Introduction à la robotique mobile
Travail pratique 2, Version 1.01
Date de remise : 9 décembre 2019, 23h55.

Instructeur : Philippe Giguère

3 décembre 2019

Les modification par rapport à la version 1.0 sont en rouge.

Instructions générales

Ce travail pratique se fait en équipe de 1 à 2. Dans l'évaluation du rapport, 10 points seront attribués à la qualité de la présentation. Les critères d'évaluations seront :

- la qualité du français ;
- la propreté de la présentation ;
- le degré d'élaboration et la présentation structurée des réponses : évitez donc de simplement donner la réponse sans un minimum de discussion ;

N'oubliez pas d'inclure TOUS les scripts/fonctions dans le fichier .zip. Vous pouvez faire le TP en matlab ou en python. Malheureusement, je ne peux fournir de fichiers scripts pour Python.

Pour les étudiants en GLO-7021, le rapport doit être rédigé en utilisant Latex.

1 Filtre de Kalman linéaire : estimation de la température d'un échantillon (15 pts)

Vous cherchez à estimer la température d'un échantillon avec une sonde électrique S, qui retourne un voltage de 2 V/K (deux volts par Kelvin), et qui a un bruit avec variance $\sigma_S^2 = 3\ V^2$. Votre estimé initial de la température est 300 K, auquel vous associez une incertitude initiale de variance $\sigma^2 = 3\ K^2$. L'échantillon baisse de température au rythme de 1.5 K par intervalle de temps. De plus, cette chute de température est légèrement bruitée, selon un bruit gaussien de variance $C_\nu = 0.04\ K^2$ (équation (2) à l'acétate 86 de 07-EstimationEtat-I.pdf) entre chaque intervalle. Les mesures de la sonde S à chaque intervalle de temps ($z_1 = 605\ V$, $z_2 = 597\ V$ et $z_3 = 586\ V$) sont reproduites dans le tableau.

1.1 a) (3 pts)

Quelles sont les valeurs des matrices Φ, Γ et Λ pour ce problème ?

1.2 b) Exécution du filtre (10 pts)

Calculez, à la main [1] pour les deux premières itérations les différentes étapes du filtre de Kalman, en complétant le tableau 1. Pas besoin d'inclure les démarches dans le rapport. Pour la troisième ligne, programmez le filtre de Kalman pour calculer les entrées. Utilisez-le d'ailleurs pour vérifiez vos calculs manuels.

Calcul	Iteration	X_t	P_t	X_{pred}	P_{pred}	Mesure z	Gain K	X_{t+1}	P_{t+1}
Manuel	1	300	3			605			
Manuel	2					597			
Programme	3					586			

TABLE 1 – Tableau à compléter pour l'exercice de Kalman.

Note : comme c'est un filtre récursif, les entrées X_t et P_t à l'itération 2 sont les mêmes que les entrées X_{t+1} et P_{t+1} à l'itération 1. Et ainsi de suite pour l'itération 3, 4, etc.

1.3 c) Convergence du gain K à l'infini (2 pts)

En ajoutant des mesures fictives à votre filtre, vers quelle valeur numérique le gain K converge-t-il pour un grand nombre d'itérations ?

1. Pour vous pratiquer, d'un coup je pose une question de ce genre à l'examen...

2 Description d'un problème d'estimation d'état

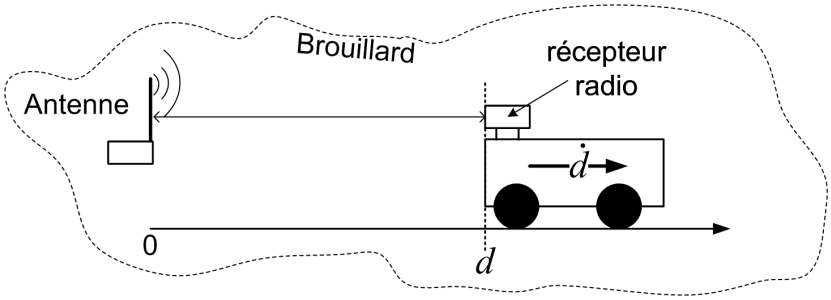

FIGURE 1 – Chariot motorisé équipé d'un récepteur mesurant la puissance d'une onde radio.

