TP2: Filtrage de Kalman appliqué à la navigation et à l'estimation d'état
RO12 - Navigation pour les systèmes autonomes

Gallego, Natalia
AST
ENSTA Paris
natalia.gallego@ensta-paris.fr

I. Introduction

Dans la navigation des véhicules autonomes, les systèmes d'estimation d'état jouent un rôle crucial pour l'intégration et l'analyse des informations provenant de divers capteurs, afin de déterminer la position et l'orientation du véhicule. Le filtre de Kalman étendu (EKF) est une technique couramment utilisée pour estimer les états non linéaires, combinant les prédictions d'un modèle dynamique avec des observations de capteurs bruyants.

Ce travail pratique vise à mettre en œuvre et à régler un EKF pour estimer la position d'un véhicule, en utilisant les informations des capteurs d'odométrie et des mesures simulées supplémentaires, telles que les plages de référence et les angles par rapport à certains points connus de l'environnement.

Le projet est structuré en plusieurs étapes, où les effets de différentes configurations EKF sont explorés, tels que la variation de la fréquence de mesure, la dynamique et le bruit du capteur, et la disponibilité de certaines mesures. De plus, le comportement du filtre est évalué par rapport aux interruptions des mesures et aux modifications du nombre de points de référence disponibles. Le code de simulation comprend un modèle de mouvement du véhicule, la génération de mesures et le filtre EKF, dont les équations clés doivent être complétées pour un fonctionnement correct.

Tout au long du TP, les résultats obtenus en ajustant les matrices de covariance du filtre seront analysés et commentés, simulant diverses conditions de mesure pour évaluer ses performances dans différents scénarios.

II. Question 1 : Explication initiale du code (simulation de véhicule, capteurs, odométrie, filtre de Kalman).

`get_odometry()` : l'odométrie fait référence à la façon dont le robot estime sa position au fil du temps en fonction du mouvement précédent.

`simulation_world()` : C'est l'évolution de la position réelle du robot : L'état réel du robot (xTrue) est mis à jour à chaque pas à l'aide d'un contrôle basé sur une trajectoire sinusoïdale.

`get_observation()` : modélise les capteurs, qui simulent les observations de caractéristiques (points de repère) dans l'environnement du robot.

`motion_model(x, u, dt_pred)` : Cette fonction est utilisée pour prédire le nouvel état du robot, étant donné l'état actuel (x), le contrôle appliqué (u) et l'intervalle de temps de prédiction (dt_pred) . L'état du robot (x) inclut sa position et son orientation, tandis que le contrôle (u) décrit les vitesses linéaires et angulaires.

`observation_model(xVeh, iFeature, Map)` : génère une prédiction des mesures observées en fonction de l'état du véhicule (xVeh) et de la position des entités sur la carte (Map). Le résultat de cette fonction est une observation simulée qui inclut la distance et l'angle de ce point de référence par rapport au robot.

`get_obs_jac(xPred, iFeature, Map)` : calcule la matrice jacobienne de la fonction d'observation par rapport à l'état prédit du robot (xPred). Il est utilisé pour linéariser le modèle d'observation.

`F(x, u, dt_pred)` : Calcule la matrice jacobienne du modèle de mouvement par rapport à l'état du robot.

`G(x, u, dt_pred)` : Cette fonction calcule la matrice jacobienne du modèle de mouvement par rapport au bruit dans les commandes du robot. Le bruit dans les commandes représente les incertitudes ou les erreurs associées aux actions du robot.

`tcomp(tab, tbc, dt)` : Cette fonction effectue une composition de transformations entre deux états. Prend l'état actuel du robot (tab) et lui applique une transformation de mouvement (tbc), en utilisant le pas de temps dt. Cela permet de mettre à jour la position et l'orientation du robot en fonction des commandes appliquées et du temps écoulé.

`plot_covariance_ellipse` : trace une ellipse qui représente l'incertitude dans l'estimation de la position du robot. A partir de la matrice de covariance estimée (PEst), l'ellipse décrit la plage probable d'erreurs dans l'estimation de la position du robot dans l'espace.

`Filtre de Kalman` : Le filtre de Kalman est une méthode d'estimation qui combine les prédictions du modèle de mouvement du robot avec les observations.

III. Question 2 : Code complet

Les codes manquants ont été complétés afin d'implémenter le filtre de Kalman sur la trajectoire du robot. Vous trouverez ci-dessous les fonctions développées et une brève explication de celles-ci.

```
def motion_model(x, u, dt_pred):
    xPred = np.zeros((3, 1))
    xPred[0, 0] = x[0, 0] + u[0, 0] * dt_pred * cos(
    x[2, 0])
    xPred[1, 0] = x[1, 0] + u[0, 0] * dt_pred * sin(
    x[2, 0])
    xPred[2, 0] = x[2, 0] + u[2, 0] * dt_pred
    xPred[2, 0] = angle_wrap(xPred[2, 0])
    return xPred
```

`motion_model` : met à jour la position en x en utilisant la vitesse en x et l'orientation actuelle. Met à jour la position y en utilisant la vitesse x et l'orientation actuelle. Met à jour l'orientation en utilisant la vitesse angulaire. Enfin, ajustez l'orientation pour qu'elle soit comprise entre -π et π à l'aide de la fonction `angle_wrap` et renvoyez l'état prédit xPred.

```
def observation_model(xVeh, iFeature, Map):
    dx = Map[0, iFeature] - xVeh[0, 0]
    dy = Map[1, iFeature] - xVeh[1, 0]
    z = np.zeros((2, 1))
    z[0, 0] = sqrt(dx**2 + dy**2)
    z[1, 0] = atan2(dy, dx) - xVeh[2, 0]
    z[1, 0] = angle_wrap(z[1, 0])
    return z
```

`observation_model` : La fonction prend trois paramètres :

- **xVeh :** l'état du véhicule (x, y, θ).
- **iFonction :** index de l'entité observée sur la carte.
- **Carte :** La carte contenant les positions de toutes les entités.

Avec cela la fonction :

1) Calcule la différence des coordonnées x (dx) et y (dy) entre la position du robot et la position de l'entité observée.
2) Calcule la distance euclidienne (z[0, 0]) entre le robot et l'entité.
3) Calcule l'angle (z[1, 0]) entre la ligne reliant le robot et la caractéristique et l'orientation du robot.

