Oblig1 INF4300

mathiaki

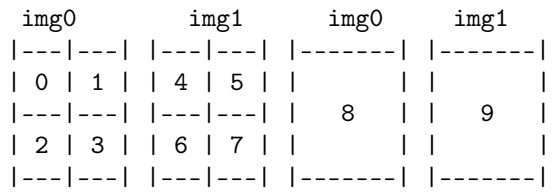
9. oktober 2017

Innhold

1 Texture description

In this part of the report I will try to analyze the different textures. I will go through every sub-texture and give a description based on: characteristics, texture direction, frequency of the texture, variance of the texture, homogeneity, and the size of the texture elements.

Before i describe the different textures, i will index the different images in the following way: (the same is true for the program)



Figur 1: Order of analyze

This means i will start in the top left corner of mosaic1 and and in the bottom right corner of mosaic2.
In the program 8 and 9 is reserved for the whole mosaic1 and mosaic2 respectively.

1.0 Texture 0

Characteristics:
Texture is mainly random noise. Somewhere in the texture you get some patters that looks like holes.

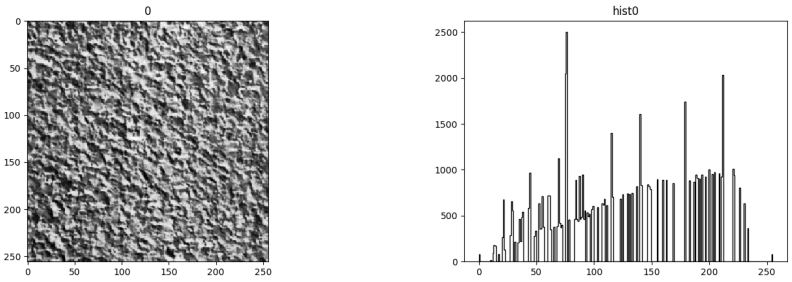
Texture Direction
There are a slight movement in the $\frac{3\pi}{4}$ rad direction, and as mentioned, some circular patterns can be observed.

Frequency
The crevasses is on a rough average 10px in diameter, and the edges between them is closer to 4px.

Variance
If we look at the histogram, we can see that it is one of the textures with the least variance. We do have some peaks int the histogram that drives down the variance a bit

Homogeneity
The texture is has very few homogeneous areas.

Texture element size
As mentioned, the texture is very similar to random noise, so the size is hard to determine. If we think of the elements as the crevasses, the texture element size is a few pixels.



(a) Texture 0 (b) Histogram of Texture 0 0

Figur 2: Texture with histogram

1.1 Texture 1

Characteristics:
Texture has a clearer pattern of squares that is roughly 8 px wide and high. It is also some random noise in the top left corner of the texture.

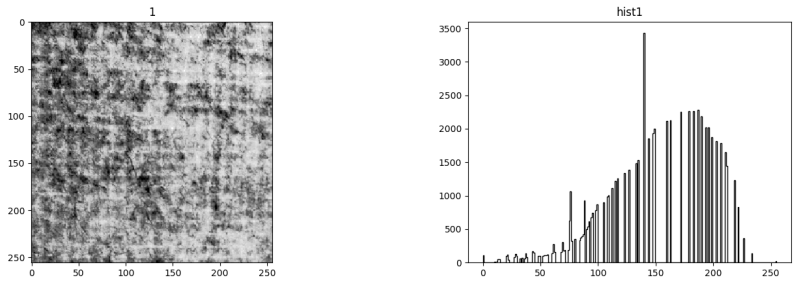
Texture Direction
The texture direction is almost horizontal (and vertical), with a slight angle clockwise.

Frequency
The frequency of the squares are approximately 1/40. Since there are that many repetitions in the image.

Variance
The histogram is fairly balanced, so the variance is in the middle if the spectrum. Especially compared to 4

Homogeneity
The texture is has no big homogeneous areas.

Texture element size
As mentioned, the squares has a diameter of approximatively 7 px.



(a) Texture 1 (b) Histogram of Texture 1

Figur 3: Texture with histogram

1.2 Texture 2

Characteristics:
The texture is a series of lines that face in roughly the same direction.

