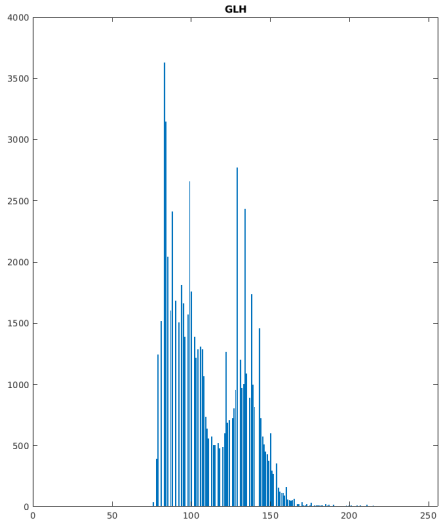
EL844 - Workshop 2 Report

Bastien Dhiver – bfrd2

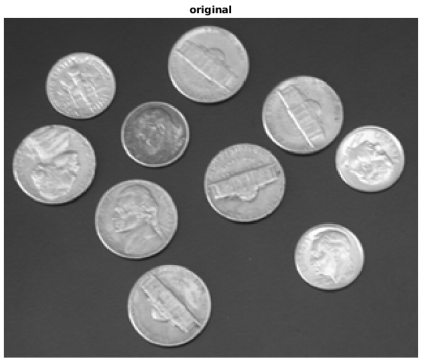
Matlab commented codes are provided with this report.

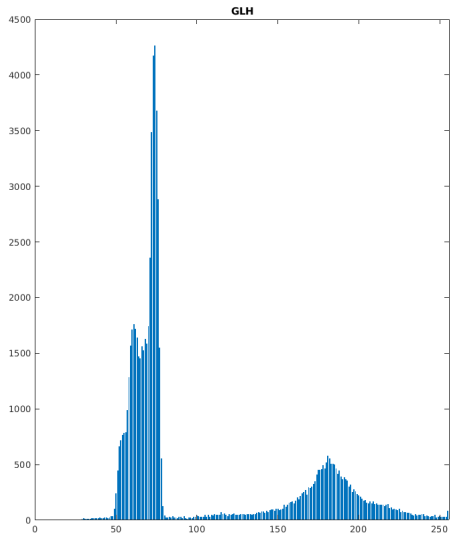
Section A:

The grey level histograms (GLH) have been generated for these images below using the script called "glh_script.m".



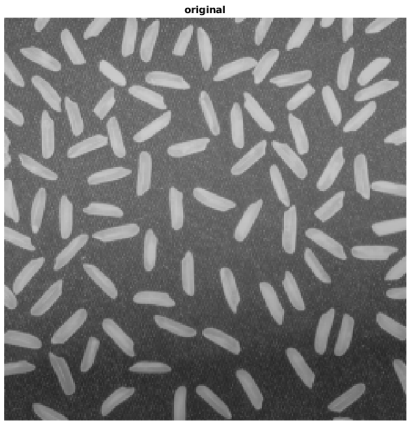
On this black and white picture, the disparity of shades is almost indistinctive which is reflected in a single narrow spike (between 75 and 160).

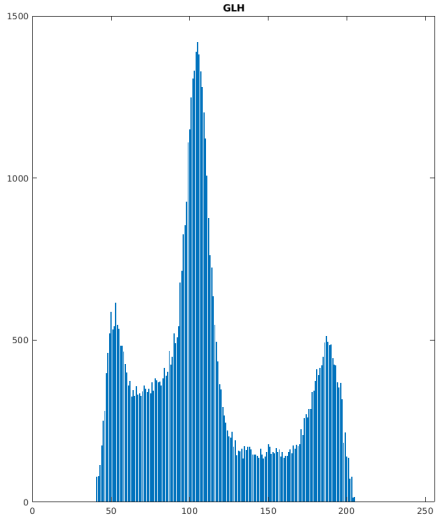




We can clearly see two spikes on this GLH, the taller one on the left (between 50 and 75) is produced by the dark image background, the dark shade is uniform which is why it's reaching heights. The brighter shades that appear

on the coins is represented by the spike on the right (between 150 and 225). It spreads out a little more because the coins textures reflect light differently.



