USI-THYMIO-SIMULATION

Università della Svizzera Italiana
Robotics 2020 | **Assignment 2**

Team Ganymede:

- **Marco Ferri**
- **Nadia Younis**

Demos

Demonstration videos of the Assignment can be found at the following links. Each Task has been done. Any comment about them is available in the next sections.

- Compulsory Tasks
- Bonus Tasks

Into Deep of Tasks

All the following tasks have to be run into an Ubuntu machine with a working ROS + Gazebo environment. The ROS package has to be built with `catkin` in order to be executed.

Compulsory Tasks

Compulsory Tasks run with the following commands:

```
roslaunch usi_myt thymio_gazebo_bringup.launch world:=wall
roslaunch usi_myt compulsory.launch
```

Robot's name is set automatically to the default value `thymio10`. Even though a second thymio (`thymio11`) will spawn due to the first command, the implemented controller will only move the first robot (the one at `0,0`), ignoring the other.

In case the simulation has to be run multiple times, then also the command `rosservice call /gazebo/reset_simulation` is needed for resetting the Gazebo simulation environment, returning the Thymio to the original position in order to be faced directly against the wall.

Additional information about what is going on during the execution can be found in the terminal shown in the demo video or in the sections below regarding Tasks 2 and 3.

Additional Info: Task 1

Task 1 has also been done and it is shown in this video.
If you want to execute it, then these two commands are required:

```
roslaunch usi_myt thymio_gazebo_bringup.launch world:=empty
roslaunch usi_myt task1.launch
```

The 8 trajectory is formed by two separates circles drawn in different directions, in which each one starts when the other circle is finished. To do so, parameters such as radius, linear and angular velocities are used to compute the time required for completing a circle. This is used to know when it is the proper moment to change the angular speed for inverting direction.

Several radius and linear velocities have been tried, but there is an intrinsic noise inside the Thymio movements which leads to unprecise circles on long time executions, regardless of the chosen velocities. As a consequence, at each direction change, the robot also loses a bit of precision with respect to the center of the 8 trajectory.

Additional Info: Task 2

Task 2 is shown in the video above. Center, right-most and left-most sensors are used to detect an obstacle 8cm in front of the robot while moving forward. Then, only the side sensors (right-most and left-most) play the role of trying to face the obstacle by minimizing the difference of ranges they are measuring (accordingly to a threshold), slowly turning the robot to the direction in which the obstacle is nearer.

We had some problems at the beginning since the wall was not detected in time for the robot to stop itself before crashing into it; this was also a problem because the sensors get mad when they are directly touching the obstacle, making the situation impossible to manage properly. After many debug sessions, we discovered that the problem was caused by an inappropriate linear velocity, that was too high for the simulation to be handled. In fact, Thymio's specification says that the maximum velocity is 14 cm/s, so values much higher than this lead to unexpected behaviors during obstacle detection, which results to be late for avoiding them.

For speeding up the testing, the original `wall` world provided has been modified to put the wall nearer to the spawn position and to be placed diagonally with respect to the x-axis.

Additional Info: Task 3

Task 3 is also part of the video, as it is complementary to Task 2.

After the wall has been faced using sensors, then the robot turns itself by 180 degrees and moves by 2 meters. The last goal is achieved by registering initial position through the usage of `odometry` and then stopping the forward-moving robot when the new pose to reach is approached (properly calculated to be 2 meters away from the wall).

No particular problems have been faced here, but once again the robot's movements tend to be unprecise, both when turning around and when moving to the goal pose. This can result in a few centimeters of error in the final position, also due to a noise into the Thymio's odometry implementation, that is done by recording the wheels actuation while moving.

Bonus Tasks

For executing Bonus Tasks, the following commands have to be run in the terminal:

```
roslaunch usi_myt thymio_gazebo_bringup.launch world:=arena
roslaunch usi_myt bonus.launch
```

Once again, robots' names are set to the default values, so you do not need to specify them manually.

The demo video clearly shows how the robots are able to detect and avoid obstacles properly while randomly exploring the available space.

Additional Info: Bonus 1

The random movement has been held by fixing the linear speed to the maximum value, according to Thymio's specification and what has been said in the previous sections, and choosing a random angular velocity (between -2 and 2) each time the robot has to move.

Collision detection is implemented as in the Compulsory Tasks, but obstacle avoidance has been slightly modified. If an obstacle is exactly in front of the Thymio, it turns completely by 180 degrees and then keeps moving; this is done to prevent the robot from getting stuck in a corner. Instead, if the obstacle is on the side (mainly detected by one left/right sensor only), then the robot just turns a little bit in the proper direction to avoid it. Furthermore, when a Thymio encounters another one, it handles it as a normal obstacle; both the two robots will stop, turn around and restart moving in such a way that would not make them crash against each other.

No particular problems have been encountered during the development of this Task. However, it must be pointed out that some obstacles can be tricky to avoid if they are sharp and the sensors cannot detect them because the obstacle spike may fill perfectly the space between two adjacent sensors. Moreover, some obstacle such as a sphere-shaped one is impossible to be avoided by the Thymio, since the proximity sensors are positioned too low to properly detect them; in fact, the Thymio crashes against a sphere with the top-mounted camera before detecting it with sensors.

Additional Info: Bonus 2

This last Bonus Task was just a matter of modifying the `launchfiles` for spawning and controlling an additional Thymio.

The two robots immediately behaved correctly without any modification of the controller made for Task Bonus 1, we just managed to spawn or move the Thymios near to each other before starting the simulation.

Please note that **the `arena` world used in the video has been slightly modified**: obstacles have been added or moved in order to create a convenient environment for demonstrating the robots' behavior more easily. The world found in the source code (which you will find when executing it) is instead the original that has been provided to us.