TP4: Simultaneous Localization and Mapping using Extended Kalman Filter
RO12 - Navigation pour les systèmes autonomes

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

I. Introduction

Dans ce travail pratique, nous explorerons une méthode de localisation et de cartographie simultanées (SLAM) qui construit une carte d'un environnement inconnu à l'aide d'un filtre de Kalman étendu (EKF). Cette approche permet à un robot de naviguer et de localiser sa position tout en estimant simultanément la position des points de repère dans l'environnement.

Tout au long des travaux pratiques, nous analyserons comment les performances du SLAM varient en fonction des différentes configurations d'environnement et des techniques d'association de données. En particulier, l'influence de la quantité et de la position des points de repère, ainsi que de la longueur de la trajectoire du robot, sur la précision de la carte et l'évolution de l'erreur du robot sera étudiée. De même, l'effet de l'utilisation de la distance de Mahalanobis pour l'association de données dans des scénarios avec et sans ambiguïtés dans la perception des points de repère sera étudié.

De plus, nous évaluerons l'impact de l'ajustement des paramètres de bruit estimés du filtre EKF par rapport aux valeurs utilisées dans la simulation (`Q` et `Py`), en observant son influence sur la cohérence du filtre et la qualité de la carte. Enfin, nous implémenterons une technique d'initialisation sans délai pour effectuer un SLAM avec uniquement l'adresse des points de repère, connue sous le nom de SLAM "adresse uniquement".

Ce travail vise à fournir une compréhension approfondie du SLAM basé sur EKF et des adaptations nécessaires pour relever divers défis dans un environnement inconnu, tels que l'association de données et l'initialisation dans le SLAM avec adresse uniquement.

II. Explication générale du code

Le code proposé par le TP implémente SLAM en utilisant un filtre de Kalman étendu (EKF), qui est une variante du filtre de Kalman particulièrement utile lorsque le système et les observations sont non linéaires. Le filtre estime à la fois l'état du robot et la position des points de landmark sur la carte.

SLAM (Simultaneous Localization and Mapping) est un processus dans lequel un robot, un véhicule ou tout système autonome construit une carte de l'environnement tout en calculant sa position à l'intérieur de celui-ci. Ceci est crucial dans les scénarios où l'environnement est inconnu ou dynamique. Les systèmes SLAM sont utilisés dans des applications telles que la robotique mobile, les véhicules autonomes et dans des environnements complexes tels que la reconnaissance spatiale intérieure.

Nous décrirons ensuite les principales parties du code :

Explication des principales étapes du Code

A. Représentation de l'Etat

L'état du système est représenté dans le vecteur ($xEst$), qui comprend à la fois la position et l'orientation du robot ((x), (y), (θ)) et les positions des points de landmark (x_ai, y_ai).

$$xEst = \begin{bmatrix} x \\ y \\ \theta \\ x_{al} \\ y_{al} \\ \dots \\ x_{aN} \\ y_{aN} \end{bmatrix} \tag{1}$$

B. Prédiction des mouvements

La fonction de mouvement ($f(xEst, u)$) est utilisée pour prédire le prochain état du robot en fonction de sa vitesse (v) et de sa vitesse angulaire (ω), ainsi que de l'intervalle de temps (dt). Cette fonctionnalité projette l'état du robot dans le temps, anticipant sa position et son orientation en fonction de son mouvement actuel.

$$f(xEst, u) = \begin{bmatrix} x + v.dt.\cos(\theta) \\ y + v.dt.\sin(\theta) \\ \theta + \omega.dt \end{bmatrix} \tag{2}$$

C. Mesure et observation

La fonction d'observation ($h(xEst)$) permet de calculer les observations du robot à partir de sa position et de celle des amers. Cette fonction permet de comparer les distances et angles mesurés avec ceux attendus. Ce calcul mesure la distance entre le robot et chaque point de landmark de l'environnement, aidant ainsi le robot à ajuster sa carte de référence et sa position relative dans l'espace.

$$h(xEst) = \begin{bmatrix} sqrt(x - x_{ai})^2 + (y - y_{ai})^2 \\ atan2(\frac{y_{ai} - y}{x_a i - x}) - \theta \end{bmatrix} \quad (3)$$

D. Mise à jour du filtre

Lors de l'étape de mise à jour EKF, les prédictions de la fonction de mouvement et de l'observation sont comparées aux mesures réelles obtenues par les capteurs. Le filtre ajuste les estimations de la position et des points de landmark du robot en fonction de cette comparaison.

En résumé, le code utilise l'EKF pour effectuer un SLAM, où il alterne entre la prédiction de l'état du robot et des points de landmark et l'ajustement de ces valeurs avec des mesures réelles.

III. Question 1 : Modifier le nombre et la position des points de landmark et la trajectoire du robot

Cette question examine l'algorithme SLAM pour voir comment les performances de cartographie et la précision de localisation varient en modifiant la configuration des points de landmark (quantité et position) et la trajectoire du robot. Les différentes configurations nous permettront d'observer comment la qualité de la carte et l'évolution de l'erreur changent au fur et à mesure que le robot boucle des boucles sur le parcours.

