Worksheet 02: Sensors and Robot Odometry

Deadline: 16.11.2022
Robot Design Lab
03-IBGA-FI-RDL
WiSe 22-23

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

1 Quiz: Sensing on Europa 15%

Congratulations! After a long evaluation process, the space agency approved a project proposal at your robotics institute. Sadly, this took so much time that the original author of the proposal already left the institute. Therefore, it is your task now to create a robot that can explore Jupiter's moon Europa[1]. Please answer the following questions:

1. Name 2 interoceptive sensors, 2 proprioceptive sensors and 2 exteroceptive sensors that you would choose for such a mission. (6%)
2. Specify for which tasks they will be used. (6%)
3. Explain why they are important for the mission. (3%)

To give you some inspiration, you can take a look at the DFKI EurEx-LUNa project at: https://robotik.dfki-bremen.de/en/research/projects/eurex-luna/.

2 Quiz: Wheel Encoders 20%

Please provide answers to the following questions:

1. What is the working principle of an optical rotary wheel encoder? What does it measure? (4%)
2. How would you classify an optical rotary wheel encoder? Is it an active or a passive sensor? Is it an interoceptive, a proprioceptive, or an exteroceptive sensor? Justify your answers. (3%)
3. Provide a formula to convert the angular resolution R_a of an optical wheel encoder into on-ground distance resolution R_d (i.e. how many meters has the robot moved on the ground per tick). (3%)
4. Compute the angular resolution R_a and the on-ground distance resolution R_d of an optical encoder disc with $N = 256$ ticks and a robot wheel radius of $r = 3.3$ cm. If the disc has turned $n = 2048$ ticks, how much on-ground distance d (in meters) did the robot wheel move? (5%)
5. What is quadrature shaft encoding? How can we determine the direction of rotation from it? (5%)

3 Quiz: Odometry 15%

1. What is odometry and why is it important for autonomous robot navigation? (3%)
2. Name three types of odometry models and state the main differences between them. (6%)
3. What is the non-holonomic robot constraint? (3%)
4. Name two scenarios that can cause odometry drift. What would be a possible solution to compensate for the errors caused by odometry drift? (3%)

[1] https://en.wikipedia.org/wiki/Europa_(moon)

4 Omnidirectional Robot 15%

You are observing an omnidirectional robot moving in a 2D world as shown in Figure 1. The robot is equipped with Swedish wheels[2], which allows it to move in any direction without any reorientation needed. You noticed that the robot moved from location (1, 0) to location (2, 1) in 10 seconds and during this time, it also turned around 45°.

Task: Calculate the linear $\left[\frac{m}{s}\right]$ and angular $\left[\frac{rad}{s}\right]$ velocities of the robot, assuming that the robot was moving at uniform speed throughout its journey (all distances are measured in meters).

Hint: We encourage you to write a short Python program to calculate these values.

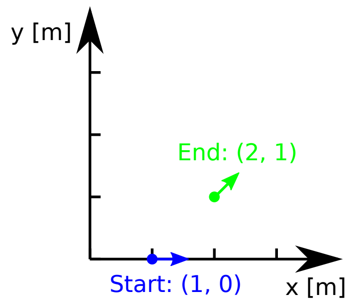

Abbildung 1: The robot starting position (1, 0) is depicted in blue, while the end position (2, 1) is depicted in green. The robot orientation at the two positions is represented by the arrow directions.

5 Differential Drive Model 35%

Our TurtleBot 3 robot is driven by a pair of wheels located on the same axis. Each wheel is driven by its own motor that has a controller attached to it, which accepts a rotational velocity as input. It is also called a differential drive configuration.

In the previous worksheet, you moved your TurtleBot 3 robot using the `teleop_keyboard` node and by pressing the associated teleoperation keys. In this exercise, you will create a ROS node that reads the robot's desired linear (forward / backward) and angular (rotational) velocities from the pressed keys, and converts them into rotational velocities for the left and right wheels of the robot. That is, you will apply the differential drive model.

This task has three parts as described below.

5.1 Use a Customized Teleoperation Node

For this exercise, we provide you an alternative `rdl_teleop_keyboard` node to teleoperate the robot. This node offers an extended set of keys to control the linear and angular velocities of the robot.

To use this node, follow these steps:

1. Download `rdl_teleop_keyboard.zip` and unzip the contents into your ROS 2 workspace (i.e. into `~/rdl_ws/src/`) on your Ubuntu PC. After this step, you will see a folder named `rdl_teleop_keyboard` under `~/rdl_ws/src/`.

