Robotic systems
Report

Sergio Castillo
Vince Jankovics
Mike Lagunes
Kevin Zhou

1 Introduction

2 Robot Design

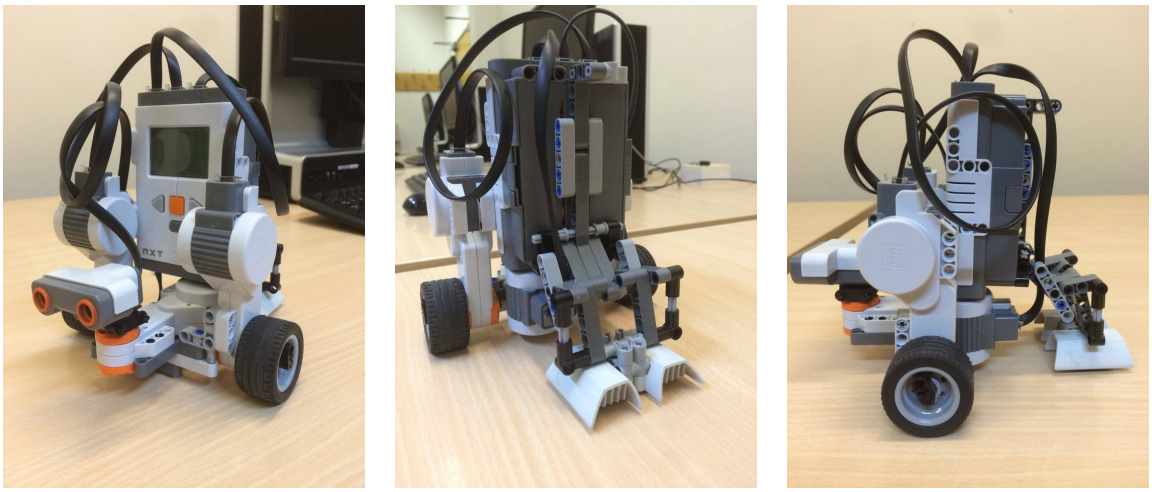
During the design and assembly stage, two models where proposed. The first prototype used caterpillar wheels, robust body and a sensor on the top with 360° of freedom. The second and current model,conversely, is lighter with the sensor suited at the bottom with 180° of freedom. It presents the following advantages:

1) Lightweight. The main robot pieces are the motors, these parts are used as frame to support the body and the NXT control.

2) Compact. The first prototype had a long body because the NXT control was in a horizontal position. This design presented two main problems, difficulties to turn and a considerable error range while rotating. In order to remove these issues the NXT control was placed in vertical position secured by few pieces to the motors.

3) Fast. Taking into account the previous points the robot's movements resulted to be quick and accurate. An improvement was made using a tail to provide more stability and reducing the friction while rotating.

Images of the final version can be seen in Figure 1.



(a) Front (b) Back (c) Side

Figure 1: Robot design views

3 Mathematical modelling

3.1 Sensors

3.2 Actuators

3.3 Overall system

3.4 Experiments, validation

4 The kidnapped robot problem

4.1 Localization

4.2 Guidance

Before and after the robot is localized in the map, it has to know what to do next according to the current objective, which are:

1. Avoid collision all time,
2. Explore map,
3. Go to target.

4.2.1 Collision avoidence

The robot's highest priority objective is avoiding collisions with the walls. This is done by a simple algorithm, which estimates the smallest distance between the robot and the scanned points based on the commanded movement. If this predicted distance is smaller than a certain threshold, it 'bounces back' from the boundary (see Figure 2). The reflected angle is perturbed with a small noise factor, so the robot does not get stuck in an area (e.g. when it goes perpendicular to the wall).

4.2.2 Exploration

The exploration is done before the PFL is converged to a location (i.e. the localization is not complete), so the robot can wonder around the map and exploit more of its features. The robot builds an internal map of the sensed points (called `knownPoints` in the code) and the points where it had been (called `beenThere` in the code), which serve as the basis of the decision making. These are based on the commanded movements and sensor data, so they are a rough estimate of the surrounding space.

