Real-time Smooth Realistic Paths & Navigation

Introduction

Autonomous navigation by a UGV or mobile robot from one location to another is a very complex process. The robot must accomplish at least four simultaneous tasks to be successful and efficient:

Perception: viewing the world and interpreting what it sees (this project: laser / sonar).
Localization: keeping track of the robot's position (this project: odometry).
Local navigation: making sure the robot doesn't drive into holes, obstacles… (this project: sonar / laser).
Global path planning: Find the safest and fastest way to go from start to goal. Different algorithms exist such as: A*, D* …

The focus in this project will be somewhere between global path planning and local navigation.

Project description

In this project the world is represented as grid, see figure 1. Planning space is broken up into discrete, non-overlapping regions (= cells). The result is a graph in which each cell is adjacent to other cells. The global planner searches for the optimal path as a sequence of cells. The global planner will give the path with the lowest cost, but it won't give the most smooth naturally looking path, as can be seen in figure 2.

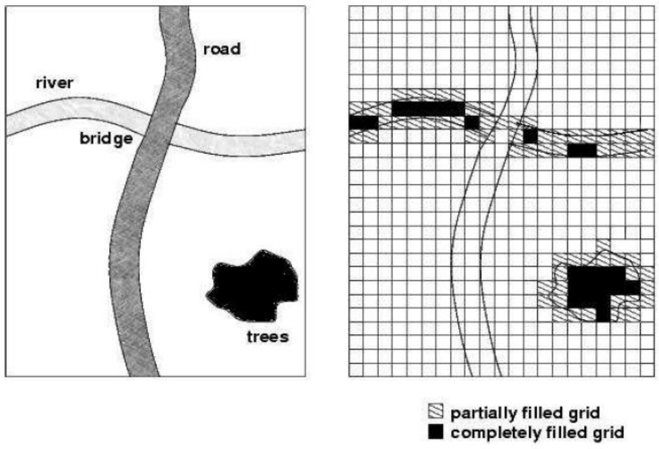


Figure 1: The map is created by laying a regular grid over the planning space. If there is an object in the cell, that cell is marked as an obstacle. Otherwise it is left as free space

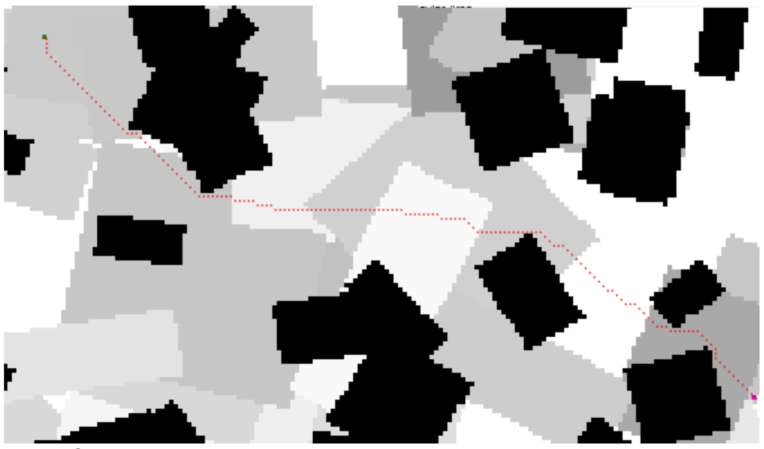


Figure 2: Planned path from start to goal, calculated using the A* algorithm. It can be seen that this path exhibits a zigzag effect and is far from smooth and natural looking.

Goals of the project are:

1. Perform post processing on the path that will remove the zigzag effect that results from the global path planning algorithm.
Each position in the calculated path is associated with a cell and therefore its coordinates are located in the center of the cell. The path resulting from post processing does not need to suffer from this constraint which will make it possible to replace a zigzag part by a straight line, as seen in figure 3.
'Turning by abrupt change of direction' must be replaced by 'turning in a curved manner', see figure 4. Things to take into account are: the dimensions of the robot, the current orientation of the robot when performing a turn, the turning radius, the vicinity of obstacles…

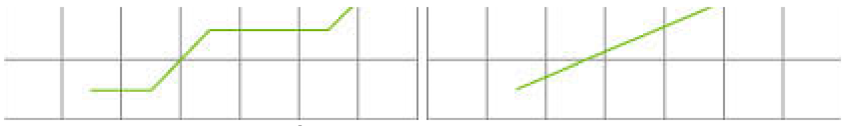


Figure 3: Transforming zigzag path into a straight path.

