Laboratory exercise 6

Bag analysis and laser visualization

Name: Paola Glavić Sanković **JMBAG:** 0036522290

Preparation and helpful instructions

- Review the lecture slides about ROS, `rosbag` and lidar sensors.
- Remember to add an appropriate shebang (`#!/usr/bin/env python3`) at the beginning of your Python scripts, to make them executable using `chmod +x my_script.py`, and to add the appropriate `__main__` wrappers around your script.
- Run RViz and familiarize yourself with the application. RViz has a concept of *displays* which visualize data on different topics.
- Write clean, readable, easy to understand code. Give meaningful names to variables. Code quality will impact your grade. **Any deviation from the defined topic names, file names or the output format will result in points being deduced.**
- Format your code according to the PEP 8 style guide. (Your IDE, e.g. PyCharm, usually has a *Format code* command for this).
- Keep a separate terminal window/tab open running `roscore`. This is considered good practice in general when developing and testing with ROS.
  If you use `roslaunch` without having a `roscore` running separately, one will be started up automatically for you. However, if you afterwards run other nodes and then exit the initial `roslaunch` session, these ROS nodes will be orphaned since the `roscore` will be shut down with the initial `roslaunch` session.

Assignments

Task 1: **Bagging a Turtle mouse follow session. Bag analysis and correction**

For this task, you will be using the code from Lab exercise 5. Make sure you have `roscore` running in a separate terminal, as explained above.

a) Run an instance of Turtle mouse following using the solution of Exercise 5, and take note of the topic names. Which command will you use to list all the topic names?

rostopic list

Paste the list of all topic names here, and shut down the ROS nodes.

```
/mouse_position
/pose
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

b) In order to record the mouse position topic, and the `pose` and `cmd_vel` topics of the turtle into a bag called `turtlefollow_20xx-xx-xx-xx-xx-xx.bag`, which `rosbag` command will you use?

`_20xx-xx-xx-xx-xx-xx` indicates the time of recording, which needs to be *automatically added* by `rosbag` when using the appropriate option – **do not add this yourself** to the bag filename.

Enter the full command line into the box.

```
rosbag record -a -o turtlefollow /chatter
```

Hint: Use the `-h` option to read the documentation of `rosbag`. You can also use `-h` to view the options for the `record` subcommand.

c) First start recording by first running the command from b), and then run the nodes from the Turtle mouse follow exercise. Make the turtle follow the mouse for around 30 seconds, and then let it catch the mouse. Stop recording (Ctrl-C) and shut down all ROS nodes.

d) Examine the sample script `process_bag_example.py` which demonstrates how to loop through all messages in a bag and write them into another bag without any changes.

e) Write a script called `process_turtle_follow.py` which will **take in a single command-line argument, an input bag filename**, and perform the following:

i) Calculate and display the distance covered by the turtle. To perform the covered distance calculation, sum up all Euclidean distances between sequential turtle positions, available in messages on the Turtlesim `pose` topic.

ii) Calculate and display the *mouse follow session duration*, defined by the time difference between the first and last `pose` topic message of the turtle. (See the sample `ellipse.py` script in Task 2 for an example of calculating time differences using `rospy.Time`.)

iii) Calculate and display the average velocities for the follower turtle by dividing the covered distance with the mouse follow session duration.

iv) Write and count the messages in a new bag called `processed_follow.bag`, which will contain two topics:

i. the turtle pose topic, renamed to `/follower/pose`;

ii. the mouse positions topic, renamed to `/mouse_positions_on_grandparents_computer`, with the mouse positions scaled down to the nearest pixel on a monitor with an 800x600 resolution. (For example, on a full HD 1920x1080 display, the position (123, 456) should be mapped into (51, 253), because $51 = round(123/1920 \cdot 800)$, and likewise for the other dimension.

The output of `process_turtle_follow.py` must look like this (the values listed are placeholder numbers for illustration purposes only and should not be considered meaningful):

