ROB312 - TD GraphSLAM utilisant la méthode ICP

BRAMBILLA Davide Luigi

9 Janvier 2019

1 Cartographie incrémentale

La méthode ICP

La méthode ICP - *Iterative Closest Point* - s'occupe de trouver la transformation qui aligne l'ensemble des données qu'on est en train de traiter avec lequel utilisé de référence.
Il est souvent utilisé pour minimiser la différence entre deux nuages de points. L'algorithme minimise l'erreur sur la distance des points des données qui reçoit comme input en essayant de les aligner et donne en sortie la combinaison de transformations qui a été effectuée.
Les étapes de l'algorithme sont les suivantes:

- pour chaque point de l'ensemble actuel il trouvera le correspondant auprès de l'ensemble de référence
- il trouvera une combinaison de transformations afin de minimiser la distance entre les points
- il transformera les points avec la combinaison de transformations trouvé au point précèdent
- il fera plusieurs itérations

Q1 - Le fonctionnement de la méthode de cartographie implémentée dans le script

Le script initialise les données et les paramètres qui seront, ensuite, utilisés pour le traitement des données:

- les paramètres *minScan* et *maxScan* donnent le numéro du premier scan et du dernier scan qu'on va utiliser
- le paramètre *step* indique les pas avec lequel nous allons considérer les scans
- *distThresholdAdd* indique la distance de la dernier position retenue au-delà de laquelle nous allons considérer un scan comme intéressant et nous allons la ajouter à la carte.

Successivement le code commence son boucle et il traitera les scans comprises entre *minScan* et *maxScan* avec un pas donné par *step*.

Parmi les scans (et les relatives positions) déjà analysés, il trouve lequel le plus proche auquel traité actuellement et performe la méthode *ICP* entre eux. Ensuite, il corrige tous les scans qui doivent encore être traités et il ajoute le scan courant au plan s'il se trouve suffisamment loin du dernier scan sauvegardé (distance donnée par *distThresholdAdd*).

Ce qu'il faut remarquer c'est que, dans ce cas, la comparaison d'un scan est faite seulement avec lequel qui reste le plus proche.

Q2 - Conséquences de la fermeture de boucle

Lors de la fermeture de la boucle, nous pouvons observer que la position n'est pas reportée une autre fois. Le robot se rende compte qu'il passe dans une position proche à une position où il était déjà passé et notre algorithme ne la reporte pas à l'intérieure du plan. Cela est dû au fait de la présence de la valeur *distThresholdAdd* qui empêche à une nouvelle position d'être ajouté si trop proche d'une position déjà présente.

Pour ce que concerne la carte, on peut observer que elle n'est pas mise à jour si on est en train de repasser deux fois dans le voisinage d'une position ancienne. Cela est donné par le fait que la version incrémentale ne corrige pas la carte et les positions actuelles mais seulement la position courant. En fait, les points qui sont déjà présents dans le graphique ne sont pas modifiés et restent dans leur position.

Q3 - La variation du seuil *distThresholdAdd*

Le rôle du seuil *distThresholdAdd* est de ne pas sélectionner des points trop proches entre eux: en fait dans la figure 1 il est possible de voir que la distance entre un position et la prochaine est constante et est fixé par cette valeur.

Nous avons reporté dans la suite les images obtenues à partir des simulations: à gauche on aura l'ensemble des données qui sont traités et à droite la reconstruction qui a été effectuée.