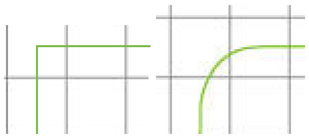


Figure 4: Transforming an abrupt change in direction in a curved turn.

2. Execute this path in a safe and natural manner. Smooth transitions from one speed to another. While executing this path speed needs to be adjusted depending on the condition of the terrain (ex.: muddy, wet...), the gradient of the terrain, the environment (ex.: small passage with lots of corners, vicinity to obstacles…)

Practical Situation: Unknown Environment
Many path planning methods assume complete knowledge about the environment. However there are situations where the map is incomplete, wrong, has too low of a resolution, or even is unknown (often in outdoor environments). In these cases path planning algorithms are needed that are able to replan based on sensor data acquired during execution of the mission. The general method is to create a global path with the available map information and then replan when new information is acquired. While the robot is replanning it cannot continue towards the goal, so it is crucial that replanning is fast. In this project we will assume no knowledge about the environment.

The student will be provided with an implementation of this replanning method (short example below). In the current implementation the robot (fig. 5) executes the current plan calculated by the global planner. The goal is to perform the post processing on this path before executing it. As all these calculation need to be performed real-time, this post processing may not introduce a bottleneck.

Tasks:

- Getting to know the robot and accompanying software ARIA (robot control software in C++).
- Literature study: smooth paths / navigation / …
- Form strategy to solve given problem.
- Implement algorithm in C++.
- Integrate your algorithm with the replan program.

- Test algorithm on custom made maps (as in fig. 2) to simulate different terrain conditions, gradients…
- Test algorithm in simulation with existing MobileSim simulator, see figure 6.
- Test algorithm on the robot Pioneer P3-DX and/or Pioneer 3-AT, see figure 5.
- Document results.

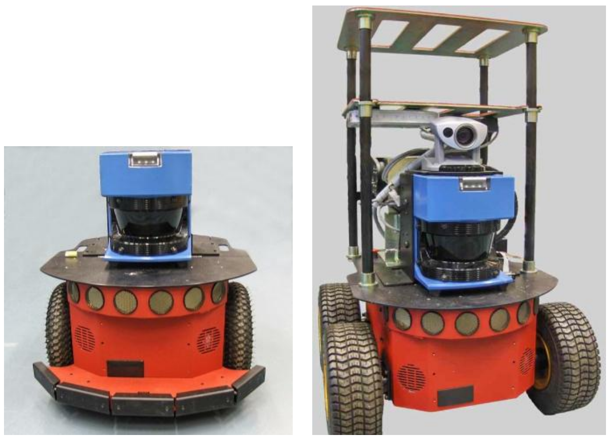


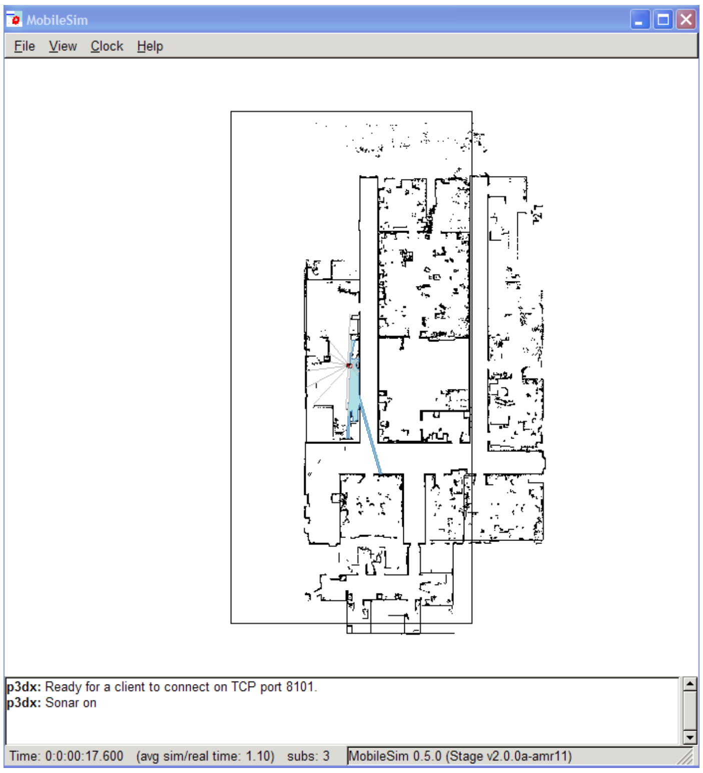
Figure 5: On the left: Pioneer P3-DX. On the right: Pioneer 3-AT