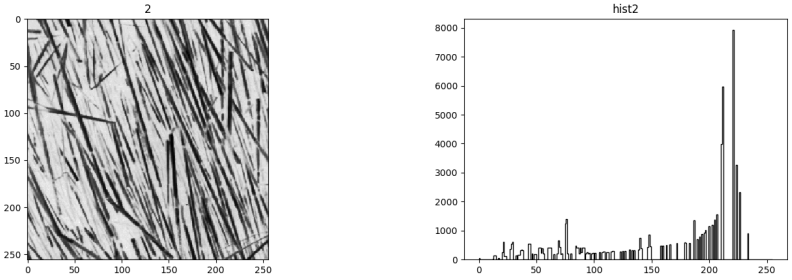
Texture Direction
The majority of the stripes has an angle of 100°. (or 1.745 rad)

Frequency
The frequency of the pattern, diagonally to the lines is 3-4 pixels in width, and they are repeating 30-40 times in the picture.

Variance
This pattern has a small variance, probably the smallest.

Homogeneity
The texture has large homogeneous areas, both in and around the stripes.

Texture element size
The element size is only a few pixels wide, and around 50 pixels high.



(a) Texture 2

(b) Histogram of Texture 2

Figur 4: Texture with histogram

1.3 Texture 3

Characteristics:
This seems like another white noise texture. Difference between this and 2 is that the first image had crevasses, and this does not.

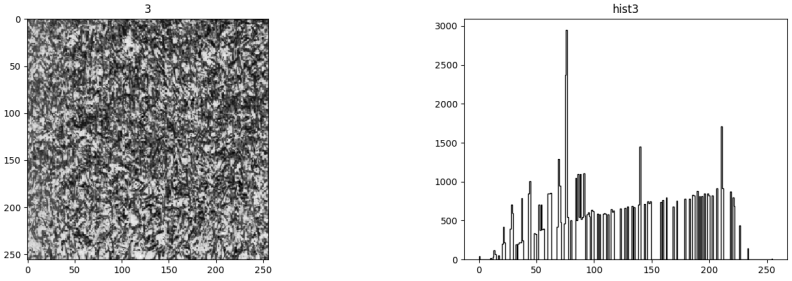
Texture Direction
There are no clear direction in the texture. This (and partly 2) is isotropic textures.

Frequency
It is hard to say anything about the frequency, but a rough guess might be 2-3px of the same grey-scale. This means a high frequency

Variance
This pattern has a low variance, but the pixel value aprox. 75 upping the variance.

Homogeneity
The texture is not homogeneous, nor does it have any homogeneous areas.

Texture element size
The texture element size is minimal.



(a) Texture 3 (b) Histogram of Texture 3

Figur 5: Texture with histogram

1.4 Texture 4

Characteristics:
This texture has a clear pattern of diagonal lines. The lines are a couple of pixels wide, but almost to thin to make a continuous line.(Almost a zigzag pattern instead)

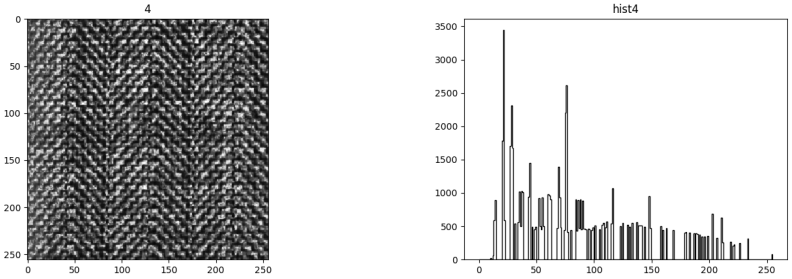
Texture Direction
We have two clear directions in this texture: $\frac{3\pi}{4}$ and $\frac{\pi}{4}$ rads.

Frequency
Frequency of the pattern is $1/3$ since the repeating pattern appears three times.

Variance
Compared to the high variance textures, the texture has a lower variance.

Homogeneity
There are not any areas with the same px value over a large area, so there are not much of homogeneous areas.

Texture element size
If you think of the texture as the diagonal lines, the size is 3-4 px in width and $1/6$ of the img in length.



(a) Texture 4 (b) Histogram of Texture 4

Figur 6: Texture with histogram

1.5 Texture 5

Characteristics:
Another Texture with a clear texture elements consisting of *blocks* that has a regular pattern thought-out the texture.

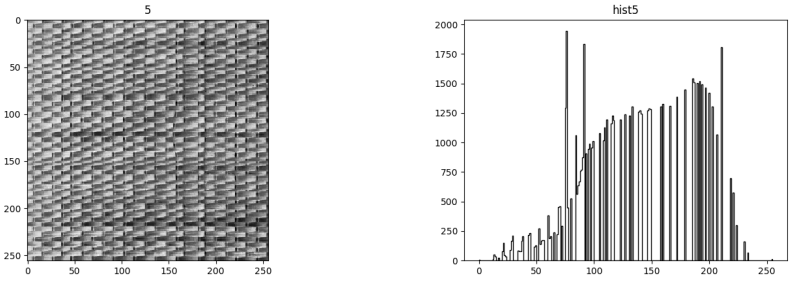
Texture Direction
We have two clear directions in this texture: horizontal and vertical. And like in 3 the texture is skewed of by a couple of degrees.