Le comportement est revu dans 3 configurations différentes :

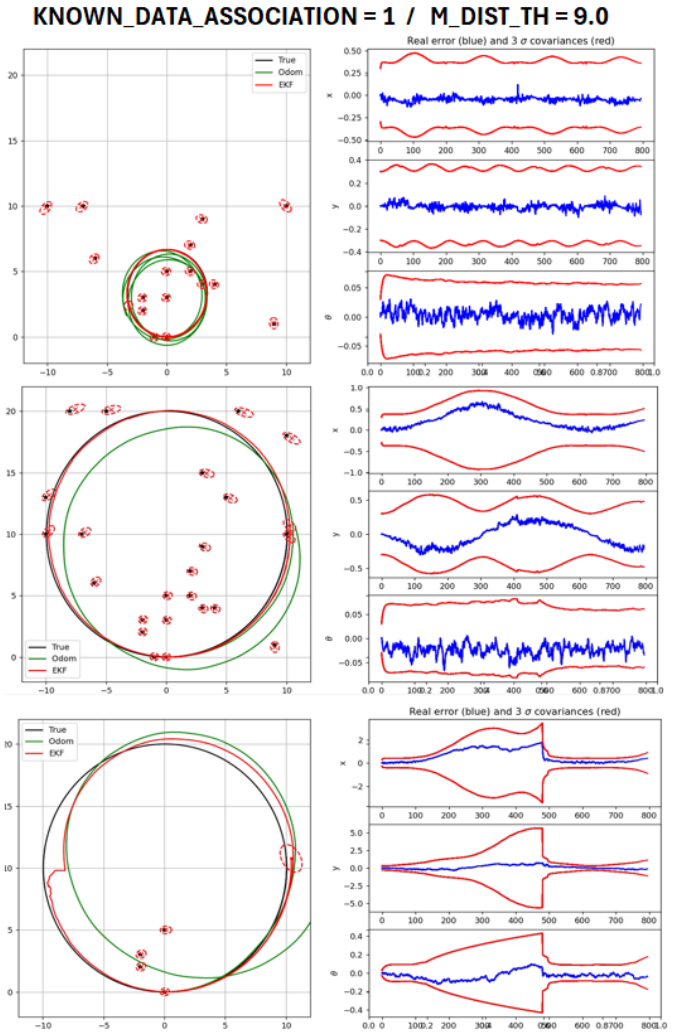

Fig. 1. Résultats avec modification du nombre et la position des points de landmark

A. Boucle courte et carte dense avec de nombreux landmarks dans le rayon de perception

Dans cette configuration, le robot suit une trajectoire qui forme une courte boucle (pour cela, `yaw_rate` a été défini dans `calc_input` avec une valeur de 0.3), tandis que les landmarks sont concentrés et se trouvent tous dans le rayon de perception du robot. Cela aide le robot à se localiser avec précision et à conserver une faible erreur d'estimation, puisqu'il détectera plusieurs points de landmark à chaque position. Comme le montrent les résultats (Figure 1), le résultat est une carte détaillée et précise, où l'erreur d'estimation reste faible tout au long du parcours, notamment lors de la fermeture de la boucle. De plus, on constate qu'à chaque fermeture d'un cycle, les erreurs d'estimation diminuent un peu lors de la redécouverte des Landmarks précédemment définis.

B. Boucle longue et carte dense avec des landmarks le long de la boucle

Dans ce cas, le robot suit un chemin en boucle plus long (pour ce `yaw_rate` a été défini dans `calc_input` avec une valeur de 0.1), mais il y a des landmarks répartis tout au long du chemin. Cela aide le robot à maintenir un niveau de référence tout au long du trajet. Bien que la boucle soit longue, les landmarks dispersés servent de points de référence, contribuant ainsi à réduire l'erreur accumulée tout au long de la trajectoire. En fermant la boucle, nous constatons que l'erreur, qui avait augmenté avec le temps, est considérablement réduite à mesure que le robot revient dans des zones avec des points de landmark connus, et l'algorithme EKF ajuste la trajectoire et la carte en conséquence.

C. Longue boucle et carte clairsemée avec peu de points de landmark à proximité de la position de départ

Ici, le robot suit un long chemin circulaire, alors qu'il y a peu de points de référence et qu'ils ne sont que proches de la position de départ. Cette configuration entraîne une erreur d'estimation plus élevée par rapport aux précédentes au début du mouvement, puisque le robot ne dispose pas de suffisamment de points de référence pour corriger son estimation de position lorsqu'il s'éloigne de la zone d'origine. La fermeture de la boucle montre une correction significative de l'erreur accumulée, qui avait atteint des valeurs élevées, à mesure que le robot revient à la zone de landmark, lui permettant d'ajuster la carte et l'emplacement de manière plus drastique. Cette correction drastique est principalement visible lorsque la trajectoire (ligne rouge) effectue un virage drastique pour s'adapter à la trajectoire réelle une fois que les points de référence entrent dans le rayon de détection du robot.

Dans toutes les configurations, on voit comment la trajectoire calculée à l'aide du filtre de Kalman étendu et de l'algorithme SLAM (ligne rouge), améliore les prédictions faites par odométrie (ligne verte) dans chacune des configurations proposées. La meilleure correction étant celle du scénario numéro 1 et la pire étant celle du scénario numéro 3, où au fur et à mesure que le robot perd ses points de référence il commence à suivre la trajectoire de l'odométrie.

