Projet expérimental - IFT-7026

William Fecteau

Supervisé par François Pomerleau

1 Introduction

En robotique mobile, la modélisation du mouvement d'un véhicule autonome est une tâche critique au succès du système. En effet, avec un modèle défaillant, plusieurs tâches comme la localisation, la planification de chemin ainsi que le suivi de chemin hériteront des erreurs de prédictions du modèle, rendant la navigation plus difficile.

Cependant, il est compliqué d'avoir un modèle qui performe bien sur plusieurs types de terrain comme la glace, le gravier, la terre et l'asphalte. Afin de résoudre ce problème, on peut récolter des données avec un véhicule sur le terrain pour ensuite en apprendre le modèle. Pour ce faire, le Norlab utilise Data-driven Robot Input Vector Exploration (DRIVE) [1] comme protocole afin de récolter des données et entraîner un modèle différentiel idéal [2] avec une correction pour le glissement sur leur plateforme Warthog.

Une future extension du protocole DRIVE vise à permettre l'apprentissage d'un modèle sur un véhicule équipé d'une géométrie Ackermann. Toutefois, le Norlab ne dispose actuellement d'aucun véhicule de ce type, ce qui limite la mise en œuvre de cette extension.

Dans le cadre de ce cours, j'ai collaboré avec Nicolas Lauzon, un autre étudiant, pour intégrer un véhicule à géométrie Ackermann au format 1/5, créant ainsi un banc d'expérimentation adapté à ce type de développement. Ensuite, j'ai effectué des expériences préliminaires afin de préparer le terrain pour l'extension du projet DRIVE. Ces premières étapes ont consisté principalement à intégrer et tester des composants clés sur le véhicule 1/5, notamment sa boucle de contrôle et la mise en place de la cartographie et la localisation. Des expériences ont été menées sur le Grand Axe afin d'évaluer la performance du modèle et identifier les ajustements nécessaires avant d'intégrer l'apprentissage de la dynamique sur un véhicule. Tout le code du robot est disponible sur le répertoire hedgehog_ws du Norlab. Ces expériences ont permis de valider les aspects techniques de la plateforme et de comparer le modèle du véhicule à la réalité terrain.

2 Modèle de déplacement Ackermann

La géométrie Ackermann est une configuration mécanique employée pour des véhicules ayant une direction à roues avant, comme les voitures. Ce modèle repose sur l'hypothèse que toutes les roues pivotent autour d'un centre de rotation instantané en commun. Ainsi, le véhicule effectue une trajectoire circulaire autour de ce point instantané. Pour y arriver, les deux roues avant possèdent des angles de braquage différents comme on peut observer sur la Figure 1

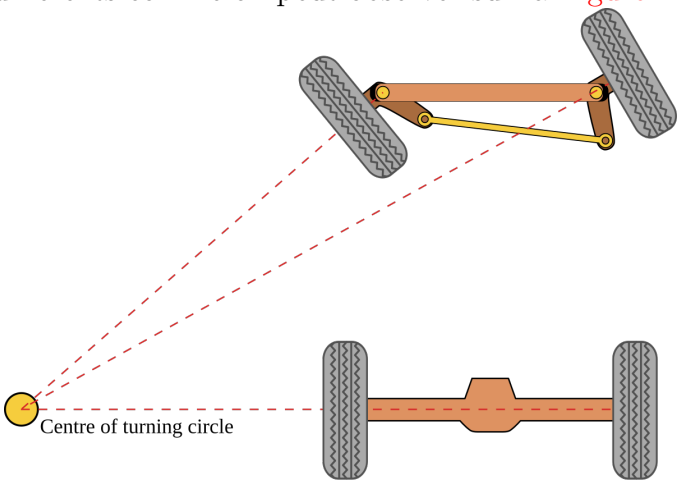

Figure 1 – Centre de rotation instantané d'un véhicule à géométrie Ackermann

Bien qu'il est possible de modéliser mathématiquement la dynamique complète de la géométrie Ackermann, celle-ci est coûteuse en temps de calcul et en identification de paramètres. Alors, une simplification courante est d'utiliser le modèle de la bicyclette cinématique, qui combine ensemble les deux roues sur un même essieu afin de former une bicyclette comme sur la Figure 2. Cette approche est particulièrement utile, car elle capture l'essentiel du comportement du véhicule avec peu d'équations et de paramètres.

