Investigator: Luke Ross
Supervisor: Dr Karen Bradshaw

Fiducial Marker Navigation for Mobile Robots

Overview

- Project Goals
- System Design
- Video Demonstration
- Results
- Limitations
- Future Work
- Questions?

Project Goals

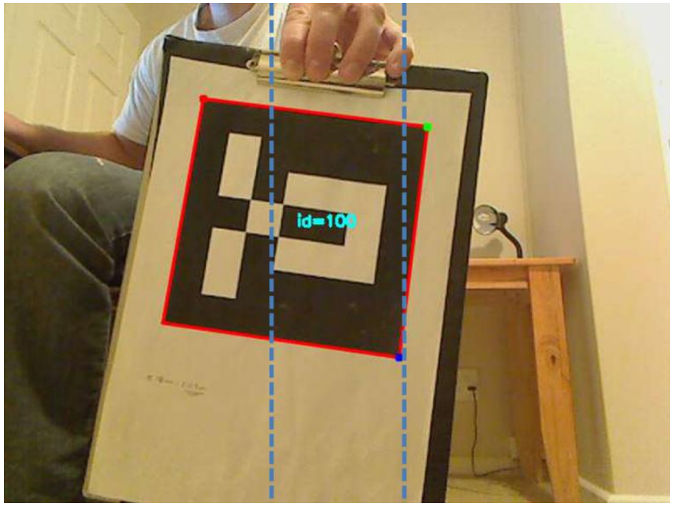
- Accurate detection of fiducial markers using a standard web camera
- Navigate Wifibot using a fiducial marker
  - In a straight line
  - With turning involved
  - Through various layouts of obstacles
- Good performance - suitable for less powerful robots

System Design – Marker Tracking

ARToolkit was first chosen for marker detection, but proved problematic. Therefore ArUco (based on OpenCv) was used instead.

Robot movement:

- When the marker is detected in front of the robot, the robot moves forward
- The webcam is panned to keep the marker in view
- Wifibot turns to face the marker once the webcam is panned by a certain amount
- Wifibot is stopped once the marker is detected within 750mm



System Design – Obstacle Avoidance

Search algorithms researched:

- A*
- D* (Original, Focussed and Lite)
- Generalized Fringe-Retrieving A*
- Moving Target D* Lite

Search algorithm criteria:

- Must be able to calculate the optimal path from the robot to the marker, avoiding obstacles
- Must function in known, partially known and completely unknown environments
- Capable of functioning effectively with a moving target
- Must be efficient

Moving Target D* Lite initially chosen!

System Design – Obstacle Avoidance

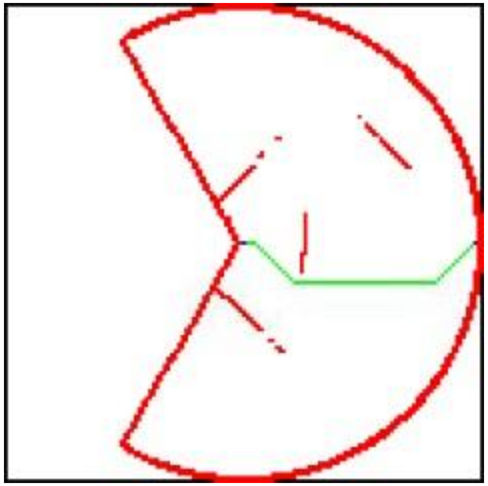
Inaccurate odometer readings → HUGE PROBLEM!!!

