# Map Construction Technique

The program creates a real-time map using the python matplotlib graphing library, the map updates as the robot moves and more data is received. The map has three main components:

* Robot path line graph (blue)
* Sonar point scatter graph (red)
* Occupancy grid image (grayscale)

The robot path line graph represented in blue shows the path the robot has taken. This is not a markable portion of the assessment, and so will not be explained in further detail.

The sonar point scatter graph represented in red displays all sonar points detected, the implementation of this section is explained in section 2.

The occupancy grid image represented in grayscale shows the probability of an object existing inside each given pixel. 1 represents a low chance of an object in the pixel, 0 represents a high chance of an object in the pixel, the default value is 0.5 when it can not be determined if the pixel contains an object. The scale with 1 as low and 0 as high was chosen so that empty pixels are white and pixels with objects in are black.

# Software Implementation

## Sonar point scatter graph

The scatter points are stored in two one dimensional arrays, each storing one set of ordinates. The points are relative to the robots starting point (0,0) this causes the output map to rotate depending on how the robot is started the map is rarely orientated correctly, although through simple coordinate manipulation the output could more closely represent the input.

## Calculating absolute positions

The absolute position for each sonar reading is calculated from the angle of the sonar, distance, and the robots current position. First the sonar absolute angle is calculated, this is the angle between the robot and the sonar reading relative to (0,0) as appose to the robots heading.

The absolute angle is then used to calculate the x and y ordinates of the distance of the sonar relative to the robot.

The relative ordinates are then used to calculate the absolute position of the sonar reading

## Occupancy grid image

The occupancy grid is stored in memory as a two-dimensional list with all values between 1 and 0, representing a grayscale image. To display the occupancy grid image on the figure, all ordinate values must be positive, images are displayed from (0,0) to (x, y). To achieve this the entire grid system must be offset (see section 2.4).

Whenever a sonar reading is collected each pixel is checked to see if it within the tolerance area (see sections 2.5 and 2.6) the relevant value is then assigned to it. A major downfall of this method is this requires a lot of computational time, this could cause a problem if running on an older robot model.

The occupancy grid changes size with the map, instead of having a fixed size (e.g. 100x100) the occupancy grid has a fixed resolution the benefit of this method is that the grid is always accurate irrelevant of the map size.

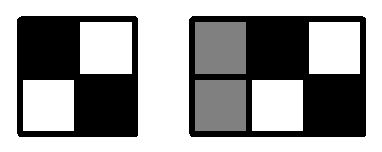
## Offsetting

Whenever a sonar reading with a negative value is detected the entire grid must be shifted to keep (0,0) as the lowest point. This must be done so that the occupancy grid image can be displayed.

### Offsetting ordinate lists

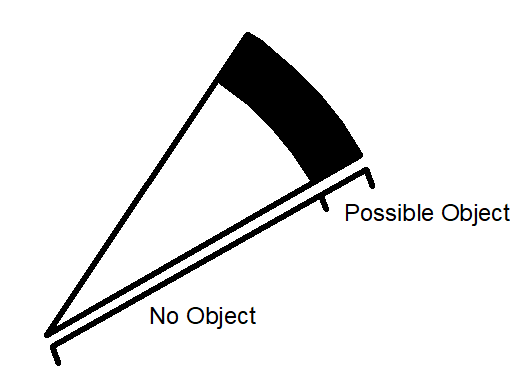
Offsetting ordinate lists is relatively simple, first the value of the sonar reading is added to all elements of relevant ordinate lists. The new offset is then saved so that all future inputs can be offset when they are read.

### Offsetting the occupancy grid

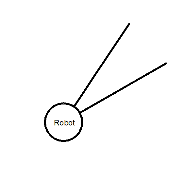


Offsetting the occupancy grid is more difficult, the entire grid must be padded with the default 0.5 value, this is required whenever a new negative sonar reading is detected, or whenever a new sonar reading that is outside of the occupancy grid is detected.