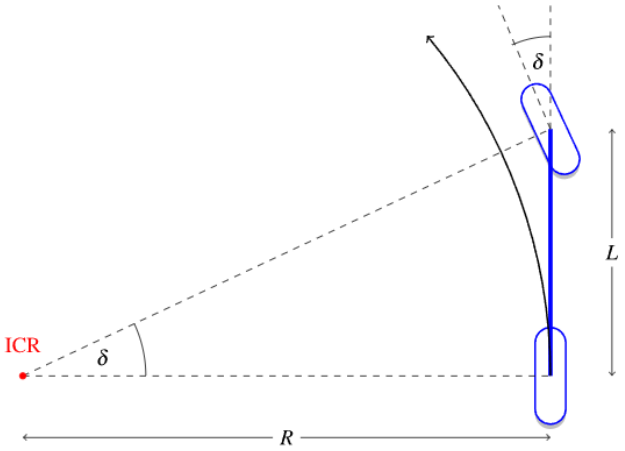

Figure 2 – Trajectoire du modèle de la bicyclette cinématique

Ce modèle possède 3 degrés de liberté (x, y et ϕ). Avec un véhicule avec une commande en vitesse v et en angle de braquage δ, on obtient :

$$\begin{aligned} R &= \frac{L}{\tan\delta} \\ \dot{x} &= v\cos\phi \\ \dot{y} &= v\sin\phi \\ \dot{\phi} &= \frac{v}{R} \end{aligned} \tag{1}$$

Cependant, la simplicité de ce modèle vient à un certain coup. En effet, on assume que le mouvement du véhicule est parfaitement géométrique, c'est-à-dire qu'on ignore complètement les interactions pneu-terrain. Ainsi, on s'attend à ce que ce modèle soit peu précis lorsque les pneus atteignent leur point de saturation et glisse sur le terrain [3].

Pour remédier à cette situation, il est possible de caractériser le véhicule sur un type de terrain spécifique et d'ajouter une correction pour le glissement, comme l'ont proposé Baril *et al.* avec un modèle à géométrie différentielle [1].

3 Vue d'ensemble de la plateforme

3.1 Mécanique

La plateforme est basée sur le 1/5 DBXL-E Desert Buggy sur lequel nous avons ajouté une plaque de plexiglass afin de supporter les capteurs. Afin de pouvoir sortir le véhicule à l'extérieur, toute l'électronique se situe dans un *Pelican Case* à l'arrière du véhicule. Sur la plateforme, nous avons gardé le moteur ainsi que le servo d'origine. Voici les différents capteurs/composantes que nous avons ajoutés :

— LiDAR : Robosense RS Helios-16
— *Inertial Measurement Unit* (IMU) : VN-100 Rugged IMU Thermal calibrated
— *Electronic Speed Controller* (ESC) : Vesc 6 MkVI
— Ordinateur de bord : Intel NUC 12 i7-1260P 16GB memory, 512GB SSD

Afin de maximiser le champ de vision du LiDAR, nous avons imprimé une pièce 3D afin de le surélever au-dessus du *Pelican Case* comme on peut voir sur la figure 3

Figure 3 – Plateforme de course format 1/5 qui a été bâtie au cours de la session

3.2 Électronique

Initialement, l'alimentation du système entier devait se faire avec seulement une batterie LiPO 4S. Cependant, lors du choix des composantes, nous avons sélectionné un ordinateur embarqué nécessitant 19V avec 6A. Puisqu'on voulait éviter le développement complet d'un *powerboard*, nous avons récupéré celui développé par le club étudiant Véhicule Autonome Université Laval (VAUL)

pour des véhicules format 1/10. Par contre, ce *powerboard* divise 6A à toutes les composantes. Alors, il était impossible d'alimenter l'ordinateur avec celui-ci, il a donc fallu ajouter une deuxième batterie LiPO 3S avec un convertisseur DC/DC pour ce faire (Voir Figure 4).

