Robotique/V2X UTAC Développement d'un nouveau mode conduite dans un convoi autonome

Joseph LOUVILLE

1^{er} Manuel de manipulation

Tuteurs :

Mme Bouchemal

M. Jun Kim

Fait à Paris, le 28/04/2022

Afin de faire compiler le projet il faut appeler la fonction **catkin_make** depuis le dossier root, ici dans notre exemple **catkin_ws**. Si les dossiers build et devel ont été importés ou déplacés, il est nécessaire de les supprimés avant de compiler le projet.

A partir de ce point normalement, on doit désigner comme source de projet le fichier **setup.bash** dans le dossier **devel** afin de pouvoir faire appel des fonctions de ROS dans la fenêtre de commande et qu'elles s'appliquent au projet compilé.

Les fonctions ROS intéressantes que j'ai pu voir sont **roslaunch [launch_package] fichier_launch.launch**, qui permet de lancer la simulation d'un package qui possède un fichier .launch associé, **rosrun nom_de_node**, qui permet de faire fonctionner une node (nœud).

Afin de lancer le projet il est indispensable que toutes les nodes soient lancées. Donc pour lancer le projet il existe plusieurs possibilité : soit faire l'appel de tous les nodes dans le fichier .launch, c'est à dire :

```
<!--in ecebot ecebot_turtle.launch-->

<node pkg="ecebot" name="path_turtle" type="path_turtle.py" output="screen"/>
<node pkg="vehicles" name="voiture_1" type="voiture_1" output="screen"/>
<node pkg="vehicles" name="voiture_2" type="voiture_2" output="screen"/>
<node pkg="vehicles" name="voiture_3" type="voiture_3" output="screen"/>
<node pkg="vehicles" name="voiture_4" type="voiture_4" output="screen"/>
<node pkg="controler" name="controler" type="controler" output="screen"/>
<node pkg="light" name="light" type="light" output="screen"/>
<node pkg="ecebot" name="following_turtle_2"
                    type="following_turtle_2.py"args="2" output="screen"/>
<node pkg="ecebot" name="following_turtle_3"
                    type="following_turtle_3.py" args="3" output="screen"/>
<node pkg="ecebot" name="following_turtle_4"
                    type="following_turtle_4.py" args="4" output="screen"/>
```

Cela permet juste d'appeler un **roslaunch** afin de lancer le projet. L'autre façon de faire est d'appeler chaque node via des rosrun.

```
roslaunch ecebot ecebot_turtle.launch
rosrun vehicles voiture_1
rosrun vehicles voiture_2
rosrun vehicles voiture_3
rosrun vehicles voiture_4
rosrun light light
```

Après avoir appelé ces commandes, Gazebo se lance et il est possible de faire fonctionner la simulation en appuyant sur le bouton play.

1 VM

Projet du PFE n°2026 composé de Margot IRLINGER, David OLIVARES, Emma PALFI, Jean PROUVOST, Alexandre SEGERAL, Jimmy VUONG.

Commande à utiliser dans un terminal :

```
$ cd /Bureau/essai_ws
$ source devel/setup.bash
$ roslaunch ecebot ecebot_turtle.launch
```

Il faut ensuite appuyer sur play sur Gazebo pour lancer la simulation.

Projet de la valorisation du groupe composé de Adrien Mailly, Timothé Bruckert, Florent Fonsalas, Benjamin Hubau, Rémi Breton, Valentin Martins.

Commande à utiliser dans un terminal :

```
$ cd /Bureau/essai_ws
$ source devel/setup.bash
$ roslaunch ecebot ecebot_turtle.launch
```

2 Copadrive

L'installation de CopaDrive nécessite de télécharger les repositories de Enioprates :

— CopaDrive :
git clone −−recurse−submodule https://github.com/enioprates/copaDrive.git

— Artery :
git clone − −recurse − submodule https://github.com/enioprates/artery.git

Dans le dossier *.../CISTER_car_simulator* et *.../CISTER_image_processing*, la commande à faire est :

```
source devel/setup.bash
```

Ensuite il faut faire la commande dans ... */Cister_car_simulator* : ... *roslaunch car_demo demo.lau*

Et il faut faire la commande dans ... */Cister_image_processing* :

... *roslaunch image_processing vehicle.launch*

et enfin pour lancer OMNeT++ :

cmake –build build –target run_gazebo-platoon