Two algorithms, designed:

- Uses simple mathematics on the data captured from the lidar to make decisions about the direction to move in - UNSUCCESSFUL
- A* is used to replan the path to goal every few ms - SUCCESSFUL

Obstacle avoidance algorithm consisted of the following steps:

- Reading from the lidar
- Building the occupancy grid
- Updating the map images with obstacle data
- Calculating the path
- Updating the grid and images with path information
- Deciding in which direction to move the robot



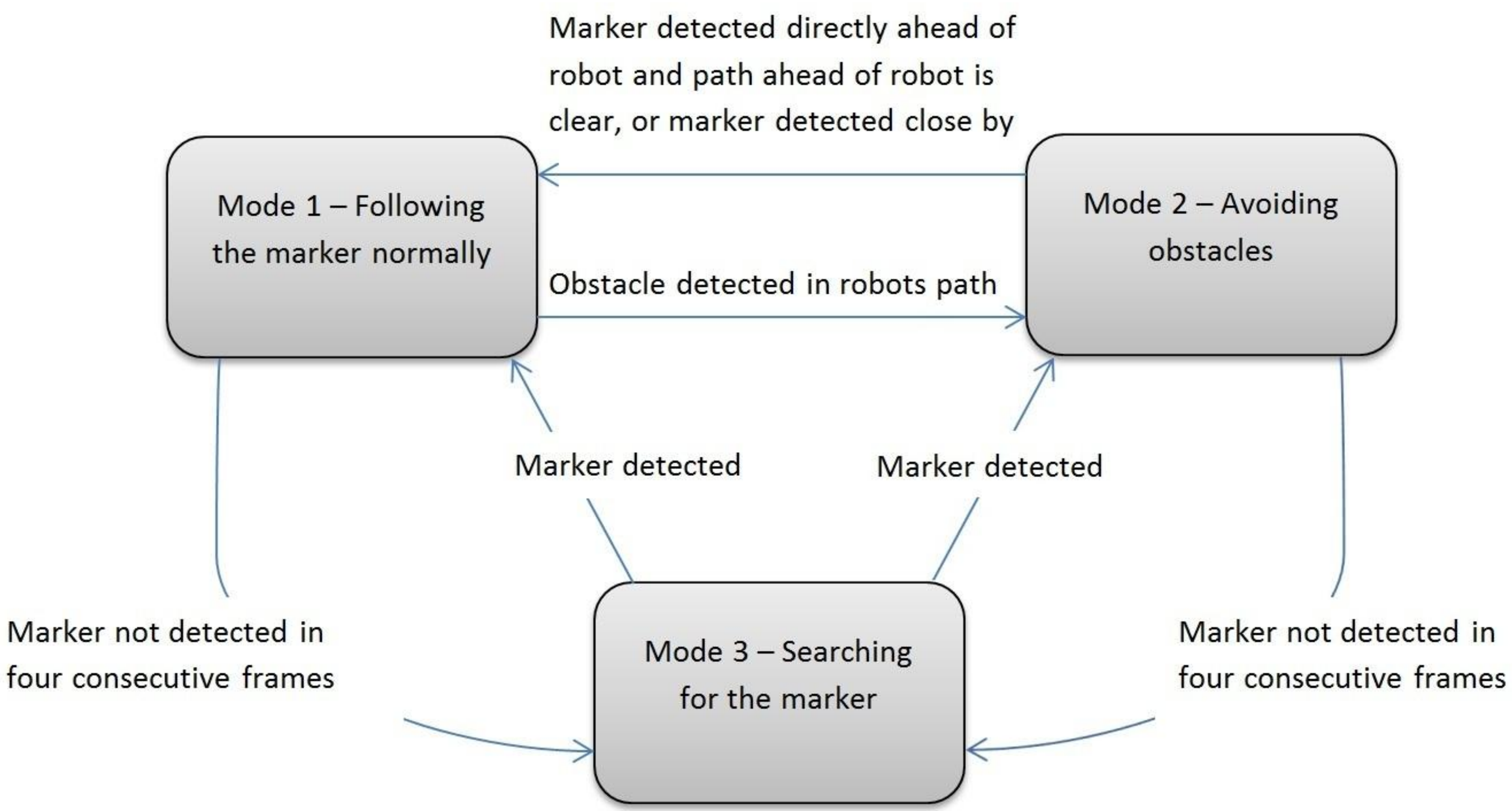
System Design – Searching

There are many options for implementing a wandering algorithm, BUT cannot rely on odometry