Pour définir ces équations, on a utilisé les informations vues au cours, où la matrice x_k est donnée par l'expression:

$$x_k = \begin{pmatrix} x_{k-1} + (\tilde{v}_k^x \cos(\theta_{k-1} - \tilde{v}_k^y \sin(\theta_{k-1}))\Delta t \\ y_{k-1} + (\tilde{v}_k^x \sin(\theta_{k-1} + \tilde{v}_k^y \cos(\theta_{k-1}))\Delta t \\ \theta_{k-1} + \tilde{w}_k \Delta t \end{pmatrix} \quad (1)$$

```
def get_obs_jac(xPred, iFeature, Map):
    dx = Map[0, iFeature] - xPred[0, 0]
    dy = Map[1, iFeature] - xPred[1, 0]
    q = sqrt(dx ** 2 + dy ** 2)

    jH = np.zeros((2, 3))
    jH = np.array([
        [-dx / q, -dy / q, 0],  # Pour rk
        [dy / (dx * 2 + dy * 2), -dx / (dx * 2 + dy
    * 2), -1]  # Pour phik
    ])
    return jH

def F(x, u, dt_pred):
    theta = x[2, 0].item()
    Jac = np.eye(3)
    u0 = u[0, 0].item()
    u1 = u[1, 0].item()
    Jac = np.array([
        [1, 0, - u0 * sin(theta) * dt_pred - u1 *
    cos(theta) * dt_pred],
        [0, 1, u0 * cos(theta) * dt_pred - u1 * sin(
    theta) * dt_pred],
        [0, 0, 1]
    ])
    return Jac

def G(x, u, dt_pred):
    theta = x[2, 0].item()
    Jac = np.array([
        [dt_pred * cos(theta), - dt_pred * sin(theta
    ), 0],
        [dt_pred * sin(theta),  dt_pred * cos(theta)
    , 0],
        [0, 0, dt_pred]
    ])
    return Jac
```

get_obs_jac :

1) Calcule la différence de coordonnées x (dx) et y (dy) entre la position du robot et la position de l'entité observée.
2) Calculez q, qui est la somme des carrés de dx et dy.
3) Calcul des dérivées partielles de la distance et de l'angle par rapport aux coordonnées du robot.
4) Calcule la dérivée partielle de l'angle par rapport à l'orientation du robot.

`F` : Calcule les dérivées partielles de la fonction de transition par rapport à l'orientation du robot (θ).

`G` : Calcule les dérivées partielles de la fonction de transition par rapport au bruit aux entrées de commande (u).

Pour cela, les définitions des matrices jH, F et G du filtre de Kalman étendu ont été utilisées.

$$F_k = \frac{\partial f}{\partial x} = \begin{pmatrix} 1 & 0 & -V_x \cdot \Delta t \cdot \sin(\theta) \\ 0 & 1 & V_x \cdot \Delta t \cdot \cos(\theta) \\ 0 & 0 & 1 \end{pmatrix} \quad (2)$$

$$G_k = \frac{\partial f}{\partial w} = \frac{\partial f}{\partial \tilde{u}} = \begin{pmatrix} \Delta t \cdot \cos(\theta) & -\Delta t \cdot \sin(\theta) & 0 \\ \Delta t \cdot \sin(\theta) & \Delta t \cdot \cos(\theta) & 0 \\ 0 & 0 & \Delta t \end{pmatrix} \quad (3)$$

```
 # Kalman prediction
xPred = motion_model(xEst, u_tilde, simulation.
    dt_pred)
F_x = F(xEst, u_tilde, simulation.dt_pred)
G_u = G(xEst, u_tilde, simulation.dt_pred)
PPred = F_x @ PEst @ F_x.T + G_u @ QEst @ G_u.T

# compute Kalman gain - with dir and distance
Innov = z - zPred         # observation error (
    innovation)
Innov[1, 0] = angle_wrap(Innov[1, 0])
S =  REst + H @ PPred @ H.T
K = PPred @ H.T @ np.linalg.inv(S)

```

```
13
14 # Compute Kalman gain to use only distance
15 Innov = z[0:1, :] - zPred[0:1, :]  # observation
       error (innovation)
16 H = H[0:1, :]
17 S = H @ PPred @ H.T + REst[0:1, 0:1]
18 K = PPred @ H.T @ np.linalg.inv(S)
19
20 # Compute Kalman gain to use only direction
21 Innov = z[1:2, :] - zPred[1:2, :]  # observation
       error (innovation)
22 Innov[0, 0] = angle_wrap(Innov[0, 0])
23 H = H[1:2, :]
24 S = H @ PPred @ H.T + REst[1:2, 1:2]
25 K = PPred @ H.T @ np.linalg.inv(S)
```

Ce code fait partie de la prédiction de l'état et de la covariance de l'état à l'aide du modèle de mouvement du robot et des entrées de contrôle. Le modèle de mouvement du robot est utilisé pour prédire l'état futur (xPred) en fonction de l'état actuel (xEst) et des entrées de contrôle (u_tilde).

Pour cela, les équations du filtre de Kalman étendu pour les systèmes nonlinéaires ont été utilisées :

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1}, u_k) \tag{4}$$

$$\hat{P}_{k|k-1} = F_k \hat{P}_{k-1} F_k^T + G_k Q_k G_k^T \tag{5}$$

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x_k}} \tag{6}$$

$$S_k = R_k + H_k \hat{P}_{k|k-1} H_k^T \tag{7}$$

$$K_k = \hat{P}_{k|k-1} H_k^T S_k^{-1} \tag{8}$$

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k(y_k - h(\hat{x}_{k|k-1})) \tag{9}$$

$$\hat{P}_k = (I_d - K_k H_k)\hat{P}_{k|k-1} \tag{10}$$

Calculer le gain de Kalman - Avec direction et distance : calcule le gain de Kalman en utilisant à la fois la distance et la direction de l'observation. La différence entre l'observation réelle (z) et l'observation prédite (zPred) est calculée. La matrice d'innovation (S) est calculée avec la covariance de l'observation et le produit des matrices jacobiennes et la covariance prédite. Calcul du gain de Kalman (K).

Calculer le gain de Kalman pour utiliser uniquement la distance : Cette section calcule le gain de Kalman en utilisant uniquement la distance d'observation. Calcul de l'innovation en utilisant uniquement la distance (Innov) : La différence entre la distance observée et la distance prédite est calculée. De plus, les matrices Innov et S ainsi que le gain K sont calculés. La matrice jacobienne (H) est réduite pour ne considérer que la distance.

Calculer le gain de Kalman pour utiliser uniquement la direction : Calculez le gain de Kalman en utilisant uniquement la direction de l'observation. La différence entre la distance observée et la distance prévue est calculée. De plus, les matrices Innov et S ainsi que le gain K sont calculés. La matrice H est également réduite.

A. *Résultats calculant Kalman avec distance et direction*

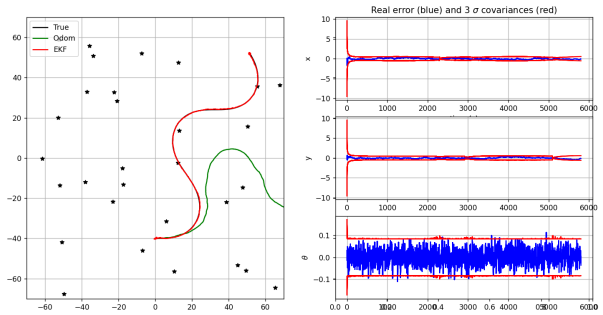

Fig. 1. Résultats calculant Kalman avec distance et direction

On constate que par rapport à la trajectoire calculée uniquement par odométrie, le filtre de Kalman améliore grandement la précision de la prédiction de trajectoire du robot. Mieux suivre la trajectoire. Concernant les erreurs des prédictions, nous pouvons voir que x et y sont tous deux en dehors des limites définies avec l'écart type. D'un autre côté, la prédiction de l'angle θ est assez précise et se comporte dans les limites de l'écart type.

