TD3: Localisation et cartographie par EKF et évitement d'obstacles

Le but de ce dernier TD est d'utiliser la boîte à outils *Robotics Toolbox*, développée par Peter Corke (Queensland University of Technology, Australia), pour mettre en œuvre sur Matlab des algorithmes basés sur le filtre de Kalman étendu (EKF) pour la localisation d'un robot mobile et pour la cartographie :

https://petercorke.com/toolboxes/robotics-toolbox/

Deux stratégies pour l'évitement d'obstacles vues en cours (la méthode basée sur le graphe de Voronoï et la méthode basée sur les champs de potentiel), seront aussi implémentées en utilisant les fonctionnalités de base de Matlab.

Introduction au Robotics Toolbox :

Robotics Toolbox est une boîte à outils Matlab qui fournit de nombreuses fonctions qui sont utiles pour l'étude et la simulation de robots industriels (modèle géométrique, cinématique et dynamique, génération de trajectoires, etc.) avec une méthode de représentation très générale. La boîte à outils supporte aussi les robots mobiles avec des algorithmes de planification de chemin (PRM, RRT), localisation (EKF, filtre particulaire), création de cartes (EKF), localisation et cartographie simultanée ou SLAM (EKF), avec plusieurs modèles Simulink de véhicules (robot car-like, véhicule de Braitenberg, quadricopter).

Les différentes fonctions de la boîte à outils sont présentées dans le fichier « robot.pdf » (rvctools/robot/robot.pdf) et sur le lien: www.petercorke.com/RTB/r9/html/index_alpha.html

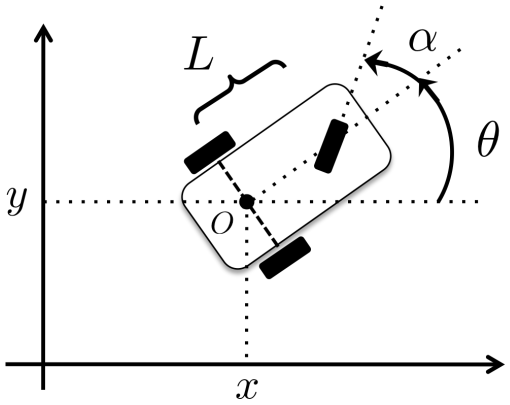
Pour utiliser la boîte à outils, téléchargez le dossier « robot-9.10.zip » (RTB 9.10) sur le site web, copiez le répertoire « rvctools » dans votre espace de travail et lancez le fichier de démarrage « startup_rvc.m » qui placera les bons répertoires dans votre chemin Matlab. La démo « rtbdemo » montre les fonctionnalités principales de la boîte à outils.

Exercice 1 : Créer et déplacer un robot mobile

Utiliser la classe « Vehicle », qui simule un robot car-like (tricycle), pour créer un robot, le déplacer suivant une trajectoire aléatoire et récupérer sa position instantanée.

- Créer un robot « veh » avec une matrice de covariance odométrique (rappel la matrice Σ_Δ) et afficher son état initial.
- Utiliser la fonction « update » pour appliquer une vitesse de 0.2 m/s avec une orientation de 0.1 rad pour un pas de temps. Afficher la nouvelle position du robot.

- Attacher un pilote (« driver ») au robot pour le faire déplacer suivant une trajectoire aléatoire dans une région $-10 \text{ m} < x < 10 \text{ m}$, $-10 \text{ m} < y < 10 \text{ m}$.
- Visualiser le mouvement du robot pour N = 1000 instants, en utilisant l'option « run », et récupérer l'état $[x, y, \theta]^T$ du robot.



<u>Exercice 2</u> : Utilisation de l'EKF pour la localisation et la cartographie

Cet exercice utilise la classe « EKF » qui peut être exploitée pour la localisation basée sur une carte, pour la cartographie et pour la localisation et cartographie simultanées (SLAM).

a) Estimation de la position d'un robot

- Créer un véhicule « veh » avec une matrice de covariance odométrique **S**.
- Ajouter un pilote avec une trajectoire aléatoire.
- Créer un filtre de Kalman avec une covariance estimée $\mathbf{V}_{est}$ et une covariance initiale de l'état $\mathbf{P}_0$.
- Exécuter le filtre pour N pas de temps.
- Afficher le chemin du véhicule, superposer le chemin estimé, et ajouter des ellipses d'incertitude tous les 10 pas de temps.

b) Localisation basée carte

- Créer un véhicule « veh » avec une matrice de covariance odométrique **S**.
- Ajouter un pilote avec une trajectoire aléatoire.
- Créer une carte avec 20 marqueurs.
- Créer un capteur qui utilise la carte et l'état du véhicule pour estimer les primitives du capteur avec une covariance **W**.
- Définir le filtre de Kalman avec les covariances estimées $\mathbf{V}_{est}$ et $\mathbf{W}_{est}$ et la covariance initiale de l'état du véhicule $\mathbf{P}_0$, puis exécuter le filtre pour N pas de temps.
- Tracer la carte, afficher le chemin du véhicule, superposer le chemin estimé, et ajouter des ellipses d'incertitude tous les 10 pas de temps.

