel qui augmente près des obstacles.

- **Oscillation des Drones** : Les drones oscillent au-dessus des robots terrestres en suivant des trajectoires sinusoidales. La loi de commande est donnée par :

$$\begin{aligned}\dot{x} &= A_x \sin(\omega_x t),\\ \dot{y} &= A_y \cos(\omega_y t),\\ \dot{z} &= k_z(z_{\text{target}} - z),\end{aligned}$$

où A_x et A_y sont les amplitudes, ω_x et ω_y sont les fréquences, et k_z est un gain proportionnel pour la descente en z.

Zoom: Fonction de Potentiel pour l'Évitement de Collision

Pour assurer une navigation sûre et maintenir l'intégrité de la formation, un mécanisme d'évitement de collision est implémenté en utilisant une fonction de potentiel. Cette fonction applique des forces de répulsion aux robots suiveurs (à l'exclusion du leader et des robots stationnaires) lorsqu'ils s'approchent les uns des autres dans un rayon défini.

La force de répulsion $u[i,:]$ pour chaque robot i est calculée comme suit :

$$u[i,:] += k_R \cdot \frac{(X[i,:] - X[j,:])}{\|X[i,:] - X[j,:]\|} \cdot \left(1 - \frac{\|X[i,:] - X[j,:]\|}{\text{avoidance\_radius}^2}\right)^2$$

Où :

- k_R: Gain pour la force de répulsion.
- $X[i,:]$: Position du robot i.
- $X[j,:]$: Position du robot j.
- $\|X[i,:] - X[j,:]\|$: Distance euclidienne entre les robots i et j.
- avoidance_radius: Rayon dans lequel l'évitement de collision est activé.

Cette fonction de potentiel garantit que la force de répulsion augmente à mesure que la distance entre les robots diminue, empêchant ainsi les collisions. L'atténuation progressive de la force de répulsion à mesure que les robots s'éloignent est réalisée grâce au terme $\left(1 - \frac{\|X[i,:]-X[j,:]\|}{\text{avoidance\_radius}^2}\right)^2$ [1].

Ce mécanisme est crucial pour maintenir la sécurité et l'efficacité de la flotte de robots pendant les opérations.

Choix des coefficients des Lois de Commande

Les lois de commandes mises en place n'ont pas été très performantes dès le début, il a donc fallu ajuster les coefficients de chaque loi de commande pour obtenir un comportement satisfaisant. La stratégie pour leur choix est la suivante:

- **Formation** : Le gain proportionnel k_p a été ajusté pour assurer que les robots suivent rapidement le leader sans oscillations excessives. Un k_p trop élevé peut entraîner des oscillations, tandis qu'un k_p trop bas peut ralentir la réponse.
- **Suivi de Trajectoire** : Les gains proportionnels k_p et dérivés k_d ont été choisis pour équilibrer la rapidité de suivi et la stabilité. Un k_p élevé permet un suivi rapide, mais peut causer des oscillations, tandis qu'un k_d élevé aide à amortir ces oscillations. Nous avons donc augmenter la rapidité du système avec un gain proportionnel élevé et pour amortir les oscillations qui sont apparues une fois la rapidité voulues atteinte, nous avons ajouté un gain dérivé k_d pour stabiliser le mouvement.
- **Évitement de Collision** : Le gain de répulsion k_R a été ajusté pour assurer que les robots maintiennent une distance de sécurité sans être trop agressifs, ce qui pourrait perturber la formation. Un k_R trop élevé peut entraîner des mouvements erratiques, tandis qu'un k_R trop bas peut ne pas prévenir les collisions efficacement.

L'enjeu du choix des coefficients a été de trouver le bon équilibre entre les valeurs de coefficients entre les différentes lois de commandes. En effet, bien que la valeur absolue des coefficients soit importante, il faut que leurs valeurs relatives soient cohérentes pour que le comportement des lois de commandes combinées n'empiètent pas les uns sur les autres.

Implémentation de la Simulation

La simulation est implémentée en utilisant une structure basée sur des classes, principalement les classes `FleetSimulation` et `Robot`. Voici une explication détaillée :

- **Classe** `FleetSimulation` :
  - **Objectif** : Gérer la simulation d'une flotte de robots, y compris leurs dynamiques, lois de commande, et visualisation.