IV. Question 3 : Modification de la fréquence des mesures

La valeur de `dt_meas` est l'intervalle de temps entre deux mises à jour de mesure, c'est-à-dire la fréquence à laquelle les observations de l'environnement sont obtenues. Par conséquent, changer cette valeur modifiera la fréquence du système et aura des implications importantes sur les performances et la précision du filtre.

Les résultats sont visibles dans les figures 2 et 3.

A. `dt_meas` *est très petit (0.1 s) :*

Les mises à jour des mesures nécessitent un calcul (calculer la matrice de gain de Kalman et mettre à jour les covariances). Avec un très petit `dt_meas`, le système peut être surchargé de calculs de mise à jour, affectant l'efficacité en temps réel. De plus, si les capteurs sont bruyants ou imprécis, des mesures fréquentes introduiront continuellement du bruit dans le système, ce qui pourrait avoir un impact négatif sur la précision des estimations si le filtre n'est pas bien réglé pour gérer le bruit.

Dans ce cas, nous pouvons voir que, selon la trajectoire calculée, le robot n'a pas bougé de manière significative entre les mises à jour, ce qui signifie que le filtre de Kalman continue de s'ajuster pour maintenir la position estimée très proche du point initial. Par conséquent, nous concluons que les très hautes fréquences peuvent affecter les performances du filtre de Kalman.

B. `dt_meas` est très grand (5 s et 10 s) :

Les mesures seront obtenues moins fréquemment, ce qui obligera le filtre de Kalman à s'appuyer principalement sur des prédictions de mouvement basées sur un modèle entre les mesures. Cela amène le système à accumuler des erreurs en raison de l'incertitude de la prédiction. De plus, si le robot subit des changements brusques ou un glissement, ces changements ne seront pas détectés tant qu'une mesure n'est pas effectuée, ce qui peut entraîner un écart important entre la position estimée et la position réelle.

Dans nos résultats, nous pouvons voir que le système gagne un peu d'erreur dans les prédictions faites, à mesure que la période entre les mesures augmente. Cependant, ce changement est très minime, ce qui nous permettrait de conclure que nous pourrions travailler à des fréquences plus basses avec le filtre de Kalman, afin de réduire le coût de calcul, tout en obtenant des résultats favorables.

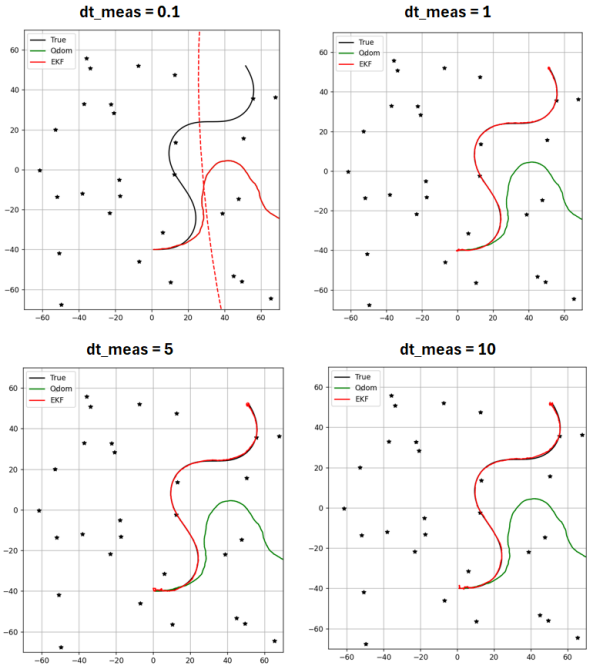

Fig. 2. Résultats de trajectoire pour différentes fréquences

V. Question 4: Variation du bruit de dynamique du filtre (matrice QEst)

La valeur de `QEst` représente l'estimation de l'incertitude dans le modèle de mouvement du robot (bruit de processus). Sa valeur peut affecter de manière significative la fusion des estimations odométriques et des mesures des capteurs :

A. `QEst` est très petit :

Dans ce cas, le filtre de Kalman s'appuiera trop sur le modèle de mouvement du robot et n'accordera pas suffisamment de poids aux mesures du capteur. Cela peut signifier que si le modèle de mouvement du robot n'est pas précis ou si l'odométrie comporte des erreurs, l'estimation du filtre peut s'éloigner de la position réelle du robot.

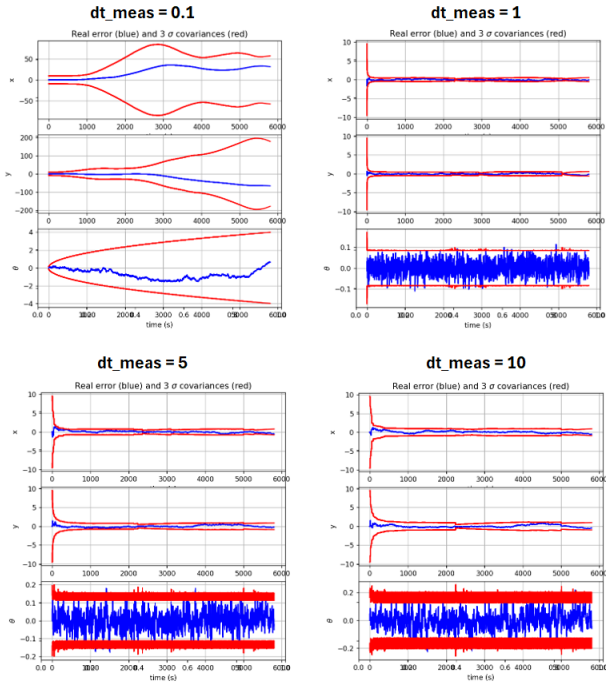

Fig. 3. Prédictions de trajectoire pour différentes fréquences

B. `QEst` est très grand :

Dans ce cas, le filtre de Kalman s'appuiera moins sur le modèle de mouvement du robot (odométrie) et davantage sur les mesures des capteurs. Le problème est que si les mesures du capteur sont bruyantes ou inexactes, le filtre de Kalman pourrait trop réagir à ces mesures, provoquant une instabilité ou des réactions excessives dans l'estimation de la position. Cependant, nous pouvons voir que dans ce cas, en augmentant `QEst`, l'estimation de la trajectoire s'améliore grandement.

En regardant les résultats, nous pouvons voir qu'en s'appuyant davantage sur le modèle de mouvement (petit QEst) nous obtenons une réponse plus douce par rapport aux autres graphiques. Ce qui nous permet de conclure que le modèle de mouvement est correct, et dans ce système particulier, les capteurs ajoutent du bruit aux résultats du filtre de Kalman, ce qui peut faire dévier un peu la trajectoire pendant le mouvement.

Les résultats sont visibles dans les figures 4 et 5.

VI. Question 5: Variation du bruit de mesure du filtre (matrice REst)

Cette question sera réalisée en utilisant un **QEst** de 1, afin de voir clairement comment la valeur de **REst** affecte les résultats.

