Assignment 2
Making a range finder using an omni-directional camera

Maarten de Jonge
Inge Becht

November 14, 2012

This assignment is about making a range finder using an omni-directional camera on a lego robot. Because most of the code was already given in the assignment, this report will mostly consist of the experimenting with values for different variables to create the best possible mapping of the camera visualisation and the real world. In the end the error will be visualised using the best possible calibration settings and variable values found.

1 The steps towards simulating a range finder

The first step towards creating the final map is calibrating the camera. For this purpose thei script `calibrate_camera_offline.m` was modified to work with a single picture taken from the omnidirectional camera, as shown in figure **??**

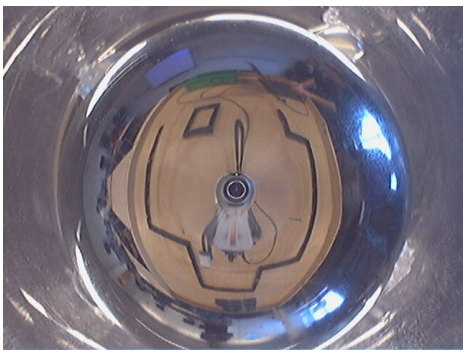

Figure 1: A snapshot from the omnidirectional camera

calibration values were set: $Rmin = 77, Rmax = 160, Centre\_x = 325.2203 and Centre\_y = 225.0$

This created the spaces as can be seen in figure 2, where the space spanned between the outer pink circle to the inner pink circle is the used data in the rest of the report. Figure 3 shows this part of the images straightened out.

To extract the walls from the original image, it needs to be unwarped and all data that does not fall between the two pink lines in 2 needs to be thrown

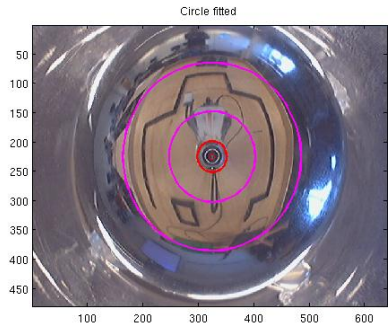

Figure 2: Circle indicating used mirror space

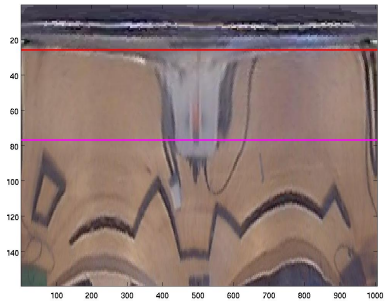

Figure 3: Flipped and straightened image

away. The premade `imunwrap.m` script does exactly this. It takes a new parameter *angstep* which determines the angular resolution of the simulated laser scanner. The bigger the angle the worse the resolution is, as can be seen in figure reffig:angstep0.05 and reffig:angstep5

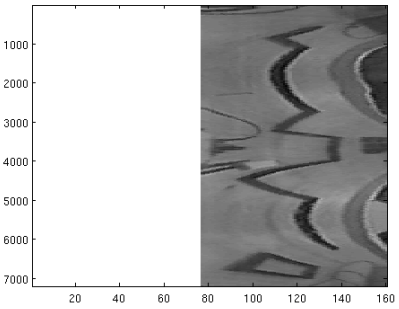

Figure 4: `imunwrap.m` applied with angstep = 0.05

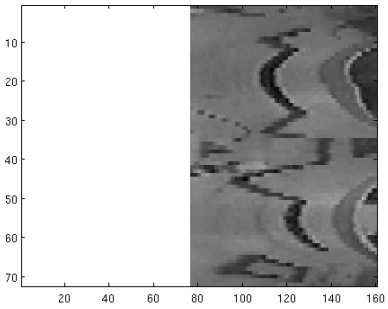

Figure 5: `imunwrap.m` applied with angstep =5

At first one would say a lower angle would be more precise in the funal mapping, but the problem with a lower angle is the great amount of detail that will be kept in the image and thus a great amount of noise that comes with it. A too high angular step, however, could make the distance estimation less accurate. *Angstep* is thus one of the parameters with wich will be experimentated to find the approximate best one.