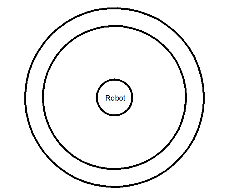
## Occupancy tolerance areas

The occupancy grid is filled with information in areas, sonars are inaccurate therefore there is a chance of the obstacle existing near where the reading was detected. The program uses a cone to fill areas, an inaccuracy angle and distance is used to find both areas. The possible object area is bounded by both sides by the inaccuracy angle and both ends by the distance from the robot, this area is tended towards 0. Tending the value towards 0 ensures that previous readings are also considered. There also exists a no object area, because a sonar reading was detected it can be assumed that there is no object between the robot and the sonar, therefore the area shown can be tended towards 1. If the sonar reading is the maximum reading for the pioneer robot sensors there does not exist a possible object area, but the entire cone can be assumed to not contain an object.

## Calculating if pixel is within sonar tolerance

There exist two tests to check if a pixel is within the sonar tolerance. First it is determined if the pixel is within the angle tolerance this is done with the following formula:

The second test determines the if the pixel is within the sonar distance this uses the following formula:

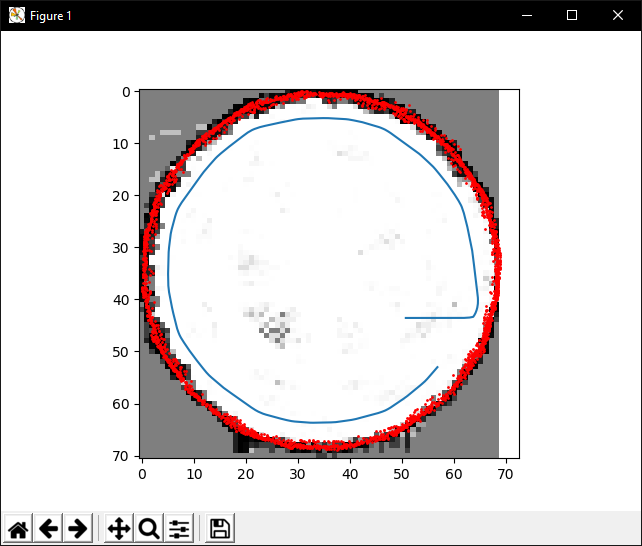
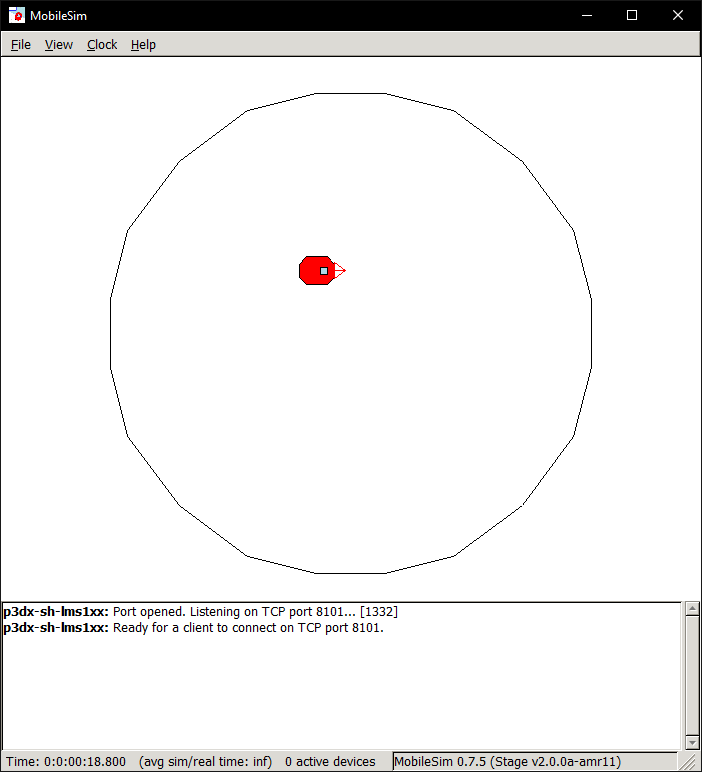


Together these two tests determine if the pixel is within the sonar tolerance and which area it is in.

