CS-7630 Autonomous Robotics

Final Project: Finding the rogue access points

VREP version

Support: cedric.pradalier@georgiatech-metz.fr
Office hours: online.

This homework is the final project for this class. This is assumes that you have done most of the other homework and will require integration of many of the components you've developed already. Normally, this would be done on the turtlebots, but we are trying a large CoppeliaSim scene similar to the second floor of GTL. The scene is located in gtl_wifi.ttt. The scene implements a large environment with static and moving objects and a number of ALVAR markers (repeated) on the wall.

Using the task management system will probably be a significant help in developing your application, even if it cost a little bit more to setup. The ros package gtlwifi in vrep4_ros_ws contains two launch files: Minimal.launch and 3dsensor.launch which emulates the eponymous file from the turtlebot.

The application we are considering is the detection of rogue access-points on the 2nd floor of GTL with the turtlebots. In an ideal scenario, we imagine that we switch the robot on on its charging station (the yellow area of the scene) and that it starts automatically to explore the environment, constructing a map, and recording the access-points visible at every location. If the robot battery goes low or if there is nothing else to explore, the robot should automatically decide to come home and charge, possibly going for another inspection when its charge is up. When the robot comes back home, it should publish a map of the environment with the intensity of the access-point signal at every location. This map could be published as an image, but that's up to you.

The objectives above imply that you need to setup a SLAM system (using the ALVAR markers), a occupancy grid/costmap mapping, a planner, path follower, obstacle avoidance and an exploration system that will explore both the occupied space and the wifi knowledge.

As a second level demonstration (but maybe an easier-to-grasp first step), the robot could run its mapping route on a manually planned path (similar to the floor_nav demonstration). An suitable floor_nav base is also available in the vrep4_ros_ws.

For each of the problem that must be solved, feel free to integrate either existing ROS package or your own implementation.

To implement the access-point sensing, you can use the ros topic /vrep/signal. To check the bubblerob battery level, check /vrep/voltage.

Deliverable:

- Demonstration of the performance of the system
- Video of the performance of the system
- Exported map in a visualisable form (e.g. image).
- All sources and launch files.