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

Maarten de Jonge
Inge Becht

November 15, 2012

This assignment was 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, the report mostly consists of experimenting with values for different variables to create the best possible mapping of the camera visualisation and the real world. In the end a map of the environment will be shown using the parameters that were found to be optimal.

1 The steps towards simulating a range finder

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

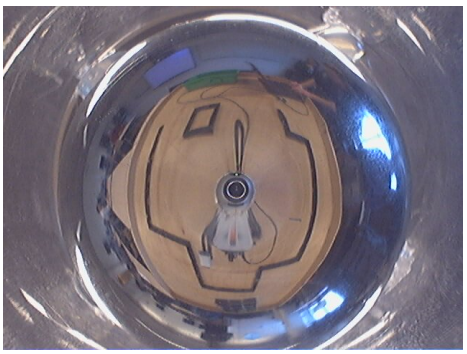

Figure 1: A snapshot from the omnidirectional camera

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

This created the areas seen in figure 2, where the area between the outer pink circle and the inner pink circle is considered 'useful information'. Figure 3 shows the same image, except unwrapped from the center.

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 away.

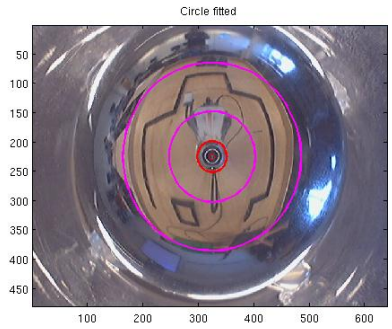

Figure 2: Circle indicating used mirror space

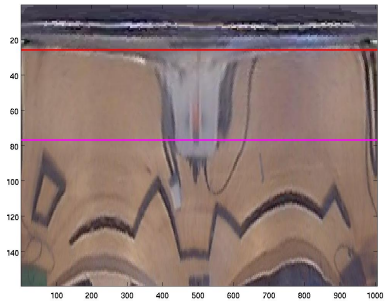

Figure 3: Flipped and straightened image

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 4 and 5.

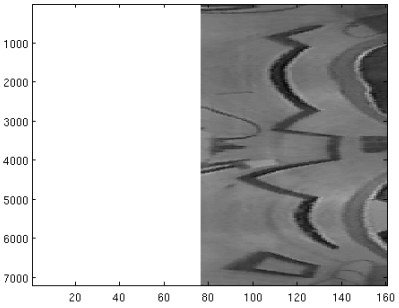

Figure 4: `imunwrap.m` applied with angstep = 0.05

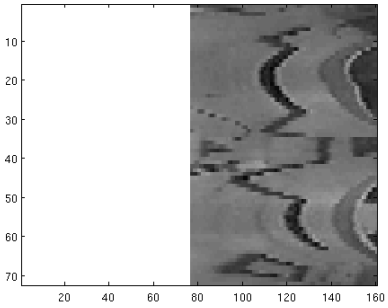

Figure 5: `imunwrap.m` applied with angstep =5

At first glance one might say a lower angle (thus, higher resolution) would lead to more precision, but the problem with a higher resolution 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 low resolution, however, could make the distance estimation less accurate. *Angstep* is thus one of the parameters with wich will be experimented with to find a suitable value.

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 range of of the input grayscale image is not normalised between 0 and 1, but instead 0 and 255, leading to the threshold values of 100 and 75.

The following step is to iterate through each horizontal line to scan for

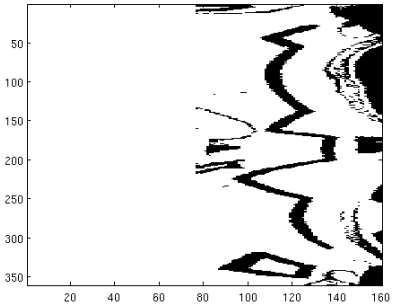

Figure 6: Threshold = 100 applied with angstep = 1

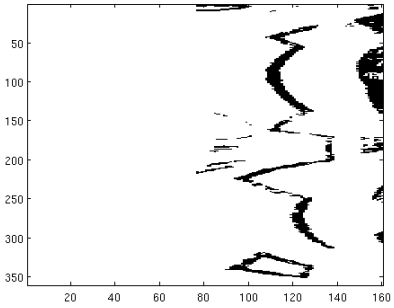

Figure 7: Threshold = 75. Applied with angstep =1

transition from white to black. Using the height of the camera (around 0.33 meters) and a certain α the distance towards these transitions 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 tested values 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 we expected a high resolution to add noise, in this particular testing setup it's not problem. 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 might not have to be a big trade-off between resolution and accuracy.

