MTE 544 LAB 3 – Pre-Lab and In-Lab Manual

Initial Setup Instructions:

- Please do not dock/undock the robot by hand
- Please do not pick-up the robot by hand

Lab 3 – Pre-Lab Steps and Development:

Problem Description

Write an A* algorithm for the turtlebot to navigate a given map from its initial position to the end goal position while avoiding obstacles (you can use the provided skeleton code or write your own, see Notes and recommendations below).

Inputs to the A* algorithm are the map (uploaded on LEARN); and the starting and desired end positions. For the pre-lab component use this as goal position: [0, -0.08] (you are also encouraged to try different start and goal positions within the map to test the performance your algorithm), but in lab you will be given different coordinates to compute.

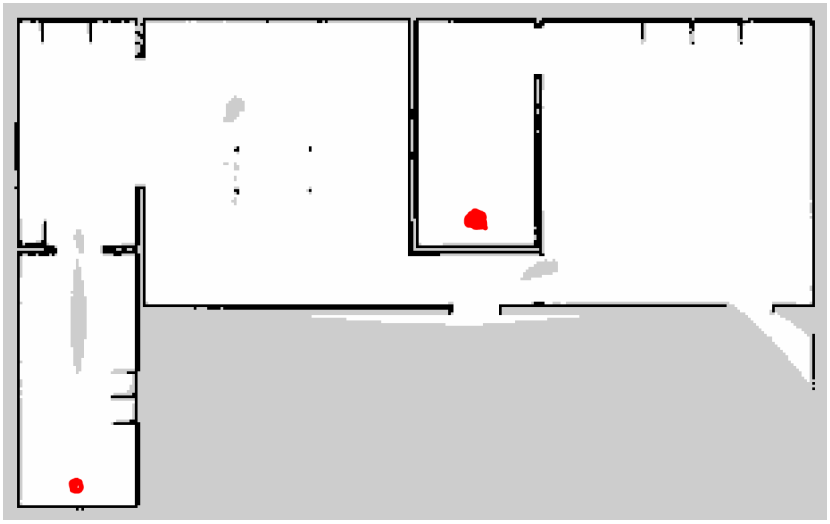
You are expected to implement a motion planner action server-client that guides the turtlebot through the map to reach end goal and pose. Your action server-client should return success when the robot reached the goal and says failed if the final goal is out of reach.

- Initialization:

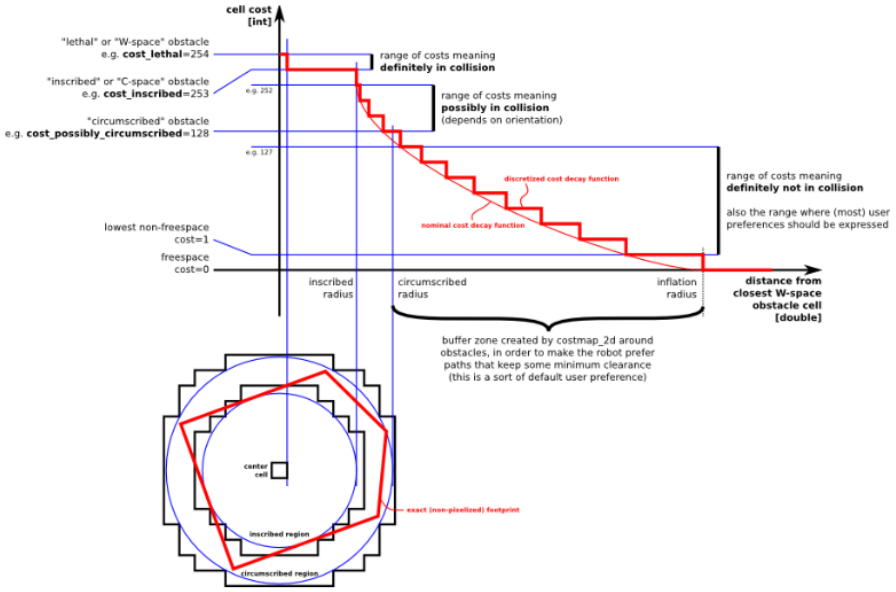
  Use the following yaml map file and map image itself for setting up and starting your virtual A* algorithm. The files are placed under a zip file called “Office Map” on Learn.