Frequency
The texture elements are repeating 12-14 times throughout the texture in the horizontal direction, so the horizontal frequency is 1/12. In the same way, the vertical frequency is 1/50.

Variance
This texture has a high variance.

Homogeneity
Inside the texture elements we find some homogeneous areas.

Texture element size
Texture element size here is the size of the rectangles. The texture element size is 5*20px**2



(a) Texture 5 (b) Histogram of Texture 5

Figur 7: Texture with histogram

1.6 Texture 6

Characteristics:
The texture has clear vertical lines that has a slight counter clockwise angle.

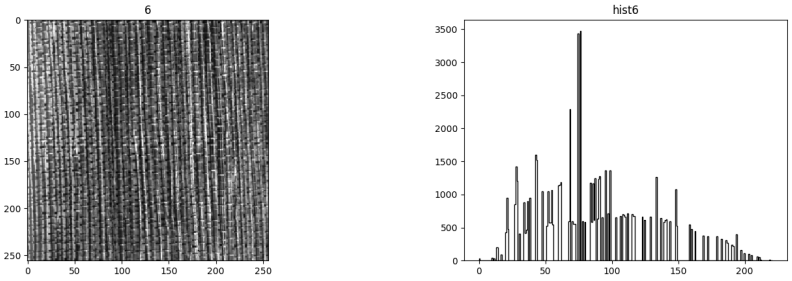
Texture Direction
The texture has an angle if $\frac{\pi}{2} + \epsilon$

Frequency
The vertical frequency is $1/50$ for the whole image.

Variance
This texture has a pretty similar variance to 6.

Homogeneity
We find some homogeneous areas if we follow the texture elements in the vertical direction.

Texture element size
Texture element size here is 3*250 px**2



(a) Texture 6

(b) Histogram of Texture 6

Figur 8: Texture with histogram

1.7 Texture 7

Characteristics:
The characteristics of this texture is pretty similar to 2. The texture does not have the same holes as the first one, but if we look at the edges of the image,10 ,here using canny edge detecting, we see a pretty similar result.

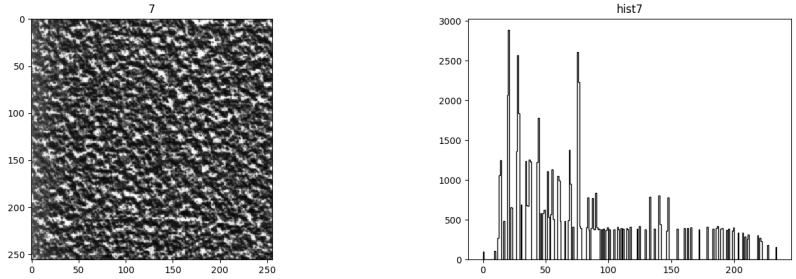
Texture Direction
You can imagine that the texture has an angle if $\frac{3\pi}{4}$ or $\frac{\pi}{4}$, but i would call it isotropic.

Frequency
Texture frequency is the same as in 3. approx 3/250.

Variance
This texture has a pretty similar variance to 3.

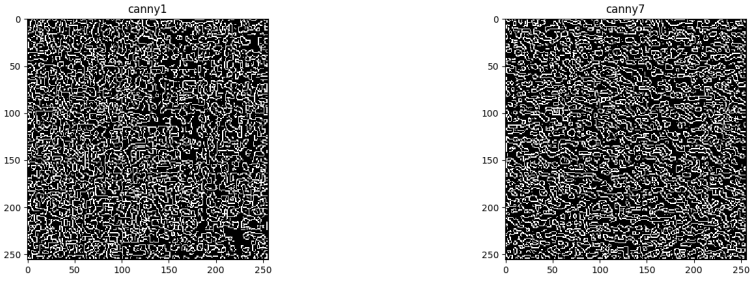
Homogeneity
We have very few large homogeneous areas. We do have a lot of small ones.