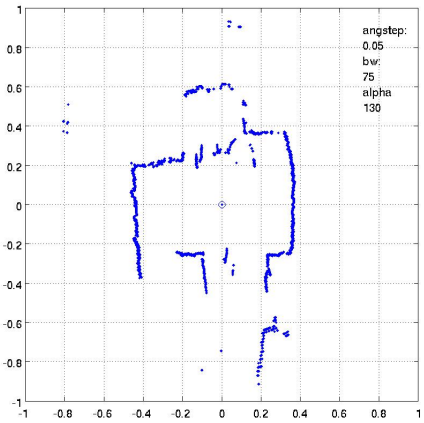

Figure 8: $angstep = 0.05$

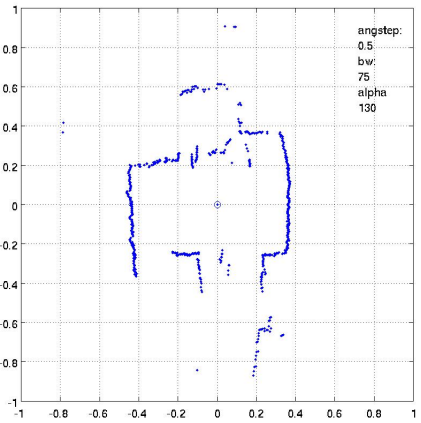

Figure 9: $angstep = 0.5$

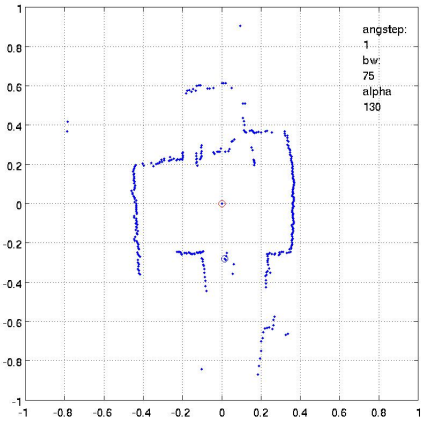

Figure 10: $angstep = 1$

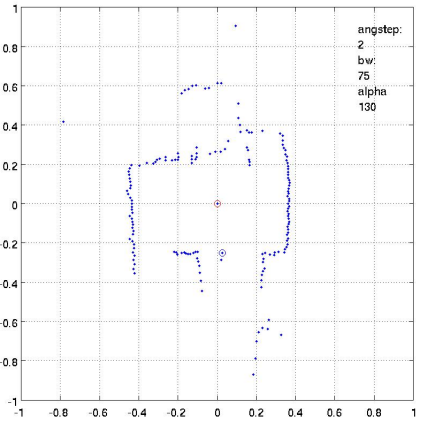

Figure 11: $angstep = 2$

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. Depending on the final use of the images a better threshold should be considered

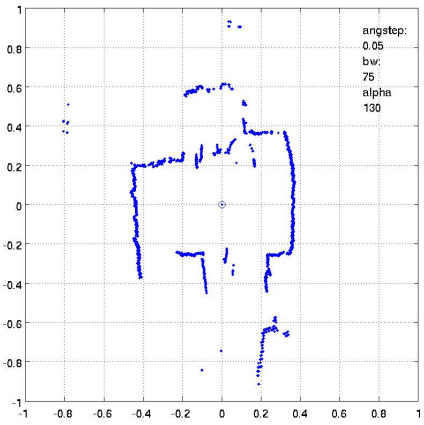

Figure 12: $BWthresh = 75$

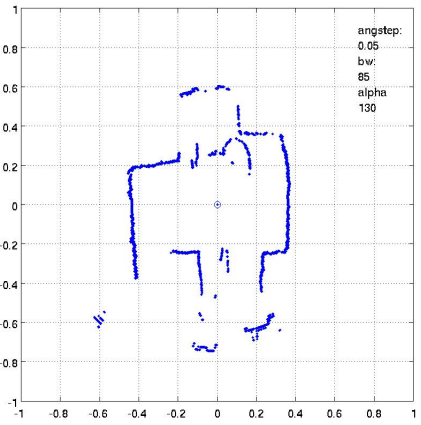

Figure 13: $BWthresh = 85$

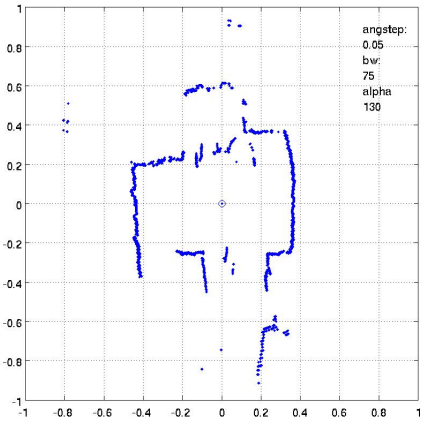

Figure 14: $BWthresh = 90$

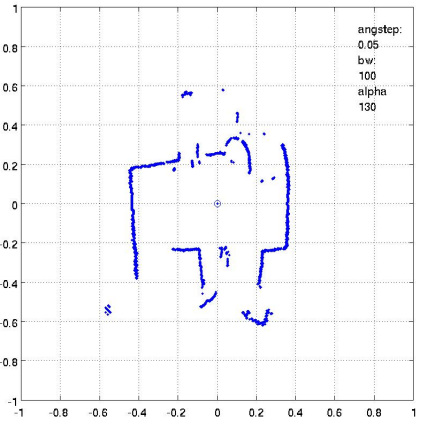

Figure 15: $BWthresh = 100$

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 or line detection. Also, the parameter should not be chosen less than 120 pixels as than the position estiation is not all that precise.

