Documentation Technique

Option Robotique Août 2017

Thierry PROST (id. 227637) - Classe Asc1 – Virtual Campus

**** Par manque de temps et une activité professionnelle intense ces derniers mois, je n'ai pu mener à terme ce devoir ****

**** Je me suis concentré essentiellement sur l'analyse et la spécification du sujet choisi ****

**** La partie conception est bien entamée mais loin d'être terminée ****

1. PARTIE 1 – Choix du Projet et Explications

Sujet choisi : Projet 5 **Formule 1**.

Technologie : ROS – Simulateur Gazebo *(projet normalement à faire sous NetLogo mais vu avec les formateurs lors de la dernière session d'accompagnement du 2 août qu'il pouvait être réalisé sous ROS / Gazebo, j'ai donc pris cette direction).*

Etat actuel du Projet : Lancement (d'un ou de plusieurs) véhicule(s) autonome(s) sur un circuit (environnement) connu à l'avance. Déroulement de la simulation fonctionnelle dans le simulateur Gazebo ou le(s) agent(s) roulant(s) partent d'un point de départ et connaissent leur point d'arrivée (x tours de circuit) avec détection de voie(s) propre (lane detection).

Parmi les choses demandées, et malgré une analyse et spécification détaillée dans le dossier UML, voici ce que je n'ai pas eu le temps d'implémenter :

- Agents pilotes, public, techniciens, dirigeants
- Gestion émission champ positif et répulsif pour dépassement
- Gestion des collisions

Déterminant de mon choix de projet et Futur : Ce n'est pas parce que j'ai manqué de temps pour la réalisation présente que je ne souhaite pas continuer son développement. Le choix de la technologie et du sujet n'est pas un hasard :

- Pour la technologie, après exploration des divers outils proposés (NetLogo, ROS, …) vus en cours, j'ai ressenti plus d'affinité avec ROS et Gazebo pour un rendu et une simulation plus élaborés.
- Pour le sujet, il propose un réel challenge et j'ai pu me baser sur quelques exemples intéressants (FormulaPi et Husky par exemple) qui me permet dans le temps d'étendre le projet à une utilisation matérielle passant par l'utilisation de composants « Arduino » ou « Raspberry ». On ne demande pas la réalisation physique du projet mais il me semble important que je puisse aller un cran plus loin en expérimentant cette solution.

D’autres projets prévus ?

Je souhaite dans un futur proche développer un système de pointage satellite très précis (utilisation bidirectionnelle) afin de bénéficier d’une connexion internet dans mon camping-car poids-lourd dans lequel je réside à l’année.

PARTIE 2 – Présentation des outils utilisés

Je me suis largement inspiré du projet FormulaPI pour plusieurs raisons :

- C’est un bon point de départ
- Une partie simulateur développé avec ROS + GAZEBO
- Utilisation des agents roulants Husky de la société Clearpath Robotics
- Complètement personnalisable
- Deux langages possibles : Python ou C++
- Possible de concrétiser la simulation avec Arduino ou Raspberry

Détails de FormulaPI : https://www.wilselby.com/research/autonomous-vehicles/formulapi-autonomous-vehicle-racing/

Dépôt GIT du Projet Robotique : https://github.com/3kynox/ROS_GAZEBO_CAR_RACING/

Récupération de la Machine Virtuelle VMWare Workstation 12 (Ubuntu startup autologin) :

Login : nox

Pass : azerty123

PARTIE 3 – Chronologie pratique

I. Mise en place de la VM et de l'environnement de développement
IMPORTANT : Désactiver « Accelerate 3D Graphics » dans VMWare

1. Installation et Utilisation Ubuntu 14.04 (Trusty) avec ROS Indiego.
2. Récupération de ROS et initialisation :

```
➢ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc
  ) main" > /etc/apt/sources.list.d/ros-latest.list'
➢ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421
  C365BD9FF1F717815A3895523BAEEB01FA116
➢ sudo apt-get update
➢ sudo apt-get install ros-indigo-desktop-full
➢ sudo rosdep init
➢ rosdep update
➢ echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc
➢ source ~/.bashrc
```

3. Création du dossier du projet et compilation avec husky (agent voiture & simulation)

```
➢ mkdir -p catkin_ws/src
➢ cd ~/catkin_ws/src
➢ git clone https://github.com/husky/husky.git
➢ git clone https://github.com/husky/husky_simulator.git
➢ git clone https://github.com/husky/husky_desktop.git
➢ cd ..
➢ catkin_make
➢
```

4. Récupération des dépendances

```
➢ sudo apt-get install ros-indigo-universal-robot
➢ sudo apt-get install ros-indigo-lms1xx
➢ sudo apt-get install ros-indigo-vocs-cmd-vel-mux
➢ sudo apt-get install ros-indigo-gazebo-ros-control
➢ sudo apt-get install ros-indigo-hector-gazebo
➢ sudo apt-get install ros-indigo-teleop-twist-joy
➢ sudo apt-get install ros-indigo-robot-localization
➢ sudo apt-get install ros-indigo-interactive-marker-twist-server
➢ sudo apt-get install ros-indigo-joint-state-controller
```

```
sudo apt-get install ros-indigo-diff-drive-controller
sudo apt-get install ros-indigo-dwa-local-planner
sudo apt-get install ros-indigo-rviz-imu-plugin
sudo apt-get install ros-indigo-twist-mux
sudo apt-get install ros-indigo-move-base
```

