MRC 2023 Navigation Assignment 1

The goal of this assignment is to apply the A* algorithm to find the shortest path from the start to the exit of a maze. A framework for the implementation has been provided, in which three parts of the code are left out. The assignment is to complete the implementation by writing these parts.

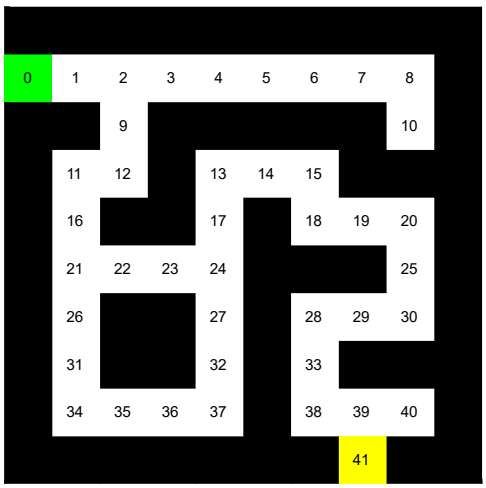
Setup

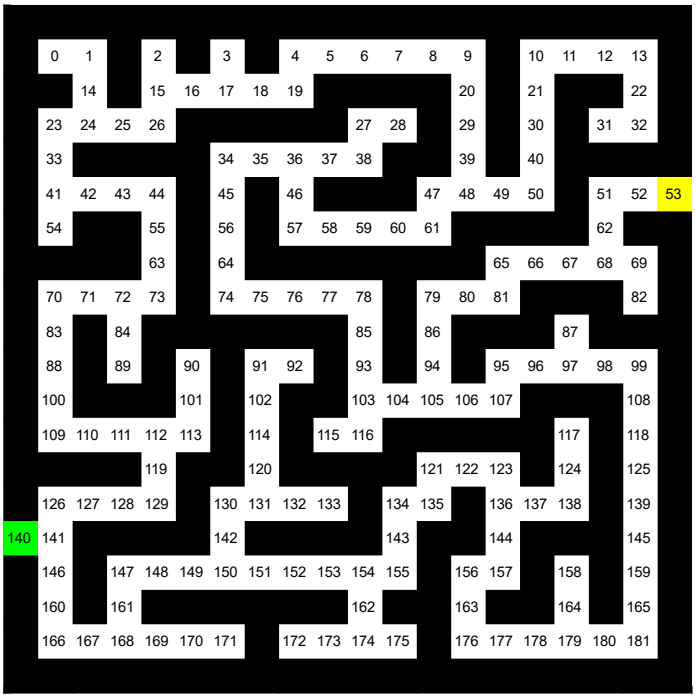
The code for this assignment can be found at:
https://gitlab.tue.nl/mobile-robot-control/mrc-navigation-assignment-1

Besides the source code (to be completed by you), it provides the files containing information about the mazes that will be used to test your implementation. For each map, these files are:

- PNG file of the maze
- YAML file with metadata for the simulator (e.g., map PNG filename and resolution)
- JSON file with configuration settings for the simulator (e.g., the robot's initial pose)
- JSON file with parameters of the maze (e.g., node grid locations, connections, and node IDs of the start and goal)

These files can be found in the subfolders `config` and `maps`. Visual representations of the mazes including the node IDs are shown below. All horizontally or vertically adjacent nodes are connected.





Assignment

The code to be completed is in the function `planPath()` in the file `'path-to-repository'/src/Planner/planning.c`. Within this function, to goal is to find the sequence of node IDs that form the shortest path through the maze from start to exit by applying the A* algorithm. The initialization step and the code's main structure have already been provided. The nodes' coordinates are given in `_nodelist`. The connections per node are given in

`_connections`, i.e., `_connections[i]` gives the indices of the nodes that are connected to the node with index `i` (which corresponds to `_nodelist[i]`). Furthermore, `_entrance_nodeID` and `_finish_nodeID` are the indices of the start and goal nodes.

The exercises/parts to be completed are indicated in the code. The first exercise is to find, in every iteration, the node that should be expanded next. The second exercise is to explore its neighbors and update them if necessary. The last exercise is to trace back the optimal path. It should not be necessary to make changes to other files or functions, but you are allowed to do so.

Testing

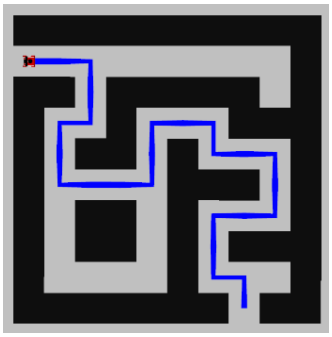
To run a test for one of the mazes, first build your code:

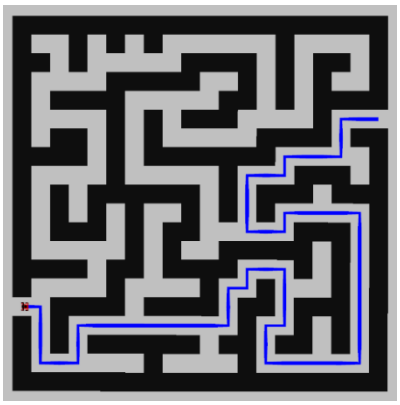
- `cd 'path-to-repository'/build`
- `cmake ..`
- `make`

If this is successful, run the test:

- Start the simulator with the YAML and JSON files corresponding to the desired maze, e.g.: `mrc-sim --map 'path-to-repository'/maps/metadata_maze_small.yaml --config 'path-to-repository'/config/config_maze_small.json`
- In a new terminal, open the visualization: `sim-rviz`
- In another terminal:
  - `cd 'path-to-repository'/bin`
  - Run the test for one of the mazes (use the JSON parameter file as an argument), e.g.: `./assignment1 ../config/params_maze_small.json`

If everything works correctly, the path as planned by your implementation of the A* algorithm will now be drawn in RViz (as shown in the images below) and the robot will start following it.





Reflection and submission

How could finding the shortest path through the maze using the A* algorithm be made more efficient by placing the nodes differently? Sketch the small maze with the proposed nodes and the connections between them. Why would this be more efficient?

Upload:

- the repository with your completed code on your group's GitLab page.
- the answers to the questions above (including the sketch) on your group's wiki page.