Vous avez un chariot motorisé, tel qu'illustré à la Fig. 1, se baladant dans un brouillard dans un monde à une dimension. Le chariot possède un récepteur radio qui mesure la puissance P, en mW, d'un signal radio émit par une antenne située à la coordonnée 0. La puissance détectée par le récepteur sur le robot mobile (en mW) suivra approximativement une loi décroissante exponentielle :

$$P = h_z(d) = P_{WiFi}e^{-d/10} \text{ (en } mW) \tag{1}$$

où d est la distance en mètre entre le robot et l'antenne et $P_{WiFi} = 10\ mW$ la puissance d'émission à l'antenne. Le bruit sur la mesure de puissance de réception sur le chariot sera $\sigma_z = 0.4\ mW$. Le capteur n'est pas biaisé, et il n'y a jamais d'obstacles entre le robot et le routeur WiFi. L'accélération (commande) du chariot u est :

$$u(t) = 0.12\cos(0.05\pi t) \text{ (en } m/s^2). \tag{2}$$

L'écart-type du bruit sur l'accélération correspond à 30% de la commande :

$$\sigma_u = 0.30u \text{ (en } \tfrac{m}{s^2}). \tag{3}$$

À titre de rappel, les équations d'un corps accéléré sont :

$$d_t = d_{t-1} + \dot{d}_{t-1}\Delta T + \frac{1}{2}\ddot{d}_{t-1}\Delta T^2, \qquad \dot{d}_t = \dot{d}_{t-1} + \ddot{d}_{t-1}\Delta T. \tag{4}$$

L'état du robot dans ce monde à une dimension est défini par :

$$X = \begin{bmatrix} d \\ \dot{d} \end{bmatrix} \tag{5}$$

où $\dot{d}$ est la vitesse du robot. Au départ, le robot est placé à $d_{init} = 6\ m$, et le chariot est au repos, i.e $\dot{d} = 0$. Le code matlab fourni `Chariot.m` utilise $\Delta t = 0.1\ s$, ce qui signifie que les déplacements sont effectués et les mesures sont prises à chaque Δt.

3 Solution par filtre à particules (15 pts)

Implémentez un filtre à particules pour le problème décrit à la section 2, à partir d'une copie du fichier `Chariot.m` et du code fournis sur le site web du cours pour les filtres à particules. Pour ce filtre, utilisez les paramètres suivants :

— $C = 4$, 10, et 40 particules ; Dans le code, `nParticules` $= C$.

— Ratio effectif $C/N_{eff} = R_{eff} = 0.5$

Pour l'estimé de la position d, prenez la moyenne pondérée des particules : `Xmoyen = sum(X(1,:).*w);`
Testez votre filtre à particules pour 400 itérations (donc de $t = 0$ s à $t = 40$ s), pour les cas suivants :

a) Vous connaissez exactement la position initiale du robot au départ. Vous initialisez donc toutes les particules avec $X = [d_{init}\ 0]^T$.

b) Vous ne savez pas exactement où se trouve le robot, mais vous savez qu'il se situe entre $3 < d < 12$. Vous initialisez donc la position des particules uniformément. En matlab, vous utiliserez l'instruction `X(1,:) = 3 + 9*rand(1,nParticules);` et `X(2,:) = zeros(1,nParticules);`

Pour ces cas a) et b), comparez la précision en fonction du nombre de particules C. De plus, pour C=40, incluez deux graphiques pour chacun des cas a) et b) suivant. Le premier montre, en fonction du temps :

— la position réelle du chariot avec le paramètre `'k-'` dans `plot` ;

— la position estimée `Xmoyen` du chariot avec le paramètre `'go'` dans `plot` ;

— la position correspondant à la mesure inversée $d = h_z^{-1}(z)$ du chariot avec le paramètre `'r*'` dans `plot`.

Le deuxième montre la variance d'échantillon de la distribution des particules.

