TP 2 Odométrie et déplacements

Objectifs

Ce second TP comporte trois objectifs :

- Programmer l'algorithme de localisation par intégration des données odométriques
- Qualifier les données de localisation par rapport à la référence
- Développer un algorithme de suivi de trajectoire

A rendre:

Vous devez rendre sur Ecampus un compte rendu de maximum 10 pages contenant votre travail et votre analyse au plus tard le **jeudi 25 Janvier 23h59**. Le rapport doit est à réaliser en monome et à rendre au format PDF. Vous joindrez au dépôt le fichier python de votre contrôleur.

I. Localisation

Votre robot thymio simulé par Webots ne contient pas de capteurs types encodeurs. Cependant, le simulateur peut nous donner la vitesse instantanée de chaque roue en utilisant la fonction « *motor_*.getVelocity()* ». Cette fonction nous retourne la vitesse du moteur en radian par seconde.

Pour ce travail, nous utiliserons le contrôle du robot par clavier que nous avons développé lors de la séance précédente.

1. Dans le simulateur, calculer le déplacement de chaque roue du robot.
2. Vérifier la cohérence de vos résultats par rapport au mouvement du robot et aux dimensions de la scène.

Le rayon de la roue du thymio est d'environ :

L'écartement de la roue du Thymio est d'environ 10.8 cm.

On redonne le modèle de déplacement d'un robot holonome :

$$\begin{pmatrix} \delta s \\ \delta\theta \end{pmatrix} = \left(\frac{\delta_r + \delta_l}{2}, \frac{\delta_l - \delta_r}{2e} \right)$$

$$(\mathbf{x}_{k+1})_{R_{glob}} = \begin{pmatrix} x_k + \delta s \cos(\theta_k + \frac{\delta\theta}{2}) \\ z_k + \delta s \sin(\theta_k + \frac{\delta\theta}{2}) \\ \theta_k + \delta\theta \end{pmatrix}$$

3. Programmer ce modèle en utilisant les déplacements calculés précédemment
4. Vérifier la cohérence de vos résultats

On souhaite tracer la trajectoire de notre robot au fur et à mesure de ces déplacements.

Il est possible de réaliser des graphiques non bloquant avec matplotlib. On utilisera comme exemple le code suivant :

```
plt.ion()
plt.plot(list_pos_x,list_pos_y)
```

```
plt.draw()
plt.pause(0.001)
plt.clf()
```

5. Tracer en temps réel la trajectoire suivie par votre robot
6. Ajouter les murs de l'environnement sur votre figure

Le principal intérêt d'utiliser un simulateur est de maîtriser entièrement l'environnement. Dans notre cas, nous pouvons connaître parfaitement la position de notre robot via le superviseur que nous utilisons.

Il faut d'abord récupérer le nœud correspondant à notre robot :

node = robot.getFromDef("Thymio");

Puis récupérer sa position et son orientation :

xyz = node.getPosition()
rotation = node.getOrientation()

7. Tracer la trajectoire réelle de votre robot sur votre figure en superposant la trajectoire mesurée.
8. Tracer l'évolution de chaque composante de votre localisation (x,y,theta) sur un graphique (subplot de 3 graphiques) ainsi que la référence de localisation.
9. Qualifier l'évolution de la précision de votre localisation.

II. Premiers déplacements

La manière la plus simple de se déplacer d'un point à un autre avec notre robot thymio est d'effectuer un déplacement en deux étapes. La première consiste à réaliser une rotation pour orienter le robot vers le point souhaité puis d'avancer jusqu'au point.

Ce type de déplacement nécessite de connaître la position du point de destination par rapport à la position et l'orientation du robot. Il est nécessaire d'exprimer la position de ce point dans le repère du robot.

1. Prendre les coordonnées d'un point de destination.
2. Exprimer les coordonnées de ce point dans le repère local du robot. On devra réaliser un changement de repère contenant une rotation et une translation.
3. Afficher les coordonnées du point dans le repère du robot et vérifier la cohérence des résultats.

III. Déplacements automatisé

On souhaite maintenant déplacer le robot automatiquement vers ce point de destination. Classiquement, la fonction de calcul de trajectoire est appelée à une fréquence plus faible que la fonction de calcul de la localisation. En effet, il n'est pas utile de calculer la trajectoire à une fréquence très élevé alors qu'il est indispensable d'intégrer très rapidement les données odométriques.

1. Calculer la rotation et la translation à effectuer pour atteindre ce point de destination.
2. Développer cette fonction de déplacements contenant d'abord une rotation puis une translation.
3. Modifier votre programme pour pouvoir suivre une trajectoire contenant plusieurs points de passage.
4. Réaliser un programme pour parcourir l'ensemble du labyrinthe automatiquement.
5. Évaluer la qualité de vos résultats en utilisant soit la localisation issue de votre intégration odométrique soit la référence de localisation.