Whatever the chosen angstep is, there still needs to be some more noise reduction to accentuate the position of the wall pieces. A simple black and white threshold is used for this. A more difficult choice is however at what value to set the threshold, something that can not be determined from a single image, but we still try to approximate it that way. In figure 6 and 7 two different tresholds can be seen. Note that the white values of the pixels have not been normalised. thresholded.

With this constructed image the following step is to iterate through all the white pixels from the left side. Now using the height of the robot(around 0.33

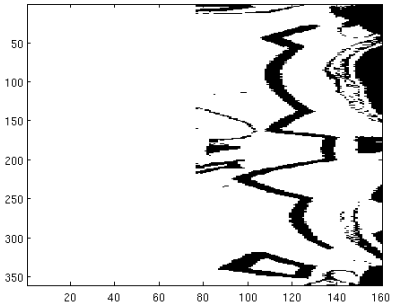

Figure 6: Threshold = 100 applied with angstep = 1

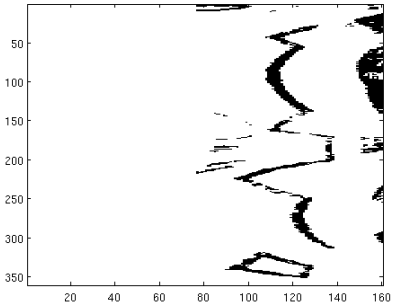

Figure 7: Threshold = 75. Applied with angstep =1

meters) and a certain α the distance towards these black points can be calculated. Different values for α were experimented with. In case of the robot in Zurich, α was given the value of 95 pixels, so we experiment around these values as well.

1.1 Varying parameter values

The data set with all experimentations of angstep, the black and white threshold and α was created by running `create_dataset.m` and can be run with or without calibration. The following different combination of values were used:

$$\begin{aligned} angstep &\in \{0.05, 0.5, 1, 1.5, 2, 5, 10\} \\ BWthresh &\in \{75, 80, 85, 90, 95100\} \\ \alpha &\in \{80, 95, 100, 120, 130, 150\} \end{aligned}$$

1.2 Varying *angstep*

When using $BWthresh = 75$ and $\alpha = 130$ and varying *angstep*, we get figure 8, 9, 10, 11. Although first concerns were given for taking a too high resolution, from this data we can see clearly that taking the smallest angular step tested does not create too much noise for the purpose of line detection. In case of the $angstep = 0.05$, we can identify 5 clear outliers and in the case of $angstep = 2$ only 3 less than this. It seems there doesn't have to be a big trade-off between resolution and accuracy.