4 Solution par filtre Kalman étendu (EKF) (25 pts)

4.1 Matrices Φ, Γ, et Λ (6 pts)

Quelles sont les valeurs des matrices pour ce système ? Important ! La matrice Γ est aussi utilisée pour propager le bruit de l'accélération vers les deux variables d'état. Ainsi, l'équation (2) du filtre de Kalman sera :

$$P(k+1|k) = \Phi P(k)\Phi^T + \Gamma C_\nu \Gamma^T \tag{6}$$

où $C_\nu = [\sigma_u^2]$.

4.2 Implémentation du filtre EKF (19 pts)

Implémentez le filtre de Kalman étendu (EKF) à partir d'une copie du fichier incomplet matlab `Chariot.m` pour estimer la position d du robot. Testez votre filtre de Kalman pour 400 itérations (donc de $t = 0$ s à $t = 40$ s), avec les scénarios suivants :

a) Vous connaissez exactement la position initiale du robot au départ. Vous initialisez donc la matrice d'état à $X = [d_{init}\ 0]^T$, et la matrice de covariance à

$$P = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

b) Vous avez une bonne idée de la position initiale du robot au départ, mais vous n'êtes pas confiant à 100 %. Vous initialisez donc la matrice d'état à $X = [d_{init}\ 0]^T$, et la matrice de covariance à

$$P = \begin{bmatrix} 25 & 0 \\ 0 & 1 \end{bmatrix}$$

c) Vous croyez connaitre exactement la position initiale du robot au départ, mais cette valeur est, dans les faits, erronée. Vous initialisez donc la matrice d'état à $X = [8\ 0]^T$, et la matrice de covariance à

$$P = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

d) Vous n'êtes pas sûr que $d = 8$ est la bonne position de départ du chariot. Vous initialisez donc la matrice d'état à $X = [8\ 0]^T$, et la matrice de covariance reflète cette incertitude car vous l'initialisez à :

$$P = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}$$

Pour tous les cas a)-d), décrivez le comportement de l'estimé de la position $X(1)$ du filtre EKF en quelques lignes, en portant une attention particulière à :

— l'erreur au début ($t < 2\ s$) du filtrage ;
— l'erreur moyenne ;
— la vitesse de convergence (ou non) de l'estimé vers la valeur réelle, surtout au début.

Faites quelques essais pour trouver des simulations concluantes [2]. Incluez pour chacun des cas ci-haut deux graphiques, pour l'une de ces simulations concluantes. Le premier graphique montrera, en fonction du temps :

— la position réelle du chariot avec le paramètre `'k-'` dans `plot` ;
— la position estimée $X(1)$ du chariot avec le paramètre `'go'` dans `plot` ;
— la position correspondant à la mesure inversée $d = h_z^{-1}(z)$ du chariot avec le paramètre `'r*'` dans `plot`.

Notez que le code matlab inclus dans le TP fait déjà ces graphiques pour vous.

2. par exemple, si la première mesure est très erronée et que votre matrice de covariance P indique une grande incertitude, vous allez voir une grande correction pour cette simulation.

5 Localisation globale par filtre à particules (25 pts GLO-4001, 45 pts GLO-7021)

Vous avez un robot initialement perdu, dans un environnement dont la carte est connue. Pour pouvoir le localiser, vous devrez donc utiliser un filtre à particules, puisque la distribution de croyance sur l'état du robot sera mulitmodale. Un aspect important de la question sera de trouver des bons paramètres, pour permettre au système de se localiser de manière robuste. Notez ici que nous ne chercherons pas à faire une solution qui puisse tourner en temps-réel ; ne vous préoccupez donc pas d'optimiser le code pour sa vitesse d'exécution.

Pour tous les cas, nous allons utiliser un monde décrit par un polygone et contenu dans le fichier `Carte.mat`. Le robot ponctuel (rayon=0) est équipé de 4 LiDAR, pointant dans les directions 0, $\pi/2$, π et 1.5π, tel qu'illustré à la Fig. 2. Le robot est soumis aux bruits gaussiens suivants (avec les noms de variable entre parenthèses) :

— LiDAR (`Lidar`) : $\sigma_L = 0.01\ m$;
— Vitesse linéaire (`V`) : $\sigma_V = 0.01\ m/s$;
— Vitesse angulaire (`omega`) : $\sigma_\omega = 0.05\ rad/s$;
— Compas magnétique (`Compas`) : $\sigma_C = 0.01\ rad$.