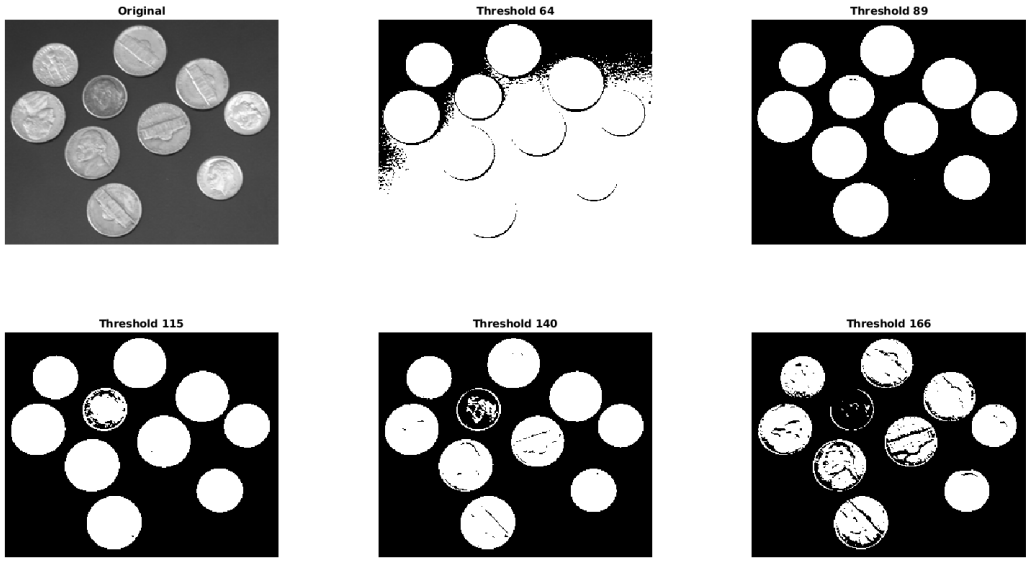


The gradient of light intensity produces a wide histogram (between 50 and 200). We can observe that the dark zone at the bottom of the image is represented by the most left peak. The background appears on the central peak with the highest peak. Finally, the rice grains that appear as the brighter part of the picture are translated into the right peak.

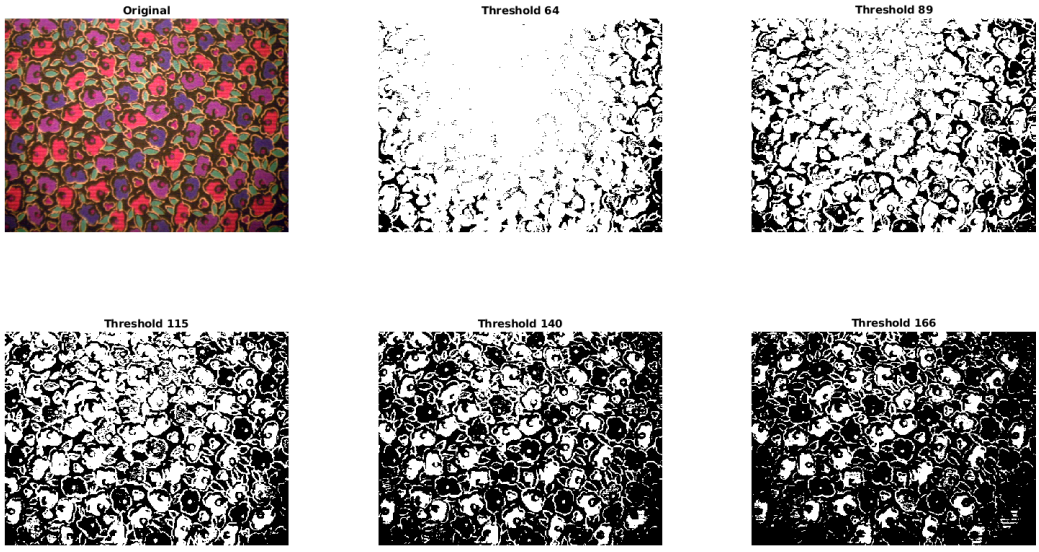
Section B:

The pictures have been binarized with the script “bzimg_script.m”. Different thresholds are generated with the `linspace()` function in order to get an overview with thresholds linearly spaced set. We started from a threshold fixed set at 64 and stopped with a threshold set at 166 so that the binarized pictures don’t get too bright or dark. (See the comments in the code for more details)

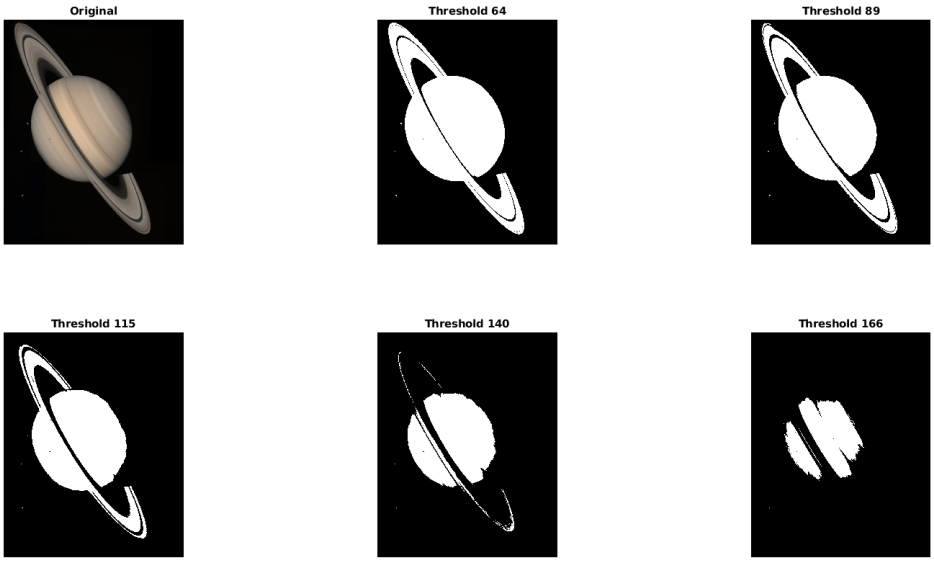
Since the picture is dark, a low threshold will give us a blank picture as shown by the picture where the threshold is set to 64. It is possible to distinguish the child on the pictures with the threshold set to 89 and 115, but we can’t when the threshold is too high.