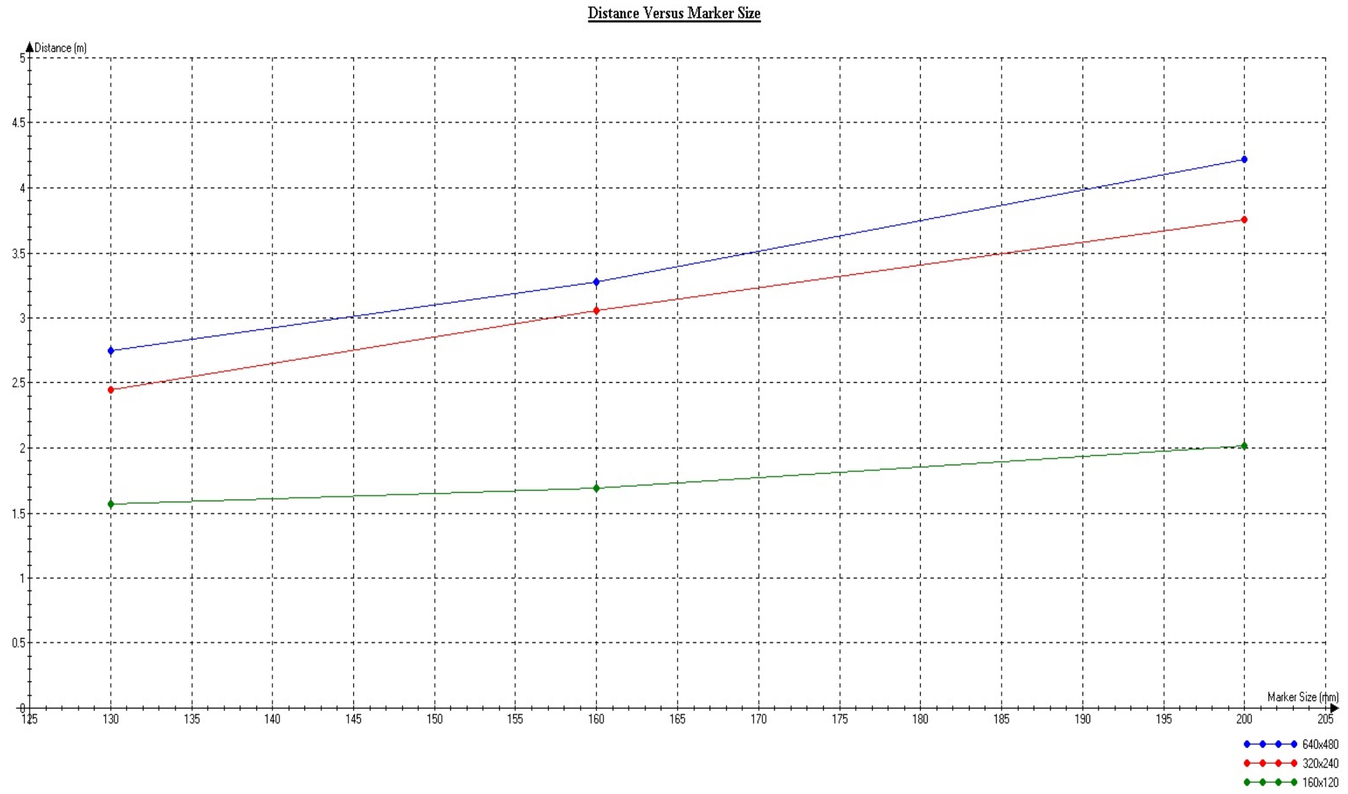
Instead, a simple 3 step process is repeated:

- Wait 5 seconds
- Pan camera, observing surroundings
- Turn robot in a full 360 degree circle