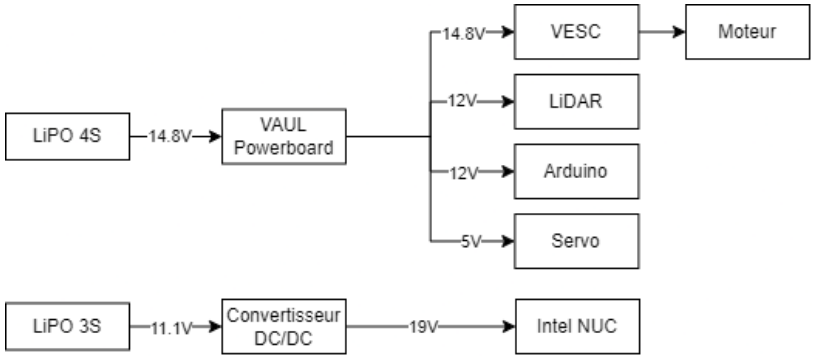

Figure 4 – Schéma électrique de la distribution des tensions dans le système

Les commandes en vitesse du système passent par le *Vedder Electronic Speed Controller* (VESC), un contrôleur de vitesse *open-source* pour moteurs sans balais (*brushless*). Le VESC reçoit les commandes en vitesse en *Electronic Rotation Per Minute* (eRPM) et il gère la communication avec le moteur via *Pulse Width Modulation* (PWM). On peut calculer le RPM final à l'aide de la formule suivante :

$$RPM = \frac{2 \cdot eRPM}{n} \tag{2}$$

Où n est le nombre de pôles du moteur, ce qui dans notre cas est $n = 6$.

Pour ce qui est des commandes de braquage, le Arduino reçoit des commandes de braquages entre $[45, 135]$ deg. Une commande de 90 deg correspond au braquage en position neutre ($\delta = 0$). À la réception de cette commande, le Arduino commande le servo via PWM.

3.3 Limites théoriques du système

Avant de commencer les expériences, il est bien de calculer les limites théoriques du système afin d'avoir une idée des capacités de la plateforme. Puisque le moteur est noté 800 kV, il produit 800 RPM/V. Sachant que le *gear ratio* est de 50: 13, que le diamètre des roues est 0.15 m et qu'on l'alimente par une batterie LiPO 4S (Voltage nominal de 14.8 V), on peut calculer la vitesse théorique maximale :

$$\begin{aligned}
\text{Shaft RPM} &= 800 \text{ RPM/V} \cdot 14.8\ V = 11840 \text{ RPM} \\
\text{Wheel RPM} &= \text{Shaft RPM} \cdot \frac{13}{50} = 3078.4 \text{ RPM} \\
v_{wheel} &= \frac{\pi \cdot 0.15 \text{ m} \cdot \text{Wheel RPM}}{60 \text{ s}} = 24.18 \text{ m/s}
\end{aligned} \tag{3}$$

Ensuite, on peut trouver une approximation pour le moment de force (*torque*) du moteur et ainsi calculer l'accélération linéaire maximale du véhicule. Sachant que le VESC peut tolérer des pics de courant de 120 A et que la masse du véhicule est 20 kg :

$$\begin{aligned}
\omega_{shaft} &= \text{Shaft RPM} \cdot \frac{2\pi}{60 \text{ s}} = 1239.88 \text{ rad/s} \\
\tau_{shaft} &= \frac{120 \text{ A} \cdot 14.8 \text{ V}}{\omega_{shaft}} = 1.43 \text{ Nm} \\
\tau_{wheel} &= \tau_{shaft} \cdot \frac{50}{13} = 5.5 \text{ Nm} \\
a_{wheel} &= \frac{\tau_{wheel}}{0.15 \text{ m} \cdot 20 \text{ kg}} = 1.83\ m/s^2
\end{aligned} \tag{4}$$

Donc, on s'attend à un véhicule pouvant aller à une vitesse maximale de 24.18 m/s, soit 87.05 km/h, avec des accélérations linéaires maximales de 1.83 m/s^2. Il est important de se rappeler que ces approximations assument aucune perte d'énergie. Ainsi, celles-ci donnent simplement des bornes supérieures sur la capacité du véhicule.