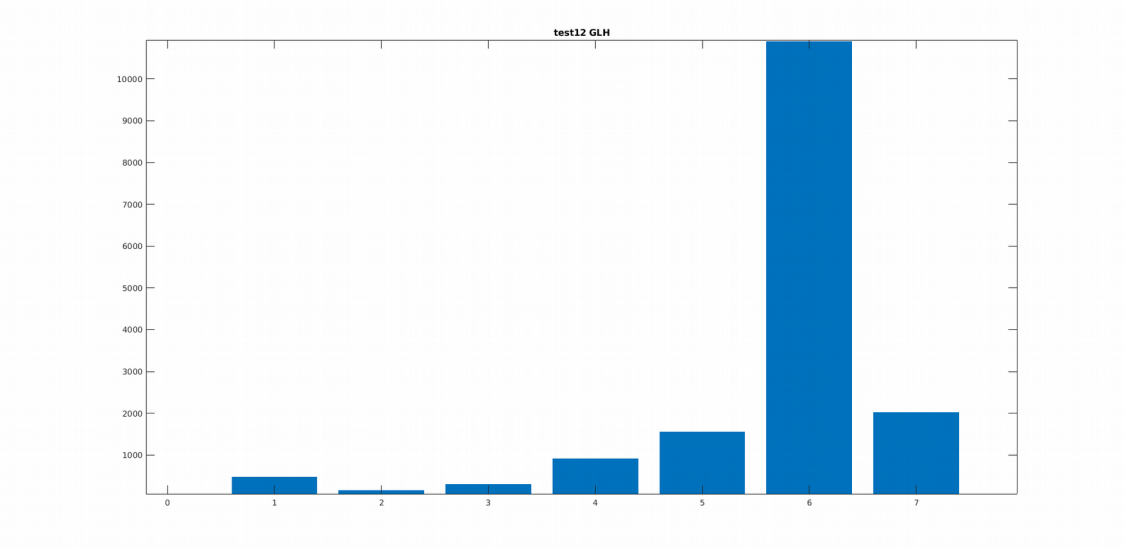
The contrast between the coins which are bright and the dark background is strong, therefore regarding on the selected thresholds, it is clearly possible to see them. With a threshold set to 89, coins appear as perfect white circles. A higher threshold enables us to perceive the details on the coins as the shades are darker.



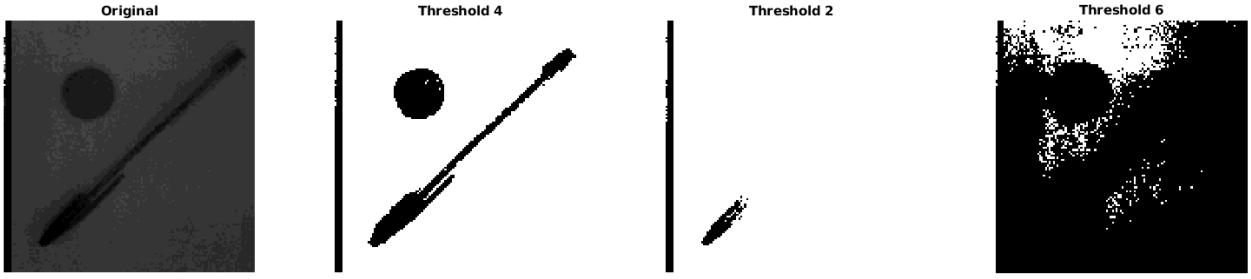
On this flower picture, a low threshold show us where the lighting was set. With a threshold set to 140, the flower outlines are visible.



Regarding of the selected threshold, the centre of Saturn is still visible as this is the brighter part of the picture. Saturn's rings are clearly distinguishable with a low threshold because of the light they reflect.



Here is the GLH of an image under relatively poor lighting conditions (test12.img) generated by the `poor_lightning.m` script. This helps us to select a threshold of 4 to binarize the image.



Our hand-picked threshold seems the best to be applied in order to clearly see the objects on the image (something circular and a pen). If a significantly higher or lower threshold is used, the objects are not visible any more.

Section C:

The black border on the left of the images (test1.img to test12.img) which is probably a sensor fault, has not been removed because it appears on every picture. If it wasn't, removing it could be done with the following code: `rawImage(:, 5:end)`.
Before extracting features of the images, we binarize them to simplify our processing on them.
The features are extracted and used to classify the images in the script called "classification.m".

Here is the extracted features values for the 12 test images:

The area is computed based on how many black pixels are in the picture. For the perimeter, we check if there are at least one neighbour that hasn't the same shade, this way we know that we are on an edge.