5. Mise à jour de Gazebo

```
sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu trusty ma
in" > /etc/apt/sources.list.d/gazebo-latest.list'
wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -
sudo apt-get update
sudo apt-get upgrade
```

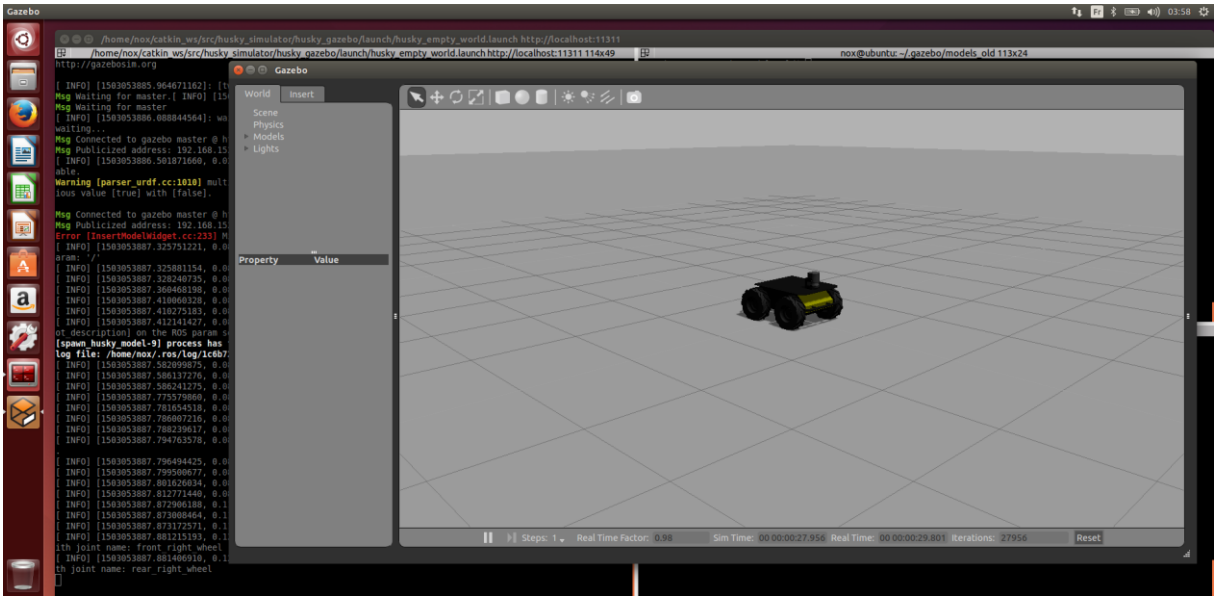
6. Correction bug Gazebo Models

```
# update models gazebo url /usr/share/gazebo(and 2-2)/setup.sh to http://models
.gazebosim.org/
cd ~/.gazebo
mkdir models
cd models
wget -r -R "index\.html*" http://models.gazebosim.org/
# wait a looooooooooonng time …
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
echo "export LC_NUMERIC=C" >> ~/.bashrc
```

II. Expérimentations

1. Premier lancement (lancement « world » vide avec l'agent « Husky »)

```
➢ roslaunch husky_gazebo husky_empty_world_launch
```



- *Video 1_intro.mp4 dans dossier « 4. Videos & Screenshots »*

2. Pilotage de l'agent « Husky » avec une manette « Playstation »

```
➢ roslaunch husky_control teleop.launch
➢ roslaunch husky_gazebo husky_empty_world_launch
```

- *Video 2_joypad.mp4 dans dossier « 4. Videos & Screenshots »*

3. Déplacement de l'agent « Husky » selon une direction & une destination

```
➢ roslaunch husky_gazebo husky_playpen.launch
➢ roslaunch husky_viz view_robot.launch
➢ roslaunch husky_navigation move_base_mapless_demo.launch
```

- *Video 3_rviz.mp4 dans dossier « 4. Videos & Screenshots »*

4. Lancement de l'agent « Husky » avec un circuit

```
➢ roslaunch formulapi_gazebo formulapi_track1.launch
```

5. Lancement de l'agent « Husky » sur le même circuit mais de façon autonome :

```
➢ roslaunch formulapi_gazebo linefollow_cam_formulapi_track1.launch
```

Reste à faire :

- Multi-Agents (20 véhicules mini)
- Implémentation des pilotes
- Doubler, changement de voie (par l'intérieur, champs positifs et répulsifs)
- Gestion des collisions
- Etude et mise en place de l'observateur
- Intégration du public, techniciens & dirigeants / sponsors
- Expérimentations avec Arduino / Raspberry physiquement

Comme indiqué, le facteur temps m'a cruellement manqué. Je compte malgré tout continuer le projet plus avant ou les détails seront disponible sur le dépôt GitHub indiqué précédemment.

Je suis inscrit sur le programme FormulaPi pour participer à une de leur compétition prochaine à l'aide d'expérimentations réelles avec des véhicules équipés avec un Raspberry et Arduino et d'une multitude de modules et capteurs. J'espère ainsi amener avec moi les résultats de cette compétition et de ces expérimentations lors de la prochaine compétition SUPINFO 2018.

Le sujet est passionnant !