Exercise n°3 – actionlib

Emiliano Gagliardi
Emanuele Ghelfi

July 27, 2017

1 The components

The components of this simple package are the client and the server, that make respectively use of the actionlib::SimpleActionClient and actionlib::SimpleActionServer classes to implement the ROS actionlib client server paradigm.

1.1 The client

The client is a simple ROS node, that when launched subsribes to the topic /draw, instantiates a simpleActionClient, and wait for the server. the client is implemented with a class, that defines:

- The callback for the topic /draw, that receives the description of the polygon and sends the goal to the server.
- The callback for the activation of the goal, that does nothing but printing that the goal have been activated.
- The callback for feedbacks from the server, that prints the integer received in the feedback message, containing the number of sides that have been drawn.
- The callback for the end of the action, that prints the status of the goal and the result received.

1.2 The server

The server is a ROS node that when launched instantiates:

- a publisher for the turtlesim cmd_vel topic
- a subscriber for the pose of the turtle
- a simple action server

The server defines the callbacks:

- For the turtle pose, to keep internally the position of the turtle.
- The execute callbacks, the core, that draws the polygon received in the goal message.
- The preempt callback, that does nothing but printing that the goal have been preempted and setting the status of the goal. When a goal is preempted is actually discarded, and there is no queue of preempted goals in the server.

2 The execute callback

The execute callback draws the polygon received:

- check that the goal have not been preempted, otherwise returns
- draw a side: the side are drawn in open loop, computing internally the space that the turtle has traveled. At each command iteration there is a chek that the turtle has no encoutered an obstacle (knowing the space that the turtle should have traveled, the position before starting the current side, and the current position). If a collision is detected the goal is aborted with result "false".
- send the feedback with the number of drawn sides
- rotate the turtle (open loop as for the sides)
- loop the previouses until finished
- rotate the turtle, until it is in the initial orientation
- send success and terminate

3 The package

turtlesim_actionlib

- action
  - DrawPolygon.action
- lauch
  - turtlesim_actionlib.launch
- package.xml
- src
  - client.h

- server.h
- client.cpp
- server.cpp

3.1 Run the package

```
roslaunch turtlesim_actionlib turtlesim_actionlib.launch
```