2. Install a custom version of `setuptools`. This command only needs to be executed once on your Ubuntu PC.
```
pip install setuptools==58.2.0
```

3. Build the package rdl_teleop_keyboard using colcon:
```
cd ~/rdl_ws/
colcon build --packages-select rdl_teleop_keyboard
```

4. Source the local and global `setup.bash` files:
```
source install/setup.bash
source /opt/ros/humble/setup.bash
```

5. Start the node:
```
ros2 run rdl_teleop_keyboard rdl_teleop_keyboard
```

[2]Swedish wheels: https://en.wikipedia.org/wiki/Mecanum_wheel

You can now familiarise yourself with the teleoperation keys. For this, you can use the documentation of the officially available `teleop_twist_keyboard` node: https://github.com/ros2/teleop_twist_keyboard. To summarize:

- The first set of keys (e.g. u, i, j, o, etc.) select a direction for linear (forward / backward) and angular (clockwise / counterclockwise) motion.
- The last set of keys (q, z, w, x, e, c) increase or decrease the speeds of linear and angular motion.
- You should first select a direction and then adjust the velocities, if needed.

Note that the `rdl_teleop_keyboard` node publishes messages of type `Twist` (from `geometry_msgs.msg`) on the topic named `/velocity`. The message type `Twist` stores linear and angular velocities. In the next task, you will need the following fields from the `Twist` message in order to calculate the wheel speed for the left and right wheel:

- The linear velocity `linear.x` in $\left[\frac{m}{s}\right]$.
- The angular velocity about the z-axis `angular.z` in $\left[\frac{rad}{s}\right]$.

Note: The z-axis passes vertically through the robot. When looking from above, positive values for angular velocity result in counterclockwise rotation. The front face of the robot is the side where the two wheels are mounted.

5.2 Write a ROS2 Node for a Differential Drive Model

In this task, you will have to write a ROS 2 node that:

- **Subscribes** to the topic:
  `/velocity` of type `Twist` (from `geometry_msgs.msg`) and
- **Publishes** on the topics:
  `/turtlebot3_burger/wheel_left_joint_controller/command` and
  `/turtlebot3_burger/wheel_right_joint_controller/command`,
  both of type `Float64` (from `std_msgs.msg`).

The diagram of the controller node is depicted in Figure 2.

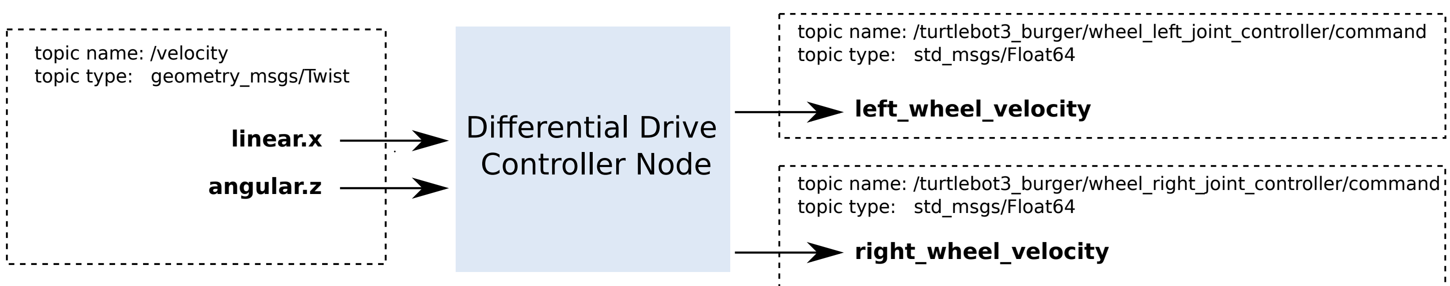

Abbildung 2: You have to implement the differential drive controller node that reads the robot's desired linear and angular velocities, computes the rotation velocities for the left and the right wheels, and publishes the wheel rotation velocities on the corresponding topics.

We provide you a skeleton code for this node. To use this code, do the following:

1. Download `worksheet02.zip` and unzip the contents into your ROS 2 workspace (i.e. into `~/rdl_ws/src/`). After this, you will see a folder named `worksheet02` under `~/rdl_ws/src/`.
2. Open the skeleton file `wheel_controller.py` under `~/rdl_ws/src/worksheet02/worksheet02/` using any editor (e.g. nano, gedit, vim, etc.). You will see that the source code in this file is incomplete. We have marked several places with the `FIXME` comment. You have to complete these parts yourself.

3. After you have completed these parts, you can build your `worksheet02` package using `colcon`.
```
cd ~/rdl_ws/
colcon build --packages-select worksheet02
```
4. Source the local and global `setup.bash` files:
```
source install/setup.bash
source /opt/ros/humble/setup.bash
```
5. If build is not successful, fix these errors and build again. After the build is successful, start the node as follows:
```
ros2 run worksheet02 wheel_velocity_controller
```
6. If errors are encountered when running the node, fix these errors, build the package again using `colcon` and then start the node using `ros2 run`.