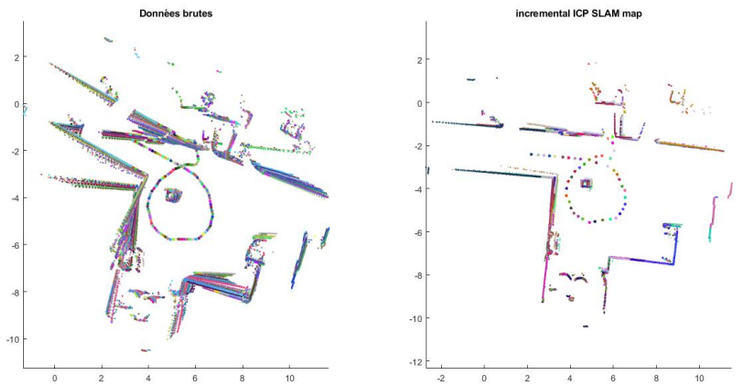

Figure 1: Résultat avec *distThresholdAdd* = 0.25

Valeur faible Nous avons essayé des valeurs très petites qui nous ramènent à la considération considérer beaucoup des plans dans la carte finale. Pour étudier le comportement avec une valeur faible nous avons choisi une valeur de 0.01. À partir des images, on peut observer que il n'y a pas des changements entre un plan capturé dans le moment *t* et le plan capturé dans le moment *t+1* et donc on va avoir encore les problèmes de superposition des murs comme on peut voir dans la figure 2.

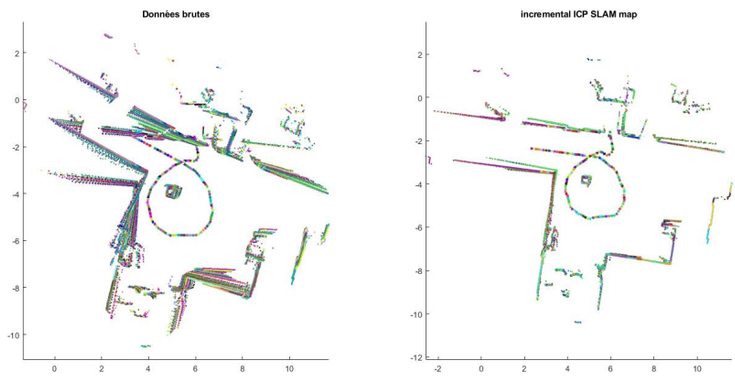

Figure 2: Résultat avec *distThresholdAdd* = 0.01

Valeur grande Pour étudier le comportement avec une valeur grande nous avons choisi d'attribuer à *distThresholdAdd* une valeur de 3 (figure 3). Ce qu'on observe c'est que le changement entre une position et la suivante est trop grand et nous obtenons que les points capturés au moment *t* ne sont plus disponibles au moment *t+1* et on a des problèmes à les unifier et à reconstituer la totalité de l'information.

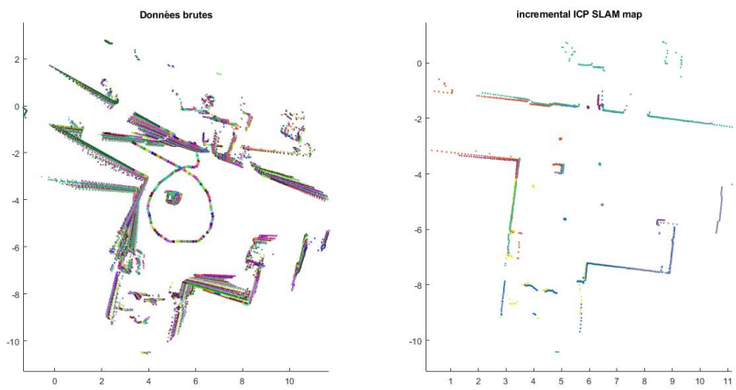

Figure 3: Résultat avec *distThresholdAdd* = 3

Si on exagère et on prend une valeur de 10 (figure 4) la carte qui vient reconstitué n'est pas utile car nous allons prendre trop peu des informations et nous n'allons pas reconstituer l'environnement.

