236609 - MULTI ROBOT SYSTEMS- Spring 2021

Assignment 3

Sarah Keren and Mohammad Masarwy

The Taub Faculty of Computer Science
Technion - Israel Institute of Technology

Logistics

Logistics

- This is the final assignments of the course.
- This assignment is separated into three dates:
  - submission of assignment presentation (without the code) in which you will describe the approaches you intend to implement: 26.01.2022
  - Submission of code + presentation: 15.03.2022
  - Demo day - in which we will meet for a final presentation of the projects and the competition (with invited reviewers): 23.03.2022
- You can work alone, in pairs, or in groups of three (in which case you will have an extra requirement).

Deliverables

You are required to submit 2 documents:

- A **single** python module containing your code named **assignment3.py** (make sure it is python2 compatible).
- A **presentation** describing your selected approaches to the problems you were required to solve. The presentation will be in **Latex** in a format we provided in *https://www.overleaf.com/read/bphsbfzyybjf* (you can add up to 10 slides).

The files will be submitted in a single zipped folder. The name of the folder is **Assignment3-ID1-ID2**.

Setup

As in exercise 2:

- You will be working with the turtlebots, but this time you will be working with 2 robots.
- We will run your solutions both in simulation and on the real robots, so we recommend that you gain some experiences working with the real robots.
- You will have 2 (or 3) tasks to fulfill within a time bound.

Each task is described below, and will be evaluated separately.

Pay attention

- **Do not !** include ANY package that is not already installed by default. If you want a package that is not installed - you need my approval.
- You need to make sure your code is python2 compatible.

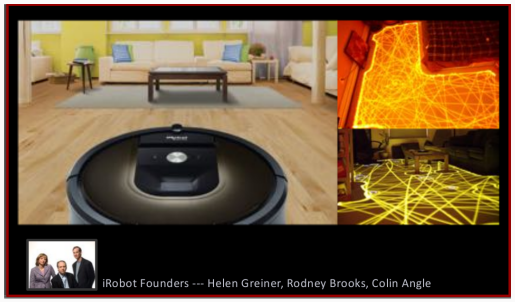
Setup

- You can choose between two options.
  - Option #1 - a single robot in an adversarial cleaning task and a collaborative inspection task.
  - Option #2 - a single robot in an adversarial cleaning task and an collaborative assistance task.

The second option allows you greater flexibility, but will require more work (and requires my approval) !

Task 1: cleaning a fully-mapped environment with an adversary agent

- The setting is the same as the cleaning task from Assignment 2, with the following differences:
  - There will be another robot in the environment that will be competing for the pieces of dirt.
  - You will receive a list of the locations of the pieces of dirt (via a yaml file).
  - The dirt_publisher publishes the updated list of dirt pieces collected by each agent, and the remaining pieces of dirt and their locations.
- You will have a fixed amount of time (2 minutes) to collect as many pieces of dirt from the room as you can.



Task 1: dirt_publisher

```
stellas@ros-melodic-u: ~/my_ws 105x28
stellas@ros-melodic-u:~/my_ws$ rosrun MRS_236609 dirt_publisher_ex3.py

current status: agent_0 collected 0 and agent_1 collected 0 remaining dirt:
[(2.4, 0.3), (1.3, -3.2), (-1.5, -2.9), (-1.0, -2.3)]

current status: agent_0 collected 0 and agent_1 collected 0 remaining dirt:
[[2.4, 0.3], [1.3, -3.2], [-1.5, -2.9], [-1.0, -2.3]]

current status: agent_0 collected 0 and agent_1 collected 0 remaining dirt:
[[2.4, 0.3], [1.3, -3.2], [-1.5, -2.9], [-1.0, -2.3]]
^Cstellas@ros-melodic-u:~/my_ws$
```

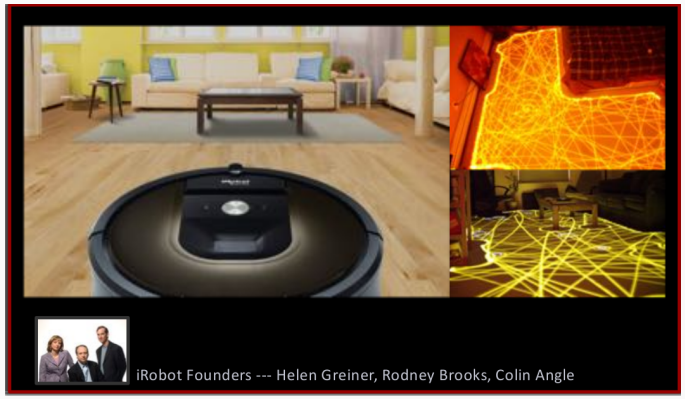
```python
class DirtPublisher:

    def __init__(self, num_of_agents, radius, dirt_pieces=None):
        if dirt_pieces is None:
            dirt_pieces = []
        self.dirt_pub = rospy.Publisher('dirt', String, queue_size=10, latch=True)
        self.dirt_pieces = dirt_pieces
        self.odom_subsribers = []
        self.num_of_agents = num_of_agents
        # initialize the array with zeros (numpy zeros)
        self.collected_per_agent = np.zeros(num_of_agents)

        for i in range(0, self.num_of_agents):
            self.odom_subsribers.append(rospy.Subscriber('/tb3_%d/odom' % i, Odometry, self.update_dirt_status))

        self.radius = radius

    def run(self):
        # Main while loop
        while not rospy.is_shutdown():

            print('\ncurrent status: agent_0 collected %d and agent_1 collected %d remaining dirt:' % (
                self.collected_per_agent[0], self.collected_per_agent[1]))
            print(self.dirt_pieces)
            self.publish_objects()
            if len(self.dirt_pieces) == 0:
                print('all dirt collected. exiting')
                exit(0)
            self.publish_objects()
            rospy.sleep(3)

    def publish_objects(self):

        self.dirt_pub.publish(''.join([str(x) for x in self.dirt_pieces]))

    def update_dirt_status(self, msg):
```