Student profile

- Knowledge of programming in C++.
- Interest in robotics.

Replan program

In this example the map is considered unknown, the complete map known to the robot is obstacle free. The actual map is composed of obstacles shown in figure 5. Under this assumption an initial path from start to goal is calculated, shown in red in figure 6.
While the robot executes this path, it discovers discrepancies between its map and the environment. This information will be updated in the map. And the path from its current position to the goal will be replanned according the newly updated map, see figure 7 and 8. It is important that this replanning is performed fast, since the robot has to stop until a new path is calculated.
The current implementation takes into account the size of the robot while planning to keep the robot at a safe distance from the obstacles. The same program can be run on the robot as well.



p3dx: Ready for a client to connect on TCP port 8101.
p3dx: Sonar on

Time: 0:0:00:17.600 (avg sim/real time: 1.10) subs: 3 MobileSim 0.5.0 (Stage v2.0.0a-amr11)

Figure 6: Screenshot of the simulator with the robot in starting position. The map, the laser range and sonar beams are also displayed.

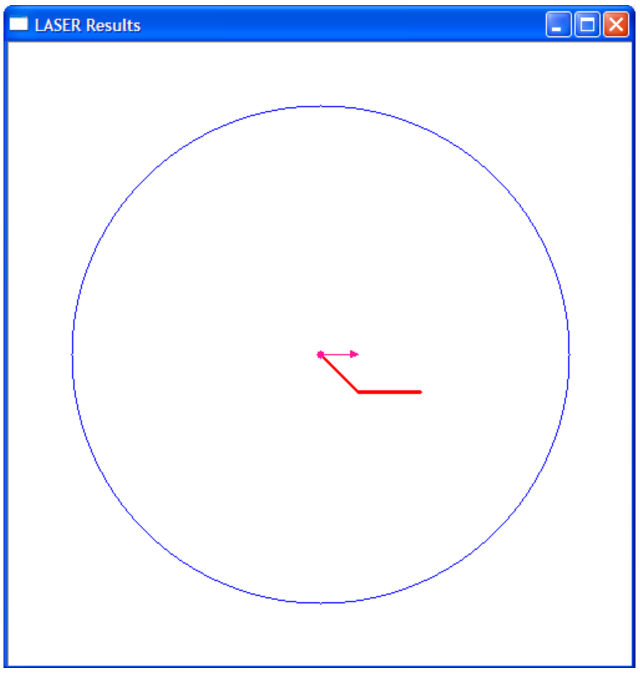
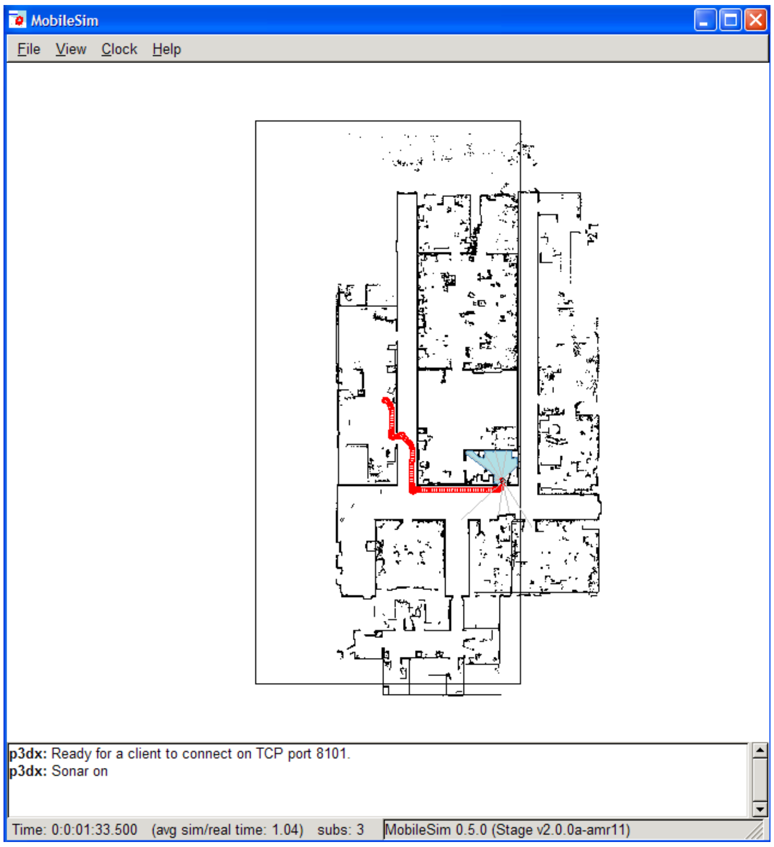