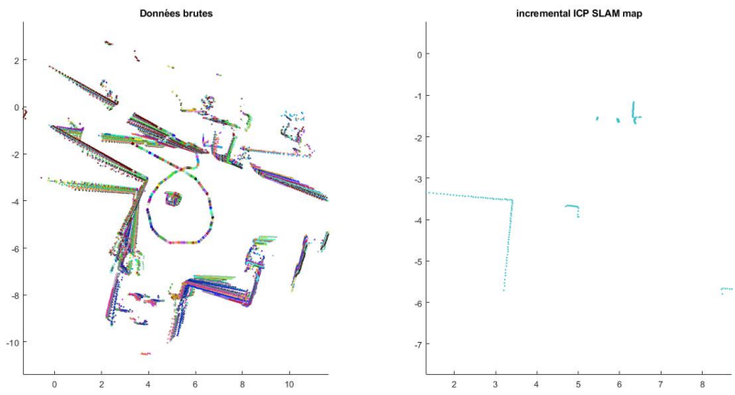

Figure 4: Résultat avec $distThresholdAdd = 10$

Conclusions Comme déjà remarqué dans les graphiques précédents, si on prend une valeur du seuil trop petit signifie que on est en train de perdre trop d'informations et on aura difficultés en reconstituer la carte selon différents angles. De l'autre coté, si on prend une valeur de seuil trop grand nous ne pourrons pas reconstituer le plan pour l'environnement car nous n'aurons pas une quantité d'informations suffisantes.

Dans ce cas, une valeur moyenne entre une valeur trop petite et une valeur trop grande nous permet de ne pas charger excessivement notre carte: nous allons avoir des points suffisamment loin afin de pouvoir résoudre les problèmes liés à la présence de trop de information qu'on peut voir dans le graphique de gauche et, au même temps, des points qui sont suffisamment proches afin de pouvoir reconstituer l'environnement en retrouvant les points correspondants en deux différents observations.

Pour trouver un bonne compromis entre ce deux valeurs nous avons varié la paramètre avec plusieurs valeurs et les résultats meilleurs nous les avons obtenus avec $distThresholdAdd = 0.75$.

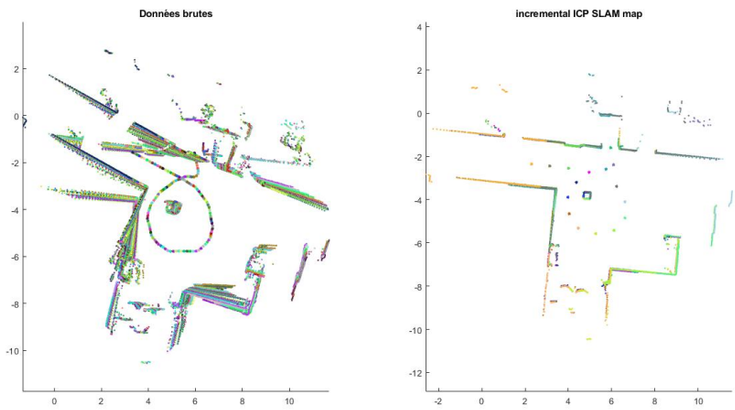

Figure 5: Résultat avec $distThresholdAdd = 0.75$

À partir de la figure ci-dessus, c'est possible de voir que les murs sont bien reconstitués et la précision sur leur position est satisfaisante. Nous ne trouvons plus les cas où les murs, pris selon différents perspectives, se superposent avec angles différents comme c'est possible de voir dans 1.

2 Cartographie GraphSLAM

Le GraphSLAM

SLAM est l'acronyme pour *Simultaneous localization and mapping*. Dans notre cas la parole *graph* indique une représentation d'un ensemble des points où des objets sont couplés au travers des relations entre eux.

L'objective de cette méthode est d'utiliser les relations entre les points d'un certain graphique afin de trouver une configuration qui minimise l'erreur introduit par les contraintes du problème.

Dans notre cas, nous chercherons de minimiser la distance entre les points correspondants parmi deux ensemble des données (c'est à dire deux différents scans).