IV. Question 2 : With data association using Mahalanobi distance. Adjusting the `M_DIST_TH`

Tout d'abord, `KNOWN_DATA_ASSOCIATION = 0` a été défini. Cela fait que l'association de données est inconnue et le filtre EKF doit effectuer un processus d'association pour identifier quelles observations correspondent à quels points de landmark. Pour cela, la distance Mahalanobis est utilisée avec le seuil `M_DIST_TH`, qui mesure dans quelle mesure une observation correspond à un point de landmark, en tenant compte de l'incertitude. Lorsque la distance entre une observation et un amer est inférieure au seuil `M_DIST_TH`, ils sont considérés comme associés.

Au cours de cette partie du travail, le paramètre `M_DIST_TH` (Threshold of Mahalanobis Distance) a été modifié, ce qui définit le seuil maximum autorisé pour la distance Mahalanobis dans le processus d'association de données dans SLAM. Cela signifie qu'il contrôle dans quelle mesure les observations actuelles peuvent s'adapter aux points de landmark déjà enregistrés sur la carte. Pour cela, les valeurs 1, 9 et 50 ont été testées et on s'attend à ce que chacune d'elles affectera la sensibilité du système en termes de laquelle les observations sont acceptées comme correspondances avec des landmarks existants ou sont considérées comme nouvelles.

A. `M_DIST_TH = 1.0 (Valeur Faible)`

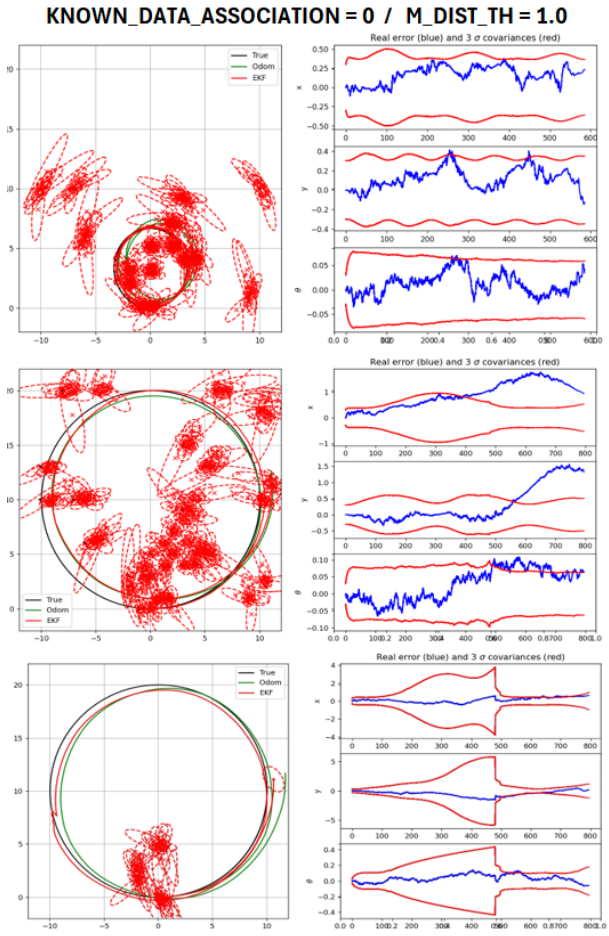

Fig. 2. Résultats pour `M_DIST_TH` = 0.1

Il s'agit d'un seuil strict, donc seules les observations très proches (selon la distance de Mahalanobis) des points de landmark déjà présents dans l'état du système sont acceptées. Dans des environnements bruyants ou dynamiques comme celui dans lequel nous travaillons, cette configuration est trop restrictive, entraînant des échecs d'association et affectant négativement les performances. Il est à noter que c'est la configuration qui met le plus de temps à être exécutée par le programme.

On peut voir dans les résultats de la figure 2, comme prévu, que pour chacune des 3 configurations il y a une augmentation prononcée de l'erreur de prédiction, sortant parfois des limites de la covariance (lignes rouges). On voit que l'effet de la taille

de la trajectoire et du nombre de landmarks est similaire à celui vu précédemment. Et contrairement à ce qui a été vu précédemment, dans aucun des cas, lors de la fermeture de la boucle, l'erreur de prédiction ne s'améliore, au contraire, elle augmente dans aucun des cas.

Un autre aspect qui ressort est l'augmentation de la taille des ellipses de covariance, qui représentent l'incertitude sur la position estimée de chaque point de landmarks. Celles-ci représentent la région de probabilité où le point de landmarks devrait se trouver, sur la base des observations et de l'état estimé du robot. Il est normal que ceux-ci augmentent en taille car nous sommes dans une configuration restrictive pour décider si la mesure correspond à un landmarks existant, ce qui fait qu'en ne recevant pas autant de mises à jour, l'incertitude sur la position des landmarks augmente progressivement à chaque pas, reflété par l'augmentation de la taille des ellipses de covariance.