La valeur de `REst` représente la matrice de covariance du bruit de mesure. Ce paramètre reflète le degré de confiance que nous avons dans les mesures du capteur.

A. `REst` est très petit :

Cela indique que les mesures des capteurs sont supposées être très précises, de sorte que le filtre de Kalman accordera

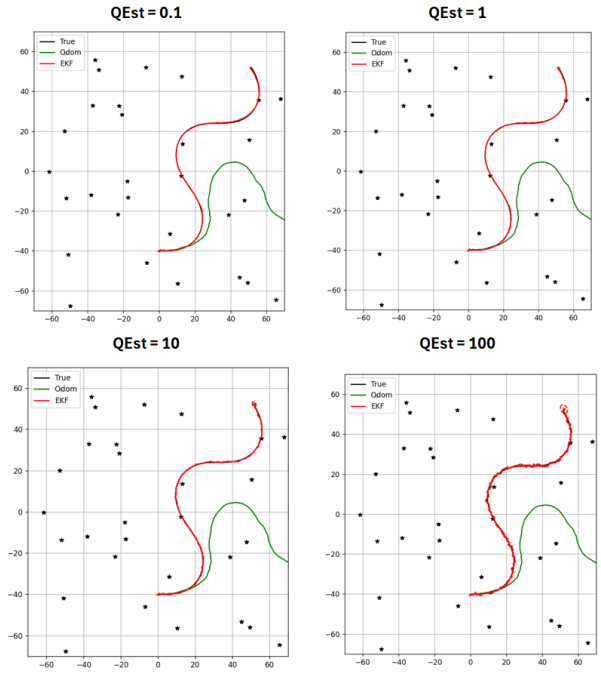

Fig. 4. Résultats de trajectoire pour différentes valeurs de QEst

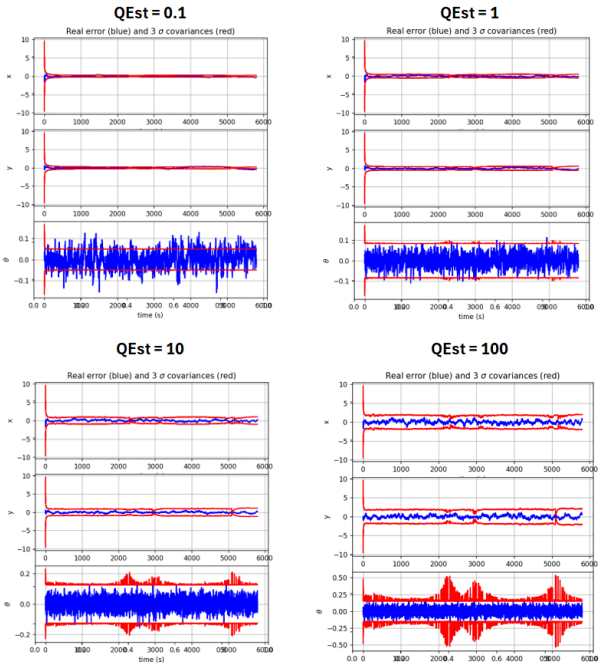

Fig. 5. Prédictions pour différentes valeurs de QEst

plus de poids aux mesures des capteurs et moins de poids à la prédiction du modèle de mouvement. Cela présente le risque que si les capteurs présentent réellement du bruit ou des erreurs, le filtre de Kalman pourrait sur-ajuster ces mesures, conduisant à des estimations erratiques.

B. `REst` est très grand :

Cela indique que les mesures du capteur sont considérées comme très bruyantes ou imprécises. De ce fait, le filtre de Kalman accordera moins de poids aux mesures des capteurs et s'appuiera davantage sur la prédiction d'état (odométrie). Cela peut avoir pour conséquence que si le modèle de mouvement n'est pas précis ou a accumulé des erreurs, le filtre peut ne pas bien corriger la trajectoire ou l'état du système.

Nous pouvons voir sur les graphiques que les prédictions s'améliorent à mesure que `REst` augmente. Nous pouvons alors conclure la même chose que dans la question précédente, où nous constatons que les résultats se sont améliorés lorsque nous nous sommes davantage appuyés sur l'odométrie du système. Cela peut être dû au fait que les capteurs sont bruyants et ne constituent pas une bonne source d'informations pour estimer l'état du système.

Les résultats sont visibles dans les figures 6 et 8.

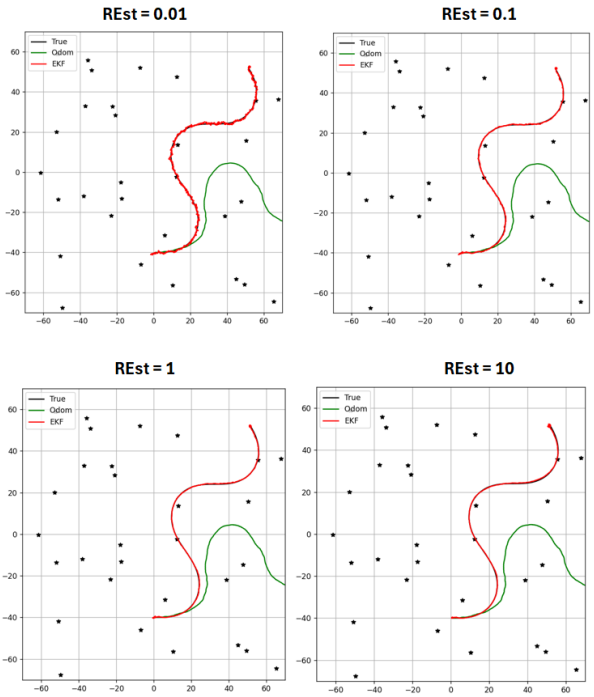

Fig. 6. Résultats de trajectoire pour différentes valeurs de REst

VII. Question 6 : Écart de mesure entre T = 2500 s et T = 3500 s avec `notValidCondition`

```python
if k*self.dt_pred % self.dt_meas == 0:
    notValidCondition = (2500 <= k*self.dt_pred <= 3500)
    if notValidCondition:
        z = None
        iFeature = None
    else:
        iFeature = np.random.randint(0, self.Map.shape[1] - 1)
        zNoise = np.sqrt(self.RTrue) @ np.random.randn(2)
        zNoise = np.array([zNoise]).T
```

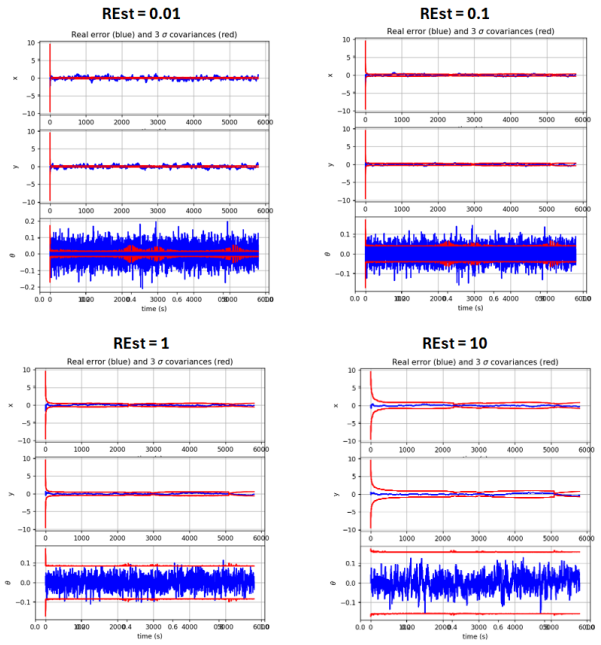

Fig. 7. Prédictions pour différentes valeurs de REst

```
        z = observation_model(self.xTrue, iFeature,
    self.Map) + zNoise
        z[1, 0] = angle_wrap(z[1, 0])
```

Pour les questions suivantes, `QEst` et `REst` ont été définis comme 1, afin de maintenir l'uniformité de l'étude.