c) Création de cartes par un robot

- Créer un véhicule « veh » avec une matrice de covariance odométrique **S**.
- Ajouter un pilote avec une trajectoire aléatoire.
- Créer un capteur qui utilise la carte et l'état du véhicule pour estimer les primitives du capteur avec une covariance **W**.
- Définir le filtre de Kalman avec la covariance estimée $\mathbf{W}_{est}$ et un véhicule « parfait » (utiliser une matrice de covariance "[]"), puis exécuter le filtre pour N pas de temps.
- Tracer la carte et superposer les ellipses d'incertitude 3-sigma estimées.

d) Localisation et cartographie simultanées (SLAM)

- Créer un véhicule « veh » avec une matrice de covariance odométrique **S**.
- Ajouter un pilote avec une trajectoire aléatoire.
- Créer une carte avec 20 marqueurs.
- Créer un capteur qui utilise la carte et l'état du véhicule pour estimer les primitives du capteur avec une covariance **W**.
- Définir le filtre de Kalman avec les covariances estimées $\mathbf{V}_{est}$ et $\mathbf{W}_{est}$ et la covariance initiale de l'état du véhicule $\mathbf{P}_0$, puis exécuter le filtre pour N pas de temps pour estimer l'état du véhicule à chaque pas de temps et la carte.
- Tracer la carte, afficher le chemin du véhicule, superposer le chemin estimé et ajouter des ellipses d'incertitude tous les 10 pas de temps.

Exercice 3 : Évitement d'obstacles : graphe de Voronoï

Soit $\boldsymbol{q}_s \in \mathbb{R}^2$ la position de départ et $\boldsymbol{q}_b \in \mathbb{R}^2$ la position à atteindre d'un robot mobile dans un environnement rectangulaire $[a, b] \times [a, b]$ m^2.

- Écrire une fonction Matlab, appelée `evit_voronoi`, qui prend en entrée les positions $\boldsymbol{q}_s$ et $\boldsymbol{q}_b$, le nombre N d'obstacles et les valeurs de a et de b. La fonction doit tracer le graphe de Voronoï qui a pour générateurs les N d'obstacles (voir la commande `voronoi` de Matlab) et un chemin sans collisions du robot superposé sur le graphe, calculé selon un critère de votre choix. Les N obstacles ponctuels sont distribués uniformément dans $[a, b] \times [a, b]$.
- Tester la fonction `evit_voronoi` avec $\boldsymbol{q}_s = [0, 0]^T$ m, $\boldsymbol{q}_b = [7, 7]^T$ m, $N = 15$, $a = 0$ m et $b = 10$ m.

<u>Exercice 4</u> : Évitement d'obstacles : champs de potentiel

Soit $\boldsymbol{q}_s \in \mathbb{R}^2$ la position de départ et $\boldsymbol{q}_b \in \mathbb{R}^2$ la position à atteindre d'un robot mobile dans un environnement rectangulaire $[a, b] \times [a, b]$ m^2.

- Écrire une fonction Matlab, appelée `evit_champ`, qui prend en entrée les positions $\boldsymbol{q}_s$ et $\boldsymbol{q}_b$, le nombre N d'obstacles, les valeurs de a et b, les gains k_a, $k_{r,i} > 0$ du potentiel attractif et des potentiels répulsifs, le rayon d'influence $\rho_{o,i}$ des obstacles et l'exposant γ qui apparaît dans l'expression du potentiel répulsif. Considérez un potentiel quadratique attractif et N obstacles ponctuels dont la position $\boldsymbol{q}_{o,i}$ est distribuée uniformément dans $[a, b] \times [a, b]$. La fonction doit afficher le potentiel total (voir les commandes `meshgrid` et `mesh` de Matlab pour tracer une fonction à deux variables) et la trajectoire sans collisions du robot calculée avec l'algorithme du gradient (on assume ici que le modèle dynamique du robot soit de type simple intégrateur).
- Tester la fonction `evit_champ` avec $\boldsymbol{q}_s = [0, 0]^T$ m, $\boldsymbol{q}_b = [7, 7]^T$ m, $N = 5$, $a = 0$ m, $b = 10$ m, $k_a = k_{r,i} = 0.2$, $\rho_{o,i} = 1$ m, $i \in \{1, 2, ..., N\}$, et $\gamma = 2$. Étudier les trajectoires du robot obtenues avec des valeurs de k_a, $k_{r,i}$, $\rho_{o,i}$, γ et α_k différentes, où $\alpha_k > 0$ est le pas discret utilisé dans l'algorithme du gradient.