  - **Caractéristiques principales** :
    * **Initialisation** : Définit la flotte de robots, le temps de début (`t0`), le temps de fin (`tf`), et le pas de temps (`dt`).
    * **Gestion des données** : Collecte et stocke les données sur les états des robots, leurs trajectoires, et les entrées de commande au fil du temps.
    * **Visualisation** : Fournit des méthodes pour animer les mouvements des robots en 2D et 3D, tracer les trajectoires, et afficher les entrées de commande.

- **Classe** `Robot` :
  - **Objectif** : Représenter des robots individuels avec des dynamiques et des états spécifiques.
  - **Caractéristiques principales** :
    * **Initialisation** : Configure les dynamiques du robot (par exemple, `singleIntegrator2D`, `unicycle`) et son état initial.
    * **Gestion des états** : Maintient l'état actuel du robot (par exemple, position, orientation) et le met à jour en fonction des entrées de commande.
    * **Conversion des commandes** : Convertit les entrées de vitesse cartésienne en vitesses linéaires et angulaires pour les robots avec des dynamiques de type `unicycle`.

- **Workflow de la Simulation** :
  - **Création de Flottes** : Les robots sont instanciés à l'aide de la classe `Robot` et regroupés en flottes. Chaque flotte est assignée des dynamiques spécifiques et des positions initiales.
  - **Algorithmes de Commande** : Les lois de commande (par exemple, formation, suivi de trajectoire, évitement de collision) sont appliquées pour calculer les entrées de vitesse pour chaque robot. Des algorithmes spécialisés comme `drone_oscillation_3d` gèrent des comportements complexes tels que les oscillations des drones.
  - **Intégration** : La classe `FleetSimulation` intègre le mouvement de tous les robots au fil du temps en utilisant les entrées de commande calculées.
  - **Visualisation** : La simulation fournit des animations en temps réel des mouvements des robots en 2D et 3D. Les trajectoires, états, et entrées de commande sont tracés pour l'analyse.

Cette structure permet de tester les capacités de coordination, de navigation, et de contrôle des robots dans des scénarios variés.

Implémentation dans ROS2

L'implémentation dans ROS2 nécessite l'adaptation des lois de commande et des dynamiques des robots pour fonctionner avec les outils et les bibliothèques de ROS2. Cela inclut l'utilisation de messages ROS pour communiquer entre les différents composants du système, ainsi que l'adaptation des boucles de contrôle pour fonctionner avec le système de temps de ROS2.

Les principales étapes de l'implémentation dans ROS2 incluent :

- **Conversion des dynamiques** : Les dynamiques des robots ont été converties en nœuds ROS, permettant leur intégration dans l'écosystème ROS2.

- **Publication et abonnement aux messages** : Les robots publient leur état (position, vitesse, etc.) et s'abonnent aux messages de commande pour recevoir les instructions du leader.
- **Utilisation des services ROS2** : Des services ROS2 ont été utilisés pour permettre une communication bidirectionnelle entre les robots et le serveur de simulation.
- **Gestion des temporisations** : Les temporisations ont été gérées à l'aide des outils de temps de ROS2 pour synchroniser les mises à jour des états des robots et l'envoi des commandes.

Conclusion

Ce projet de robotique a permis d'explorer la coordination de robots terrestres et de drones dans un environnement simulé. Les principales réalisations incluent :

- L'implémentation réussie de différentes dynamiques de robots (unicycle, single integrator)
- La mise en place de stratégies de formation et d'évitement de collisions
- Le développement de lois de commande robustes pour la coordination des robots
- La transition vers une implémentation ROS2 pour des applications réelles

Les défis rencontrés, notamment dans la gestion des dynamiques différentes et la coordination inter-flottes, ont été surmontés grâce à des solutions techniques appropriées. Ce travail pose les bases pour de futures applications dans des scénarios réels de robotique collaborative.

Bibliographie

[1] Fabio. *LNR p4*. 2018. URL: https://home.mis.u-picardie.fr/~fabio/Eng/documenti/Teaching/LNR18-19/LNR_p4.pdf.

[2] Le Monde. *Ukraine : Kiev lance une vaste attaque de drones contre quatre bases russes*. 2025. URL: https://www.lemonde.fr/international/article/2025/06/03/ukraine-kiev-lance-une-vaste-attaque-de-drones-contre-quatre-bases-russes_6236000_3210.html.