B. `M_DIST_TH = 9.0 (Valeur modérée)`

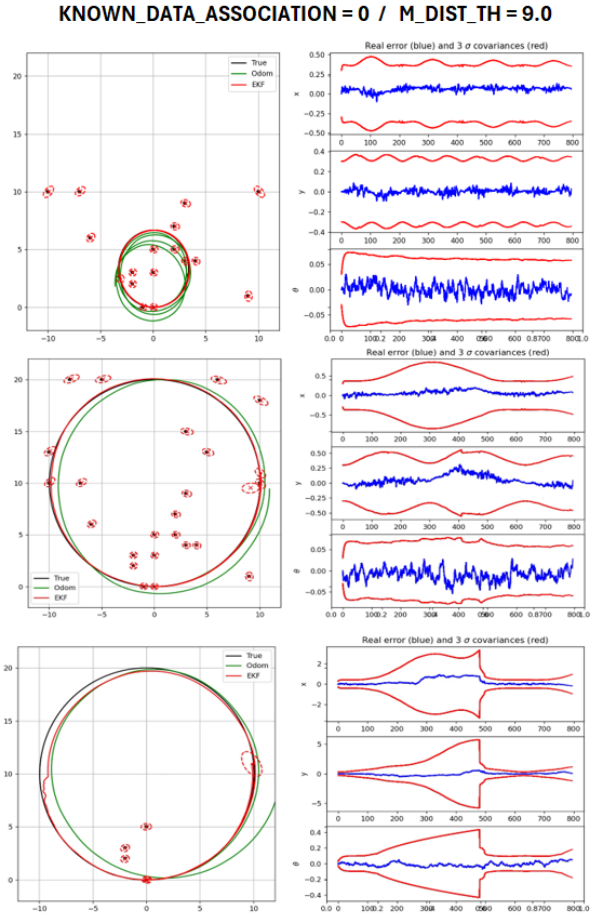

Fig. 3. Résultats pour `M_DIST_TH` = 9.0

L'augmentation du seuil donne un peu plus de tolérance dans l'association des données, permettant d'associer des points de landmark nouveaux ou mal observés à des observations légèrement en dehors de la distance minimale. Cela améliore l'association dans les environnements où le bruit est modéré ou où les points de landmark peuvent être plus dispersés, établissant ainsi un équilibre entre précision et flexibilité. Nous pouvons voir ces résultats sur la figure 3, où nous pouvons voir des résultats d'erreur de prédiction pour les trois scénarios qui sont assez faibles par rapport à ceux vus précédemment. Cette configuration permet au système d'être guidé correctement et de corriger la trajectoire, en maintenant les valeurs d'erreur dans les limites établies par la covariance.

On peut également constater un comportement moins bruyant tout au long du mouvement du robot et une correction odométrique adéquate tout au long du trajet. De plus, cette configuration permet de fermer la boucle de manière appropriée, en reconnaissant les LandMarcs qui indiquent le début et la fin de la boucle.

Les résultats de la variation du nombre de points de landmark et de la taille de la trajectoire restent les mêmes que ceux observés précédemment.

C. `M_DIST_TH = 50,0 (valeur élevée)`

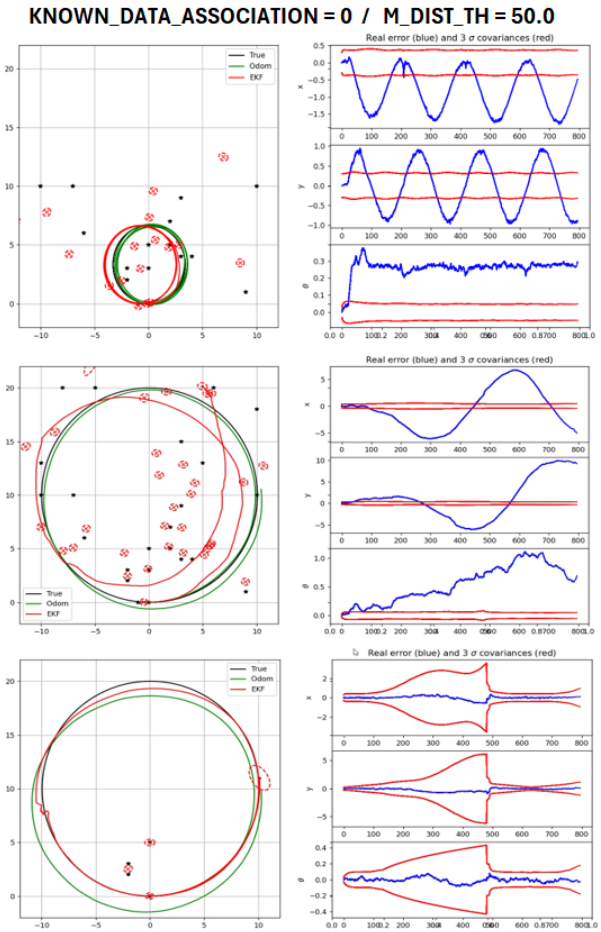

Fig. 4. Résultats pour `M_DIST_TH` = 50.0

Avec un seuil également élevé, l'association de données devient très permissive, acceptant des observations relativement éloignées (selon la distance de Mahalanobis) des points

de référence déjà existants dans l'état du système. Cela peut potentiellement augmenter les erreurs d'association, ce qui peut introduire des erreurs significatives dans la localisation et la cartographie.

Dans les résultats illustrés sur la figure 4, observez que pour les trois configurations de trajectoire et de distribution de points de référence, l'erreur de prédiction est bien supérieure aux cas précédents, sortant des limites proposées par la covariance, indiquant un dégradation de la qualité de l'estimation. La boucle de fermeture ne parvient pas à corriger efficacement la trajectoire et l'erreur de localisation augmente, ce qui affecte la précision du SLAM.

