COMP47130 Final Project Description

Code Submission Deadline: 25/04/14 00:00

Project Competition: 25/04/14 11:00

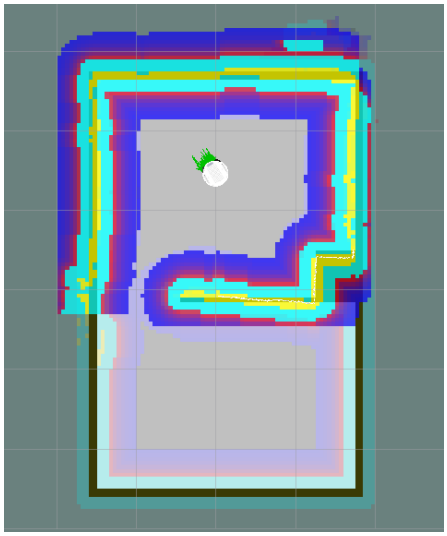
In this project we would like to build upon what has previously been done in the Potential Field assignment to perform a “Three Laws” service. The project will involve the following elements:

- Localization
    - Functionality to accurately localize the Turtlebot within the arena
- Navigation
    - Functionality to allow the Turtlebot move from one location to another reliably
- Image Processing
    - Functionality to allow the Turtlebot to recognize and distinguish between QR codes
- Text-to-Speech
    - Functionality to allow the Turtlebot to announce events

Localization

The task for implementing the localization is not an easy one, and will not be feasible before the deadline. In order to aid in this, we are permitting the use of the ROS amcl to provide you with an accurate method of localizing the Turtlebot. This will provide you with (x, y, theta) values (Pose).

A map of the arena will be generated by the TA and provided to you along with the yaml configuration. You will also be able to import this map into your simulators for testing. The TA will show you how to import this map during a coming practical.



The Turtlebot stack comes with a package that will include many of the launch files and packages that you will need, this packages is called turtlebot_navigation. Though no changes to the Turtlebot will need to be made by the student.

List of Necessary ROS Packages for Implementation:

- set_initial_pose

List of ROS Topics for Utilization:

- /amcl_pose
- /initialpose

Navigation

A navigation algorithm will be designed and implemented by you to allow the Turtlebot to move from one location to another. The Potential Field code previously implemented in your assignment will be integrated into this functionality to guarantee no collisions with the arena walls and allow for rudimentary path finding. You are **not** permitted to use the ROS Navigation stack for this functionality. You may however use material from the lectures as well as methods described by other sources to implement an algorithm of your choosing.
Note: Your code will be inspected for any inclusion of third-party code.

List of Necessary ROS Packages for Implementation:

- generate_random_waypoint
- generate_path
- set_waypoint
- go_to_waypoint
- path_execution
- pf_avoidance
- navigate (path_execution + pf_avoidance)
- save_waypoint

List of ROS Topics for Utilization:

- /map
- /scan
- /cmd_vel_mux/input/teleop

Image Processing

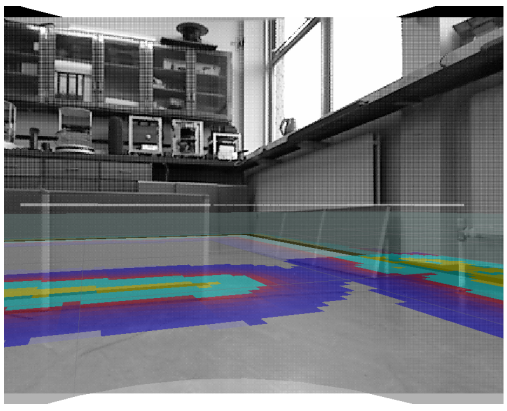
Three QR codes will be randomly placed the arena and you will need to be able to distinguish between each. The three QR codes the Turtlebot is looking for are uploaded in the moodle. There is no standard ROS package for QR code scanning, so complete freedom is given to the student in implementing an accurate scanner. The Turtlebot will be publishing on a /camera/rgb/image_raw topic. The included image is from the rviz view augmented with obstacle heat-maps and floor plan, image_raw will be devoid of such details and provide a image quality sufficient for the student's needs.

List of Necessary ROS Packages for Implementation:

- qr_code_search
- go_to_qr_code
- save_qr_code_waypoint
- announce_law

List of ROS Topics for Utilization:

- /camera/rgb/image_raw



Text-to-Speech

This is probably the most straightforward part of the project. It is **not** necessary to have a node running on the Turtlebot in order to do this, and the packages to allow for this are already installed (espeak). A single node that will subscribe to a \text-to-speech topic on your workstation is what we are looking for here. You can then simply push the command via ssh to the Turtlebot for execution there, thus making the Turtlebot speak instead of your workstation.

An example of using espeak: *espeak "hello world"*
You can remotely set the volume with: *pactl set-sink-volume 0 20%*

List of Necessary ROS Package for Implementation:

- tts_turtlebot

The Challenge

A number of QR codes will be placed at random around the arena provided. The Turtlebot will have to search the arena for each of the QR codes. Once it has found a QR code, it must recent a corresponding law from [Isaac Asimov's Three Laws of Robotics].

The Three Laws:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Each of the above laws correspond to each of the QR codes on the right. The order of discovery is totally depending on the implementation of the student and the placement, but on discovery, the Turtlebot should drive to 0.5 meters of the QR code and recite the corresponding law. On discovering each law, the Turtlebot must return to each QR code and recite the corresponding laws in the correct order.

The Turtlebot must complete it's task within 10 minutes. Three opportunities to demonstrate your algorithm will be allowed. No modification of code will be allowed during the demonstration. It has often been the case in previous years that students make modifications to an already partially working solution only to make it not work at all. Tested 10 times? Test it 3 more times.

Setup

In order to run the Turtlebot from your workstation, you'll need to set a few environmental variables. Add the following to your .bashrc file:

export ROS_MASTER_URI=http://192.168.1.110:11311
export ROS_HOSTNAME=<WORKSTATION-IP>

Once you have that done, source your .bashrc:

$ source .bashrc

A single roslaunch file has to be run in order to make available all of the Turtlebot features, you can do that with:

$ ssh turtlebot@192.168.1.110
$ roslaunch turtlebot_bringup all.launch map_file:=/home/turtlebot/arena_map.yaml

If there are errors after running this script, run the following script in the home directory:

$ sudo sh serial.sh

Then check the base is powered on with a green light lit.

Once the all.launch file is running, on your workstation, run the following to check connectivity:

$ roslaunch turtlebot_teleop keyboard_teleop.launch

If you are able to move the Turtlebot around the arena, everything else should be OK.

Now, run rviz on your workstation, it is configured to show you all the Turtlebot details:

$ roslaunch turtlebot_rviz_launchers view_navigation.launch