En écrivant `notValidCondition = (2500 <= k*self.dt_pred <= 3500)`, nous définissons une plage de temps spécifique pour marquer une condition comme invalide. Cela permet de simuler l'absence de mesures de capteur dans cet intervalle de temps.

Cela divise nos graphiques en trois moments différents, qui seront expliqués.

A. *Avant l'intervalle de temps ($t < 2500$ s) :*

Les graphiques d'erreur montrent les corrections du filtre de Kalman basées sur les observations des capteurs et de l'odométrie, comme vu dans les questions précédentes.

B. *Pendant l'intervalle de temps ($2500 \text{ s} <= t <= 3500$ s) :*

Aucune observation ne sera disponible, le filtre de Kalman fera donc uniquement des prédictions basées sur le modèle de mouvement. Nous pouvons voir que l'incertitude dans les estimations augmente en raison du manque de corrections basées sur les observations, et nous voyons également que les graphiques d'erreur montrent une augmentation de l'écart de l'erreur réelle par rapport à l'estimation. Par conséquent, dans les délais impartis (lignes verticales vertes), les courbes de prédiction commencent à s'écarter rapidement de la valeur attendue.

C. *Après l'intervalle de temps ($t > 3500$ s) :*

Les observations des capteurs reprennent, permettant au filtre de Kalman d'effectuer à nouveau des corrections en utilisant ces données. Par conséquent, nous voyons que le système retrouve immédiatement sa trajectoire et que l'incertitude des estimations diminue à mesure que davantage de corrections sont apportées sur la base des observations.

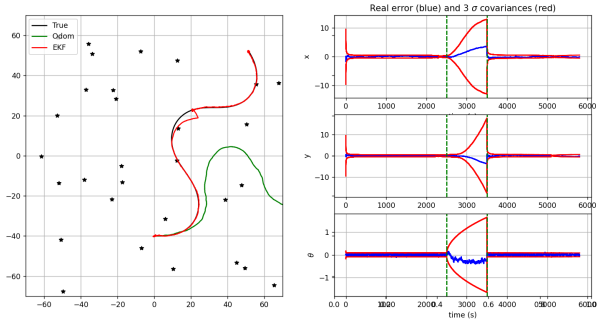

Fig. 8. Erreurs de prédiction pour différentes valeurs de REst

VIII. Question 7 : Variez le nombre de points de référence.

Pour les questions suivantes, la modification apportée à la question précédente a été éliminée, de sorte que le système obtient à nouveau des mesures pendant l'ensemble des temps.

Les `nLandmarks` représentent le nombre de points de repère ou de caractéristiques de l'environnement que le robot peut observer. Le filtre de Kalman s'appuie sur ces éléments pour réduire l'incertitude de l'état du robot en fusionnant les données des capteurs avec la prédiction du modèle de mouvement du robot.

Par conséquent, le nombre de points de référence affecte le compromis du filtre de calme, comme le montrent les figures 9 et 10. Les résultats sont expliqués ci-dessous.

A. *Réduction des `nLandmarks` :*

Avec moins de points de référence, le robot est moins capable d'observer une caractéristique et de corriger sa position. Cela peut conduire à de plus grandes incertitudes dans l'estimation de l'état du robot, surtout si le robot se déplace pendant de longues périodes sans observer de point de référence.

B. *Augmentetation des `nLandmarks` :*

Lorsqu'il y a plus de points de repère dans l'environnement, le robot a plus de possibilités d'observer une caractéristique, ce qui entraîne des corrections plus fréquentes de sa position estimée, ce qui permet d'espérer une précision de localisation améliorée.

Contrairement à l'explication précédente, en regardant les résultats obtenus, on constate que le filtre de Kalman fonctionne parfaitement dans les 4 cas proposés, où, quel que soit le nombre de points de référence, le système prédit avec précision la trajectoire. Les changements entre les différents scénarios sont minimes voire nuls entre eux, je conseille donc de garder ce nombre le plus minimal possible, dans notre cas 5, afin de réduire le coût de calcul. Cela peut être dû au fait que les points de repère supplémentaires sont redondants ou ne fournissent pas de nouvelles informations.

