Robotic systems
Report

Sergio Castillo
Vince Jankovics
Mike Lagunes
Kevin Zhou

1 Introduction

2 Robot Design

During the first stage of the design one beta version using caterpillar wheels and the sensor on the top to turn 360° was created finding some disadvantages that were solved in the last version.

The current version of the robot was developed based on the findings found on the first version and contains the next advantages:

Compact design. The robot uses the motor pieces as structures to support the NXT control, the decision to place the control vertically was to reduce the length o

Tail to turn.

Sensor in the front.

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 1). 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).

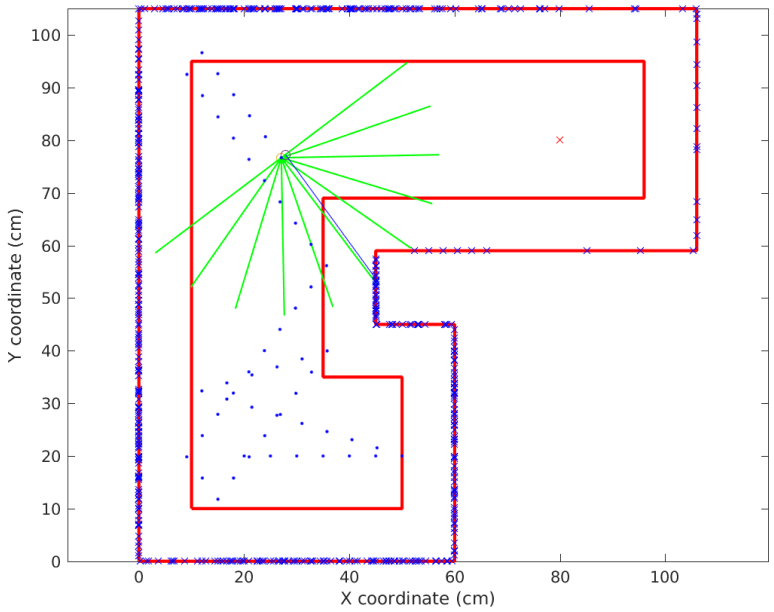


Figure 1: 'Wall bouncing', with a distance threshold of 10cm (inner boundary).

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 2.

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

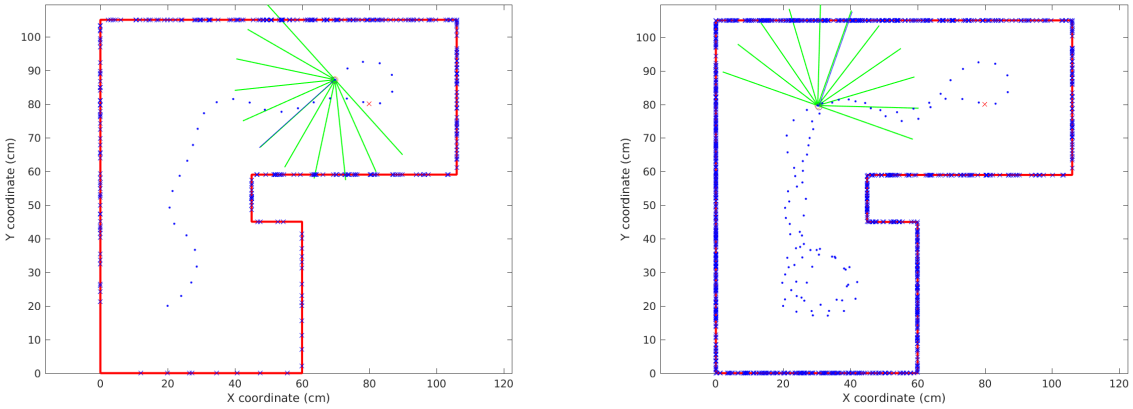
$$\delta\theta = \epsilon \cdot \arg(-\nabla U), \tag{1}$$

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 3.

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



(a) (b)

Figure 2: Simulation with artificial potential field as guidance ($\epsilon = 0.3$). (a) Initial path. (b) Path after some time.

is greatly reduced and the movements are smoother compared to a grid splitting of the map.

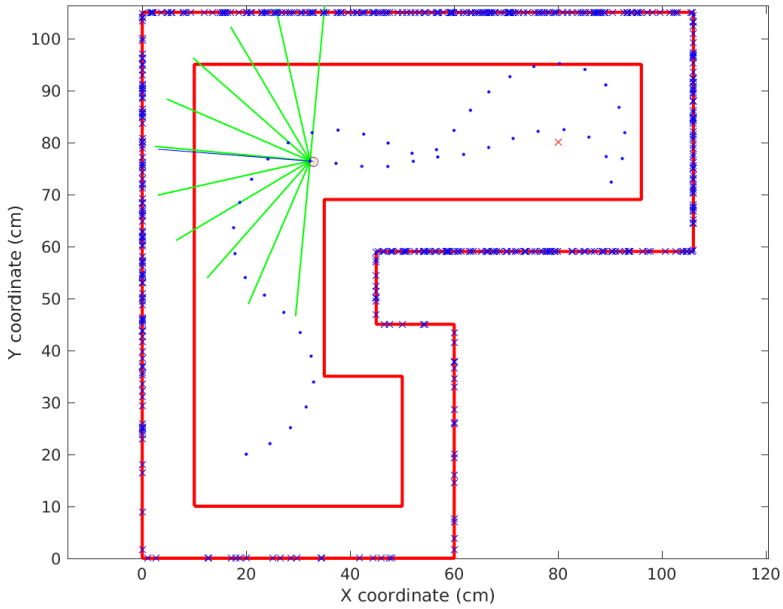


Figure 3: Combined exploration (artificial potential field with $\epsilon = 0.2$ and collision avoidance), the former explores the space, a latter makes sure that the robot does not collide with the wall in any circumstances.

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