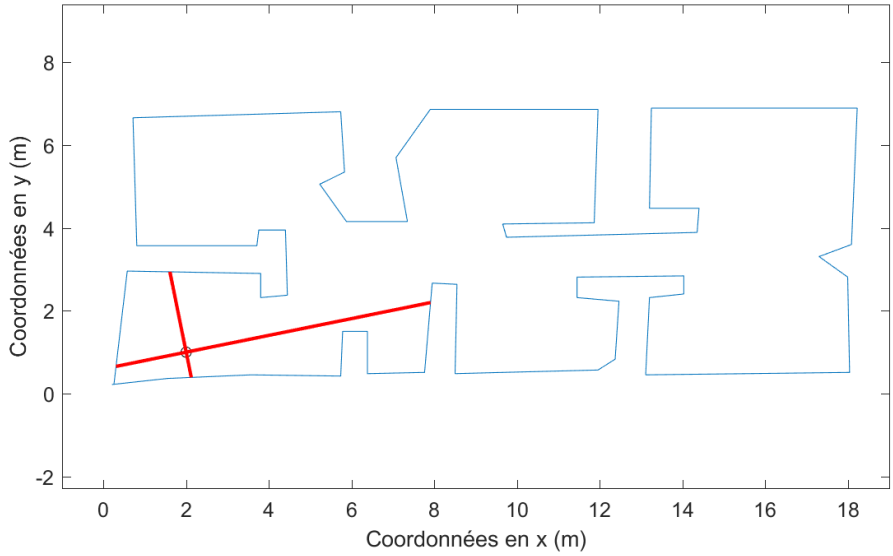

FIGURE 2 – Robot ponctuel dans un environnement décrit par un polygone. Les 4 lignes rouges indiquent les faisceaux laser.

Pour prédire les mesures LiDAR, vous allez utiliser des fonctions géométriques. Le faisceau LiDAR sera simplement un segment de droite partant du centre du robot et allant à une distance maximale (disons 20 m), selon l'angle du LiDAR (et bien sûr l'angle du robot, puisque le LiDAR est fixé sur ce dernier). Trouvez tous les points de cette droite qui interceptent le polygone de la carte (fonction `polyxpoly` dans matlab, ou la fonction appropriée dans le package `Shapely` dans Python). La distance mesurée par le LiDAR sera celle correspondante au point le plus proche du centre du robot.

Pour l'initialisation des particules, vous pouvez les distribuer de manière uniforme dans un rectangle couvrant la carte au complet. Ne vous préoccupez pas des particules en dehors du polygone (essentiellement en dehors de l'environnement). Ils finiront par avoir des poids quasi-nuls et seront donc éliminé lors de la phase du resampling. (Si vous insistez, la fonction `inpolygon` permet de trouver

les points à l'intérieur d'un polygone. D'ailleurs, il serait en théorie possible de se localiser uniquement à partir de cette contrainte. Mais je diverge...)

5.1 Cas 1 : l'angle est connu (GLO-4001 : 25 pts, GLO-7021 : 15 pts)

Afin d'y aller en douceur, nous allons commencer par un problème d'estimation à deux dimensions, où l'état est simplement $X = [x\ y]^T$. Pour l'angle du robot, vous allez simplement prendre l'angle `Compas` mesuré par le compas magnétique (en rad, de $-\pi$ à π), et ajouter un bruit σ_{angle} aléatoire sur l'angle. Ce bruit permettra de "brasser" les particules un peu, pour permettre une convergence plus rapide, en plus de tenir compte de l'incertitude du compas. Faites un filtre à particules qui exploite les informations suivantes, **contenues dans le fichier `Q1Trajectoire.mat` :**

- les 4 mesures LiDAR (`Lidar`), en m ;
- la vitesse linéaire (`V`), en m/s ;
- l'angle du compas magnétique (`Compas`), en rad.