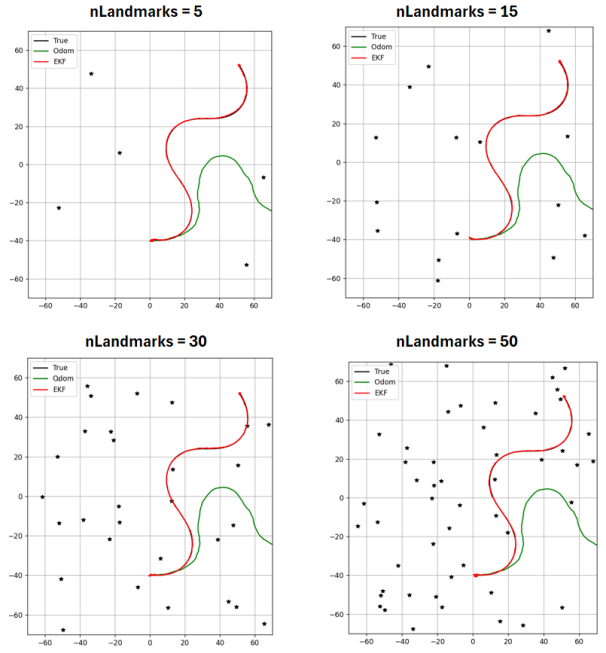

Fig. 9. Résultats de trajectoire pour différentes valeurs de nLandmarks

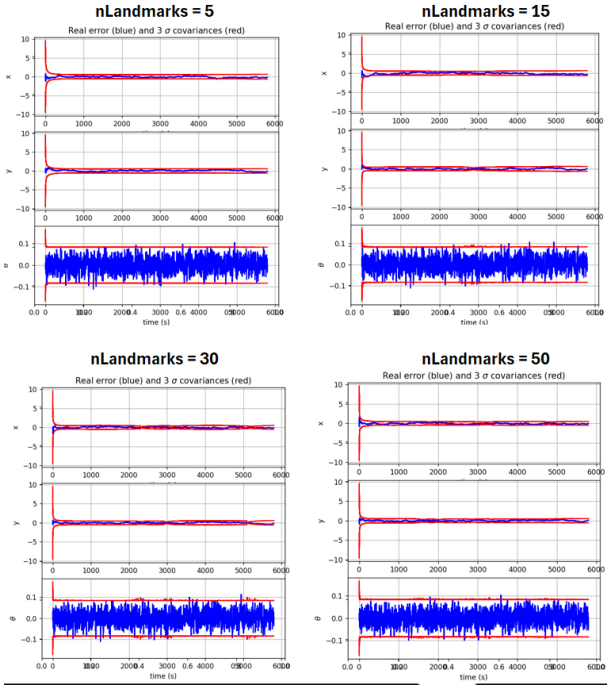

Fig. 10. Prédictions pour différentes valeurs de nLandmarks

IX. Question 8 : Résultats calculant Kalman uniquement avec la distance

Dans cette méthode on se concentre uniquement sur la distance pour calculer le gain de Kalman, on utilise donc uniquement la première partie de l'observation (`z[0:1, :]`), qui correspond à la distance. En conséquence, nous pouvons deviner qu'ignorer la direction peut conduire à des erreurs d'estimation, surtout si le robot se trouve dans un environnement où la direction est cruciale pour l'interprétation correcte des données.

Comme prévu, en comparant les résultats de la figure 11 avec les résultats de la figure 1, on constate que la courbe du système serpente davantage par rapport à la trajectoire réelle lorsqu'on utilise uniquement la direction dans le filtre de Kalman. Ceci est raisonnable, car en ignorant les informations de direction, le système dépend uniquement des corrections x et y, ce qui peut conduire à des oscillations plus importantes de la trajectoire.

Cependant, même si la courbe est un peu plus éloignée de la trajectoire réelle par rapport au cas initial où la direction et la distance étaient utilisées, le système présente tout de même un comportement suffisamment correct. Cela indique que même si la distance seule ne suffit pas à maintenir une précision de trajectoire élevée, elle est capable de fournir une estimation acceptable, surtout si le modèle du système (reflété dans les matrices de covariance) est bien calibré.

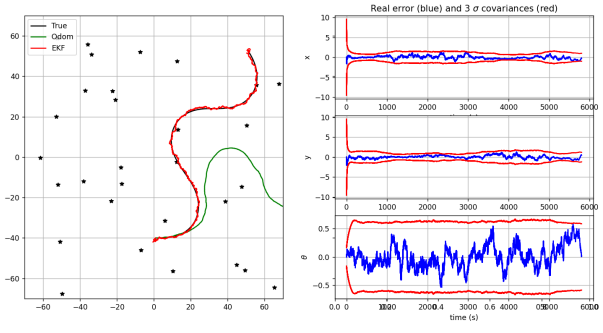

Fig. 11. Résultats calculant Kalman uniquement avec la distance

A. *Variez le nombre de points de référence*

De la même manière que précédemment, nous avons voulu voir comment le nombre de points de référence affectait le résultat du filtre de Kalman.

On voit que les résultats sont très similaires à ceux obtenus avec le filtre de Kalman utilisant à la fois la distance et la direction. Dans ce cas, le nombre de points n'améliore pas grandement le suivi de la trajectoire originale, donnant des performances similaires ou égales dans tous les scénarios. Pour cette raison, il est recommandé de prendre en compte le temps de calcul que peut entraîner l'ajout de points supplémentaires au système.

Pour voir les résultats rendez-vous sur les figures 12 et 13.

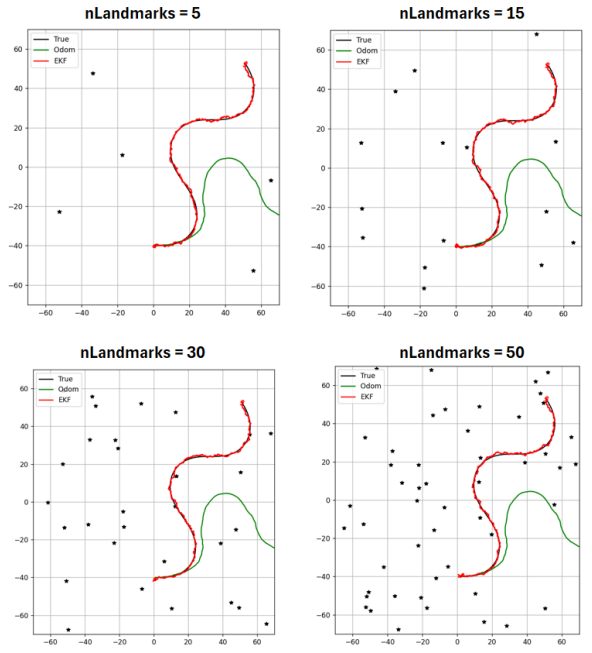

Fig. 12. Résultats utilisant uniquement des points de repère à distance variable

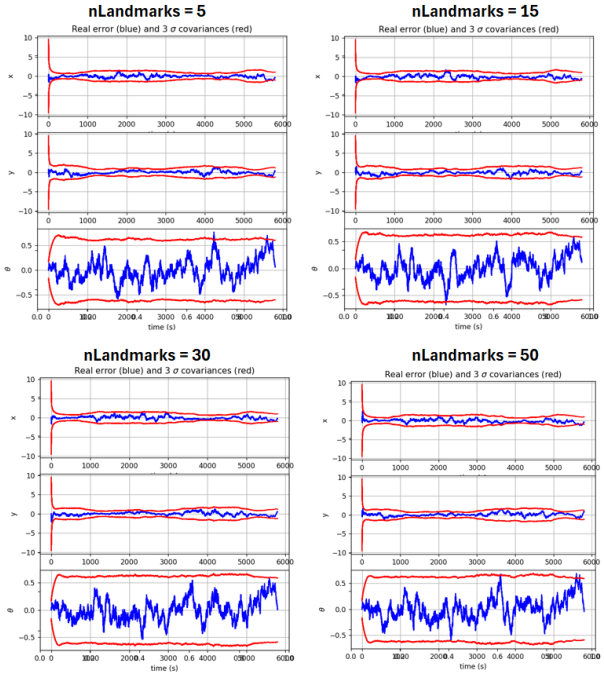

Fig. 13. Prédictions utilisant uniquement des points de repère à distance variable

B. Variez les valeurs de QEst et REst

De la même manière que précédemment, nous avons voulu voir comment le changement des valeurs de confiance `QEst` et `REst` affectait le comportement du filtre de Kalman.

Dans ce cas particulier, nous pouvons observer que varier les valeurs de `REst` produit des changements significatifs dans les résultats obtenus, comme auparavant. Cela suggère que la suppression de la direction rend le niveau de confiance dans les capteurs plus critique pour la trajectoire calculée par Kalman.

D'autre part, en faisant varier `QEst`, des changements dans les résultats sont également observés, bien que dans ce cas, pour obtenir de meilleurs résultats, une petite valeur de `QEst` soit préférée. En effet, une incertitude plus faible dans le modèle du système signifie une plus grande confiance dans le modèle lui-même, ce qui est bénéfique lorsque toutes les observations sensorielles ne sont pas utilisées (seule la distance est utilisée).