The points are used to build an *Artificial potencial field* (U), and drive the robot downhill (i.e. $-\nabla U$) [1]. Since the points where the robot had been is also used for this field (with different weight than the walls), the robot is driven towards unexplored locations. The result can be seen on Figure 3.

The effect of the potencial field can be tuned by the constant ϵ, so the commanded turning angle is:

$$\delta\theta = \epsilon \cdot \arg(-\nabla U), \tag{1}$$

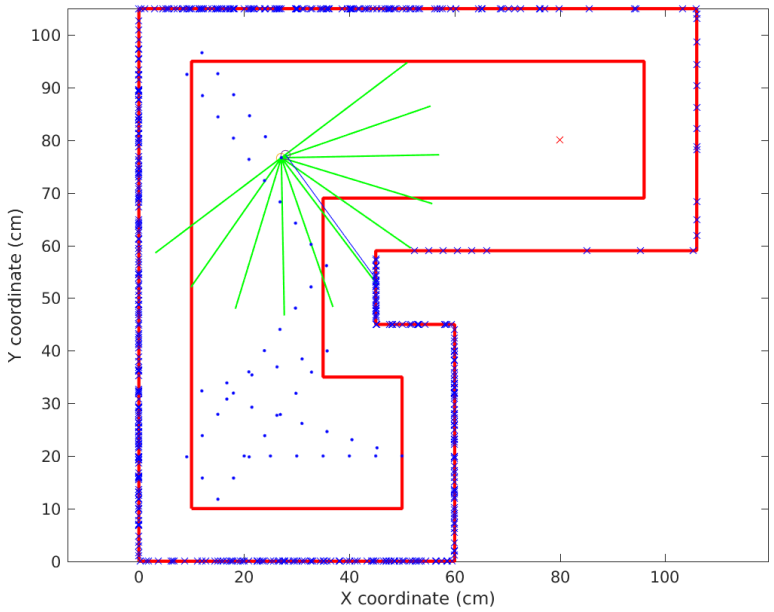


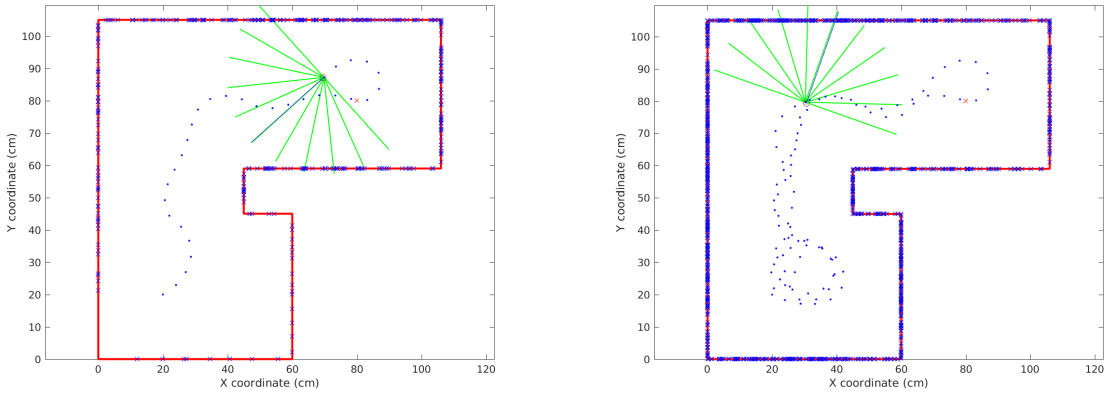
Figure 2: 'Wall bouncing', with a distance threshold of 10cm (inner boundary).

meaning that with lower ϵ the robot has higher inertia (goes more uphill). The commanded forward movement is kept constant, so the robot gives more consistent response even on higher gradient values (e.g. close to a wall).

This feature improves the robot's movement compared to the simple bouncing, but it needs the collision avoidance as a simple backup. The result is shown on Figure 4.

4.2.3 Path planning

After the PFL is converged, the robot can plan its path from its current location to the target. The planning algorithm was a standard A* search [2] on a visibility graph [1] defined by the map, robot and target. This way the search space for the optimal path is greatly reduced and the movements are smoother compared to a grid-word representation. Figure 5 shows the visibility graph with the optimal path between the robot and the target.