De plus, on constate qu'en comparant la trajectoire réelle (ligne noire) avec celle estimée (ligne rouge), cette dernière ne suit pas adéquatement la référence. Surtout dans le deuxième scénario, où nous constatons qu'un plus grand nombre de points de référence affecte cette configuration du système, car comme nous l'avons mentionné précédemment, cela augmente les erreurs d'association.

V. Question 3 : Modification des valeurs de bruit Q et P

Dans un filtre EKF SLAM, `Q` et `P_y` (et leurs équivalents `Q_Sim` et `Py_Sim` pour la simulation) sont des matrices de covariance qui représentent respectivement l'incertitude dans les modèles de mouvement et d'observation.

La matrice Q décrit l'incertitude du modèle de mouvement du robot. Capture les erreurs ou variations possibles qui peuvent survenir lorsque le robot se déplace, en raison de facteurs tels que le bruit dans les moteurs, les roues ou dans les capteurs qui mesurent le déplacement. Cette matrice intègre ces erreurs dans le processus de prédiction d'état.

La matrice P_y représente l'incertitude du modèle d'observation du robot, c'est-à-dire le bruit dans les mesures des amers détectés dans l'environnement. Cette matrice prend en compte l'erreur qui peut survenir dans les mesures de distance et d'angle par rapport au point de landmark en raison du bruit dans les capteurs qui le détectent (par exemple, un LIDAR).

Nous verrons ci-dessous les conséquences de la variation de ces deux valeurs.

A. `Q` et `P_y` sont inférieurs à `Q_Sim` et `Py_Sim` (0.2)

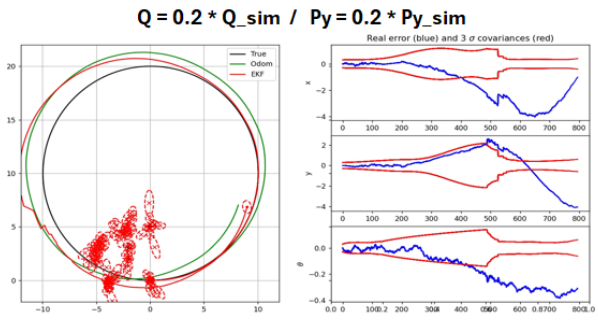

Fig. 5. Résultat avec 0.2

La réduction des valeurs de `Q` et `P_y` indique que le filtre estime moins d'incertitude dans le processus (mouvement du robot) et dans la mesure (détection de landmark) qu'elle n'est réellement présente dans la simulation. En conséquence, le filtre devient trop confiant dans les prédictions de mouvement et les observations de points de landmark, générant davantage d'estimations et donc plusieurs ellipses avant d'avoir des points de landmark solidement confirmés.

Cette faible incertitude, selon les résultats de la figure 5, conduit le filtre à effectuer des mises à jour agressives, en s'appuyant excessivement sur des observations qui ont en réalité plus de bruit que ce que le filtre suppose. Cela conduit à une accumulation rapide d'erreurs dans l'état estimé et éventuellement à des problèmes de dérive dans la position estimée du robot. Cette dernière peut être constatée en voyant les erreurs de prédiction de ce scénario, où ladite erreur dépasse les valeurs délimitées par la covariance.

La cohérence du filtre se détériore car la matrice de covariance d'état P sous-estime la véritable incertitude du système. Lorsque le filtre sous-estime l'incertitude, les observations futures sont moins susceptibles de se situer dans les ellipses de confiance calculées. Cela signifie que le filtre est moins susceptible de réussir les tests de cohérence (tels que ceux basés sur la distance de Mahalanobis), ce qui indique que ses estimations sont trop optimistes et ne reflètent pas la réalité.

La sous-estimation de l'incertitude affecte négativement la qualité de la carte. Le filtre estime les positions des points de landmark avec trop de confiance. Une confiance excessive peut également rendre difficile l'identification et la correction des erreurs de position des points de landmark, ce qui dégrade la précision globale de la carte. Ce qui précède a pour conséquence que l'estimation de la trajectoire est erronée, comme le montre la figure 5, et qu'elle suit l'itinéraire odométrique, étant plus évidente dans les endroits où il n'y a pas de landmarks pour le guidage.

B. `Q` et `P_y` sont égaux à `Q_Sim` et `Py_Sim` (1.0)

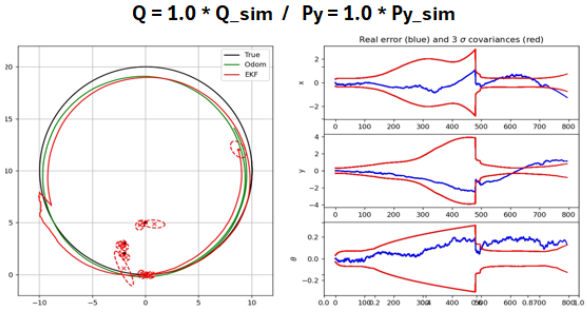

Fig. 6. Résultat avec 1.0

Régler `Q` et `P_y` sur la même amplitude que `Q_Sim` et `Py_Sim` signifie que le filtre a une représentation réaliste de l'incertitude à la fois dans le processus et dans la mesure. Cela permet au filtre de mettre à jour les estimations avec un équilibre approprié entre les prédictions de mouvement

et les observations de points de landmark, minimisant ainsi l'erreur dans l'état estimé et facilitant la convergence du filtre. Les performances s'améliorent car le filtre ne s'appuie pas trop sur les observations ou sur ses propres prédictions d'état. Cependant, comme on peut le voir sur la figure 6, lorsqu'on tente de boucler la boucle, les erreurs de prédiction augmentent légèrement, sortant des limites de la covariance.