DirtPublisher
def update_dirt_status

Task 1: cleaning a fully-mapped environment with an adversary agent

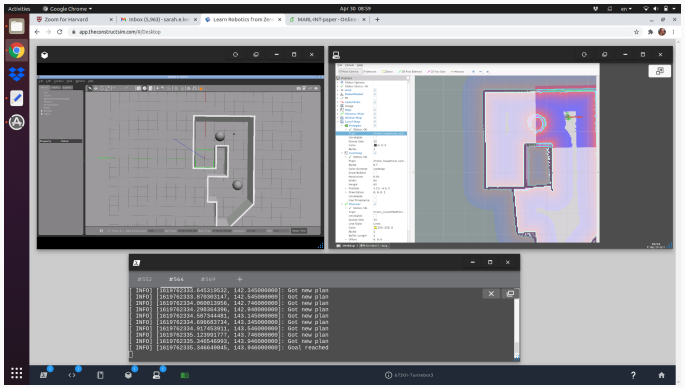
Note that:

- The two robots will be running under the same ROS master.
- This means you will have access to the publications of the other agent, but you will not be allowed to publish messages on its channel.
- The winner is the robot that collects the highest number of dirt pieces.
- The maximal allowed speed is limited to 0.22.



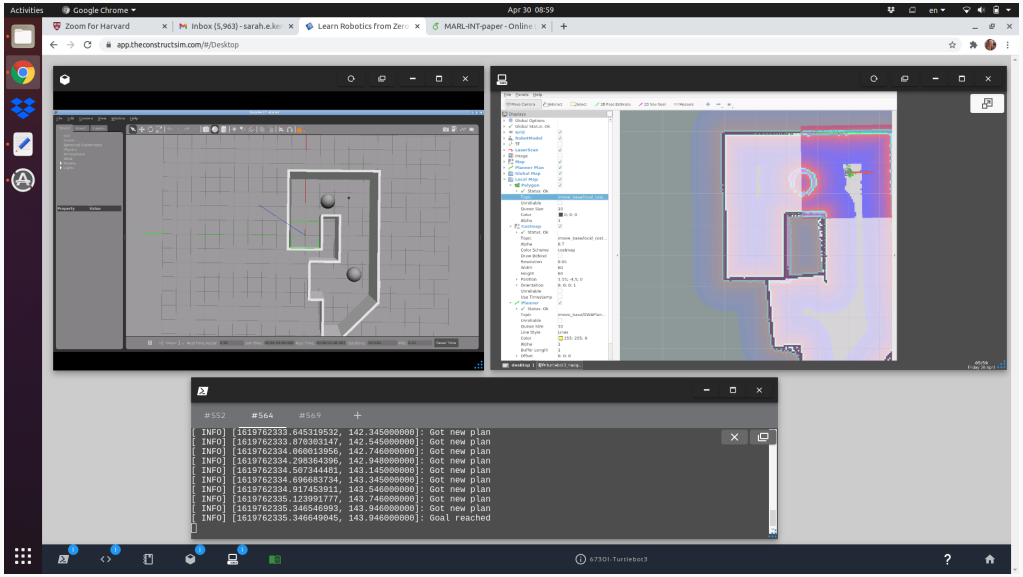
Task 2: inspection task

- As with exercise 1, your task will be to detect the number of unmapped objects (spheres) that are distributed in the environment.
- You will have an unknown amount of time to detect the number of spheres in the room (the program will end unexpectedly at some time bound we will set).
- The novelty here is that you will have two robots for the task.
- Note! We will provide as input the maximal speed that is allowed for each agent.



Task 2: inspection task - continued

- Since you don't know how much time you have for the task, you need to publish every 10 seconds a message with the following format:
  - topic: inspection_report
  - structure: a string with the format: "X spheres detected at time TIMESTAMP"
  - We will check your answer at 3 time intervals (the last one will be of at least 4 minutes).



Option #2

Setup

- You will complete Task 1 as described above with the Turtlebots.
- Instead of the collaborative inspection task, you will complete a ’request for help’ + ’offer help’ task.

An Autonomous Agent Requesting Help

- **Main agenda**: In assignment #2 you equipped your robot with the ability to understand its own limitations with regards to its ability to accomplish a task and ask for help.
- In this assignment, you will integrate the ‘helper’ robot that will perform the assistance action.
- The helper will communicate with the ‘beneficiary’, and report success or failure.
- You can choose which robots you want to work with.
- You will need to account for some form of uncertainty both robots needs to ’deal with’: localization uncertainty, navigation cost estimation, uncertainty regarding grip poses, communication noise, etc.
- You will need to **randomly** generate various possible configurations, in which the robots will intetract.

An Autonomous Agent Requesting Help

- Note: This is a task that requires my approval.
- You will only be allowed to choose this option if you have a rich and interesting representation of uncertainty and an innovative way to quantify it.

Note!

This is probably not the final description of the assignment.

As you start working on it, you will have questions, and we will updated the description accordingly (make sure to monitor the forum).

Good luck and don’t forget to enjoy the process.