4 Analyse des commandes du système

4.1 Commandes de vitesse

Pour commander correctement la vitesse au VESC, il est nécessaire de convertir la vitesse désirée (m/s) en vitesse de rotation du moteur (eRPM). Cette conversion peut être réalisée à l'aide de la formule suivante, dérivée des équations 2 et 3 :

$$\text{eRPM} = \frac{60v}{0.15\pi} \cdot \frac{50}{13} \cdot \frac{6}{2} = 1469.12v = Qv \tag{5}$$

Ainsi, la transformation de vitesse en m/s vers eRPM revient à multiplier par un terme constant. Cependant, cette valeur théorique de 1469.12 représente une borne inférieure, car il est nécessaire de commander une vitesse de rotation plus élevée pour compenser les pertes énergétiques dues aux frottements et autres phénomènes dissipatifs. Une identification expérimentale de ce gain (Q) est donc requise pour refléter les spécificités du véhicule réel.

4.1.1 Identification expérimentale du gain Q

Pour déterminer le gain Q, un nœud d'odométrie en boucle ouverte (*open loop*) a été développé en utilisant les équations 1. À chaque envoi de commande (vitesse et angle de braquage), celle-ci est intégrée et donne une estimation du déplacement. Sur une surface plane avec un faible glissement, cette odométrie est précise localement, mais dérive avec le temps.

Ensuite, on mesure une distance de 2 m en ligne droite dans les corridors de l'université et, par essais et erreurs, on trouve la valeur de $Q = 3850$ de manière à ce que l'odométrie estime correctement la distance parcourue. Cette calibration a été réalisée à une vitesse constante de 1 m/s afin de minimiser les effets de glissement.

4.1.2 Validation expérimentale du gain Q

Pour vérifier que l'équation 5 reste valable pour différentes vitesses avec un gain constant, on conduit une autre expérience. Au lieu d'utiliser l'odométrie à boucle ouverte (*open loop*), on mesure le temps nécessaire pour parcourir une distance de 5 m avec différentes vitesses commandées.

À partir des données présentées dans le tableau 1, on observe une erreur relative maximale de 9 %. Ces écarts peuvent être attribués en partie à l'imprécision humaine lors de l'utilisation du chronomètre, particulièrement à des vitesses élevées comme 3 m/s, où le chronométrage précis devient plus difficile à réaliser. Il aurait été intéressant de collecter des données à plus de 3 m/s, mais de telles vitesses dans un corridor de l'université sont trop difficiles à maintenir et à chronométrer.

Table 1 – Comparaison des vitesses commandées/calculées pour valider le gain Q à différentes vitesses

Vitesse commandée (m/s)	Essai 1 (s)	Essai 2 (s)	Essai 3 (s)	Temps moyen (s)	Vitesse calculée (m/s)
1	4.89	4.94	4.89	4.91	1.01
2	2.26	2.29	2.32	2.29	2.18
3	1.44	1.59	1.60	1.54	3.25

4.1.3 Limite de vitesse

En utilisant le gain Q déterminé expérimentalement, il est possible de raffiner l'estimation de la vitesse théorique maximale du véhicule. En effet, le gain Q permet d'intégrer les pertes énergétiques dans le calcul de la vitesse. On obtient alors l'expression suivante pour la vitesse maximale :

$$v_{max} = \frac{\text{Max shaft eRPM}}{Q} = \frac{35520}{3850} = 9.23 \text{ m/s} \tag{6}$$

Cela donne une vitesse maximale de 9.23 m/s, soit environ 33.21 km/h, une valeur bien plus réaliste et représentative des capacités réelles de la plateforme.

4.2 Commandes de braquage

Dans le système d'axes du véhicule, l'axe des x pointe vers l'avant du véhicule, y vers la gauche du véhicule et z vers le ciel. L'angle de braquage des roues, noté δ, est mesuré en radians par rapport à l'axe des x dans le plan xy. Un angle de braquage positif correspond à un virage vers la droite, tandis qu'un angle négatif indique un virage vers la gauche.

Cependant, le servo du véhicule accepte des commandes d'angle dans la plage de $[45, 135]$ deg. Par conséquent, il est nécessaire de convertir l'angle de braquage δ (en radians) afin de le faire correspondre à cette plage. La transformation se fait selon la formule suivante :

$$\text{Commande servo} = \frac{-180\delta}{\pi} + 90 \tag{7}$$

où $\delta \in [\frac{-\pi}{4}, \frac{\pi}{4}]$

Toutefois, en tenant compte de la friction et du poids du véhicule, il est probable que le servo ne puisse pas atteindre toute cette plage en pratique. Par conséquent, il est essentiel d'identifier la plage réelle de commandes exploitables par le robot en fonction de ces limitations mécaniques.