Avec cette configuration, le filtre présente une cohérence raisonnable. En termes de distance de Mahalanobis, les observations se situent systématiquement dans les ellipses de confiance, ce qui indique que la matrice de covariance P reflète adéquatement l'incertitude du système. De plus, contrairement au cas précédent, les amers estimés sont cette fois plus proches de leurs positions réelles, puisque le filtre ajuste le niveau de confiance à chaque mise à jour. Cette configuration se traduit par une disposition assez précise et stable des points de landmark sur la carte.

C. `Q` et `P_y` sont supérieurs à `Q_Sim` et `Py_Sim` (2.0 and 20)

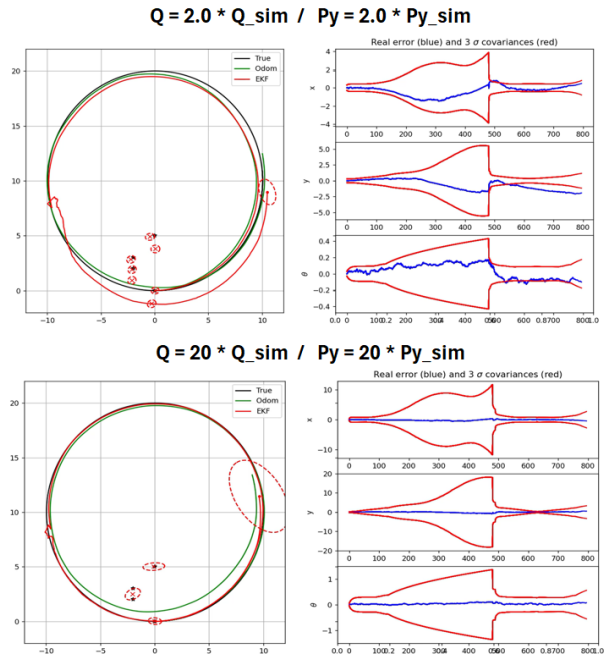

Fig. 7. Résultat avec 2.0 et 20

En augmentant `Q` et `P_y` , le filtre suppose qu'il y a plus de bruit ou d'incertitude dans le processus de mouvement et dans les observations qu'il n'y en a réellement. Cela amène le filtre à moins faire confiance à chaque mise à jour individuelle et à devenir plus conservateur dans l'incorporation de nouvelles informations. Les mises à jour seront moins agressives et le filtre ajustera lentement les estimations de position et les landmarks du robot. Comme nous le voyons dans les résultats, l'augmentation entraîne des bénéfices dans le résultat final, car à mesure que nous augmentons cette valeur, nous obtenons une estimation de trajectoire non seulement plus précise mais aussi avec moins de bruit.

Ce paramètre améliore la cohérence, car le filtre a tendance à sous-estimer sa confiance et offre une plus grande tolérance à la variabilité des observations. Le filtre réussit plus facilement les tests de cohérence basés sur la distance de Mahalanobis, puisqu'une observation est moins susceptible de sortir des zones de confiance. Cependant, il peut arriver que le filtre doute trop de chaque observation et accumule des erreurs.

Cette configuration génère une carte avec des landmarks précis, ce qui améliore la façon dont le chemin est estimé. On voit en comparant les cas où `Q` et `P_y` sont égaux à 2.0 et lorsqu'ils sont égaux à 20, qu'avec une valeur plus élevée le suivi de la trajectoire s'améliore considérablement, et à la fois En fermant la boucle, le système agit plus correctement dans la deuxième configuration, avec des performances presque parfaites.

D. Conclusion

En observant les résultats, on peut conclure que la meilleure configuration pour le filtre EKF SLAM, en termes de cohérence du filtre et de stabilité de la carte, a été obtenue lorsque les valeurs de bruit estimées `Q` et `P_y` ont été fixées à 20 fois les valeurs utilisées dans la simulation `Q_Sim` et `P_y,Sim`.

Dans cette configuration, bien que le filtre devienne plus prudent dans la mise à jour de ses estimations, l'augmentation de l'incertitude permet au système de maintenir la cohérence de ses prédictions, évitant ainsi les problèmes d'excès de confiance qui peuvent détériorer la qualité de la carte. Cette augmentation des valeurs de bruit permet au filtre d'être moins sensible aux erreurs d'association de données et offre une meilleure représentation des points de landmark sur la carte.

Par conséquent, la configuration avec Q et `P_y` à 20 fois les valeurs de simulation semble offrir un équilibre favorable, privilégiant la cohérence du filtre et produisant une carte fiable.

VI. Question 4 : Use only direction in Kalman correction. Implement a simple delay-free initialization for new landmarks by adding multiple landmarks along the sensing direction with increasing covariances.