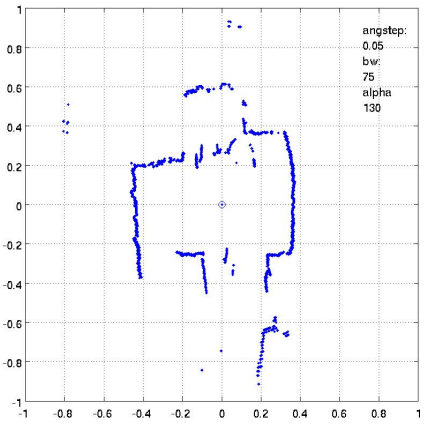

Figure 8: $angstep = 0.05$

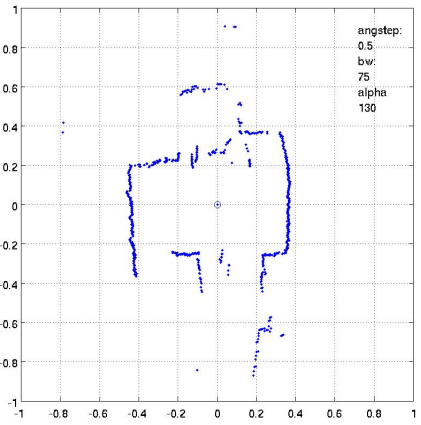

Figure 9: $angstep = 0.5$

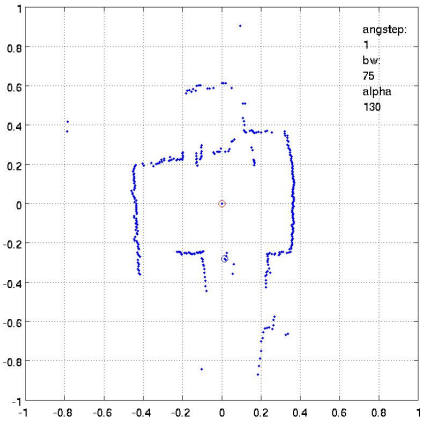

Figure 10: $angstep = 1$

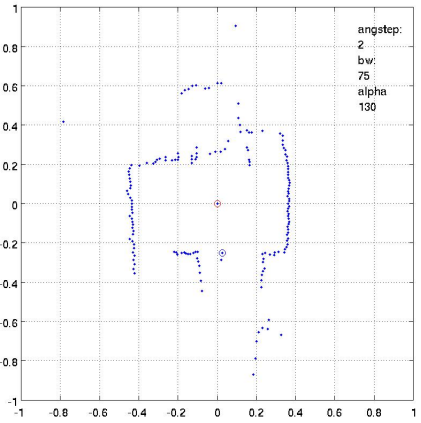

Figure 11: $angstep = 2$

1.3 Varying $BWThresh$

When varying the values of $BWthresh$ and keeping $angstep = 0.05$ and $\alpha = 130$ we get the results seen in figure 12, 13, 14 and 15. It seems that in the chosen threshold range it does not really matter that much what value you use. But as stated earlier, this result should be taken lightly, as it is only constructed with a single image with only one kind of lighting condition