  Choose an arbitrary point in the rooms. Initialise that point as the start for the robot to travel from to the goal point [0, -0.08] (you are also encouraged to try different start and goal positions within the map to test the performance your algorithm), and use using any form of position controller (bang-bang or proportional).

  The cost map of your system can be generated by the methods provided below but for a simple map such as this, generating your own cost map would give you an idea of how to avoid collision, and what those algorithms are doing.



- Loop:
    1. This section should hold your A* method, that determines the coordinates of each of the intermediate goal poses/nodes within the graph. Any “occupied” or “collision” space as determined by the cost map should not be considered.
        1. For your occupation map, you could use something similar to this asymptotic formula that is generated from the 2d-cost-map (http://wiki.ros.org/costmap_2d), and scaling your distance from it



    2. P-Controller method, that:
        1. Subscribes to the */tf* topic to determine the turtlebot current pose.
        2. Calculates the error between the current pose and the intermediate goal pose.
        3. Generates the control signals, i.e., required velocities to reach the intermediate goal pose.
        4. Publishes the commanded velocities to drive the robot from its initial position to the intermediate goal pose over the */cmd_vel* topic.
    3. The P-controller should be performed after the A* algorithm calculates the path

Notes and recommendations:

- The given maze in this pre-lab scenario is not the same as the one you will be approaching in the lab; a hardcoded solution will not work.
- You are free to use any ROS library, but not a direct A* algorithm.
- **It is recommended**: use the provided skeleton code to create a simple algorithm in python or write your own version of A* (keep it simple at this stage) that creates a point that travels between the points using the A* and the image. Have it draw on top of the image before you start creating clients and packages.
    - This will improve bug identification and reinforce cost and motion planning from a smaller data set and without having to wrangle with ros2 at the same time.
- You can re-use the P-Controller **core** from Pre-Lab 2.

Testing

To test your A* motion planner:

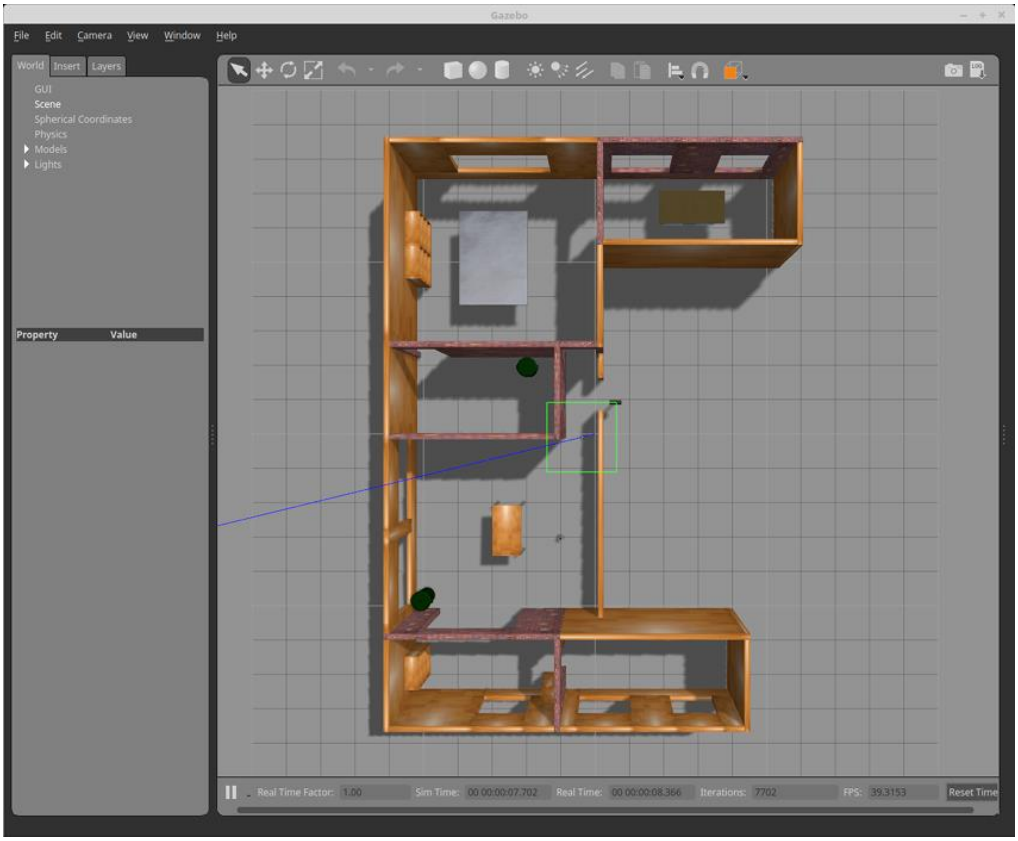
1. Open a new terminal and make sure you have turtlebot3 packages (you should have installed this from Prelab1):

```
$ sudo apt update && sudo apt install ros-galactic-turtlebot3*
```