(a) (b)

Figure 3: Simulation with artificial potential field as guidance ($\epsilon = 0.3$). (a) Initial path. (b) Path after some time.

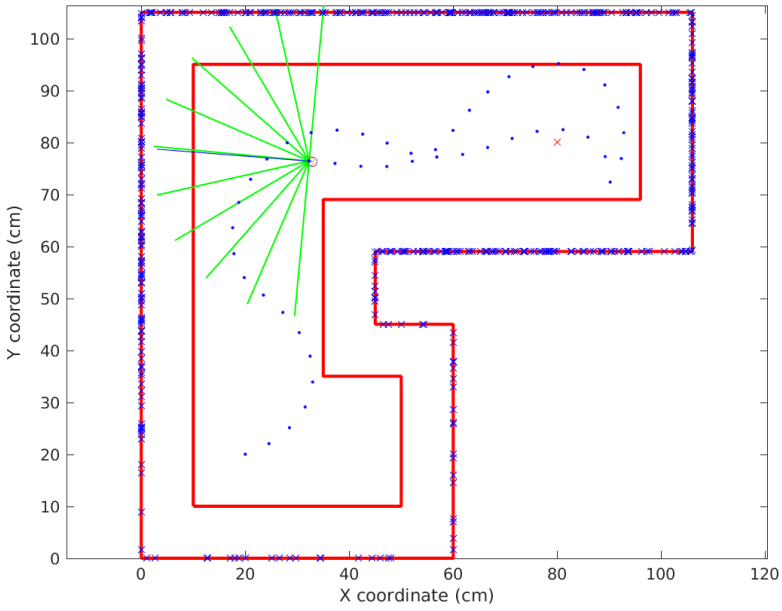


Figure 4: Combined exploration (artificial potential field with $\epsilon = 0.2$ and collision avoidance), the former explores the space, a latter makes sure that the robot does not collide with the wall in any circumstances.

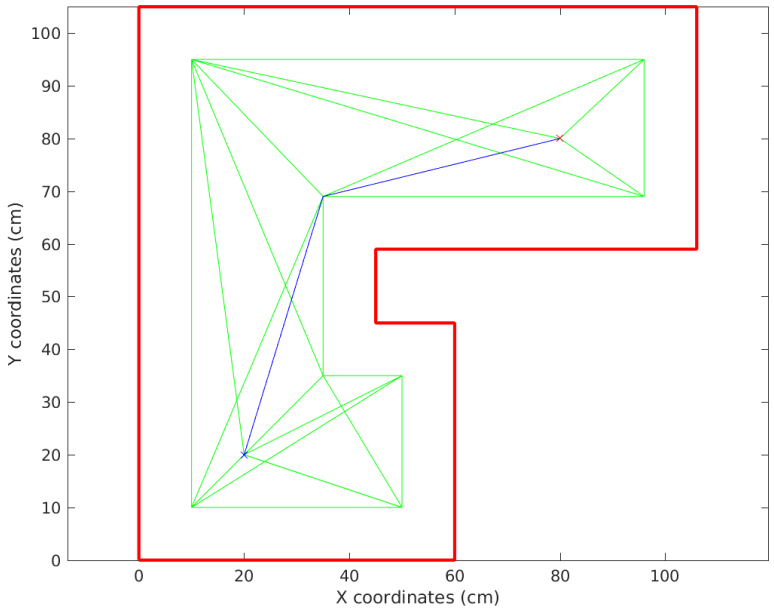


Figure 5: A* search on the visibility graph finds the shortest path (blue line) between the robot and the target (blue and red cross respectively). The modified map is used, so the robot does not go closer to the walls than 10cm.

5 Results

5.1 Simulation

5.2 Experiments

6 Discussion

References

[1] Howie M. Choset, ed. *Principles of robot motion: theory, algorithms, and implementation*. Intelligent robotics and autonomous agents. Cambridge, Mass: MIT Press, 2005. ISBN: 978-0-262-03327-5.

[2] *Introduction to A**. http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html. (Visited on 04/16/2016).

Appendix