4.2.1 Identification expérimentale de la plage δ

Pour déterminer cette plage de commandes, on peut faire exécuter au véhicule un demi-cercle vers la gauche et vers la droite en utilisant un angle de braquage maximal. Ensuite, on mesure le diamètre du cercle tracé. À partir de cette mesure et en utilisant le modèle de la bicyclette (1), on peut calculer l'angle de braquage correspondant à l'aide de la formule suivante :

$$\delta = \arctan \frac{2L}{d} \tag{8}$$

Où d est le diamètre du cercle mesuré.

À l'issue de cette expérience, nous obtenons la plage d'angles de braquage $\delta \in [-0.585, 0.532]$ rad. On observe une asymétrie entre les mouvements à gauche et à droite. Par conséquent, pour simplifier, on peut restreindre la plage d'angles de braquage atteignable à $\delta \in \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]$ rad. On doit donc ajuster la formule de commande au servo pour la suivante :

$$\text{Commande servo} = \frac{-270\delta}{\pi} + 90 \tag{9}$$

où $\delta \in [\frac{-\pi}{6}, \frac{\pi}{6}]$

5 Analyse de la performance du modèle bicyclette

En nous basant sur les limites théoriques du système et sur notre modèle de commande, nous sommes désormais en mesure d'évaluer les performances du modèle de la bicyclette cinématique sur la plateforme. Pour ce faire, nous nous inspirons du protocole DRIVE [1] en envoyant une série de commandes aléatoires d'une durée de 6 secondes. Cela permet de capturer les dynamiques du mouvement, tant en régime transitoire qu'en régime permanent. À partir des données recueillies, il devient possible de calculer l'erreur en translation et en rotation sur la trajectoire réalisée.

5.1 Cartographie et localisation

Pour obtenir la trajectoire parcourue par le véhicule, il est indispensable qu'il puisse se localiser dans son environnement. Pour cela, la *stack* logicielle de cartographie et de localisation développée par le Norlab a été utilisée. Cette *stack* repose sur l'algorithme d'alignement de nuages de points *Iterative Closest Point* (ICP), en s'appuyant sur les données d'un LiDAR 3D. L'algorithme d'ICP permet d'ajuster et d'aligner successivement les nuages de points acquis par le capteur afin d'estimer la position et l'orientation du véhicule dans l'espace, tout en construisant une carte précise de l'environnement.

Étant donné la complexité des tâches de cartographie et de localisation en temps réel, des *rosbags* ont été enregistrés afin de réaliser tous les calculs en post-traitement. Cette approche permet d'éviter les simplifications généralement nécessaires pour assurer une exécution en temps réel des algorithmes.

5.2 Collecte de données

Dans un premier temps, le véhicule est conduit à une vitesse de 1 m/s autour du Grand Axe afin de collecter les données nécessaires à la création d'une carte de l'environnement. Une fois cette étape complétée, le véhicule est positionné au centre de la carte, et le nœud de contrôle pseudo-DRIVE est activé. Ce nœud génère aléatoirement une commande parmi les valeurs de vitesse linéaire $v \in [0.5, 2]$ m/s et d'angle de braquage $\delta \in [\frac{-\pi}{6}, \frac{\pi}{6}]$ rad. La commande ainsi générée est exécutée pendant une durée de 6 secondes lorsqu'un bouton de la manette est pressé. La plage de vitesse ne se rend pas jusqu'à 9.22 m/s comme la limite théorique calculé puisqu'à cette vitesse, la localisation du robot échoue et l'exécution de ce protocole devient plus risquée.

Si, au cours de l'exécution, le véhicule s'éloigne trop de la zone couverte par la carte ou si l'exécution des 6 secondes est interrompue, le véhicule est ramené manuellement au centre de la carte et le processus est répété. Cette méthodologie permet de tester le comportement du système dans diverses configurations de commande, tout en restant dans les limites de la carte générée.

Sur la figure 5, on peut observer le sol du Grand Axe au moment de la collecte de données. Le sol était couvert de neige et bossu.

Figure 5 – Condition du sol du Grand Axe au moment de la collecte de données

5.3 Traitement des données

Pour débuter, une carte est générée à l'aide des données enregistrées lorsque le véhicule se déplace lentement autour du Grand Axe. Pour ce faire, un *rosbag* est rejoué à une vitesse de 0.5x avec le nœud d'odométrie et le *package* norlab_icp_mapper_ros.

