Yellow Project

Robotics 2 – BTF6420

Spring 2019
Moodle: GNG1-BTF6420-19

1 Objective

In this project students shall

- implement a program to control a mobile robot,
- **apply the theory** of the **robotics course** to solve the assignment,
- use existing software libraries for the robots' hardware,
- implement **collision avoidance** on a mobile robot using sensor data,
- implement and use **mapping and localization** algorithms with a mobile robot, and
- write a **clear and concise documentation** of their work.

As a part of the Robotics 2 course on mobile robotics students will work in teams of two persons with the *Yellow* robot. The goal of the project is to program the robot to navigate autonomously through a labyrinth. This includes avoiding collisions and dynamically generating a map.

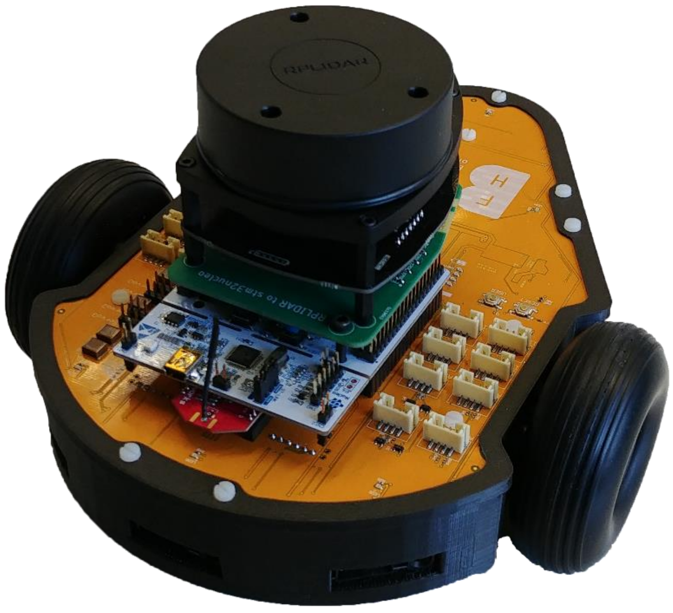

Figure 1: Mobile robot *Yellow*

2 Resources

From the class' exercises the students are already familiar with the programming environment and available libraries and hardware.

2.1 Programming Environment

A C++ project is edited in an IDE of choice (e.g. *QtCreator*, *Eclipse* or *Visual Studio*). The code is cross-compiled to a binary executable, which can be run on *Yellow*. Refer to the Software/README.md for instructions on how to open, compile and execute the project.

2.2 Libraries and Hardware References

The robot is controlled using the *mbed-os* library that offers classes for I/O peripherals of the STM NUCLEO F446RE microcontroller. Refer to the ARM mbed-os documentation.

For details on Yellow's hardware, refer to the components' datasheets and the summary on moodle.

3 Project Description

The project consists of two tasks. In the first task Yellow shall autonomously find its way out of a labyrinth. In the second task Yellow shall collect LIDAR data to construct a map of the labyrinth and then use it for planning navigation.

3.1 Task 1: Reactive Navigation

In this task the robot shall drive autonomously through an unknown labyrinth without collisions with walls or other obstacles. To detect obstacles, Yellow may use its infrared sensors.

Yellow will be placed at the start of the labyrinth. Yellow shall autonomously find the way out of the labyrinth **as fast as possible without collision.**

For task 1, the labyrinth will have the following properties:

- Corridor **width ≥ 0.5 m**.
- The **inner** wall **corners are ≥ 90°.**
- The wall material is Styrofoam blocks, as known from class.
- The labyrinth has **only one exit.**
- The labyrinth does **not have any bifurcations.**

