Examen de Middleware/ROS

1h, documents autorisés

1 Initialisation de l'environnement

1.1 Si ROS n'est pas initialisé sur le compte

Si votre environnement n'a pas encore été utilisé pour ROS, exécuter le script `ros_user_setup.sh` à télécharger depuis mon site web. Refermez ensuite le terminal et ouvrez-en un nouveau.

1.2 Téléchargement du package

L'examen est téléchargeable sous forme de package ROS avec la commande suivante :

```
cd ~/ros/src
git clone https://github.com/oKermorgant/ecn_ros2015.git
```

2 Description du package

2.1 Le simulateur

Lancez `midwa.launch`:

```
roslaunch ecn_ros2015 midwa.launch
```

RViz est lancé avec trois robots mobiles plans.

- Un robot type BB-8 qui suit une trajectoire prédéfinie
- Deux robots type R2-D2 qui sont pour l'instant immobiles

Ces robots évoluent en (x, y, θ) et sont commandés avec une vitesse linéaire v_x et angulaire ω_z.

2.2 Nodes et topics

Le node `simulator` rend compte du déplacement des trois robots. Celui de BB-8 est imposé, les autres dépendent d'une consigne de vitesse.
Chaque robot publie sa position sur le topic `robot#/pose` (`#` étant un nombre entre 1 et 3) et souscrit à une consigne de vitesse sur le topic `robot#/cmd`.
Pour plus d'information, les nodes et topics sont visualisables avec `rqt_graph` et accessibles via les commandes `rosnode` et `rostopic`.

Les topics qui nous intéressent sont :

1. Les topics de position (`robot#/pose`), sur lesquels sont publiés des messages de type `geometry_msgs/Pose2D` dont la structure est :

```
float64 x
float64 y
float64 theta
```

2. Les topics de consigne (`robot#/cmd`), sur lesquels le simulateur attend des messages de type `geometry_msgs/Twist` de structure :

```
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

Le simulateur ne prend en compte que les champs `linear.x` (vitesse linéaire dans le repère du robot) et `angular.z` (vitesse angulaire).

3 Objectifs

Le travail demandé est d'écrire :

- Un node qui permet à un robot d'en suivre un autre à une distance donnée
- Un launchfile pour exécuter deux fois le node ci-dessus, afin que les trois robots se suivent

3.1 Écriture du node

Le node principal est à écrire en C++[1]. Créer le fichier directement dans le dossier `ecn_midwa`. Ce node doit :

1. Souscrire à deux topics de pose (type `geometry_msgs/Pose2D`)
   (a) Le premier est nommé `current` et correspond à la pose du robot piloté
   (b) Le deuxième est nommé `target` et correspond à la pose du robot visé
2. Publier sur un topic de commande en vitesse (type `geometry_msgs/Twist`), nommé `cmd`

[1] on modifiera le fichier `CMakeLists.txt` afin de le compiler

Les deux poses courante (x_c, y_c, θ_c) et désirée (x_t, y_t, θ_t) sont représentées en Fig. 1.

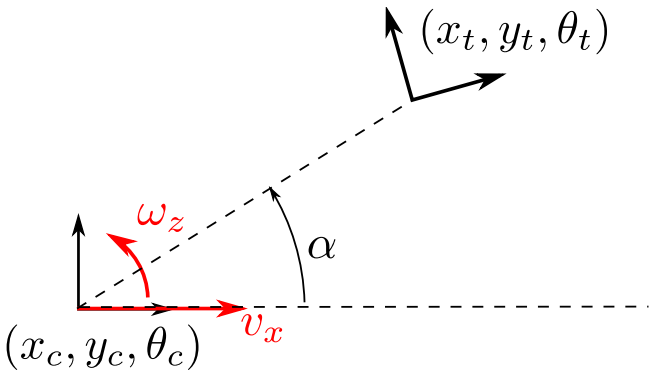

Figure 1: Poses courante et désirée. Vitesses linéaire v_x et angulaire ω_z.

La loi de commande (v_x, ω_z) permettant au robot contrôlé de s'approcher à une distance d du robot cible est calculée avec l'algorithme suivant :

1. Calcul de l'erreur cartésienne entre la cible et le robot courant :

$$\begin{cases} dx &= x_t - x_c \\ dy &= y_t - y_c \end{cases}$$

2. Calcul de l'erreur angulaire (à remettre dans $[-\pi, \pi]$ si besoin) :

$$\alpha = \arctan 2(dy, dx) - \theta_c$$

3. Calcul de l'erreur suivant x dans le repère courant :

$$\rho = dx \cos\theta_c + dy \sin\theta_c - d$$

où d est la distance à laquelle on veut se placer par rapport à la cible.

4. Loi de commande proportionnelle :

$$\begin{cases} v_x &= k_\rho \rho \\ \omega_z &= k_\alpha \alpha \end{cases}$$

où k_ρ et k_α sont des gains.

On utilisera les valeurs suivantes pour cette application :

$$\begin{cases} d &= 1 \\ k_\rho &= 2.7 \\ k_\alpha &= 2.9 \end{cases}$$

La consigne (v_x, ω_z) est à écrire ensuite dans un message de type `geometry_msgs/Twist` et à publier sur le topic `cmd`.

3.2 Écriture du launchfile

Le node ci-dessus utilise des topics génériques, il convient donc de le lancer via un launchfile en utilisant la fonction `remap` pour l'application qui nous intéresse.

Écrire un launchfile permettant au robot 2 de suivre le robot 1, et au robot 3 de suivre le robot 2.

Pour simplifier le développement du node on pourra dans un premier temps écrire en dur les topics permettant au robot 2 de suivre le robot 1.