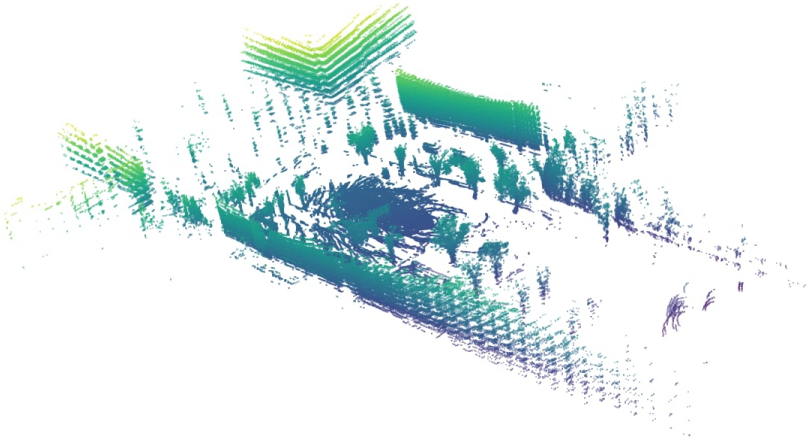

Figure 6 – Carte 3D du Grand Axe utilisée pour la localisation du robot

À l'aide de la carte obtenue (figure 6), la localisation du robot est maintenue en tout temps lors de l'exDes exp´eriences ont ´et´e men´es sécution d'un second *rosbag*, également rejoué à une vitesse de 0.5x. Parallèlement à la localisation, un nœud de planification est activé pour prédire les mouvements attendus du robot sur des périodes de 6 secondes. Grâce à cette configuration, il est possible de comparer les mouvements prévus avec ceux réalisés, comme sur la figure 7.

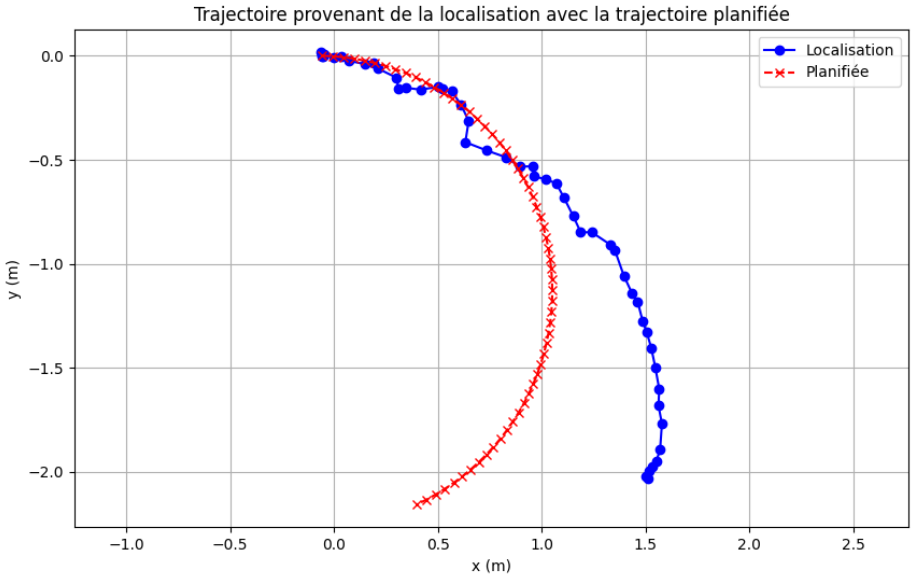

Figure 7 – Trajectoire planifiée et réalisée suite à l'exécution d'une commande de 6 secondes

5.4 Résultats

L'analyse des résultats révèle un comportement attendu, mais néanmoins significatif concernant la précision entre la trajectoire planifiée et celle réellement suivie par le robot. Comme illustré dans la figure 7, au début de la trajectoire, le mouvement réel du robot suit de très près la planification réalisée. Cependant, après un certain temps, les erreurs causées par le glissement sur la neige s'accumulent et séparent les deux trajectoires. On peut observer ce phénomène de manière analytique avec les graphiques des erreurs au fil du temps sur la figure 8.

