Université de Mons

TP4

Défis en Intelligence Artificielle

Auteur :
Maxime De Wolf
Dimitri Waelkens

14 janvier 2019

1 Configuration de *Cartographer*

La configuration de cartographer est réalisée via son fichier de configuration : *uafr_ cartographer.lua*. Dans ce fichier nous avons modifier trois paramètres :

— *num_ accumulated_ range_ data* définit le nombre de messages requis pour reconstruire un *scan* complet (une image de l'espace autour du robot au temps t). En augmentant cette variable nous augmentons la précision d'un *scan*.

— *num_ subdivisions_ per_ laser_ scan* définit le nombre de points utilisés par *scan*. Ces points sont ensuite utilisés pour reconstituer la carte. Augmenter ce nombre de points permet à *cartographer* de mieux reconstituer une sous-carte.

— *TRAJECTORY_ BUILDER_ 2D.submaps.num_ range_ data* correspond à la taille des sous-cartes. Ce paramètre doit être plus grand que *num_ accumulated_ range_ data*.

Augmenter ces paramètres permet d'augmenter la qualité de la carte calculée. Nous avions initialisé ces trois paramètres à 200. Et nous avons obtenu le résultat illustré par la Figure 1.

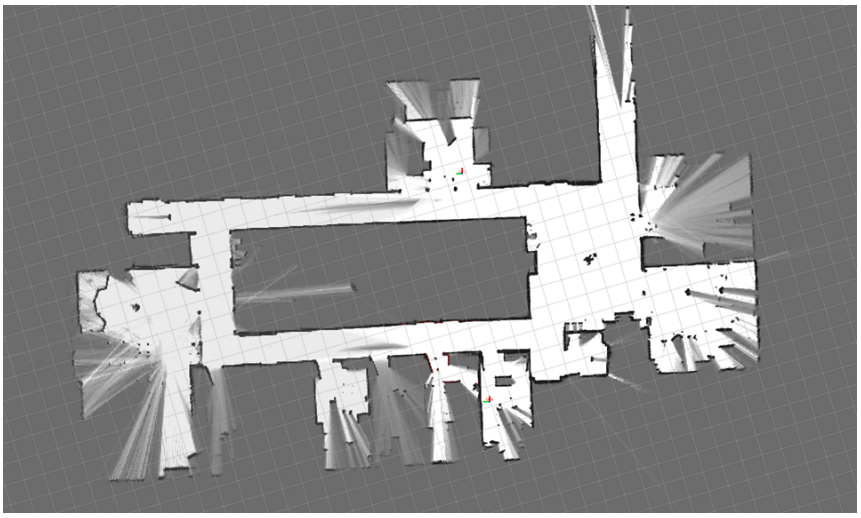

FIGURE 1 – Carte obtenue grâce à *Cartographer*

Ce résultat est assez bon. Chaque pièce est bien reconstituée, modulo les meubles qui ne permettent pas de réaliser une représentation complète de chacune des pièces. Aucune pièce n'en chevauche une autre. De plus les mur et les angles sont bien droits. Augmenter au-delà ne permettrait pas d'améliorer sensiblement le résultat et risque de causer de la latence à cause de calculs inutiles.

2 Localisation des objets

Cette section présente les résultats obtenus pour la détection d'objets que le robot a filmé. Nous expliquons donc dans un premier temps la logique que nous avons suivie pour obtenir ces résultats. Nous justifions ensuite la manière dont ils pourraient être améliorés. Cette détection des objets est réalisée grâce au fichier *object_finder.py*.

2.1 Technique utilisée

Pour résoudre ce problème, nous utilisons une technique très simple. Effectivement, à chaque fois que le robot détecte un objet, nous repérons la position du robot ainsi que la distance à laquelle l'objet à été filmé. Cependant, nous avons également besoin de l'orientation du robot pour calculer la position de l'objet.

Afin d'avoir l'orientation du robot, nous repérons également l'emplacement précédent duquel le robot voyait également cet objet. Nous pouvons donc calculer l'angle formé par la droite passant par ces deux points dans le repère. Ensuite, grâce à la distance de l'objet filmé, nous pouvons déterminer la position de celui-ci.

Le robot travaille sur un grand nombre d'images similaires. Nous calculons donc plusieurs fois la position du même objet.

Pour pallier à ce problème nous utilisons un algorithme de *clustering* simple afin de regrouper les positions qui sont susceptible de représenter un même objet.

Cet algorithme fonctionne de la manière suivante :

1. Pour chaque position, on regarde si elle est proche du centre d'un *cluster* existant ;
2. Si c'est le cas, on l'ajoute à ce *cluster* ;
3. Si cette position n'est proche d'aucun *cluster*, on en crée un nouveau qui contient cette position ;

Finalement pour retrouver la position de chaque objet, on calcule le centre de chaque *cluster*. Cette technique nous donne les résultats exposés par la Table 1.

Objet	Position 1	Position 2
teddy bear	(2.399926850060335, -0.008283207609849293)	(12.604331235659147, 14.238862468344005)
motorbike	(8.366541118657906, 14.943175542227172)	/
backpack	(4.832979494372698, 2.7004257263585636)	/
suitcase	(5.446312729568893, 16.221012094608028)	/
pottedplant	(12.343678479509247, 14.322766597693517)	(0.9148871784197752, 23.25876369786711)
stop sign	(7.87594829417553, 15.93296178462322)	/
refrigerator	(6.239195085908132, 13.789541846887905)	/

Table 1 – Positions obtenues pour chaque objet détecté

2.2 Discussion sur les résultats

Grâce à la Table 1, nous pouvons voir que nos résultats sont assez mauvais par rapport aux résultats attendus car nous ne détectons pas le bon nombre d'objets. Nous dressons donc une liste non-exhaustive de tous les facteurs qui pourraient être responsables.

Pour retrouver la position du robot précédant la détection de l'objet, nous prenons la position du robot durant la détection précédente d'un objet. Cela implique que nous prenons éventuellement la position du robot pendant la détection d'un autre objet. Cela peut donc causer la création de *clusters* artificiels.

Une solution pour régler ce problème est de tester si l'objet précédemment détecté est le même que celui en cours de détection. On pourrait notamment faire ceci en se basant sur la technique de *clustering* que nous avons utilisée.

Une autre solution serait de ne plus prendre la position du robot pendant la détection de l'objet précédent mais de sélectionner la position du robot juste avant de détecter l'objet. Cette solution est selon nous la plus logique à appliquer.

Notre technique de *clustering* est très (trop ?) simple. Dès lors, on peut se demander si d'autres techniques donneraient de meilleurs résultats. Effectivement, chaque position que nous trouvons découle de la formation d'un *cluster*. Un problème à ce niveau aurait donc des conséquences directes sur nos résultats au niveau du nombre d'objets détectés.

Le robot nous fourni la distance à laquelle se trouve ce qu'il voit. Cependant, il arrive qu'il détecte plusieurs objets sur la même image. Dans ce cas nous utilisons la même distance sur tous ces objets ce qui implique que tous les objets présents sur une même image auront d'office la même position. Or, cela est loin d'être vrai. Ce problème influe donc sur la position des objets détectés, ce qui a une incidence sur les *clusters* formés et donc sur les résultats finaux.