Q4 - Le fonctionnement de la méthode de cartographie implémentée dans le script

Le code commence, comme dans la section précédente, en initialisant les donnés à traiter et les paramètres à utiliser. En particulier il faut remarquer la présence non seulement du seuil *distThreasholdAdd* mais aussi lequel du *distThresholdMatch*. Le premier a le même but du cas précèdent, c'est à dire, lequel de distancier

deux positions consécutifs et le deuxième permettra de corriger la position non seulement par rapport à la position précédente mais par rapport à toutes les positions qui se trouvent à la distance *distThresholdMatch* de la position courante: en particulier, la nouvelle position sera corrigée en utilisant une moyenne sur les différentes positions dans le voisinage.

Finalement, ce code se différencie par rapport au cas précèdent par la possibilité qu'il donne de modifier la carte et les positions précédentes.

Q5 - Conséquences de la fermeture de boucle

Lors de la fermeture du boucle, c'est possible de voir les effets sur la carte et sur les positions mentionnés dans la section précédente.

Pendant l'exécution, il est possible d'observer que, au moment où la position courante se rapproche des positions précédentes, comme dans le cas d'une fermeture du boucle, les positions antécédents et la carte se modifient et s'ajustent.

On peut donc affirmer que, pour ce que concerne la position, le comportement est similaire au cas précèdent: la position n'est pas ajouté si trop proche des positions déjà présentes.

Pour ce que concerne la carte, par contre, nous avons dans ce cas une mis à jour et un ajustement au moment de la fermeture de boucle et cela est dû au fait que la position courante se trouve à une distance *distThresholdMatch* des autres positions: ce qui rend possible la moyenne sur plusieurs points et donc une correction plus efficace et évidente.

Q6 - La variation des seuils *distThresholdAdd* et *distThresholdMatch*

Si nous allons étudier le cas où *distThresholdAdd* > *distThresholdMatch* nous allons retourner dans le cas précèdent où nous allons considérer seulement la position précédente et nous ne ferons pas une moyenne avec toutes les positions avec une distance inférieure à *distThresholdMatch*. Dans ce cas, les avantages données par ce méthode, comme la correction de la position des points antécédents n'est plus valable.

Nous allons maintenant essayer plusieurs valeurs pour notre algorithme afin d'étudier le comportement pour des valeurs petites ou grandes et de chercher des valeurs adaptes à notre cas. Nous allons toujours maintenir *distThresholdAdd* < *distThresholdMatch* afin d'exploiter les avantages de l'algorithme.

Valeurs faibles Pour étudier cette possibilité, nous avons donné la valeur de 0.01 à *distThresholdAdd* et la valeur de 0.1 à *distThresholdMatch*.
Comme c'est possible de voir dans la figure 6, nous allons avoir en sortie un graphique qui n'est pas trop précis et cela est du au fait que il y a trop de valeurs disponibles.

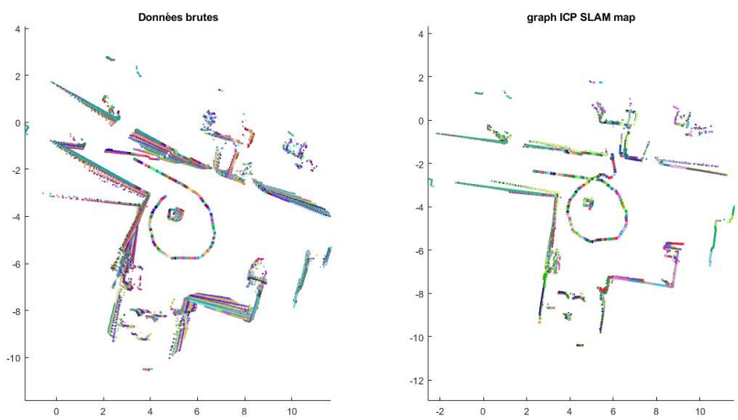

Figure 6: Résultat avec *distThresholdAdd* = 0.01 et *distThresholdMatch* = 0.1

Comme précédemment, il est possible de remarquer le fait que les murs ne sont pas bien définies et nous avons plusieurs incertitudes.