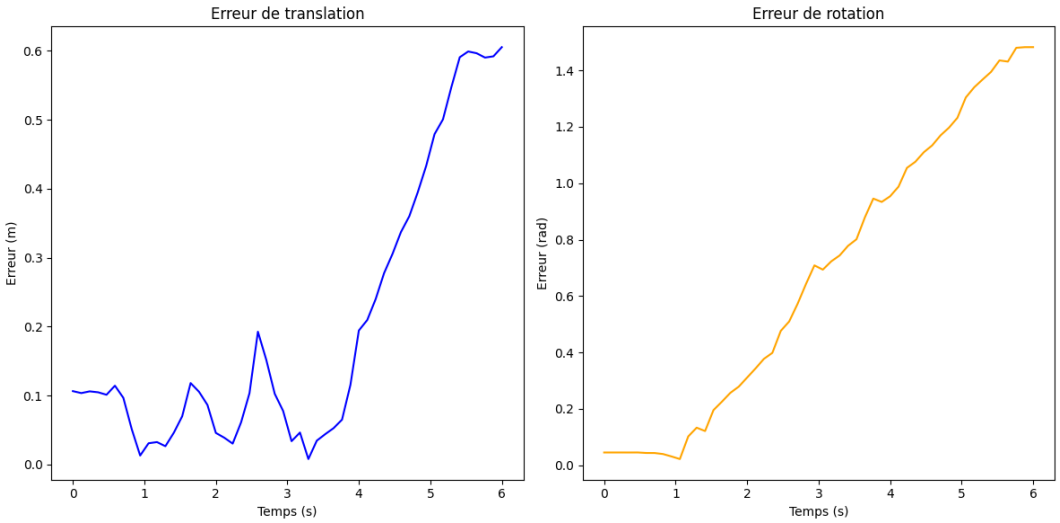

Figure 8 – Erreur en translation et rotation entre la trajectoire planifiée et la trajectoire réalisée

À partir des erreurs en translation et en rotation, on peut calculer le *Root Mean Square Error* (RMSE) pour chacune d'elles. Pour la figure 8, on obtient un RMSE en translation de 0.279 m et un RMSE en rotation de 0.842 rad.

Sur toutes les étapes réalisées, le plus grand RMSE en translation est de 0.758 m et le plus grand RMSE en rotation est de 1.276 rad. Ces deux résultats ont été obtenus avec une vitesse supérieure à 1.5 m/s. Ainsi, on peut s'attendre à ce que plus on augmente la vitesse, plus il y aura de glissement, ce qui rend le modèle bicyclette de moins en moins précis.

6 Problèmes connus

6.1 Odométrie sans encodeurs

L'absence d'encodeurs sur la plateforme constitue un défi majeur pour obtenir une estimation précise de l'odométrie. Sans encodeurs, le système doit reposer sur des sources alternatives, telles qu'une centrale inertielle ou une caméra. Dans notre cas, les données de l'IMU sont fusionnées avec l'odométrie d'ICP à l'aide du *package* imu_odom développé par le Norlab. Cependant, comme la position est obtenue par une double intégration des accélérations, la dérive accumulée est quadratique. En conséquence, l'odométrie basée uniquement sur l'IMU ne peut fonctionner de manière fiable sur de longues périodes sans un recalage fréquent via ICP, faute de quoi la précision se dégrade rapidement.

De plus, il serait pertinent de positionner l'IMU plus près du LiDAR. Actuellement, l'IMU est fixé sur la plaque de plexiglass alors que le LiDAR est situé 21 cm plus haut. En rapprochant l'IMU du LiDAR, on permettrait une intégration des accélérations plus cohérente avec la position du LiDAR, ce qui améliorerait la précision du recalage effectué par l'ICP.

Donc, l'ajout d'encodeurs de roues permettrait d'introduire un *feedback* extéroceptif capable de réduire la dérive de l'IMU. Avec de tels encodeurs, il serait alors possible d'utiliser le nœud imu_and_wheel_odom, actuellement employé sur le robot Warthog du Norlab, pour bénéficier d'une estimation plus robuste et précise de l'odométrie.

6.2 Contrôle du servo par Arduino

Initialement, le contrôle du servo de direction était assuré par le VESC, offrant une solution intégrée et centralisée pour gérer les commandes du robot. Cependant, une défaillance du connecteur PPM sur le VESC a rendu cette approche inutilisable, nécessitant la mise en place d'une solution alternative pour maintenir le contrôle du servo.