2. Launch turtlebot3 simulation in the house/office world (note that depending on your setting, you may not be able to see the walls properly rendered):

```
$ export TURTLEBOT3_MODEL=burger
```

```
$ ros2 launch turtlebot3_gazebo turtlebot3_house.launch.py
```



3. In another terminal, open rviz2 and make sure "Map", and "TF" or “RobotModel” are added under "Displays":

```
$ export TURTLEBOT3_MODEL=burger
```

```
$ ros2 launch turtlebot3_navigation2 navigation2.launch.py use_sim_time:=True map:=<path/to/map>/map.yaml
```

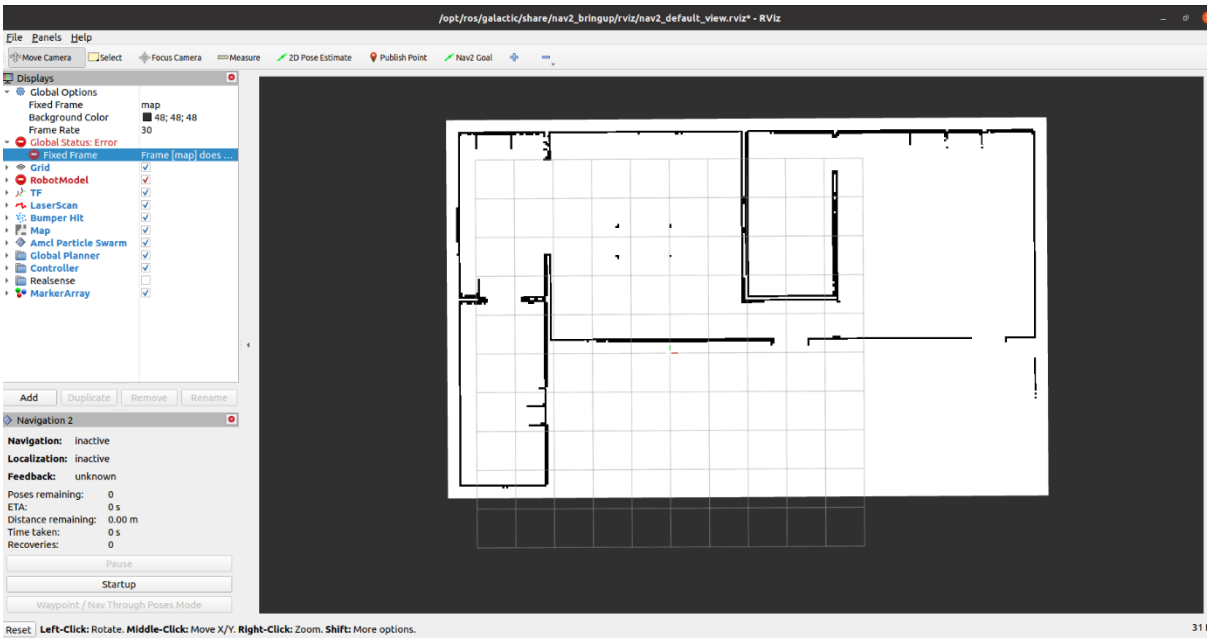
For more information on the documentation for the nav2 toolbox which is generating the occupancy grid, see here: https://navigation.ros.org/tutorials/docs/navigation2_on_real_turtlebot3.html

In another terminal, activate the map_server to be published for rviz2:

```
$ ros2 lifecycle set /map_server configure
```

```
$ ros2 lifecycle set /map_server activate
```