Figure 7: Screenshot of the laser results. Shown in red is the current planned path to go from the current position to the goal position. The circle and the arrow represent the robots current position and orientation respectively. The large blue circle represents the range of the laser. In black the results from the laser are shown. As this is the start of the mission nothing has been detected yet.



p3dx: Ready for a client to connect on TCP port 8101.
p3dx: Sonar on

Time: 0:0:01:33.500 (avg sim/real time: 1.04) subs: 3 MobileSim 0.5.0 (Stage v2.0.0a-amr11)

Figure 7: Screenshot of the simulator during execution of the current path. The traversed path of the robot is shown in red.

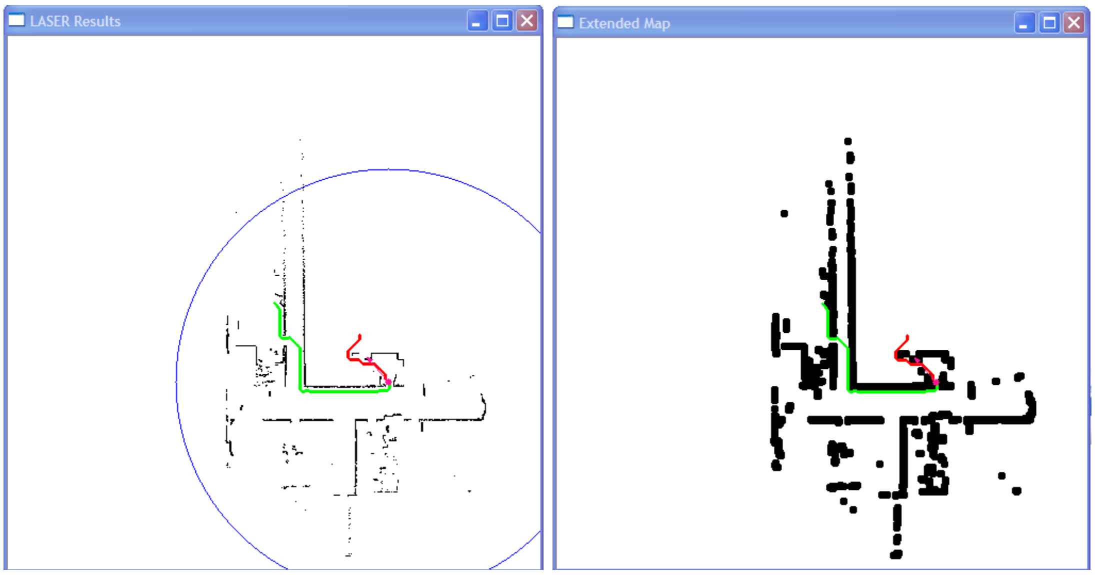


Figure 8: On the left: screenshot of the laser results. The part of the path that is traversed by the robot is shown in green; the part that still has to be executed is shown in red. As the robot attempts to follow this path, it discovers new information, updates the map and replans. In black, the information the robot has gained at this time in execution is shown. On the right: screenshot of the extended map. This figure shows the same information as the laser results on the left, except that each laser point is extended with the robot dimension radius. Planning is performed in this map, as it ensures that the robot will stay at a safe distance from obstacles.