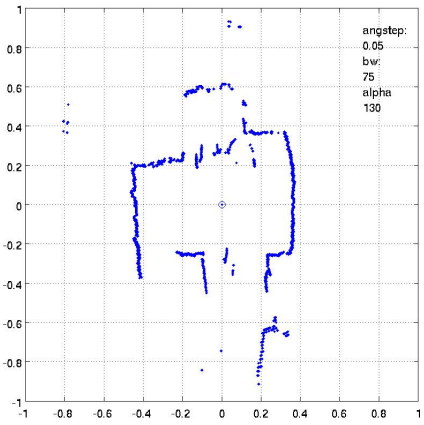

Figure 12: $BWthresh = 75$

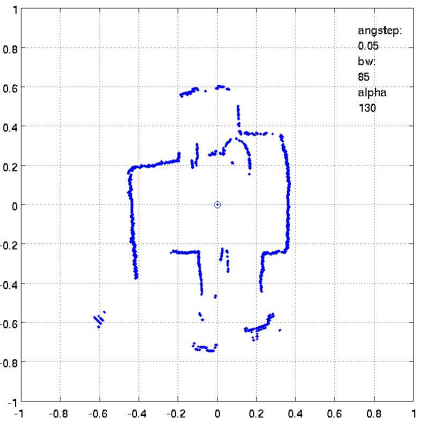

Figure 13: $BWthresh = 85$

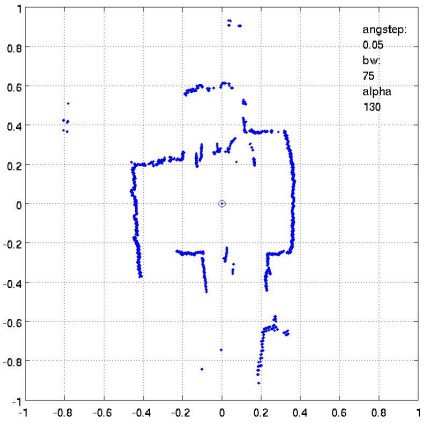

Figure 14: $BWthresh = 90$

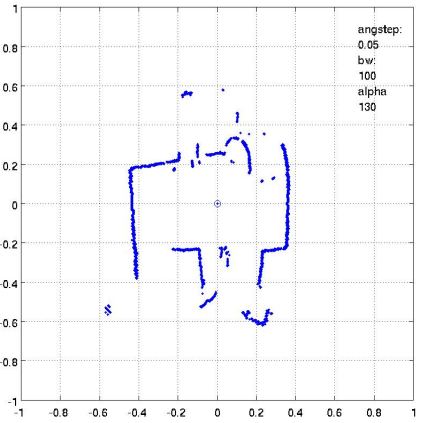

Figure 15: $BWthresh = 100$

1.4 Varying α

When varying the values of α and keeping $angstep = 0.05$ and $BWthresh = 80$ we get the results seen in figure 16, 17 18, 19.These images show that α should not be taken higher than 130 pixels as than the lines begin to deform in the shape of the mirror, which has probably to do with a less than optimal calibration. This would make it harder to do edge ore line detection. Also, the parameter should not be chosen less than 120 pixels as than the position estiation is not all that precise.

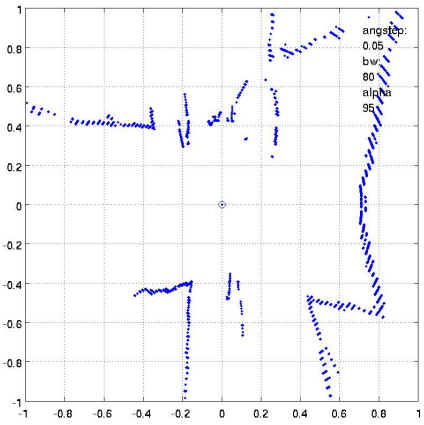

Figure 16: $\alpha = 95$

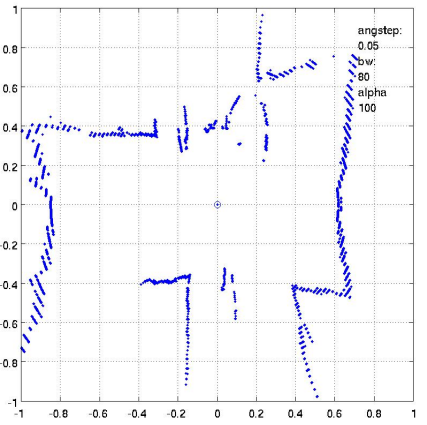

Figure 17: $\alpha = 100$

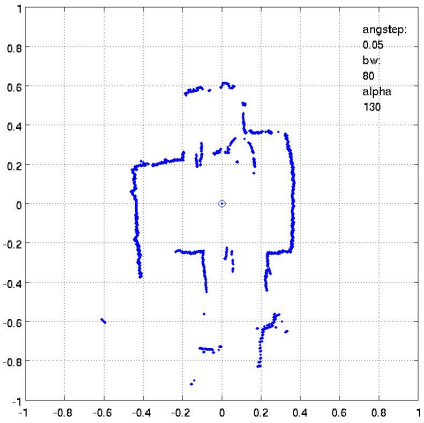

Figure 18: $\alpha = 130$

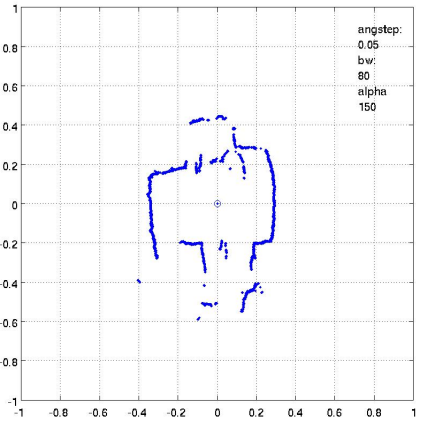

Figure 19: $\alpha = 150$

So now we know that the optimal value for $angstep = 0.05$, for $\alpha = 130$ and for $BWthresh$ it does not matter. This can be used for calculating the error using function `compute_uncertainty`.