Worksheet 04: Path Planning and Task Planning

Universität Bremen
FB 3 – Mathematik und Informatik
Arbeitsgruppe Robotik
Prof. Dr. Dr. h.c. Frank Kirchner
Dr. rer. nat. Teena Hassan
M. Sc. Mihaela Popescu

Deadline: 14.12.2022
Robot Design Lab
03-IBGA-FI-RDL
WiSe 22-23

---

1 Path Planning

1.1 Quiz on Path Planning and Obstacle Avoidance [25%]

1. Name 3 path-planning algorithms and describe their pros and cons. [9%]
2. Breadth-first search is optimal, that is, if a path exists, then it always returns the shortest path. Why? [5%]
3. Which condition should be satisfied by the heuristic so that the A* algorithm finds the least-cost path from start to goal? Give two examples of heuristics that satisfy this condition. [3%]
4. Why do we need obstacle avoidance? [2%]
5. How are candidate trajectories (or velocities) generated in the dynamic window approach? [6%]

1.2 Having Fun with Path Planning [10%]

Try out the interactive web interface that visualizes the results of different path planning algorithms in a grid world: https://qiao.github.io/PathFinding.js/visual/. Using this interface, do the following:

1. Create obstacles in the map by clicking on the cells. Select a cell as the start cell (green) and another cell as the goal cell (red). Experiment with the grid design so that you create an interesting map with several obstacles. Submit a screenshot of your grid world. [2%]
Note: The cells in this grid can only be either free or occupied.
2. Now, use the menu to select Dijkstra's algorithm. De-select the option "Allow Diagonal". Click on "Start Search". Take a screenshot showing the map, the found path, all the visited nodes (light blue) and all the frontier nodes (light green). [2%]
Note: You should use the same grid map that you created in the first subtask.
3. Now select the A* algorithm with Manhattan distance as heuristic. De-select the option "Allow Diagonal". Click on "Restart Search". Take a screenshot showing the map, the found path, all the visited nodes and all the frontier nodes. [2%]
Note: You should use the same grid map that you created in the first subtask.
4. What differences do you observe in the above results produced by A* and Dijkstra's algorithms? Describe the reasons for these differences. [4%]

2 Launch Your Nodes! [20%]

In this task, you will create a ROS 2 launch file to configure and launch multiple nodes that belong to different packages.

2.1 Setup Your ROS 2 Workspace

Please follow the instructions given below, in order to set up your ROS 2 workspace to solve this task.

1. Create a new ROS2 package named `worksheet04` in your workspace:

```
cd ~/rdl_ws/src
ros2 pkg create --build-type ament_python worksheet04
```

2. Inside worksheet04 package, create a directory for the launch file:

```
cd ~/rdl_ws/src
mkdir -p worksheet04/launch
```

3. Download the file `example.launch.py` from Stud.IP and copy it to the above directory.

4. Rename the launch file to `teleop_wheeled_robot.launch.py`:

```
mv example.launch.py teleop_wheeled_robot.launch.py
```

5. Add the launch file to `setup.py` so that the launch file can be installed into the ROS 2 run-time environment by the `colcon build` process. For this, add the following two lines at the beginning of the `setup.py` file:

```
import os
from glob import glob
```

   Afterwards, add the following line to the list named `data_files` in `setup.py`:

```
(os.path.join('share', package_name), glob('launch/*.launch.py')),
```

2.2 Write the Launch File [20%]

The example launch file given to you contains an example showing how to rename a node, rename the topics of the node, and set values for the node's parameters.

In worksheet 2, you wrote a node to control the wheel velocities based on teleoperated commands. You started these nodes manually, one in each terminal. In this task, you will launch them in one go!

1. Extend the launch file to **start the following 3 nodes**. Each node should be started in a separate terminal. [3%]
   - rdl_move_base node.
   - rdl_teleop_keyboard node.
   - wheel_velocity_controller node.
2. In the launch file, change the name of wheel_velocity_controller node to a name of your choice. [1%]
3. In the launch file, change the name of the following topics [4%]:
   - Change `turtlebot3_burger/wheel_left_joint_controller/command` to `rdl_left_wheel_velocity`.
   - Change `turtlebot3_burger/wheel_right_joint_controller/command` to `rdl_right_wheel_velocity`.

   Note: The changes should be made for the node that publishes to these topics and for the node that subscribes from these topics.
4. Modify the node wheel_velocity_controller in `worksheet02` package so that it takes `track_width`, `wheel_diameter`, `max_abs_linear_vel` and `max_abs_angular_vel` as parameters [8%].

   Hint: For this task, do the following:
   - Use `self.declare_parameter()` to declare the parameters inside the `__init__()` method of your `wheel_velocity_controller` node. Here is an example of how parameters can be declared along with default values:

```
self.declare_parameters(
        namespace='',
        parameters=[
                ('parameter1', 0.0),
                ('parameter2', 1.0),
                ('parameter3', 2.5),
                ('parameter4', 3.0),
        ]
)
```

- At the places where you need these parameters, use the following to retrieve the value of the required parameter: `self.get_parameter('parameter_name').value`

For more information, see:

https://roboticsbackend.com/rclpy-params-tutorial-get-set-ros2-params-with-python/.