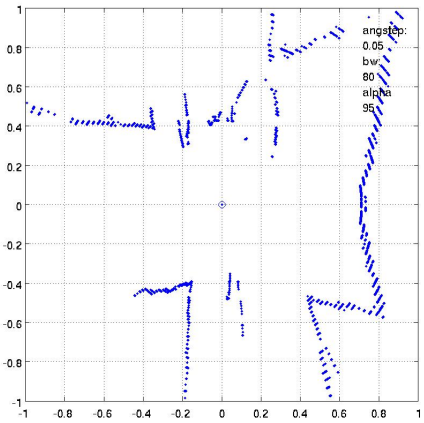

Figure 16: $\alpha = 95$

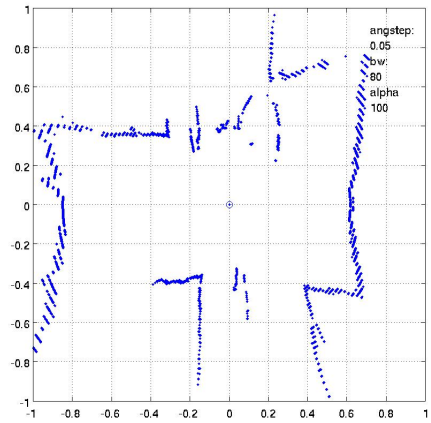

Figure 17: $\alpha = 100$

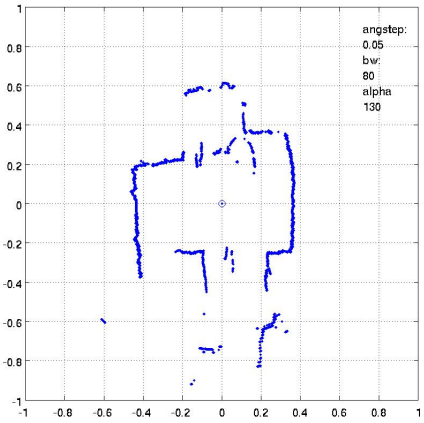

Figure 18: $\alpha = 130$

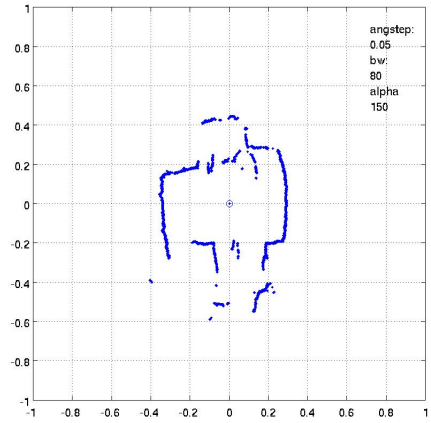

Figure 19: $\alpha = 150$

1.5 Calculating the error

So now we know that the optimal value for $angstep = 0.05$, for $\alpha = 130$ and for $BWthresh$ it does not matter in the tested threshold. This can be used for calculating the error using function `compute_uncertainty`. The final result of this can be seen in figure 20 in case of the optimal parameters and in figure 21 in case of less than optimal parameters. We are not sure why the camera position creates such large errors in case of figure 21. It does not have to do with camera calibration, because it never happens in case of good parameter choices.

the case with ...

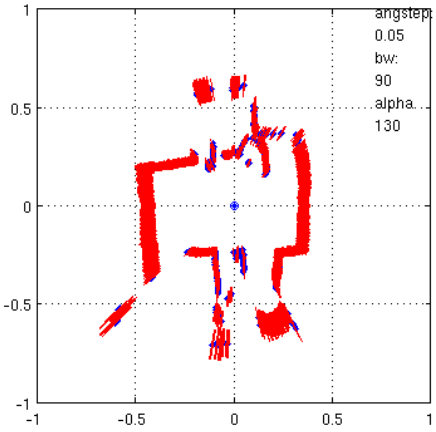

Figure 20: The image showing the distance error in case of $\alpha = 130$, $angstep = 0.05$ and $BWthresh = 90$

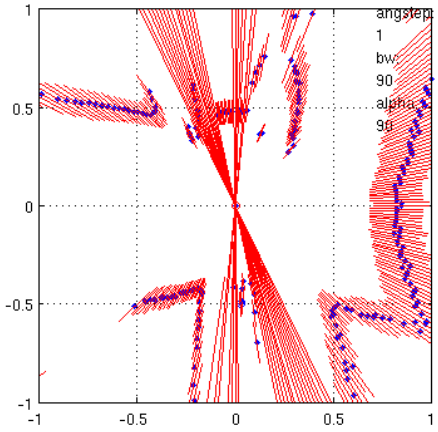

Figure 21: The image showing the distance error of$\alpha = 90$, $angstep = 1$ and $BWthresh = 90$