# Testing

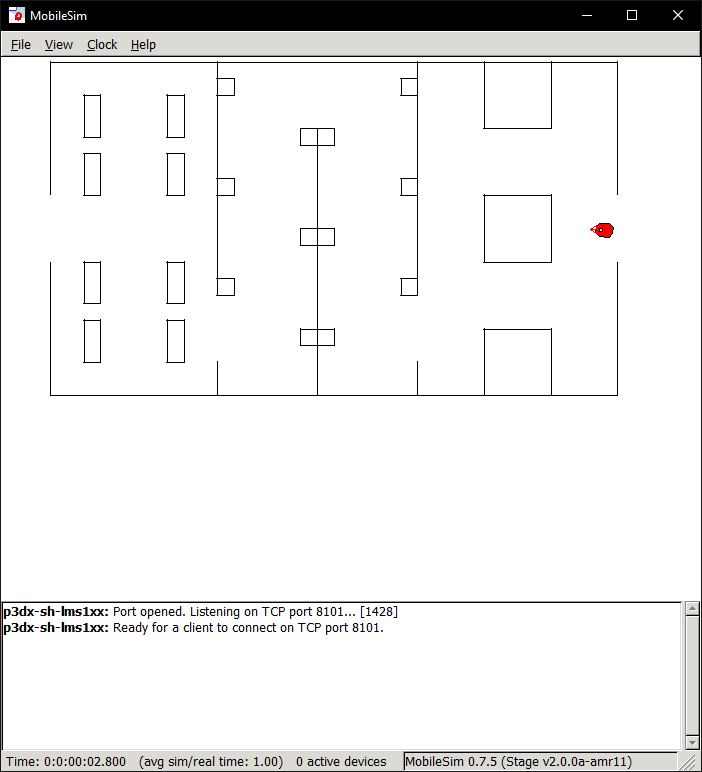
Map: Circle

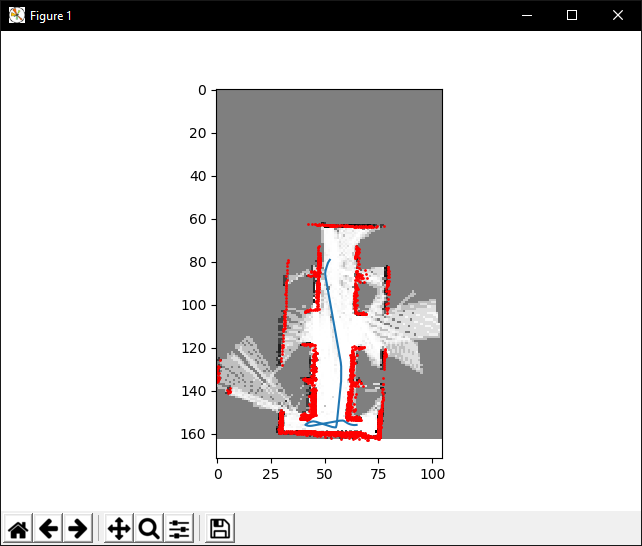
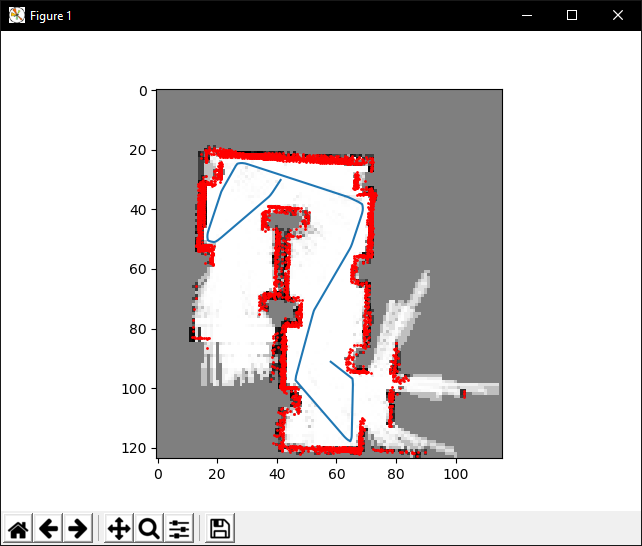
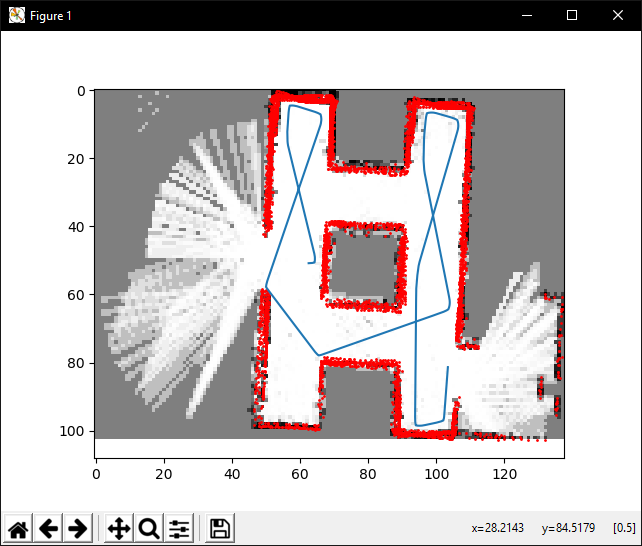
The robot circles around the edge of the map, the output map accurately reflects that of the input map, some inner pixels are still grey, however over time these will tend towards white



