Ludovic Chavalarias, Lab-STICC, 2022

Table des matières

[Débogage 1](#_Toc101939831)

[Marquer les zones inconnue d’une carte comme infranchissable 1](#_Toc101939832)

[Augmenter la sécurité du robot 1](#_Toc101939833)

[Paquet waypoints saute des points 2](#_Toc101939834)

[Le robot est bloqué à un point et les resets ne fonctionnent pas 3](#_Toc101939835)

# Débogage

## Marquer les zones inconnue d’une carte comme infranchissable

Dans le fichier « base\_local-planner\_default\_params.yaml », ajouter la ligne *allow\_unknown : false.* Cela a pour effet d’interdire la planification de trajet à travers les zones inconnue de la carte. Dans le labo, cela est utile au niveau des vitres.

Dans le fichier « cosmap\_common\_params.yaml » ajouter une « layer » :

obstacle\_layer :

observation\_sources : scan

scan :

data\_type : LaserScan

topic : scan

marking : true

clearing : true

track\_unknown\_space : true

## Augmenter la sécurité du robot

Jouer avec les parametres *inflation\_radius* et *cost\_scaling\_factor* afin d’augmenter la distance au obstacles.

Augmentation aussi de l’empreinte du robot pour ajouter une distance de « sécurité ». De manière général, l’empreinte devrait toujours être un peu plus grande que le robot lui-même.

## Paquet waypoints saute des points

On détermine à quoi serve les topics :

|  |  |
| --- | --- |
| /waypoint\_server/statusgoal | Indique si le l’objectif actuel est un succès ou non. Reçois un message seulement pour indiquer si est but est accomplis ou non. Ne reçois pas de message pendant l’accomplissement de l’objectif. |
| /waypoint\_server/waypoints |  |
| /waypoint\_server/waypoints\_gr | Nom du groupe et liste des points que le robot est en train de parcourir |
| /move\_base/current\_goal | Position que le robot veut atteindre. Reçois un message à chaque nouvelle objectif reçu |
| /move\_base/result | Indique qu’un objectif ait été atteint (status = 3) |
| /move\_base/status | Liste de status de l’objectif actuel :   * 1 : objectif accepté * 4 : abandon après que toutes les tentatives de déblocages aient échouées |
| /move\_base/recovery\_status | Reçois des messages lorsque des tentatives de déblocage sont en cours. |

**Observation** : Dans le fichier tour\_bug\_skipEtRecovery.bag, le bug peut être observé, c’est le point au niveau de la pause IRDL qui est sauté. Lorsque le robot arrive au point de passage le précédent, le topic **/move\_base/status** comporte deux entrées : reached et goal accepted. Cependant il comporte ensuite **3 entrées**: deux reached et une goal accepted. Cela confirme que le point est accepté puis directement considérer comme atteint

**Résolution** :

Problème similaire : <https://github.com/ros-planning/navigation/issues/949>

On fait les modifications du lien au-dessus dans les fichier move\_base.cpp et move\_base.h

## Le robot est bloqué à un point et les resets ne fonctionnent pas

* amcl.launch
  + laser\_likeihood\_max\_dist = 4 from 2
  + laser\_siglma\_hit = 0.1 from 0.2
  + laser\_z\_hit = 0.9 from 0.5

Il se peut que les costmap ne soit pas réellement reset lors des tentatives de recovers. Dans le fichier tour\_bug\_skipEtRecovery.bag le bug est observable au niveau du bureau des ingés. Au moment du bug les costmap ont été reset de façon manuelle via un appel au service */move\_base/clear\_local\_costmap*. On peut voir que le robot reprend sa route.

**Reproduire le bug**

Lorsque le robot se déplace, il update sa carte avec les obstacles qu’il voie. Le problème est que les obstacles qu’il a vus reste dans la carte une fois que le robot s’est éloigné. C’est pour cela qu’il bug parfois au niveau du bureau des ingé.

Pour le reproduire, il suffit de se tenir dans l’encadrement de la porte du bureau pendant que le robot s’éloigne. Il va considérer que des obstacles bloque le passage. Arrivé au bureau des ingés, il ne pour donc pas trouver de chemin vers le bureau SHAKER.

Pour faire le tour sans problème, il suffit de ne pas se tenir dans l’encadrement de la porte pendant que le robot s’éloigne.

Ce comportement est observable dans les enregistrements « tour\_bureau\_2.bag » et « tour\_bug\_recovery.bag »

**Correction**

Le problème vient des configurations pre-hydro utilisées par défaut. Pour corriger le phénomène, il faut modifier les fichiers suivant :

* *local\_costmap\_param.yaml* en ajoutant:

plugins :

{name : obstacle\_layer, type : “costmap\_2d::ObstacleLayer”}

{name : inflation\_layer, type : “costmap\_2d::InflationLayer”}

obstacle\_layer :

observation\_sources : scan

scan :

data\_type : LaserScan

topic : scan

marking : true

clearing : true

track\_unknown\_space : true

inflation\_layer:

inflation\_radius: 1.2

cost\_scaling\_factor: 3

* *global\_cosmap\_param.yaml* en ajoutant :

plugins :

{name : static\_layer, type : “costmap\_2d::StaticLayer”}

{name : obstacle\_layer, type : “costmap\_2d::ObstacleLayer”}

{name : inflation\_layer, type : “costmap\_2d::InflationLayer”}

obstacle\_layer :

observation\_sources : scan

scan :

data\_type : LaserScan

topic : scan

marking : false

clearing : true

track\_unknown\_space : true

* *costmap\_common\_params.yaml* en déclarant les « layers » :

static\_layer :

map\_topic : map

inflation\_layer:

inflation\_radius: 1.2

cost\_scaling\_factor: 3

Effet : de cette façon les couches sont déclarées et utiliser grâce à la déclaration des plugins. Cela permet de les configurer comme on le souhaite, étant donné que le problème venait de la configuration par défaut.

Le résultat est que les obstacles observés par le Lidar sont supprimés de la costmap une fois que le robot s’est éloigné.