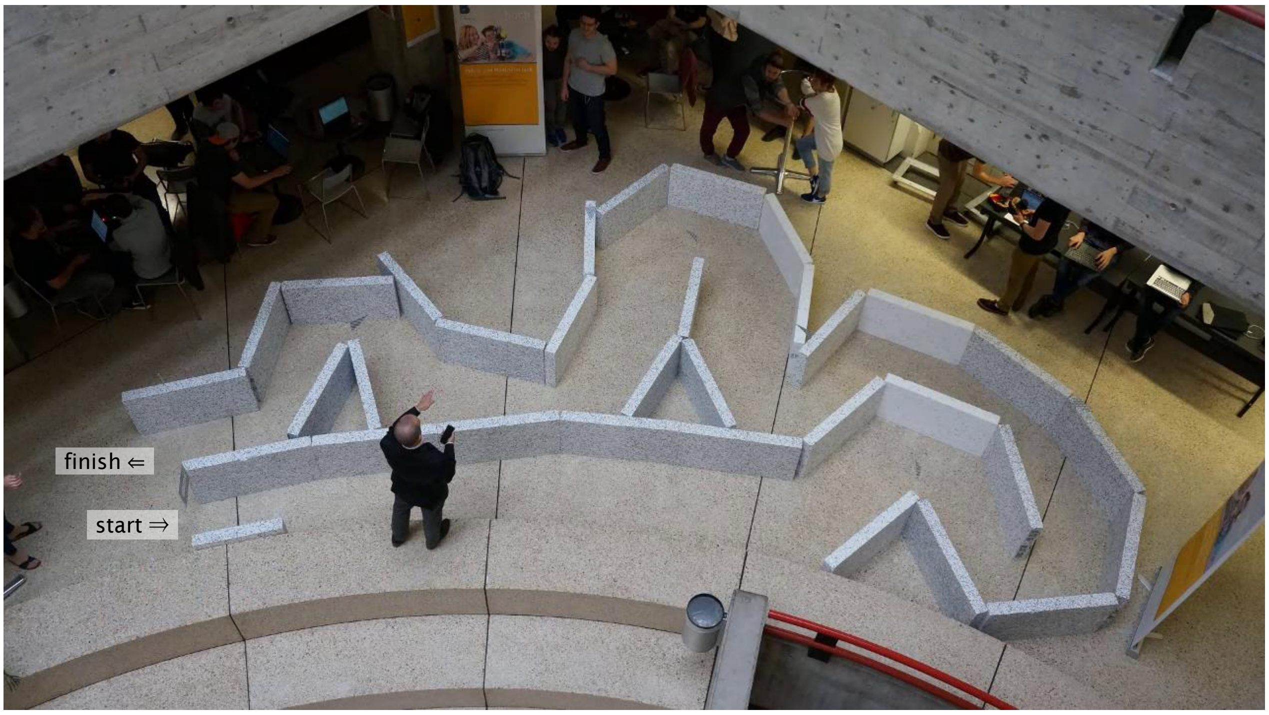

Figure 2: Example of a labyrinth for reactive navigation

For an example of the task refer to the video of last year's edition on moodle.

3.2 Task 2: Map-based Navigation

In this task Yellow shall collect data with its LIDAR. The LIDAR delivers a 360° scan with distances to objects around the robot for every degree. Additionally, for every point it delivers a quality *level* that relates to the amount of light reflected by the point.

To process the LIDAR scan data, a line finding algorithm may be implemented. The line data may be requested from a *Matlab* script running on a PC to construct a map.

For task 2, the labyrinth will have the following properties:

- Corridor **width ≥ 0.5m**.
- The **corners** will have an angle of **90°**.
- The wall material will be the same as in task 1.
- The labyrinth **may contain bifurcations**.
- The labyrinth **will be closed** (no exit available, different to task 1)

3.2.1 Local Occupancy Grid

In a first step, a local map of the robot's environment shall be displayed graphically on a PC in the form of an occupancy grid. The occupancy grid shall display **occupied cells** (i.e. from visible obstacles), **free cells** and **unknown cells**. The resulting map will display the robot's environment in the robot's frame. To complete this task, a single LIDAR scan suffices, since the robot shall not move.

Each cell in the occupancy grid shall have the size 10x10 cm.

3.2.2 Global Map with Odometry

Yellow shall autonomously **drive through the complete labyrinth** and construct a **complete map** based on the LIDAR (line) data. The mapping operation may be performed on a PC instead of on Yellow's onboard microprocessor. The mapping operation shall be autonomous, any human intervention will result in a point deduction.

You shall use the robot's odometry data as it drives to find a relation between LIDAR scans from different positions and continuously construct a global map.

Algorithms and existing libraries and toolboxes may be used to solve this task. In any case the team will have to be able to explain the methods used.

Save the global map
Once the map is complete, the corresponding occupancy grid may be saved as a bitmap with each cell saved as one pixel.

Read a global map
If a team fails to create a reliable global map, it may receive the *official* global map as a bitmap image file (boolean occupancy grid). It is therefore a good idea for all teams to write software that reads a bitmap and converts it to an occupancy grid usable for the following tasks.

A sample bitmap occupancy grid is provided on Moodle.

3.2.3 Localization

Yellow will be placed on an arbitrary location inside the labyrinth. Yellow shall localize itself **using the global map**.

You may use algorithms as *Markov* or *Kalman Localization* as seen in class to localize your robot in the global map. Note that localization usually may only be completed if the robot drives around to gather LIDAR data.