A. Bearing-Only SLAM [1]

Le code mis à jour implémente le filtre de Kalman étendu (EKF) pour SLAM (Simultaneous Localization and Mapping) pour le robot mobile basé uniquement sur la direction (relèvement uniquement). Pour cela nous nous sommes inspirés des travaux de Joan Sola, André Monin, Michel Devy et Thomas Lemaire [1] qui ont proposé une initialisation non retardée pour améliorer les performances des filtres dans ce type de scénarios.

La première étape d'EKF SLAM consiste à prédire l'état et la covariance du système. Cela se fait en calculant les Jacobiens de la fonction de mouvement du robot et en mettant à jour `xEst` à l'aide de u. `PEst` est également mis à jour en fonction de l'incertitude et de `Q`.

Ensuite, la phase de correction ajuste l'estimation d'état en fonction des observations. Pour ce faire, toutes les observations

sont revues et on calcule s'il s'agit d'un point de référence connu ou s'il s'agit d'un nouveau. En fonction de la valeur de `KNOWN_DATA_ASSOCIATION`, une association de données est effectuée. Si l'association est connue, l'index correspondant est recherché. Dans le cas contraire, un algorithme de recherche est utilisé pour trouver la correspondance du point de repère. S'il s'agit d'un nouveau point de repère, le code étend la matrice d'état et de covariance du filtre de Kalman pour inclure le nouveau point de repère. L'innovation est calculée et l'estimation de l'état et la covariance sont ajustées à l'aide du gain de Kalman.

Il garde une trace du nombre de fois où chaque point de repère a été observateur (`landmark_vote`). Si un repère a été observé plusieurs fois, sa valeur de "vote" augmente. Si la fréquence des observations d'un point de repère tombe en dessous d'un seuil et si le nombre total de points de repère dépasse un seuil, le code supprime ce point de repère, ajustant à la fois l'état et la matrice de covariance en conséquence.

Enfin, l'angle d'orientation du robot (`xEst[2]`) est ajusté pour s'assurer qu'il se trouve dans la bonne plage à l'aide de la fonction `pi_2_pi`.

Le code renvoie l'état mis à jour du robot (`xEst`), la matrice de covariance mise à jour (`PEst`) et le vecteur de votes marquants (`landmark_vote`).

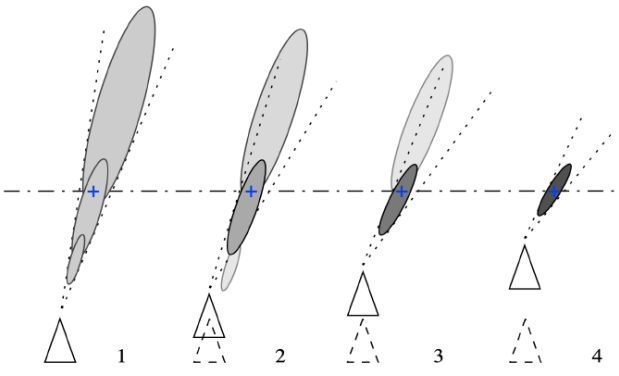

Fig. 8. Réduction de l'incertitude de position d'un landmark dans le Bearing-Only SLAM [1].

B. *Résultats*

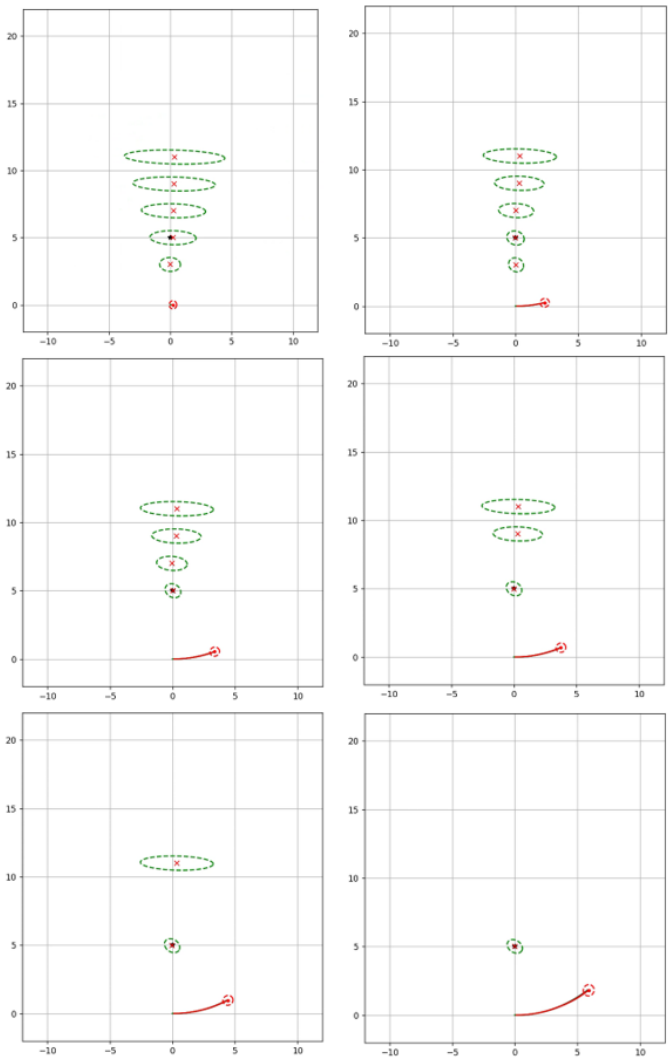

Fig. 9. Itérations avec initialisation non retardée en roulement uniquement en claquement [1].