Instructions for implementing the wheel_controller.py node:

- The distance between the wheels (known as **track width**) is 16 cm and the **diameter** of each wheel is 6.6 cm (see Figure 3). You can also check "Section 1.1.2.1 Data of TurtleBot3 Burger" of the e-manual for a diagram showing the robot's dimensions (link here).
- Note that all distance should have the unit `meters` and all angles should be in `radians`.
- To compute the wheel velocities based on the received linear and angular velocities, the following formulae can be used (differential drive model):

$$\texttt{vl} = \left(\texttt{linear.x} - \texttt{angular.z} \cdot \frac{\texttt{track\_width}}{2}\right) \cdot \frac{2}{\texttt{wheel\_diameter}}$$
$$\texttt{vr} = \left(\texttt{linear.x} + \texttt{angular.z} \cdot \frac{\texttt{track\_width}}{2}\right) \cdot \frac{2}{\texttt{wheel\_diameter}}$$

- **Important!** For device safety reasons, if the received linear and angular velocities exceed the following limits, then do not calculate or publish the wheel velocities:
  - **max. absolute value of linear velocity** `linear.x`: $\mathbf{0.18\frac{m}{s}}$.
  - **max. absolute value of angular velocity** `angular.z`: $\mathbf{2.5\frac{rad}{s}}$.

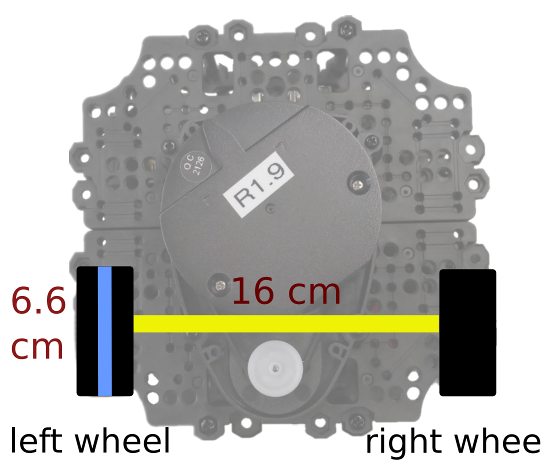

Abbildung 3: TurtleBot 3 Burger robots have a track width of 16 cm and each of its wheels has a diameter of 6.6 cm.

5.3 Test the ROS2 Node for a Differential Drive Model

After successfully testing your `wheel_velocity_controller` node on your local machines, use this node to move your TurtleBot. For this, we provide you with a ROS 2 package named `rdl_move_base`. This contains the node `rdl_move_base` that will connect your own `wheel_velocity_controller` node with the wheel motor drivers running on the robot. To use `rdl_move_base`, do the following:

1. Download `rdl_move_base.zip` and unzip the contents into your ROS 2 workspace on your local machine (i.e. into `~/rdl_ws/src/`). After this, you will see a folder named `rdl_move_base` under `~/rdl_ws/src/`.
2. Build `rdl_move_base` package using `colcon`:
   ```
   cd ~/rdl_ws/
   colcon build --packages-select rdl_move_base
   ```
3. Source the local and global `setup.bash` files:
   ```
   source install/setup.bash
   source /opt/ros/humble/setup.bash
   ```
4. Start the node:
   ```
   ros2 run rdl_move_base rdl_move_base
   ```

To test your `wheel_velocity_controller` node, you should first connect to your robot and launch `turtlebot3_bringup`. After this, on your local machine, you should start the given `rdl_teleop_keyboard` and `rdl_move_base` nodes. Finally, you should start your own `wheel_velocity_controller` node on your local machine. Now, you should be able to move the robot via teleoperation.

Please submit the following:

1. The completed source code for your ROS 2 node `wheel_controller.py`. (25%)
   Note: You do not need to embed code snippets in the PDF.
2. Screenshots showing (i) the received linear and angular velocities (from the topic `/velocity`) and (ii) the published left and right wheel velocities. (5%)
3. Attach a short video showing the robot being teleoperated using the given `rdl_teleop_keyboard` node and your own `wheel_velocity_controller` node. (5%)

6 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?

7 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 Stud.IP under *Files/Submissions/ex01*.
- 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.
- **Hint.** We recommend you to solve the tasks in the following time frame:
  - Tasks 1, 2, 5.1 and (5.2) in the first week of this worksheet.
  - Tasks 3, 4, 5.2 and 5.3 in the second week of this worksheet.