Those values have been cross-checked with the values provides by Matlab functions `bwarea()` and `bwperim()`. Black and white have been reversed thanks to `imcomplement()` function because the `bw*` functions are focussing on the white part.

```
test1.img: area: 3509, perimeter: 409
test2.img: area: 2469, perimeter: 400
test3.img: area: 3363, perimeter: 501
test4.img: area: 2323, perimeter: 440
test5.img: area: 3487, perimeter: 409
test6.img: area: 3196, perimeter: 526
test7.img: area: 2356, perimeter: 458
test8.img: area: 3450, perimeter: 398
test9.img: area: 2988, perimeter: 367
test10.img: area: 2132, perimeter: 405
test11.img: area: 2914, perimeter: 423
test12.img: area: 1858, perimeter: 766
```

	correct	guess
t1	'circle'	"circle"
t2	'triangle'	"triangle"
t3	'nail'	"nail"
t4	'screw'	"screw"
t5	'circle'	"circle"
t6	'nail'	"nail"
t7	'screw'	"screw"
t8	'circle'	"circle"
t9	'unknown'	"unknown"
t10	'screw'	"unknown"
t11	'triangle'	"unknown"
t12	'unknown'	"unknown"

The algorithm I devised to identify unlabelled objects regarding the extracted features is based on the reference values from `test1.img` to `test4.img`.
When reference values are compared with the newly extracted ones, I add a percentage that allows to create an interval. This means that there is a more or less margin added. The margin percentage has been set to 7%.

We do not have wrong classification as shown by the results but there are two objects that can't be assigned by this algorithm.

The effectiveness of the algorithm is influenced by the static reference values set but also the margin that we allow ourselves. This could be improved by having more tests images to adjust the these values.

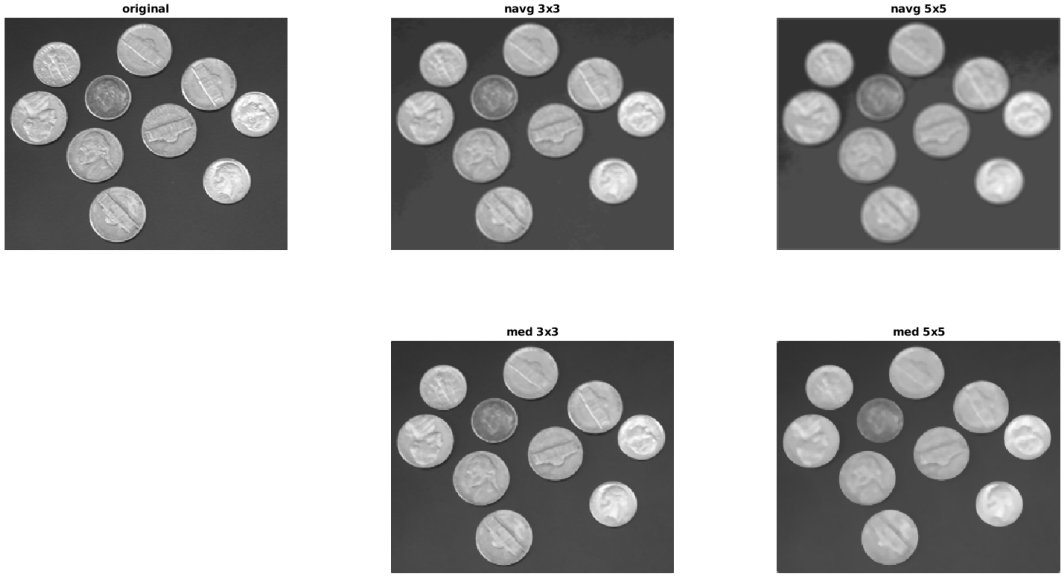
Section D:

Four filters have been created to help us improve noisy images. The filters are used in the script called "filtering.m".

We deal with the image edges by adding a border with a value set to `127` around the original image so that the edges doesn't get too black or white.

The filters did not affect the binary images (test1.img to test12.img), therefore I tested with the black and white given images.