System Design - Mode Switching



Results – Marker Detection

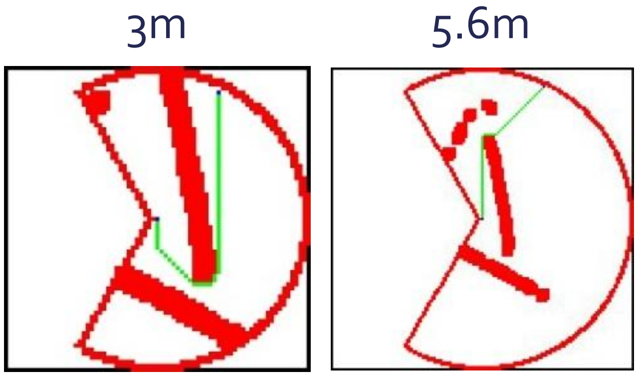


Resolution	Time (*seconds*)	Frames Per Second
640x480	90.51	11.05
320x240	33.34	29.99
160x120	33.32	30.01

Results – Occupancy Grid Design

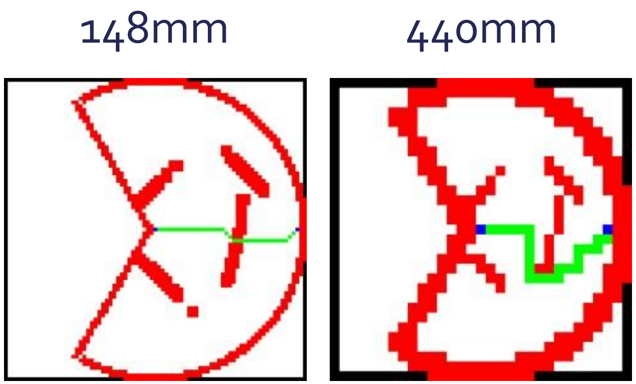
- The area that the grid represents

Distance (mm)	Total Cells	Test 1 Ave (ms)	Test 2 Ave (ms)
5588	131x131=17161	145.1	248.8
3036	73x73=5329	33.8	135.9
1012	27x27=729	1.6	102.6



- Grid resolution

Cell Size (mm)	Total Cells	Test 1 Ave (ms)
88	131x131=17161	145.1
148	79x79=6241	34.4
440	29x29=841	1.6

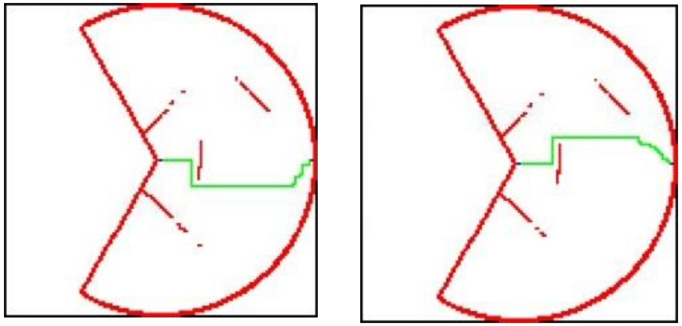


Results – A* Design

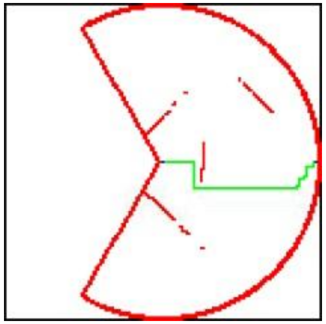
- The number of adjacent cells considered and the heuristic used

Cells Considered	Heuristic	Test 1 Ave (ms)
8	Chebyshev Distance	145.1
4	Manhattan Distance	19.2
4	Euclidean Distance	5.7

