Localization report

We will provide a rosbag to test our localization.

On one page, provide some snapshoots showing some positions after localization during the path performed by robair and a figure showing the complete path of robair in this bag file.

Lab Workflow and Testing Process

Our goal with this course and the labs is not for you to implement state of the art high-level algorithms with fancy toolboxes, but rather to implement simple, practical algorithms which you will be able to fully understand and explain, and which will allow you to get hands-on experience with the variety of challenges and problems in robotics.

We provide you with a codebase for the labs, so that you do not have to implement everything from scratch. In each lab, crucial parts of the implementation are missing, and it is your job to complete it. The algorithms we ask you to implement in the labs are the same as the ones covered during the lectures, so we assume you have reviewed the lecture material before starting each lab.

You have three tasks during each of the labs:

- **Part I: understand** the provided code structure in terms of the ROS Nodes, the messages they exchange, and the processing pipeline inside of each Node.
- **Part II:** understand which parts of the algorithm are missing and **implement** the rest of the algorithm.
- **Part III: experiment** with running the algorithm on the real robot (and recorded data files (rosbags) when applicable). Design **test** cases to **analyse** the algorithm's behaviour, its limitations.

Follow-me /3 points (3 tests: 1 point per test, 1 page max per test).

Given that the algorithms are already explained in quite a lot of detail in the lectures, we do not consider the implementation to be the main part of the workload. The main difficulty lies in getting a good understanding of *how* these algorithms work, and what their *limits* are. This is why we insist heavily on the testing process, which we describe below.

Testing Process

In any field of computer science, testing your code is essential. It is especially useful once you work on more complex projects such as robotics, where you have multiple processes running simultaneously, each implementing a different sub-module and exchanging data with the others. Below, we detail what we mean by "testing" your robotics algorithms.

How to design a test case?

In order to create test cases, you can use the following questions as a guide:

- Which **parameters** do we have in our algorithm?
- What are their respective roles, and **what do they influence**?
- **When** / **how** do they **change**?
- How can we design a **situation that makes this parameter vary**?
- **What happens** when this parameter changes / gets too large / small?
- If there are multiple "steps" in the algorithm, **what happens if one of the steps encounters a problem**?

We give two examples of how you could proceed at the end of this document.

How to perform a test?

1. **Design your test case** in order to evaluate a part of the algorithm, or a specific situation (see above for test-case design);
2. **Design a scenario** that allows you to put the robot into that situation;
3. Perform that scenario several times, **record** data and **write** down observations;
4. **Analyze** the data and your own observations to **understand** what happened, and why it happened;
5. **Write about the test case** to keep a trace (see below)

How to write about the test case:

1. Describe the goal of the test (refer to point 1 above);
2. Describe how your scenario enables you to test what you want to test;
3. Report on how many times you tried the scenario, in which conditions, and explain data recordings. State your results / observations;
4. Detail your analysis of the results (what conclusions can you draw from them?);
5. (optional, but encouraged) If the behaviour displayed by the algorithm is not ideal, discuss possible ways to improve it, or things that could be explored further.

Test case design examples:

1) Starting from the algorithm,and determining a situation that affects a given part of the algorithm: *“We create a new cluster based on a distance threshold. What is the consequence of this method? What impact does this parameter have? Let’s try having a person put both their legs very close to each other, or put their leg close to a wall, and analyse what happens.”*
2) Starting from an interesting / problem situation, and then backtracking through the algorithm to determine what is the cause / parts responsible for the behaviour: *“When we were running the person-detection, sometimes the person wasn’t detected. It seemed to happen when their legs were close together, or when they were near another object… What part of the algorithm could cause this to happen? Let’s try to trace what happens through the algorithm’s steps in this situation.”*