Pour résoudre ce problème, un Arduino a été employé comme interface de contrôle. Bien que cette solution ait permis de rétablir le fonctionnement du servo, elle a introduit une complexité supplémentaire en ajoutant une nouvelle couche de communication et de câblage. Un ajout d'une composante ajoute par le fait même une autre source de problèmes. Par exemple, lors d'un déploiement sur le Grand Axe, une soudure défectueuse sur l'Arduino a cédé, obligeant à interrompre l'opération.

Afin d'assurer une fiabilité accrue et de simplifier l'architecture de la plateforme, il serait pertinent d'envisager la réparation ou le remplacement du connecteur PPM du VESC. Cette démarche permettrait de retirer l'Arduino et de revenir à une solution plus robuste et centralisée.

6.3 Asservissement par deux batteries

Actuellement, le système utilise deux batteries distinctes : l'une alimente le moteur et les capteurs, tandis que l'autre est dédiée à l'ordinateur de bord. Cette séparation offre l'avantage de protéger les composants sensibles contre les fluctuations de tension causées par le moteur. Cependant, elle engendre également certains inconvénients.

L'un des principaux défis réside dans la gestion de la charge des deux batteries. En pratique, la batterie de l'ordinateur de bord se décharge généralement plus rapidement que celle alimentant les capteurs et le moteur. Par conséquent, même si l'une des batteries reste opérationnelle, il est souvent nécessaire d'interrompre le fonctionnement du système pour remplacer les deux batteries simultanément. Cette approche est préférable, car une décharge excessive de l'une des batteries peut entraîner une diminution des performances globales du système.

Par ailleurs, la présence de deux batteries dans un même système peut créer des boucles de masse susceptibles de provoquer des interférences dans les signaux des capteurs ou des contrôleurs. Ces interférences peuvent altérer la précision ou la stabilité du système. Il est donc crucial de s'assurer que les *grounds* sont correctement partagés et configurés pour éviter ces problèmes.

Bien que cette configuration soit fonctionnelle, une alternative prometteuse consisterait à développer un *powerboard* sur mesure afin d'utiliser une seule batterie LiPO 8S. Une telle solution simplifierait la gestion de l'alimentation, réduirait la masse totale du système et limiterait les risques associés à l'utilisation de deux batteries distinctes.

7 Conclusion

En conclusion, ce projet a permis d'intégrer et de tester une plateforme robotique dans des conditions réelles, tout en mettant en lumière les forces et les limites du système actuel. À travers l'intégration de la *stack* de localisation et de cartographie du Norlab, combinée à des protocoles d'expérimentation inspirés du protocole DRIVE, il a été possible de valider partiellement la précision du modèle cinématique et d'évaluer les écarts entre la trajectoire théorique et réelle du robot. Ces travaux ont également révélé des défis techniques, tels que l'absence d'odométrie basée sur encodeurs, le passage au contrôle du servo via Arduino, et la complexité liée à l'utilisation de deux batteries distinctes pour l'alimentation. Malgré ces obstacles, les solutions mises en œuvre ont permis de maintenir la fonctionnalité du système et de poursuivre les expérimentations prévues.

En somme, ce projet a non seulement permis de développer une meilleure compréhension de la dynamique du robot, mais a également offert une opportunité d'apprentissage face à des défis réels, renforçant ainsi les compétences en intégration, modélisation et résolution de problèmes. Ces enseignements serviront de base pour les itérations futures, qui viseront à maximiser les performances et la fiabilité de la plateforme pour poursuivre l'extension de DRIVE pour une géométrie Ackermann.

Références

[1] D. Baril, S.-P. Deschênes, L. Coupal et al., *DRIVE: Data-driven Robot Input Vector Exploration*, 2023.

[2] D. Baril, V. Grondin, S.-P. Deschênes et al., *Evaluation of Skid-Steering Kinematic Models for Subarctic Environments*, 2020.

[3] T. Zhang, Y. Sun, Y. Wang, B. Li, Y. Tian et F.-Y. Wang, "A Survey of Vehicle Dynamics Modeling Methods for Autonomous Racing: Theoretical Models, Physical/Virtual Platforms, and Perspectives", *IEEE Transactions on Intelligent Vehicles*, t. 9, nº 3, p. 4312-4334, 2024.