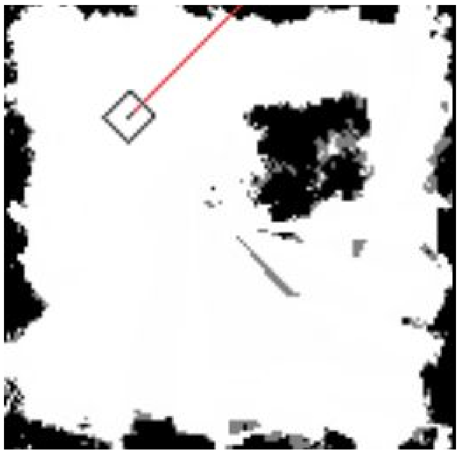
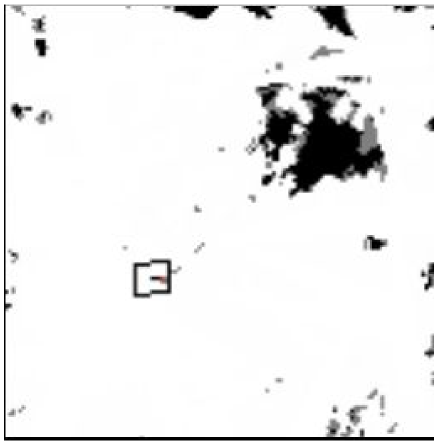
Homework 3: Mapping

i. Modify the get_first_hist() function in ObstacleMap so that it simulates rangefinder noise. You can use uniform noise on the range [-n,n] or you could use Gaussian noise (see the numpy.random module). Is the mapping algorithm able to cope with this noise? What happens as you increase the noise?

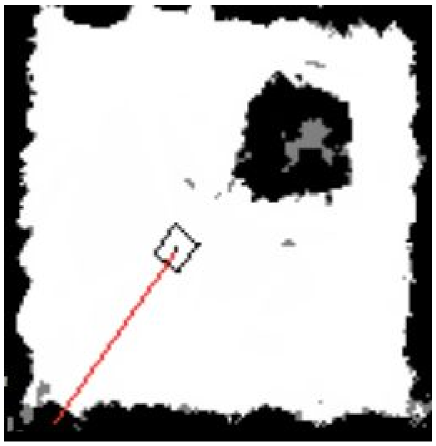
We used Gaussian noise using the function numpy.random.normal(0, 10, 1) which means that the random value generated has a standard deviation of 10 centered at 0. Adding this noise makes the robot detect objects further, closer or in the real position in the map. The mapping algorithm is able to cope with this noise if the robot scans the zone several times because it does the mean and we are using a Gaussian noise. We can see this in the following image:



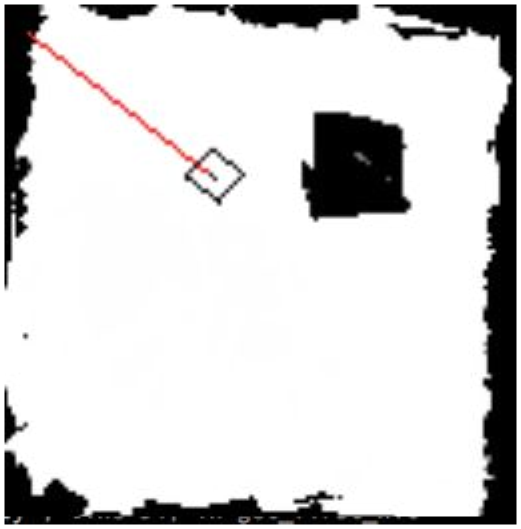
If we increase the noise the map gets very dirty. In the following image, it was tested with a standard deviation of 20:



If we decrease the standard deviation to 5 we can see that we have better results:

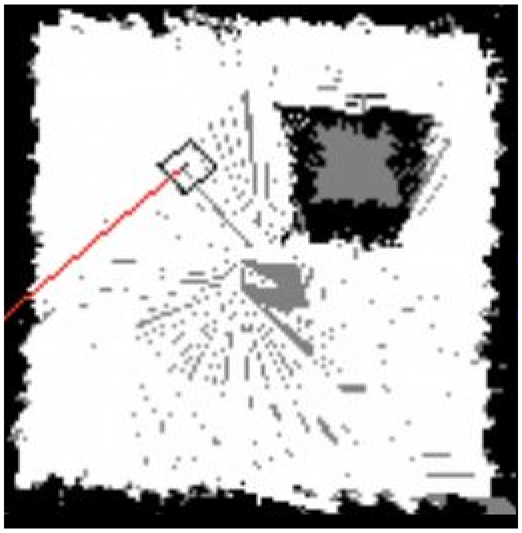


Also if we go through the map a lot of times as the noise is gaussian we will get at the end a cleaner map like the following:

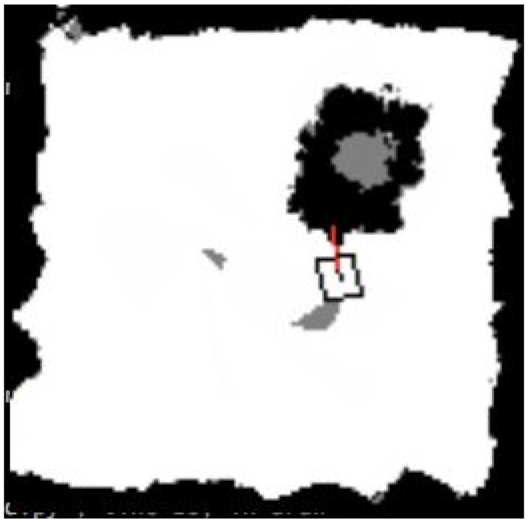


ii.Test different cone widths and obstacle thicknesses for the rangefinder model. How does this affect the mapping results?

All the tested above was with a cone of 10. With a small cone of 4 degrees:

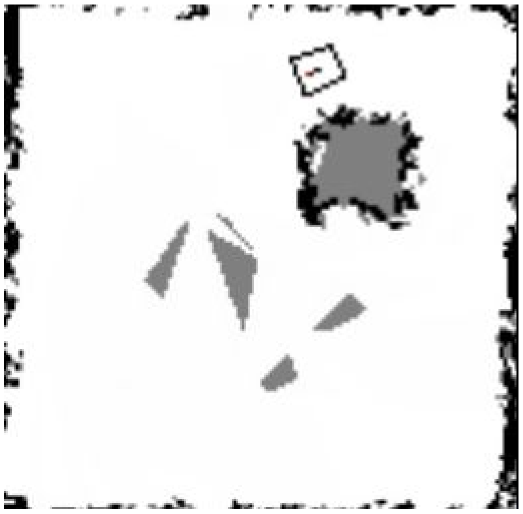


With a big cone of 20:

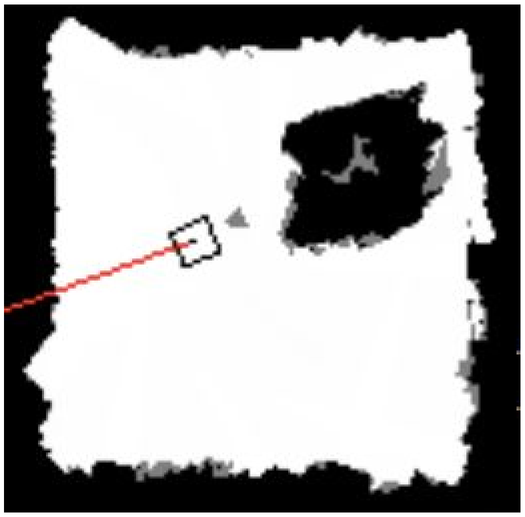


Our conclusions are that is easier to represent obstacles if the cone is bigger, but is not that precise as small cones. So a good option is something in between, like 10 degrees. So the following tests will be with that cone width.

All the above was tested with a thickness of 10 cm. If we try a thickness of 2 cm:



With a thickness of 20 cm:



We can see that if we use a big thickness is easier to represent the obstacles very fast but the obstacles will seem bigger, which is good in order to not crash with them. If the thickness is smaller then we will know more precise where the objects are but we have to run the robot a lot of more time to register the full obstacle. So we have to find an intermediate cone width and thickness in order to have the best representation.

Iii. Try your mapping system using the real robot. Does it seem to work? Why or why not?

We tried it with the real robot and the simulation rotates much faster than the real one so the results are not accurate. Also there is a lot of noise and sometimes prints and sometimes it doesn't. The range finder started pointing to random directions and it didn't detect correctly what it had in front of it.