En résumé, ces résultats ont du sens, puisque lorsque seule la distance est utilisée, la valeur de `REst`, liée aux capteurs, devient plus pertinente, et un grand `REst` permet d'obtenir de meilleurs résultats. Au contraire, il est important de garder `QEst` petit, car une incertitude plus faible dans le modèle de processus garantit que la confiance dans l'odométrie et d'autres aspects du système influence positivement la précision de la trajectoire calculée.

Pour voir les résultats, rendez-vous sur les figures 14 et 15.

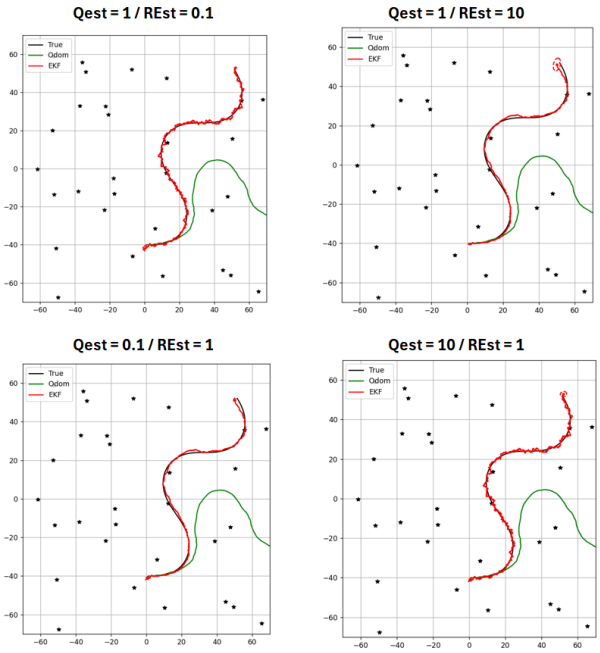

Fig. 14. Résultats utilisant uniquement la distance variant QEst et REst

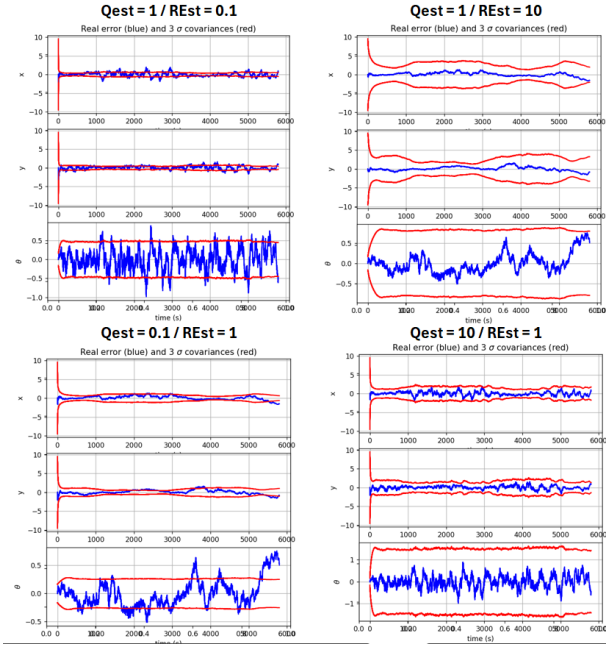

Fig. 15. Prédictions utilisant uniquement des distances variables QEst et REst

X. Question 9 : Résultats calculant Kalman uniquement avec la direction

Cette méthode se concentre uniquement sur la direction, en utilisant la deuxième partie de l'observation (`z[1:2, :]`), qui correspond à l'angle. Comme dans le cas précédent, l'omission de la distance peut rendre l'estimation inexacte, en particulier dans les scénarios où la distance est importante pour la localisation.

Contrairement au cas précédent, comme on peut le voir dans les résultats de la figure 16, en les comparant avec ceux de la figure 1, la courbe se comporte de manière très similaire à la situation originale. Dans la figure actuelle, nous pouvons voir que la trajectoire du robot suit les mêmes mouvements fluides, ce qui indique que le robot suit avec précision la trajectoire réelle.

Ces résultats sont surprenants, car en supprimant la distance et en utilisant uniquement la direction pour les corrections de Kalman, on s'attendrait à un impact plus important sur la précision de la trajectoire estimée. Cependant, le fait que les résultats soient presque identiques à ceux obtenus lorsque la distance et la direction sont utilisées suggère que la direction seule fournit suffisamment d'informations pour guider le système avec une précision proche de la totalité.

Cela peut être dû au fait que la direction est une variable critique dans ce scénario, et son estimation correcte permet de compenser le manque d'information sur la distance, surtout si le modèle du système est bien configuré et que les incertitudes de l'odométrie sont faibles.

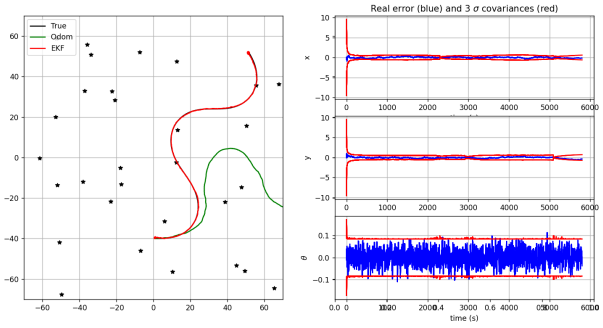

Fig. 16. Résultats calculant Kalman uniquement avec la direction

A. Variez le nombre de points de référence

De la même manière que précédemment, nous avons voulu observer comment le nombre de points de référence affectait le résultat du filtre de Kalman.

Comme dans les cas précédents, dans cette situation le nombre de points de référence n'influence pas beaucoup le résultat final de la trajectoire. Qu'on réduise le nombre de points (5 ou 15 points) ou qu'on l'augmente (30 ou 50 points), le système se comporte de manière similaire. La trajectoire calculée reste très proche de la trajectoire réelle, sans écart majeur.

Cela indique que dans ce scénario particulier, le nombre de points de référence utilisés par le filtre de Kalman n'influence pas la capacité du système à estimer correctement la position du robot. Il est possible que les informations fournies par les capteurs ou le modèle de mouvement soient suffisamment robustes pour compenser un faible nombre de points de repère, rendant leur nombre moins crucial pour le calcul final.