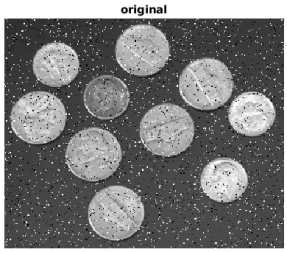
The neighbour averaging filter (navg) 5x5 blurs the image. On the other hand, the 3x3 median filter is keeping the image faithful to the original.

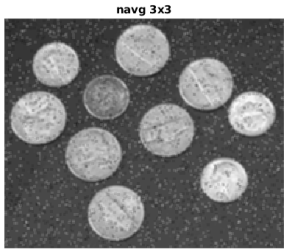


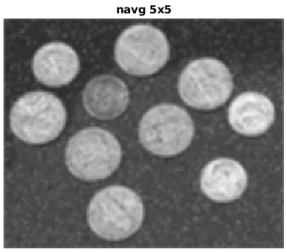
We can notice the same effects with the coins. With the 5x5 filters, the details of the coins fade.

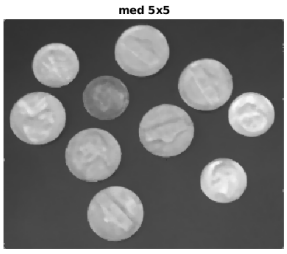
After adding some “salt and pepper” noise on the images, we can really see the effects of the filters.

The neighbour averaging filters reduces the effects of noise but the image gets instantly blurry. On the other hand, the median filter is performing very well, the 3x3 version gives us pretty much the original without the added noise.

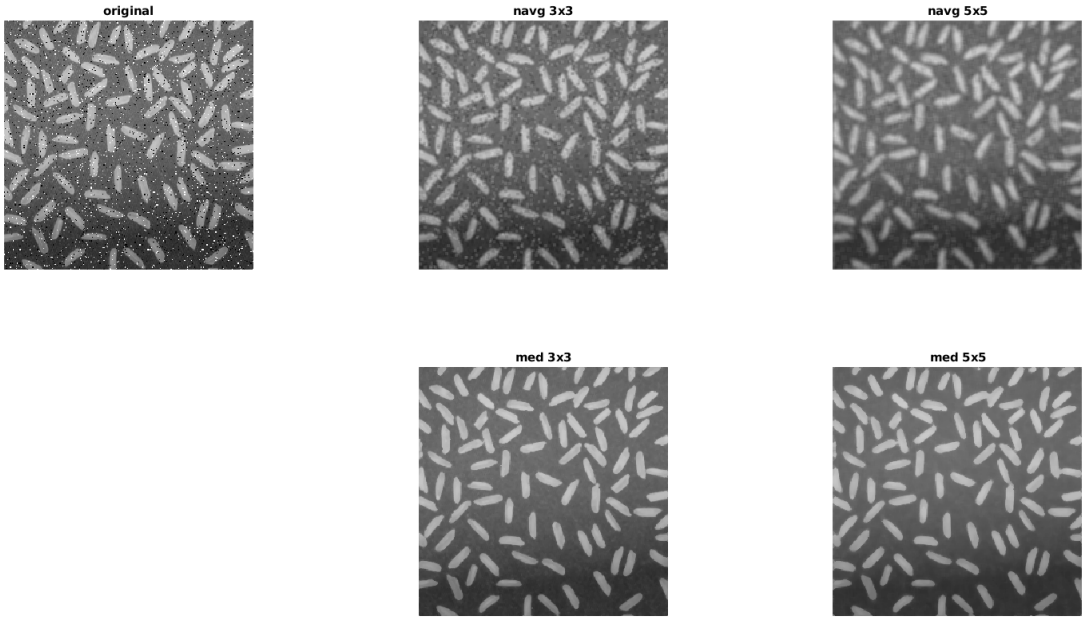








The same effects are happening with the coins.



And with the rice picture as well. This time the median filter 3x3 and 5x5 are performing the same way.

The median filter is well suited to remove the added “salt and pepper” noise.