Dans votre rapport, discutez des éléments suivants :

- une justification de votre choix σ_{angle} ;
- les divers paramètres utilisés dans le filtre (pour GLO-7021, la question 5.2 vous demande d'élaborer sur certains de ces paramètres, alors ne pas dupliquer) ;
- description qualitative de la distribution des particules au fil du temps.

Incluez deux tracés de trajectoires estimées par votre filtre, l'une pour une exécution réussie du filtre et l'autre pour une exécution qui n'a pas convergée. La trajectoire est basée sur le calcul, à chaque itération, de la moyenne des particules (pondérée par leur poids respectif). Dans le cas où le filtre a échoué, donnez une explication qualitative. **Incluez aussi dans vos graphique la trajectoire vérité-terrain (`xPose`,`yPose`,`anglePose`) contenue dans le fichier `Q1Trajectoire.mat`. Bien entendu, vous ne pouvez pas utiliser cette information vérité-terrain dans le filtre à particules, seulement à titre de débuggage et du calcul des erreurs dans le rapport. N'oubliez pas de faire le wrap2pi lorsque vous calculez des erreurs sur les angles.**

Pour les étudiants de GLO-7021, votre note dépendra de la qualité des explications et justifications. Par exemple, il serait intéressant de voir des graphiques d'erreur en fonction du temps vs. choix de σ_{angle}.

Pour les étudiants en GLO-4001, le TP se termine ici !

5.2 Impact du bruit utilisé dans la vraisemblance LiDAR (GLO-7021 seulement, 15 pts)

Lors de la mise-à-jour, vous devez choisir l'écart-type σ_{LiDAR} pour évaluer la vraisemblance des mesures du LiDAR dans la fonction $p(z|x)$. Quel est l'impact de cette valeur, en terme *a)* du nombre de particules nécessaire pour converger et *b)* de la vitesse de convergence du filtre. Pour*a)*, faites 10 essais par nombre de particules, et rapportez le pourcentage de succès. Vous devrez trouver, par essai et erreur, des valeurs "intéressantes" de nombres de particules pour ces tests, i.e. pour lesquels le taux de succès change. Pour aller plus vite, ne laissez pas tourner le filtre sur toutes les itérations : dès que vous voyez que les particules ont condensé en une région, vous pouvez établir si le filtre a divergé ou non par rapport à la vérité-terrain.

Prenez des mesures sur la vitesse d'appauvrissement, afin de voir l'impact de σ_{LiDAR} sur cette vitesse.

Selon vos résultats, en quoi serait-il intéressant d'augmenter ou de diminuer graduellement σ_{LiDAR} au fil du temps ? Si vous comprenez bien cette question, cela vous sera utile pour la question 5.3 pour accélérer le filtrage.

5.3 Cas 2 : l'angle est inconnu (GLO-7021 Seulement, 15 pts)

Pour ce cas-ci, la dimensionalité du problème va augmenter, car il vous faudra maintenant estimer l'état en y incluant l'angle θ du robot : $X = [x\ y\ \theta]^T$. Tel que vu en classe, le nombre de particules est exponentiel en dimension de l'état... Alors vous devrez probablement être patient lors de l'exécution de votre filtre ! Le fichier de donnée pour ce cas d'estimation, `Q2Trajectoire.mat`, contient les commandes de vitesses linéaires et les commandes angulaires, les mesures LiDAR et la vérité-terrain. Notez ici que vous n'avez plus de données du compas. Vous pouvez partir de votre implémentation du filtre à particule de la section 5.1.

Refaites les mêmes expérience qu'à la section 5.2 et commentez sur ces résultats.

À partir de votre expérience de l'impact de σ_{LiDAR} et du nombre de particules sur les chances de convergences, adaptez ces valeurs dynamiquement afin d'accélérer l'exécution du filtre à particules. Expliquez votre stratégies, notamment à l'aide de graphiques montrant le nombre de particules, la valeur de σ_{LiDAR} et les erreur en distance/orientation en fonction du temps. Appuyez ausis votre réponse avec des histogrammes d'erreur correspondant au régime pour lequel le filtre a convergé.

Comment pouvez-vous justifier votre approche, d'un point de vue théorique ?