Alternative given start Location
If a team's robot fails to localize itself, its position may be specified by hand. For example, the **pixel position** on the map may be typed or clicked on a graphic on the PC.

3.2.4 Navigation

Yellow shall navigate from its current position to a defined goal on the map. The goal may be given to the robot by hand. For example, the **pixel position** on the map may be typed or clicked on a graphic on the PC. Yellow shall drive **as fast as possible and without collisions** to the goal. The arrival tolerance shall be ±5 cm. **The closer** Yellow arrives **to the goal**, the more points you will get.

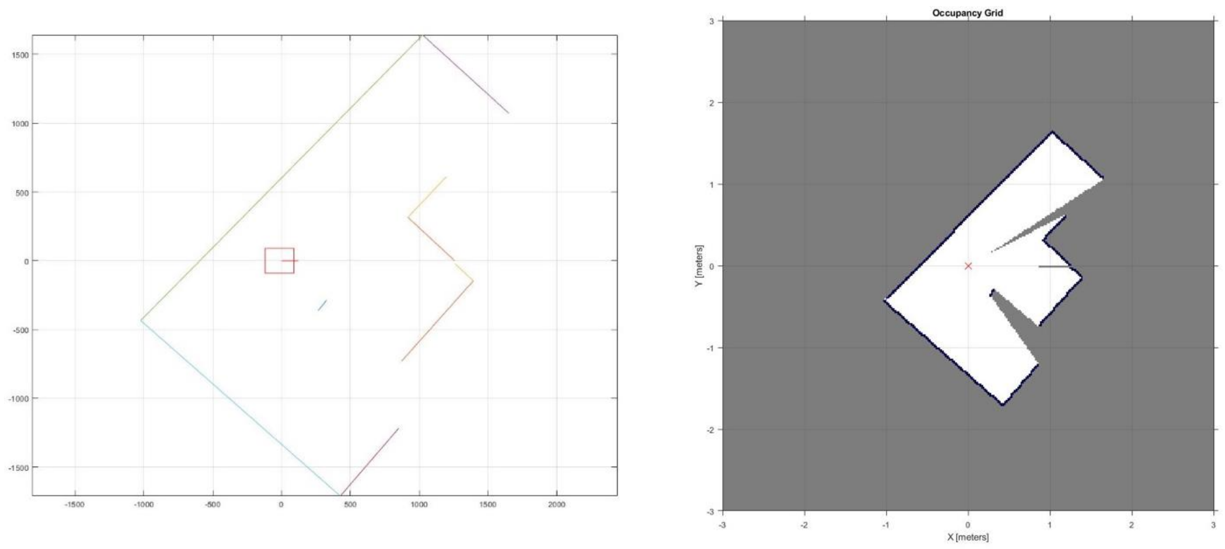

Figure 3: Left: LIDAR Lines. Right: Occupancy grid of a local LIDAR scan.

