CENG 460 - Introduction to Robotics
Fall 2018

Middle East Technical University
Uluç Saranlı

Homework 4 - Sensor-based Navigation and Potential Fields

Assigned - Dec 22, 2018, **Due - Jan 6, 2018**

In this homework, you will be working with a point robot on the plane (configuration space $\mathbb{R}^2$), equipped with a 360^o range sensor. The following questions will require you to implement different navigation problems with this robot. In doing so, you can also assume that you have access to the robot's own position $q \in \mathbb{R}^2$, as well as the position of a desired goal position $q^* \in \mathbb{R}^2$. You are not allowed to use any information other than these. In particular, your navigational controller should not have access to the global map, but you can of course build your own map internally. Matlab scripts accompanying this homework provide a skeleton implementation of the range sensor and facilities to build and simulate a polygonal environment. The following utility and example scripts are provided:

- `build_arena(n)`: Allows you to interactively create n polygonal obstacles, recording their vertex coordinates into the global variable `arena_map`.
- `draw_arena`: Draws the current arena defined by the global variable `arena_map`. Does not clear the figure.
- `distfn(angle, position)`: Returns the distance to the closest obstacle in the direction `angle` from the horizontal axis for when the robot is located at `position`
- `draw_range_map(position, steps)`: Draws an illustration of the range map with the number of uniformly spaced rays specified in `steps` for when the robot is located at `position`. Clears the current figure, and shows two subfigures, one with the arena, second with the range map for $\theta \in [0, 2\pi]$.
- `run_planner(planner)`: A skeleton example that calls a planning algorithm implemented by the script with the name supplied in the argument and visualizes the result. As an example, try running `run_planner('linear_path')`.
- `linear_path(qstart, qgoal)`: A simple "planner" that simply returns a linear path from `qstart` to `qgoal`.

For each of the questions below, you should submit a separate Matlab file with your planning algorithm, together with an associated section in your homework PDF submission explaining your algorithmic choices and solutions (no code!), as well as results from your example simulations on different obstacles. Make sure to discuss challenges you faced, failures you encountered and to include different obstacle examples. For all questions, you can assume a sensor range of $1.5m$, and you can use 360 angular samples for your range sensors. I would recommend that you use fewer samples in your initial development to increase speed of debugging.

1. Implement sensor-based boundary following for convex polygonal obstacles. An example of such an obstacle is provided in `example_obs1.m`. You can use a sensor range of $1.5m$, and assume that the obstacle is visible from the starting position of the robot. Your algorithm should follow the obstacle at a distance of $0.5m$ and stop when the robot comes back to the same point on the obstacle boundary. Your planner implementation should be supplied in the a separate file `e1234567_convex.m`.

2. Extend your boundary following algorithm to also handle concave obstacles, wherein the same obstacle might now have more than one closest points, so turning around inner corners will be trickier. An example of such an obstacle is provided in `example_obs2.m`. You can use a sensor range of $1.5m$, and assume that the obstacle is visible from the starting position of the robot. Your algorithm should follow the obstacle at a distance of $0.5m$ and stop when the robot comes back to the same point on the obstacle boundary. Your planner implementation should be supplied in the a separate file `e1234567_concave.m`.

3. Implement a potential field based motion planner using attracting and repulsing fields with a sensor range of $1.5m$. You can assume that the "safe distance" from obstacles is $0.5m$, beyond which repulsing field components may be zero. An example arena for such a scenario is given in `example_obs3.m`. Your planner implementation should be supplied in the a separate file `e1234567_potential.m`.

Submission

Submitted PDF reports for your solutions must be typeset in a word processing environment such as LaTeX. Submissions are expected to be in the form of a ZIP file named `hw4_e1234567.zip`, including a PDF report with answers to theoretical questions with your name and student ID indicated clearly, as well as Matlab or other source files that are requested in the homework text. Late submissions will be penalized with a deduction of $10n^2$ points where n is the number of late days.

Note: You can discuss your discoveries and knowledge with your classmates but you must write your own answers and code for all questions above. If any significant similarities are found between your answers and other homeworks, you will be audited on your understanding of your own solutions.