Map: Mine

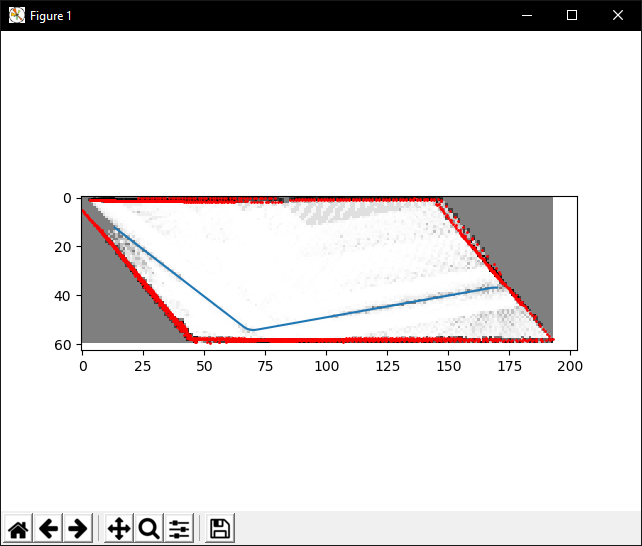
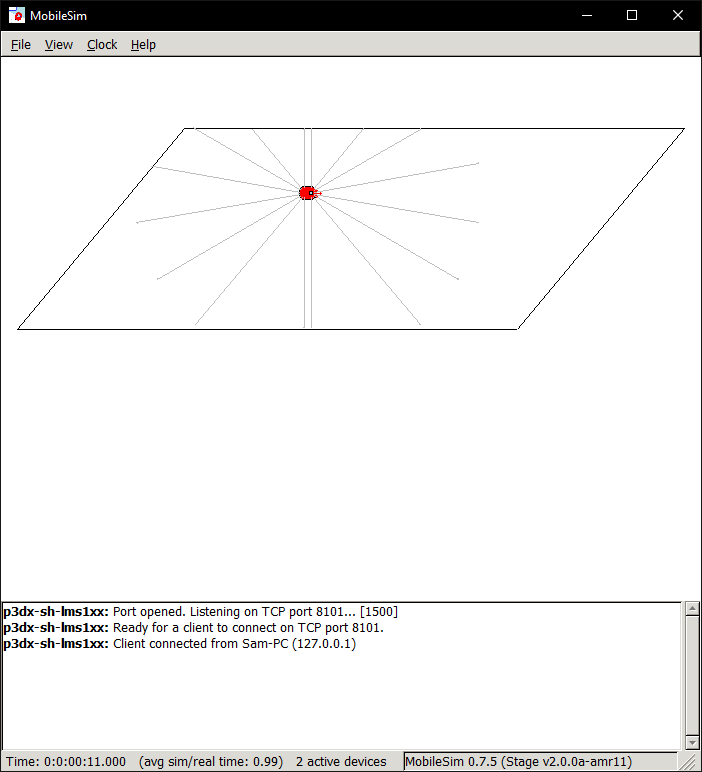
The robot does not navigate the map very well and was manually moved a few times, the occupancy grid is created of the path followed, it is interesting to see the white lines of the grid trail out of the door.





Map: Parallelogram

The robot navigates all of the parallelogram and a clear map is produced extremely quickly.



Map: Map1

The robot navigates the map and produces the following output, a video of this has been included in the submission. An interesting effect happens here, when the robot moves out from behind the smaller square, its sides are parallel to the robot, if a sensor picks up the reading of the far wall then because of the angle tolerance all of the small square wall cells are coloured white. This is an error with the tolerance values, in this case it is hard to fix: smaller angle tolerances will lessen the effect but also dramatically increase the amount of time taken to fill out the occupancy grid.