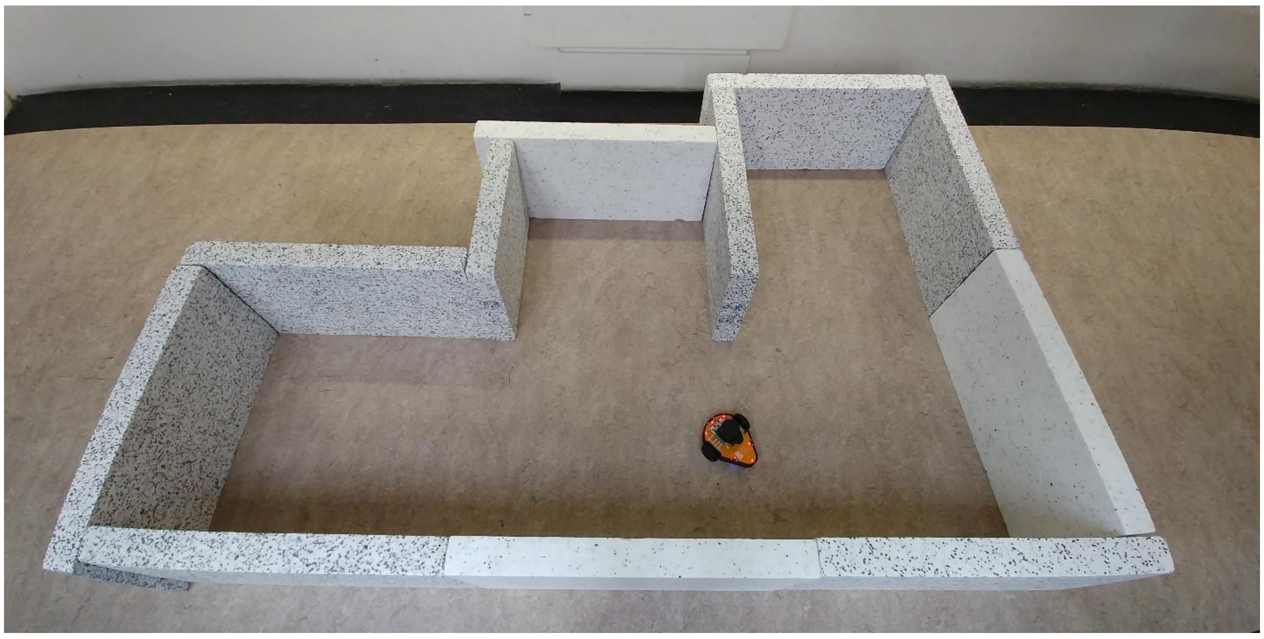

Figure 4: Top view of the Lidar sample for the robot pose shown in Figure 3 left

4 Project Documentation

The project work shall be briefly documented. Clean sketches may help to concisely describe concepts.

The documentation shall not exceed a maximum of 10'000 characters (including white spaces).

The report shall contain the following points:

- The most important algorithms, methods and design decisions so that an outsider (who knows the assignment) can understand the solution strategy.
- A diagram of the software's state machine.
- A list of all threads used, including their execution period.
- A short outlook describing possible enhancements to the current solution.
- A screenshot of the generated local and global maps.

5 Due Date

The project presentation will take place on **Thursday, May 2, 2019**. Refer to section 7 below for details.

The **documentation** (one single **PDF**-file) as well as the **source code** (C++ and Matlab) shall be **committed** (and pushed) to the team's git repository (project-19_groupNumber) until:

Friday, May 3, 2019, 8:00.

6 Grading

The project accounts for **45% of the Robotics 2** module's final grade.

Task	Weight [%]
1. Reactive Navigation	20
2. Map-based Navigation	60
3. Documentation	20
Total	**100**

7 Project Presentation

7.1 Task 1: Reactive Navigation

From 10:00 to 11:00 next to the cafeteria, Quellgasse 21, 3rd floor.

Every team will drive through the labyrinth **one after the other** as in a ski downhill race. The official race consists of **two runs** of the exact same labyrinth. Each team will keep the best score of the two runs.

The labyrinth will be available for **test runs from 9:30 to 9:55**. During this time, teams are allowed to use the labyrinth and fine-tune their algorithms.

During the race, robots must be ready to run one right after the other without interruption. There will be no break between the two race rounds.

The score will be assessed using the following two criteria:

- No collisions during the drive.
- Time to complete the run – from start until crossing the finish line.

Bonus points:

- Robots dressed/decorated in creative ways that differentiate them from the others will receive bonus points.

7.2 Task 2: Map-based Navigation

From 11:00 to 16:30, Quellgasse 21, room 510.

Each team receives **15 minutes** to show their ability to create a map of the labyrinth, localize the robot and navigate to a specified goal. Time management is important: no additional time will be given.

The following tests will be performed:

1. The robot will be placed somewhere in the labyrinth
2. Yellow **generates a local map** (occupancy grid) without moving.
3. Yellow **drives autonomously** through the labyrinth and **generates a global map**.
4. Yellow is **placed at an unknown location of the labyrinth** and it **localizes itself**.
5. Yellow **drives from a known location** (from 4 above) to a **specified map goal**.

In the best case, the 5 tests will be performed in sequence using information from a previous test to solve the next. Note that you may use an external PC (e.g. with Matlab) to help Yellow solve the task.

A team may skip or abort a particular test: the information generated during the skipped test required to perform the following test will be provided.

The assessment of task 2 has the following distribution:

Test	Weight [%]
Local map	15
Global map	15
Localization	15
Navigation	15
Total	**60**

7.2.1 Time table for task 2 – Map-based Navigation

The table below shows the time slots randomly allocated to each team to present Map-based Navigation.

Time	11:00 – 11:15	11:20 – 11:35	11:40 – 11:55	13:00 – 13:15	13:20 – 13:35	13:40 – 13:55
Group	8	10	5	11	12	2

Time	14:15 – 14:30	14:35 – 14:50	14:55 – 15:10	15:30 – 15:45	15:50 – 16:05	16:10 – 16:20
Group	1	7	6	3	9	4