Mobile Robot Control 2024 - Local Navigation Assignment

This assignment is about safe navigation on a real robot. The goal is to let the robot move through a corridor without colliding with obstacles that are not known in advance.

Setup

For this assignment, you can use the same setup as for the exercise ‘The art of not crashing' in week 1. A lot of the code, such as for obtaining laser and odometry data, can be re-used here.

The challenge is to move 6 meters through the corridor without collisions. Slightly touching the walls or obstacles is allowed, however, bumping (i.e., driving head-on into a wall) is not allowed. The robot receives a target position relative to its initial position and must navigate there while avoiding obstacles using the odometry and laser data. Once the robot is approximately near the target position (according to odometry), let the robot stop.

Assignment

Divide your group into two groups of equal size, where each subgroup must implement a **different** solution (e.g., artificial potential fields and dynamic window approach) to tackle this challenge. The approach can be inspired by the methods explained during the lecture, methods you find in literature or online, or ideas that you come up with yourself. Every method is fine **as long as you can explain how it works!**

Four maps have been provided with increasing difficulty. You should be able to build these maps on the soccer field.

To easily test your code and make it more modular (very helpful for the final challenge!), ensure to create a `cpp` function of each implementation (the function can be in a class if desired). In this way, it is easy to switch between the two solutions since you only have to change your function call. Additionally, it can be called easily for future developments (e.g., linking it to other components like global navigation). For information on `cpp` functions, please see this link.

Testing

Test your code both in simulation and on the real robot using the provided maps (you can always create more maps if desired). If your solution works on the real robot, also try to walk through the corridor yourself while the robot is on its way and see how it reacts to you!

Make sure to test the two algorithms in the same conditions to get a good comparison.

Questions

1. What are the advantages and disadvantages of your solutions?
2. What are possible scenarios which can result in failures (crashes, movements in the wrong direction, ...) for each implementation? For example, set the target position in an obstacle and let the robot try to get to the target; what happens? Try to think of more examples yourself!
3. How would you prevent these scenarios from happening?
4. For the final challenge, you will have to link local and global navigation. The global planner will provide a list of (x, y) (or (x, y, θ)) positions to visit. How could these two algorithms be combined? (Implementing this will be part of next week's assignment.)

Submission

Upload your code with comments/documentation on your group's GitLab page on **one** git branch, meaning that both solutions should work. Additionally, on your group's wiki page you should:

1. Describe the main idea behind your approaches,
2. Upload screen recordings of the simulation results: show videos of your two solutions with the same target position. Comment on the behavior of the robot, e.g., if it is oscillating. If the robot does not make it to other end, describe what is going wrong and how you would solve it (if time would allow).
3. Upload video's of the robot's performance in real life (same as simulation videos), and comment the seen behavior.
4. Answer the questions given above.