Valeurs grandes Nous allons maintenant étudier le comportement de l'algorithme dans le cas où on considère des valeurs très grands. En particulier, nous avons donné la valeur de 3 à *distThresholdAdd* et la valeur de 6 à *distThresholdMatch*.

Ce qu'il est possible de remarquer c'est que l'incertitude sur les murs que on avait dans le cas des *valeurs faibles* est disparu mais nous continuons à avoir le problème de perte d'informations donnée par le fait que nous sommes en train de traiter trop peu de données. Nous avons une bonne amélioration par rapport à la figure 3 de l'algorithme précèdent mais nous ne pouvons pas être satisfaits car la perte sur l'information est significative.

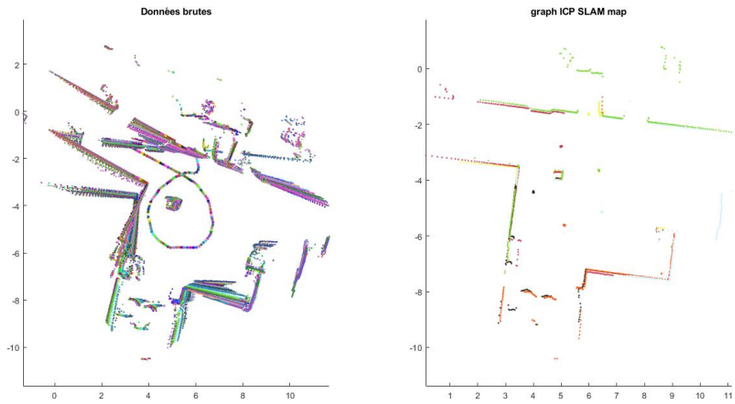

Figure 7: Résultat avec *distThresholdAdd* = 3 et *distThresholdMatch* = 6

Conclusions Enfin, nous nous sommes concentrés sur la recherche des valeurs qui nous donnent un résultat satisfaisant dans notre environnement. Nous sommes parti des résultats obtenu dans le cas précèdent en attribuant la valeur de 0.75 à *distThresholdAdd* et nous avons obtenu de très bon resultats en attribuant la valeur de 1.25 à *distThresholdMatch*.

Nous avons reporté ci-dessous le graphique obtenu:

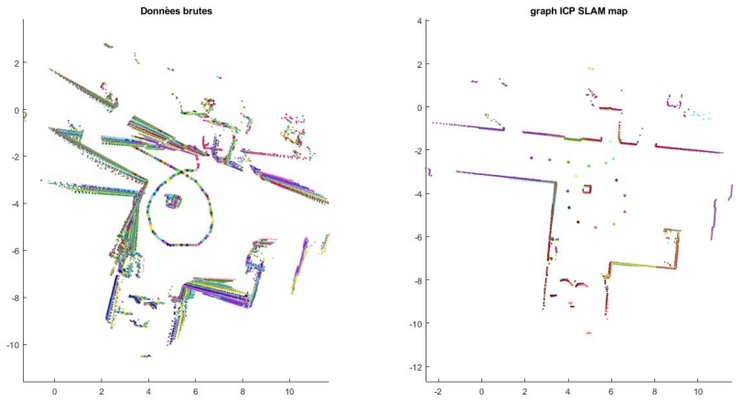

Figure 8: Résultat avec *distThresholdAdd* = 0.75 et *distThresholdMatch* = 1.25

3 Influence de l'environnement

Pour la simulation, nous avons essayé de donner des valeurs similaires aux deux méthodes et, pour cela, nous avons choisi de leur attribuer les valeurs qui nous ont permis d'obtenir des résultats satisfaisants dans les sections précédents: on donnera la valeur de 0.75 à *distThresholdAdd* et la valeur de 1.25 à *distThresholdMatch*.