5. After successfully building your modified `wheel_velocity_controller`, set its parameters `wheel_base`, `wheel_diameter`, `max_abs_linear_vel` and `max_abs_angular_vel` to their correct values through the launch file. [4%]

Hint: You can use the following link as reference to complete the above subtasks:
https://roboticsbackend.com/ros2-launch-file-example/.

2.3 Build and Launch

After you completed the launch file as required in Section 2.2, you can follow the steps below to build the package and start the launch file:

1. Update the `package.xml` file in `worksheet04` by listing the required packages as execution-time dependencies. E.g. `<exec_depend>rdl_move_base</exec_depend>`

2. Build `worksheet04` and `worksheet02` packages using `colcon`:

```
cd ~/rdl_ws
colcon build --packages-select worksheet04 worksheet02
```

3. Source the local and global setup files.

```
source /opt/ros/humble/setup.bash
source install/setup.bash
```

4. Start the launch file.

```
ros2 launch worksheet04 teleop_wheeled_robot.launch.py
```

5. If you get an error saying that `xterm` is not recognized, then you can install `xterm` using `sudo apt install xterm`

6. Optional: Check if all the nodes have started properly. Teleoperate your robot with the `rdl_teleop_keyboard` and see if it is moving as expected.

Submit the following:

1. After finishing the above-mentioned sub-tasks, run the following commands and insert their outputs in your PDF file as a screenshot.

```
ros2 node list
ros2 topic list
ros2 param list
ros2 param dump <new-name-of-wheel-velocity-controller-given-inside-launch-file>
```

2. Embed relevant parts of your code in the PDF and explain briefly the changes you made.
3. Submit the contents of your ROS 2 package `worksheet04` which contains your completed `teleop_wheeled_robot.launch.py` launch file.
4. Submit the contents of your ROS 2 package `worksheet02` which contains your modified `wheel_velocity_controller` node.

3 Task Planning

3.1 Theory: Task Planning and Acting (25%)

Please answer the following questions:

1. Describe briefly any four assumptions of classical task planning. (4%)
2. What is the difference between an operator and an action? (1%)
3. Distinguish between planning domain and the statement of a planning problem (4%).
4. How would you determine if an action 'a' is applicable in a state 's'? (1%)
5. How would you determine if a state 's' satisfies the goal 'g'? (1%)
6. What do you understand by the term 'state transition'?(2%)
7. Given below is an operator `fly(plane, from, to)`. `fly-plane` is the operator symbol and `plane`, `from` and `to` are parameters of the operator. The planning domain contains an Airbus `A320` and a list of domestic airports in Germany. Construct the action to fly `A320` from Hamburg (`HAM`) to Frankfurt (`FRA`). (6%)

```
fly(plane, from, to)
       precond: can-fly(plane), is-airport(from), is-airport(to), at(plane, from)
       effects: !at(plane, from), at(plane, to)
```

8. Represent the above action as a behaviour tree by applying the postcondition-precondition-action rule. (4%)

   (Note: For simplicity, you can replace the negation symbol '!' with the prefix 'not-', e.g. 'not-at(locationA)', and use this as the label for the corresponding condition node. Remember that you should use grounded predicates and action names for labeling the leaf nodes in the tree.)
9. Describe briefly the control logic of the above behaviour tree. (2%)

3.2 Applying Task Planning to Deliver Medicines (20%)

In this video https://www.youtube.com/watch?v=AWtIybYU-20, you will see an application of autonomous drones to deliver medicines to remote places. After watching the video, solve the following subtasks.

3.2.1 Planning Domain (10%)

Here, you will define a simple planning domain for the above application scenario. For this, do the following:

1. Define **constant symbols** to represent a package, the drone, the base station of the drone, and the location of the hospital. (2%)
2. Define **predicates** representing where the drone is located and whether a package is attached to the drone or has been released from the drone. (2%)
3. Define **operators** for the drone to take-off, land, hover, fly and release package. If your operators need additional predicates, add them to the list of predicates that you defined in the previous subtask. (6%)

3.2.2 Planning Problem (10%)

Using the above planning domain, formulate a planning problem to deliver a package of medicines at the hospital. For this, answer the following:

1. Define an **initial state** in which the drone is located at the base station and the package is attached to the drone. (2%)
2. How would you represent the **goal** in which the drone is back at the base station after delivering the package? (2%)
3. Write down a **plan** to solve the above planning problem. (3%)
4. Write down the sequence of **state transitions** that would be obtained when applying this plan to solve the above planning problem (3%).

4 Feedback

Your feedback is very important to us. Please briefly answer the following questions:

1. How much time did you spend on doing this sheet per person? Anonymize your answer!
2. Was it too easy, easy, ok, hard, too hard?
3. Please tell us what you liked in this exercise sheet.
4. Did you face any difficulties? What should be improved?
5. Any other general remarks?

5 Submission Procedure

- Please use the LaTeX template provided in *StudIP/Wiki* to write your solutions. Upload the PDF file together with source code and other additional materials as a .zip file in StudIP.
- The naming style of your submission should follow the pattern **Gxx_0y_lastname1_lastname2_lastname3.zip**, where xx stands for the group number and y stands for the exercise sheet number.