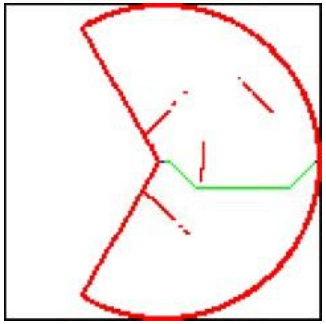
Excessive Jumping



4 Cells

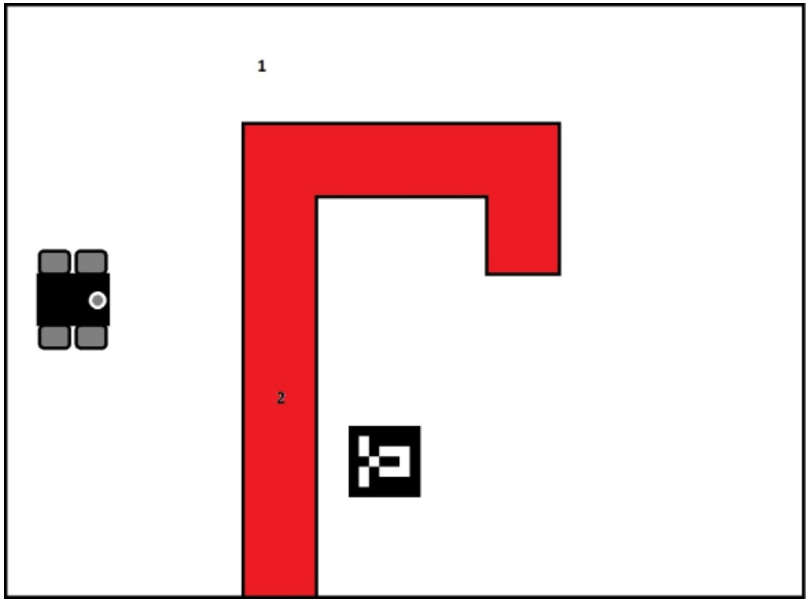


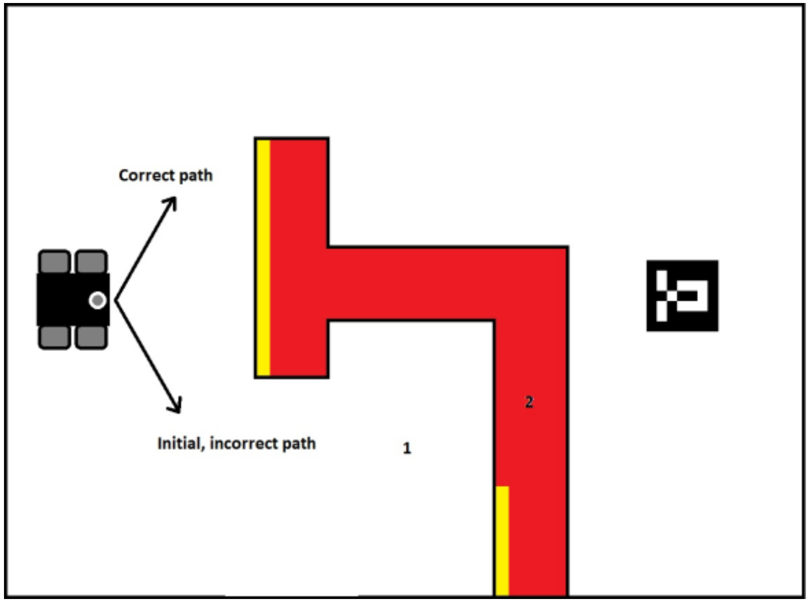
8 Cells



Limitations

Unsuccessful layouts:





Future Work

- Solving the inaccurate odometry issue:
  - Using a different robot
  - Adding two trailing wheel encoders onto the current robot
  - Adding a single wheel encoder to the current robot as well as a digital compass

  The search algorithm, perhaps now changed to Moving Target D* Lite, can be implemented in the manner it was intended, resulting in a huge performance increase.

- Allowing for marker detection at any angle:
  - Using a camera with 360 degree panning capabilities
  - Using multiple cameras

Questions?