3.1 Environnement U2IS

Cartographie incrémentale

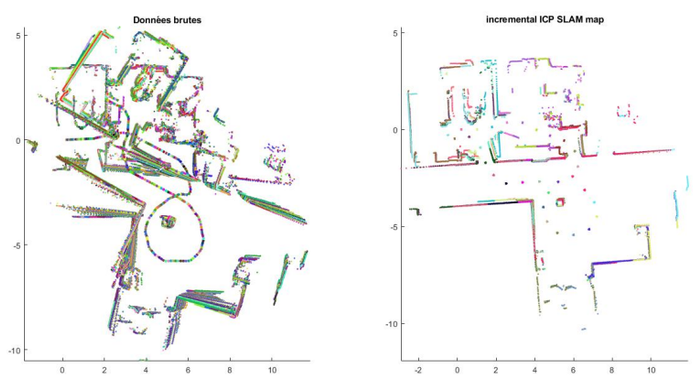

Figure 9: Résultat pour Cartographie incrémentale avec *distThresholdAdd* = 0.75

Cartographie GraphSLAM

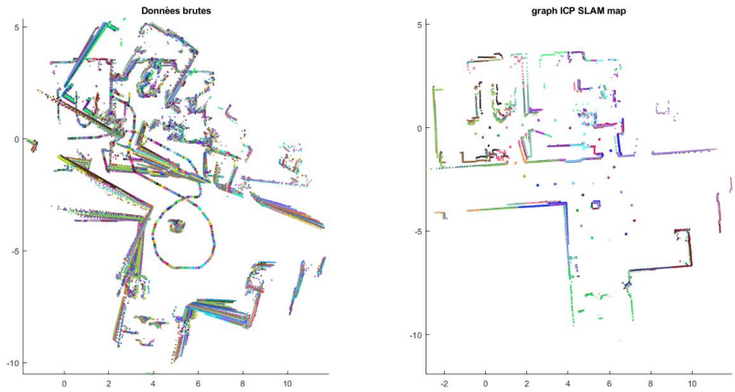

Figure 10: Résultat pour GraphSLAM avec $distThresholdAdd = 0.75$ et $distThresholdMatch = 1.25$

3.2 Environnement CSE550

Cartographie incrémentale

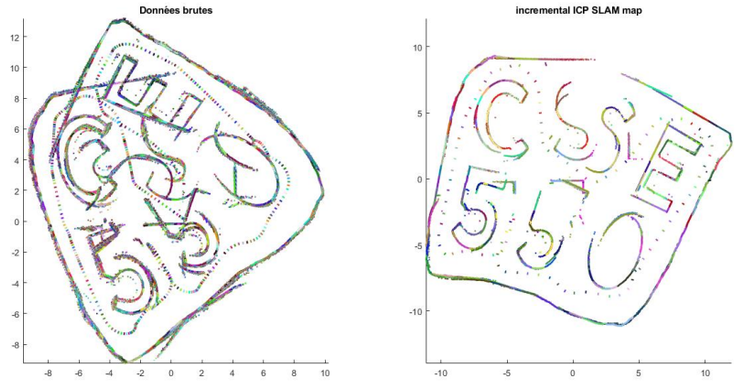

Figure 11: Résultat pour Cartographie incrémentale avec $distThresholdAdd = 0.75$

Cartographie GraphSLAM

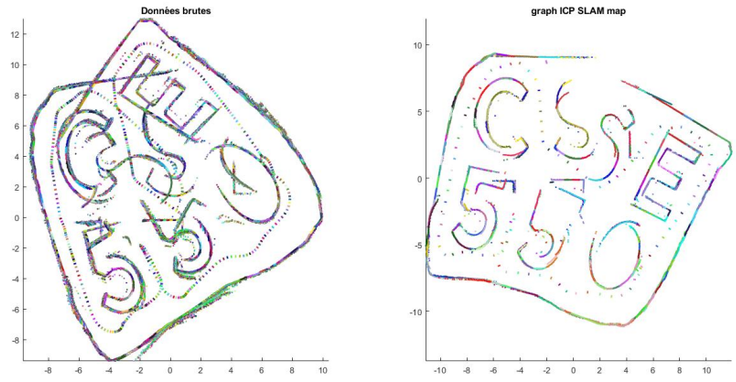

Figure 12: Résultat pour GraphSLAM avec $distThresholdAdd = 0.75$ et $distThresholdMatch = 1.25$