```
$ ./process_turtle_follow.py turtlefollow_2021-12-01-20-38-01.bag
Processing input bagfile: turtlefollow_2021-12-01-20-38-01.bag
Follower turtle
  Covered distance: 12.34 m
  Average velocity: 56.78 m/s
Follow session duration: 34.56 s
Wrote 1234 messages to processed_follow.bag
```

Task 2: **Laser sensor visualization**

In this task, you will use Matplotlib and RViz to visualize range data from real-world 2D laser sensor data captured in the Faculty building. The data is in the bag file provided with the exercise, `b-zgrada-minimal.bag`.

a) Run and thoroughly examine the two provided ROS node scripts, `ellipse.py` and `visualizer.py`. These two scripts are sample code to help you get started in solving this task.

The `ellipse.py` script calculates a set of points from an ellipse. The lengths of ellipse axes are animated to help illustrate that we are visualizing data which is changing in real time. The points are stored in the `points` field of a message of type `visualization_msgs/Marker` and published on the topic `point_positions`.

The `visualizer.py` script opens a Matplotlib window, subscribes to the `point_positions` topic, and plots the received points in the Matplotlib window.

Besides viewing the marker data in the Matlotlib window, visualize it in RViz as well. Run RViz (`rviz`), set the camera type to *TopDownOrtho* in the right panel, on the left panel in *Global options*, set the *Fixed frame* to match the frame id in `header.frame_id` in the marker message published in `ellipse.py`. Finally, add a *Marker* display using the *Add* button on the bottom of the left panel.

If adding display by type (first tab in the *Add* window), you need to set the topic to `point_positions` afterwards in the left panel, under the options of the Marker display you added. Alternatively, in the *Add* window, you can go to the second tab (*By topic*) and select the topic directly, which will automatically determine the display type (Marker).

Tip: After setting up everything RViz, press Ctrl-S or *File -> Save Config* to save the current settings as default when starting up RViz.

b) Examine the `sensor_msgs/LaserScan` message (http://docs.ros.org/en/api/sensor_msgs/html/msg/LaserScan.html) and the provided bag `b-zgrada-minimal.bag` using `rosbag info`. What is the name of the laser scan topic in the provided bag?

/pioneer/scan

Helpful tips for using `rostopic echo`: you can use `rostopic echo` to examine only parts of a message. For example, to view timestamps in the header of a `LaserScan` message on a topic named `scan`, you can use the following command: `rostopic echo scan/header/stamp`. To print only a single (1) message, you can pass `-n1`. To print messages straight from a bag without having to play it using `rosbag play`, you can pass `-b bagname.bag` to `rostopic echo`.

c) In the following subtasks, let j be the last two digits of your JMBAG + 100. For example, if your JMBAG is 0036465831, $j = 131$.

$j =$ 190

d) Write the full `rostopic echo` command for printing the `angle_min` field of the `LaserScan` message, for **first** j messages in `b-zgrada-minimal.bag`. Note: all angles are in radians.

rostopic echo /pioneer/scan/angle_min -b b-zgrada-minimal.bag -n190

What is the value of `angle_min`?

-2.356194496154785

What is the value of `angle_increment`?

0.008726646192371845

The formula for calculating the angle of the range measurement with an index i (where the first measurement has the index of 0) is: $angle = angle\_min + i * angle\_increment$. What would be the angle for a range measurement with the index j as defined in c)?

-0.6981317196041346

e) Note: **before running any nodes (or RViz)** which will be visualizing data **from a played bag**, make sure you run `rosparam set use_sim_time true` to use the simulated clock from the bag instead of the wall (system) clock. Make sure you have a `roscore` running in a separate tab.

(For running the sample `ellipse.py` from a) which animates the ellipse based on the system clock, `use_sim_time` should be set to `false`, which is the default value after after starting up `roscore`.)

f) Write a script for a ROS node named `laserscan_to_points.py` based on the sample `ellipse.py` script. This ROS node must subscribe to a `sensor_msgs/LaserScan` topic named `scan` (**not directly to the topic from** `b-zgrada-minimal.bag`). Instead of publishing a marker containing an ellipse, calculate the Cartesian (x, y) points using the range readings from received laser scans. See subtask d) for calculating the angle of a range reading.

Set the marker frame id to the frame id from the received `LaserScan` message. (**Do not hardcode it, read it from the received message.**) Also copy the stamp from the received `LaserScan` message. You can accomplish both of these at once by simply copying the entire `header`.

Increase the plot extents in `visualizer.py` to be able to fit the range data.

Which topic remapping will you use when running `laserscan_to_points.py` to visualize the scans from `b-zgrada-minimal.bag`? Enter the full command line.

./laserscan_to_points.py /scan:=/pioneer/scan

g) Start the `laserscan_to_points.py` node with the command line above, along with `visualizer.py`. Play the provided bag using the command `rosbag play --clock b-zgrada-minimal.bag`. Verify that your visualization is working correctly.

Tip: you can pass `--loop` to keep looping the bag until you quit with Ctrl-C.

h) What is the frame id (in the header) of the `LaserScan` messages in `b-zgrada-minimal.bag`?

pioneer/laser

How did you examine this?

in process_bag_example.py printed msg.header while iterating through bag

Start RViz with the setup **as described in a)**, and change the *Global frame* to the above frame id.

Verify that the Marker display for the `point_positions` topic looks identical to the `visualizer.py` Matplotlib window.

Finally, in RViz, add a native `LaserScan` display for the `LaserScan` topic directly. Make sure it matches the markers published by `laserscan_to_points.py` on the `point_positions` topic. **If it looks different, you have incorrectly solved the subtask f) and will not receive any points for Task 2!**

Feel free to adjust the `LaserScan` display colors in the left panel, and the `Marker` scale and color in `laserscan_to_points.py`.

Save the RViz config containing both the `LaserScan` and `Marker` displays using *File -> Save Config As...* in your exercise folder as `exercise_06.rviz`.

Notes:

- The command line remapping specifier `:=` should not have spaces around it.
- Do not change the `point_positions` marker topic name.
- Until closed with Ctrl-C, `laserscan_to_points.py` must keep working when the bag playback loops.
- The marker should be published from the laser scan message callback.
- The timer and the initial time are only used for animating the ellipse in the example. They should not be used in any way for visualizing laser scans.
- **In case you need help with solving the exercise by yourself, have any difficulties or questions, contact the course staff. Sanctions will be imposed for plagiarism.**

Exercise submission

Create a zip archive containing **this pdf with the filled out answers** and **all other exercise files** (`turtlefollow_20xx-xx-xx-xx-xx-xx.bag`, `process_turtle_follow.py`, `processed_follow.bag`, `laserscan_to_points.py`, `visualizer.py`, `exercise_06.rviz`) and upload it to Moodle.