Nous pouvons voir sur la figure 9 que ce qui est attendu est réalisé lors de l'utilisation de la méthode "d'initialisation sans délai".

Comme nous pouvons le constater, au départ, la position exacte du point de repère est incertaine, notamment en ce qui concerne la distance au robot. Cette incertitude se reflète dans les grandes ellipses.

À mesure que le robot se déplace et effectue de nouvelles observations du même point de repère sous différents angles, l'incertitude quant à la position du point de repère diminue. De ce fait, les ellipses de covariance deviennent moins allongées, indiquant une diminution de l'incertitude. C'est ce processus de raffinement qui permet d'initialiser et de localiser le point de repère sans avoir à attendre une grande précision initiale.

Finalement, après suffisamment d'observations, l'ellipse d'incertitude devient petite et bien définie. À ce stade, la

position du point de repère est estimée avec une grande précision et son incertitude est minime.

Cette convergence progressive illustre comment l'approche d'initialisation sans délai permet d'incorporer et d'affiner les points de repère dans la carte dès le premier moment d'observation, en tirant parti des observations successives pour réduire l'incertitude sans avoir besoin d'estimations initiales parfaites.

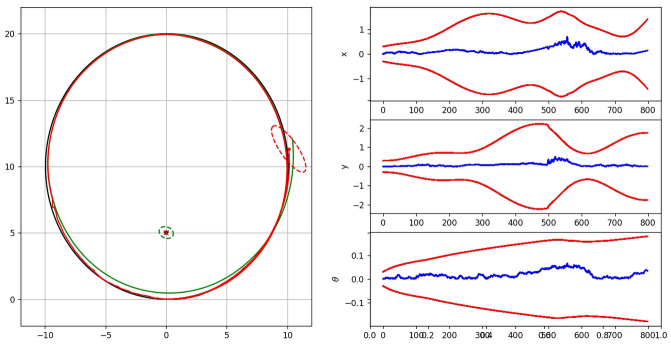

Fig. 10. Résultats utilisant l'initialisation non retardée en roulement uniquement en claquement [1].

Le résultat de la prédiction de trajectoire est visible sur la figure 10. Il est à noter que pour cela le filtre utilise uniquement la direction du repère dans la correction de Kalman

Le graphique montre comment EKF-SLAM s'adapte à la position réelle du robot et réduit les incertitudes dans l'estimation des points de repère. Les ellipses de covariance dans le graphique de gauche illustrent comment l'incertitude sur la position des points de repère diminue à mesure que le robot effectue des observations répétées à partir de différents points de la trajectoire. De plus, les graphiques de droite démontrent que l'estimation des états du robot devient de plus en plus précise, car les intervalles d'incertitude en rouge ont tendance à se stabiliser et à se rétrécir par rapport à la valeur réelle.

Si l'on compare ce résultat avec tous ceux obtenus précédemment, on voit que c'est la configuration qui génère le moins de bruit dans les prédictions de mouvement et c'est aussi celle qui donne un résultat de prédiction le plus précis, suivant presque parfaitement la courbe de trajectoire réelle.

Ce comportement est cohérent avec l'approche utilisée, dans laquelle l'initialisation sans délai permet d'ajouter des points de repère avec une incertitude élevée dans un premier temps, mais cette incertitude est rapidement réduite à mesure que le robot accumule des observations.

VII. Conclusions

En conclusion, le SLAM basé sur un filtre de Kalman étendu (EKF) est un outil puissant pour la robotique mobile, car il permet aux robots de construire une carte et de se localiser en temps réel dans un environnement inconnu. Cette capacité est essentielle pour la navigation autonome dans des environnements où les cartes prédéfinies ou les systèmes de positionnement global ne sont pas disponibles.

Tout au long de ce travail, nous avons vu comment les performances SLAM peuvent être améliorées en affinant les configurations d'environnement, l'association de données et les modèles de bruit probabilistes. Par exemple, en utilisant des associations de données plus avancées, comme la distance de Mahalanobis, ou en ajustant les paramètres de bruit du filtre, il est possible de réduire les erreurs d'estimation de la position du robot et des points de repère, augmentant ainsi la précision de l'évaluation. la carte et la fiabilité de la localisation. Même dans des configurations plus complexes, telles que le SLAM directionnel uniquement, l'EKF est capable de s'adapter à l'aide de techniques d'initialisation sans délai, obtenant ainsi de bons résultats dans des situations perceptuellement limitées.

Ces résultats démontrent que le SLAM basé sur l'EKF est non seulement efficace dans son objectif de construire des cartes et de localiser simultanément des robots, mais qu'il permet également une grande flexibilité pour s'adapter à différentes conditions environnementales et exigences de précision. La possibilité d'optimiser les paramètres de filtre et d'ajuster l'association de données en fonction de conditions spécifiques rend SLAM encore plus robuste.

VIII. GitHub

Vous pouvez vous référer à la référence suivante pour voir les codes de nœud du projet: GitHub/RO12/TP4

References

[1] Joan SOLA, André MONIN, Michel DEVY, and Thomas LEMAIRE. Undelayed initialization in bearing only slam. *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2499–2504. IEEE, 2015.*