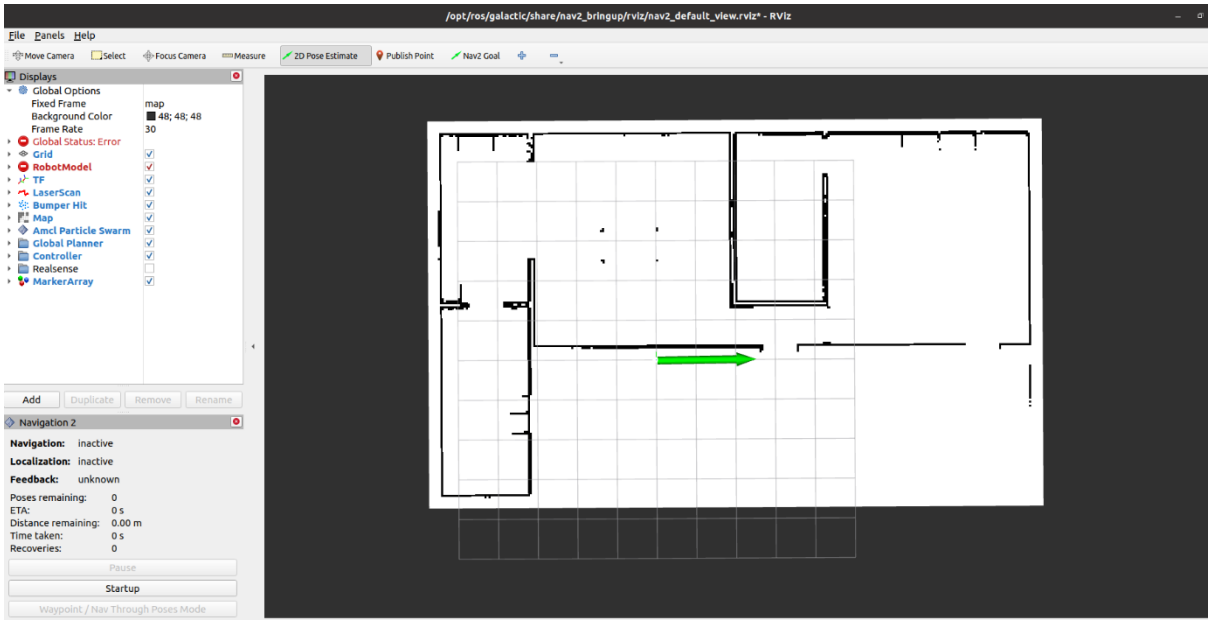
RViz2 should display the given map.



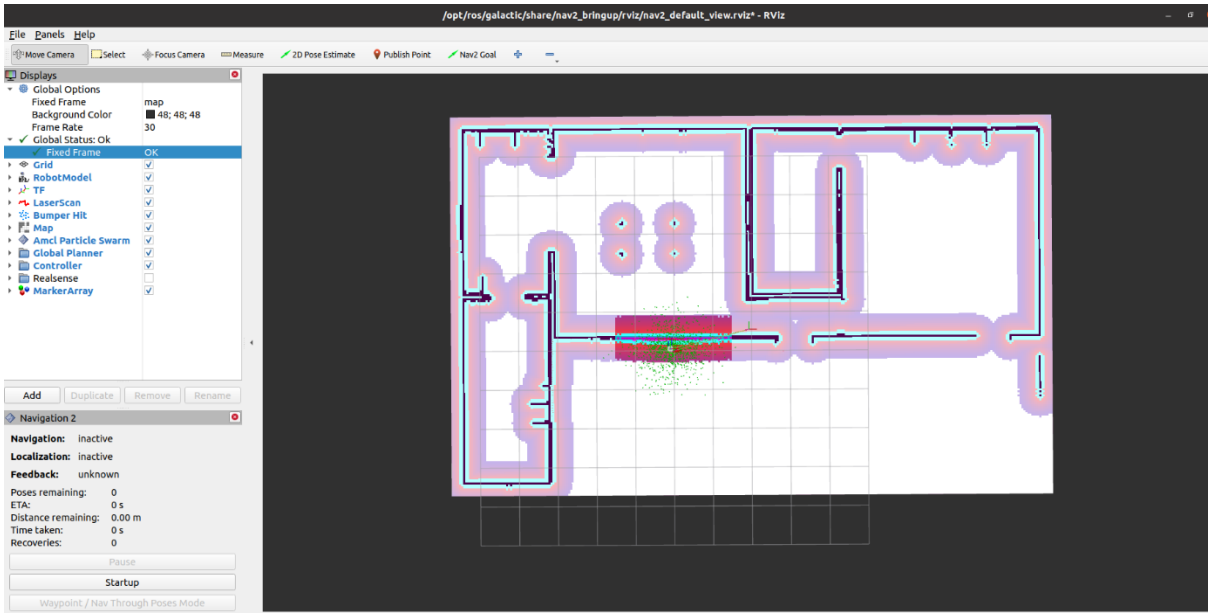
4. In RViz2:

Click the *2D Pose Estimate* button in the RViz2 menu.

Click on the map where the actual robot is located, which is at the shown frame. Drag your mouse toward the direction where the robot is facing, as shown in the image below. While dragging, a large green arrow will appear instantly to help you direct the robot.



After identifying the robot's direction, RViz will provide you the cost map automatically. It should look like something as the below screenshot:



5. In another terminal, control your robot using teleop_keyboard node to make sure everything is alright, and place your robot at the initialised start position:

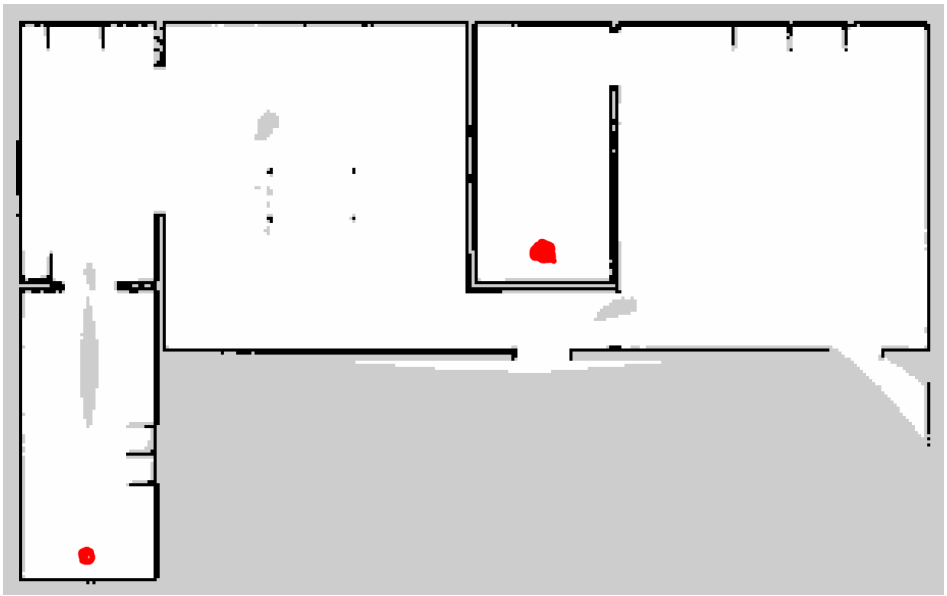
```
$ export TURTLEBOT3_MODEL=burger
```

```
$ ros2 run turtlebot3_teleop teleop_keyboard
```

6. Kill the teleop_keyboard node:

   Ctrl + c

7. Run your planner and make sure it reaches the given end goal by subscribing or taking the exported map occupancy information as shown below (note that your given goal might not match the one on this figure):



Lab 3 – Report:

Lab 3 report will be submitted on 2 parts:

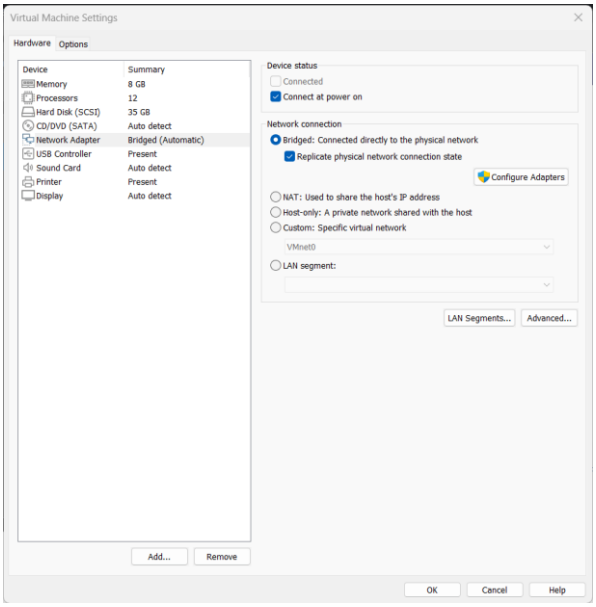
1. **Part 1 due 1st of December (70%):**
    - The report should be 3 pages (excluding Appendix) and should include Introduction, Methods, Results, Discussion, Conclusion, and Appendix section on the simulation you performed.
    - Submit your report as a single PDF on Crowdmark, and your code on Learn
2. **Part 2 due 6th of December (30%):**
    - It should be 1 page (excluding Appendix) and include Results and Discussions from the experiments you performed in the lab.
    - Submit your second part of the report as single PDF on Crowdmark, and your code on Learn (if it applies, i.e. if you had to change your code for the in-lab experiment).

Lab 3 – In Lab Steps:

Connecting to the Robot:

1. Turn on the robot by docking the robot
2. After the robot is on, ensure that the screen on the base has what looks like an IP address. (If the lcd was turned off, the IP address is simply 100+bot_number, for example 3, is 103)

3. Switch your network to the private Turtlebot network (**SSID: turtlebot, password: tobeltrut)**
    a. If you are using a virtual machine, you will change the network on the host OS
4. Ensure that everything is up to date, and you are running the correct version of Ubuntu (this should not be necessary if you did this in the previous prelabs/labs)
    a. $ sudo apt update
    b. $ sudo apt install git (just in case you haven't installed git)
    c. $ sudo apt install ros-galactic-rmw-fastrtps-cpp
    d. $ sudo apt install iputils-ping
    e. $ sudo apt install net-tools
5. Setup connection environment
    a. **If you are using a virtual machine**, please ensure that your network connection in the virtual machine settings is set to bridged and replicate physical network connection state has been selected



b. Create a directory called lab3 in your home directory to clone a repository in.
c. In the lab3 directory, run clone the turtlebot4 repo with the following command:
    i. $ git clone https://git.uwaterloo.ca/robohub/turtlebot4.git

WARNING: Next step will overwrite your .bashrc. If you want to keep your original .bashrc file, copy it to another directory before proceeding.

d. From the newly cloned repo, enter the **configs folder** and copy both **.bashrc** and **.fastdds.xml** to your **Home** directory from the configs folder with the following commands:
    i. $ sudo cp ~/lab3/turtlebot4/configs/.bashrc /home/$USER/
    ii. $ sudo cp ~/lab3/turtlebot4/configs/.fastdds.xml /home/$USER/
e. Make sure robot is alive on the network by pinging it (example below is for robot 3)
    i. $ ping 192.168.23.103
f. If you saw the packets keep on, if not tell one of the TAs.
g. After the repo is cloned and the files have been copied, move into the vpn subfolder within the terminal with the command:
    i. $ cd ~/lab3/turtlebot4/vpn
h. Now within the terminal in the vpn directory run:
    i. $ ./vpn.sh 192.168.23.XYZ
    ii. Where XYZ is the number for the robot (robot 3 is 103)
    iii. It will prompt for your password first if sudo was used before in the shell
    iv. Then it prompts for another password, which is "**turtlebot**"
i. Set the ROS_DOMAIN_ID=X (the value of your robot (robot 03 is 3)) with the following command:

    $ export ROS_DOMAIN_ID=X

6. Verify the connection
    a. Once you set the ROS_DOMAIN_ID you should be connected to the robot and can verify by viewing the topics

    $ ros2 topic list

7. Undock the robot:

```
$ ros2 action send_goal /undock irobot_create_msgs/action/Undock {}
```

8. Take the map information given to you for the maze in lab and generate offline a cost map for it, or have the navigation toolbox and initialise a cost-map from there.
9. Run your A* motion planner.