3.3 Environnement FR079

Cartographie incrémentale

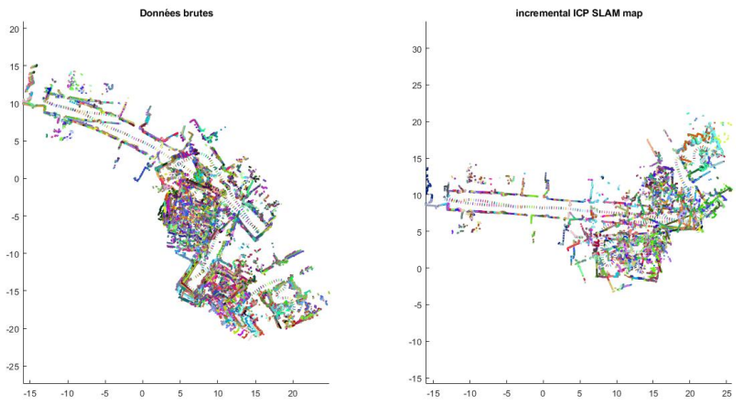

Figure 13: Résultat pour Cartographie incrémentale avec $distThresholdAdd = 0.75$

Cartographie GraphSLAM

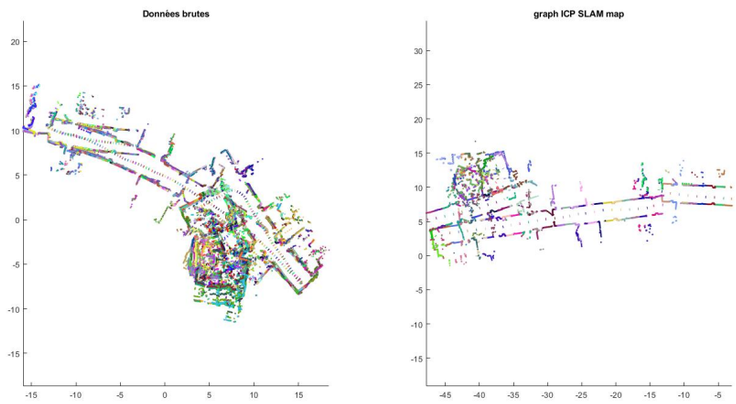

Figure 14: Résultat pour GraphSLAM avec $distThresholdAdd = 0.75$ et $distThresholdMatch = 1.25$

Conclusion

Nous pouvons observer que les méthodes ont des comportements similaires sur les deux environnements *U2IS* et *CSE550* mais que nous permettent d'arriver à des résultats différents sur l'environnement *FR079*.

Dans les deux environnements *U2IS* et *CSE550* nous avons des parcours qui présentent des caractéristiques similaires comme par exemple le fait d'avoir des trajectoires qui effectuent peu fermetures de boucle et mettent à disposition des positions qui sont souvent éloignées entre eux. Cela ne permet pas d'exploiter l'avantage du moyennage donnée par la cartographie de type *GraphSLAM*. En particulier, dans l'environnement *CSE550*, l'on peut remarquer que la trajectoire ne permet pas de faire trop de corrections et les résultats des deux cartographies sont très similaires.

C'est dans le cas de l'environnement *FR079* que nous avons l'amélioration plus intéressante entre les deux méthodes. Les données de cet environnement représentent une trajectoire qui se superpose en continuation: ce qui permet à l'algorithme d'utiliser la moyenne avec les positions proches afin de recaler et trouver la meilleure position des scans.

On peut voir que dans la figure 13, l'algorithme, après un certain temps commence à avoir des problèmes à réorganiser les données sur la droite du graphique. Par contre, dans l'image 14 les problèmes sont beaucoup moins importantes et la carte devient beaucoup plus claire.