Pour voir les résultats rendez-vous sur les figures 17 et 18

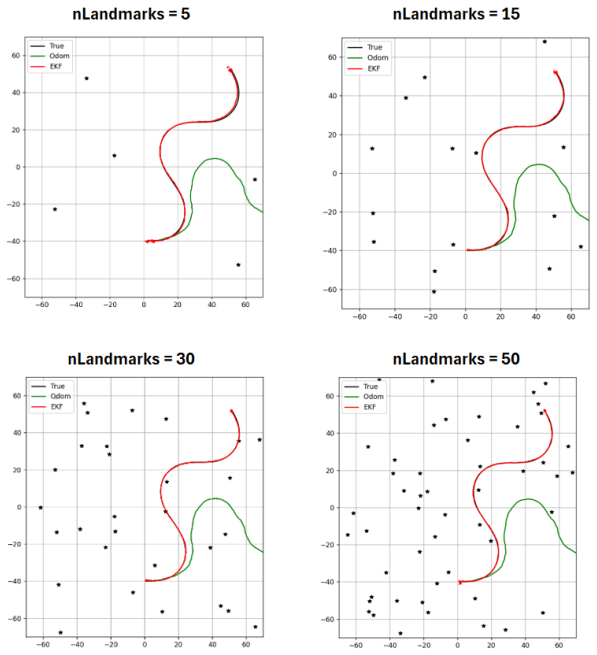

Fig. 17. Résultats utilisant uniquement l'adresse des différents points de repère

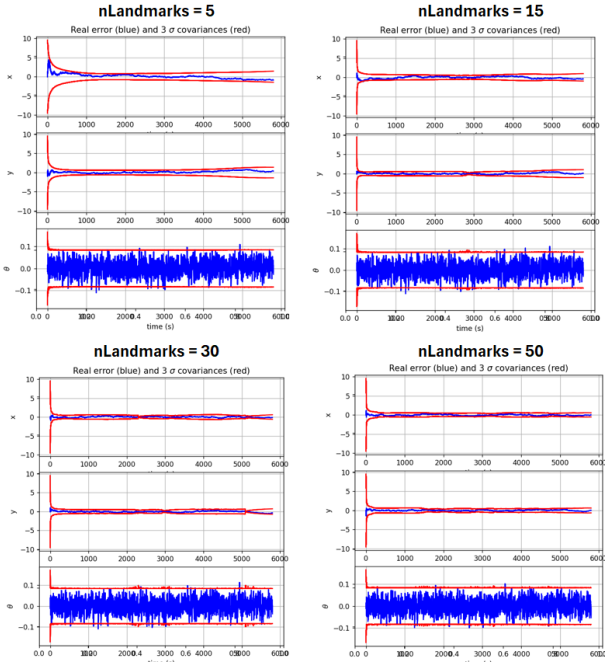

Fig. 18. Prédictions utilisant uniquement la direction variable des points de repère

B. Variez les valeurs de QEst et REst

De la même manière que précédemment, nous avons voulu voir comment le changement des valeurs de confiance `QEst` et `REst` affectait le comportement du filtre de Kalman.

Contrairement à l'époque où l'on utilisait uniquement la distance, faire varier les valeurs de `REst` n'a pratiquement même pas d'impact sur les résultats obtenus.

En revanche, nous observons un comportement similaire avec `QEst`. Changer cette valeur modifie légèrement les erreurs de prédiction dans les variables x, y et . En augmentant les valeurs de `QEst`, on remarque une petite variabilité dans les erreurs de prédiction.

Cela suggère que, dans cette configuration particulière, la confiance du capteur (`REst`) a un impact minimal sur la trajectoire calculée. Pendant ce temps, la confiance dans le modèle du système (`QEst`) affecte modérément les résultats, générant de petites variations dans les prédictions, mais sans compromettre de manière significative la précision globale du système.

Pour voir les résultats, rendez-vous sur les figures 19 et 20.

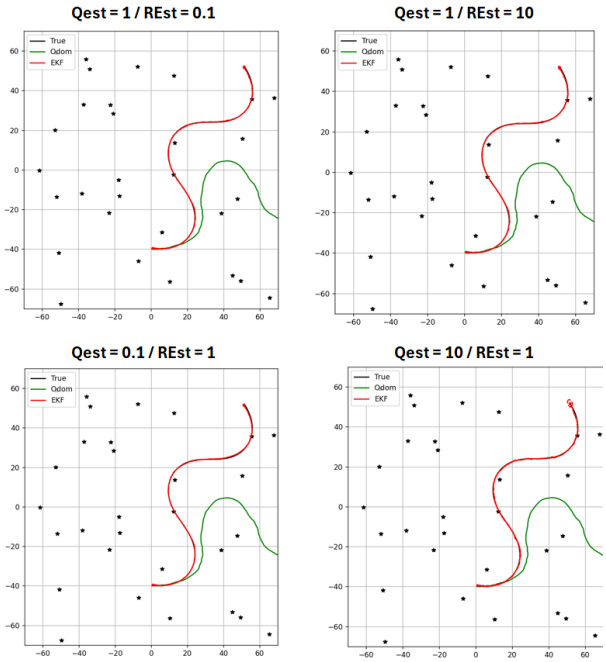

Fig. 19. Résultats utilisant uniquement la direction variant QEst et REst

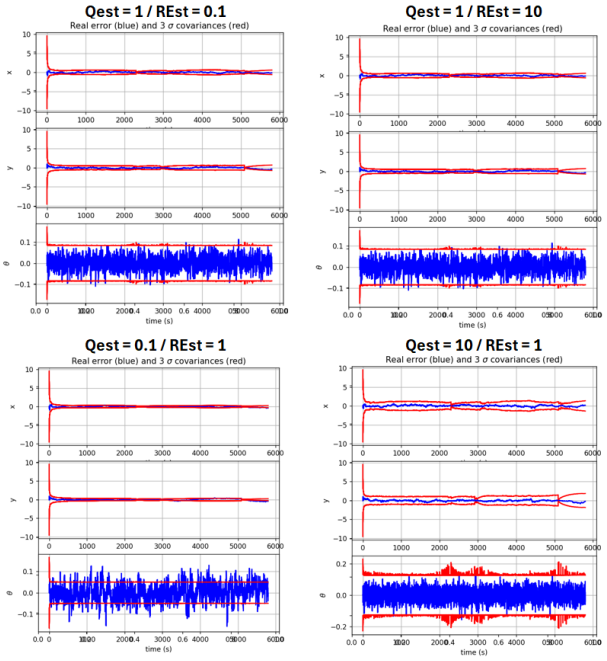

Fig. 20. Prédictions utilisant uniquement des direction variables QEst et REst

XI. Conclusions

Le filtre de Kalman s'est avéré être un outil exceptionnellement robuste pour l'estimation de trajectoire. Tout au long de nos tests, il a été évident que, quelles que soient les variations clés telles que la fréquence de mise à jour, le nombre de points de repère disponibles, l'absence temporaire de mesures sensorielles, ou encore les ajustements des matrices de covariance du système (`QEst `) et de la capteurs (`REst`), les résultats obtenus sont encore suffisamment précis.

Remarquablement, le système suit correctement la trajectoire même lorsque des observations importantes telles que la distance ou la direction sont supprimées. Cela souligne la capacité du filtre de Kalman à s'adapter à un large éventail de conditions tout en fournissant des estimations fiables. En conclusion, le filtre de Kalman est non seulement efficace, mais également très robuste face aux limitations et modifications du modèle et des observations, ce qui en fait un choix idéal pour les systèmes nécessitant une estimation en temps réel.

XII. GitHub

Vous pouvez vous référer à la référence suivante pour voir les codes de nœud du projet: GitHub/RO12/TP2