Texture element size
Texture element size here is 3*3 px** 2



(a) Texture 7 (b) Histogram of Texture 7

Figur 9: Texture with histogram



(a) Canny 0 (b) Canny 7

Figur 10: Comparison of the edges of 2 and 9

2 Visualizing the GLCM matrices

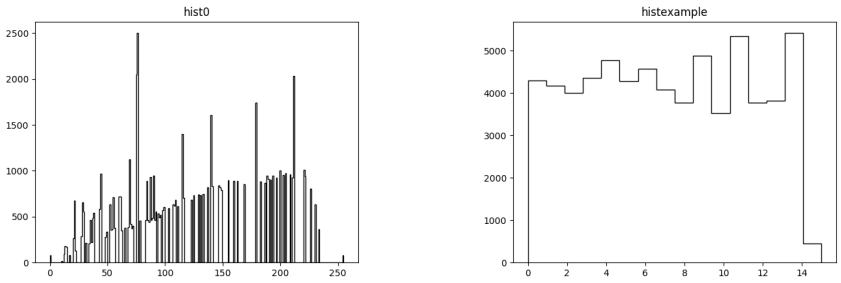
Now that we have a description of each image we want to work on each one of the images separately. And the first thing we want to do is to find a suitable GLCM. The problem with running a GLCM on the images is that we have 255 different grey-levels. So there are two operations we want to do before we continue.

- rescale the histogram[1]
- reduce the number of grey levels.

This can be done with the cv2 and skimage packages:

```
1
2 equalized_hist=cv2.equalizeHist(original_img)
3
4 equalized_rescaled_hist=(ski.exposure.rescale_intensity(
      equalized_hist, out_range=(0, 15)))
```

At the end we get the Original picture histeq-ed with 16 grey-levels.



(a) Old Histogram

(b) New Histogram

Figur 11: Transformation of the histogram (just an example, not actual transformation)

[1]The reason that we rescale the histogram is to ensure that the histogram isn't skewed. With a rescaled histogram we can be more certain that the GLCM is better represented.

2.1 The GLCM vals

2.1.0 Texture 0

As we remember from 2 we should use the angle $\frac{3\pi}{4}$ and the dist 4px.

2.1.1 Texture 1

As we remember from 3 we should use the angle 0 and angle π rad the dist 4px.

2.1.2 Texture 2

As we remember from 4 we should use the angle 1.745 rad and the dist 8 px. (length can vary on this one)

2.1.3 Texture 3

As we remember from 5 it was isotropic , so we use dist 2px. (isotropic is here just the average of linspace(0,2π,4))

2.1.4 Texture 4

As we remember from 6 we should use the angle $\frac{3\pi}{4}$ and the angle $\frac{\pi}{4}$ the dist 7px.

2.1.5 Texture 5

As we remember from 7 we should use the angle 0 rad and angle π rad the dist 4 and 12 respectively

2.1.6 Texture 6

As we remember from 8 we should use the angle π and the dist 8px (here we can use any length, preferably something that is not in the frequency of the horizontal component)

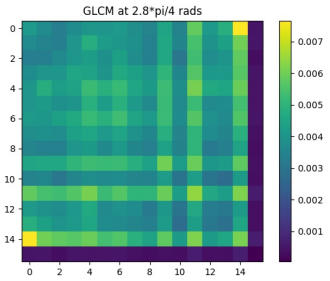
2.1.7 Texture 7

As we remember from 9 it was isotropic , so we use dist 3px. (isotropic is here just the average of linspace(0,2π,4))

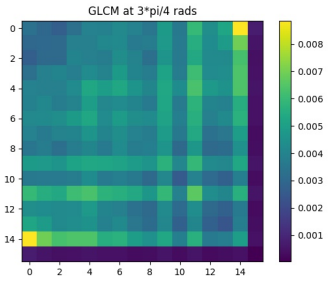
2.2 The GLCM Results

Keep in mind that the normalization of the textures often cut out most of the highest values, so it looks like we have no 15 value pixels. In truth they do exist, but we have only a few of them.

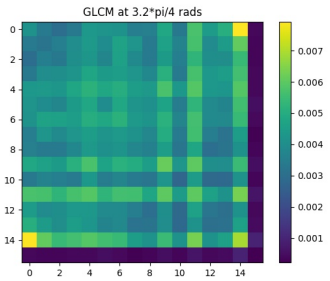
2.2.0 Texture 0



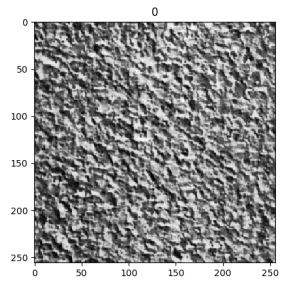
(a) GCLM 0 0



(b) GCLM 0 1



(c) GCLM 0 2

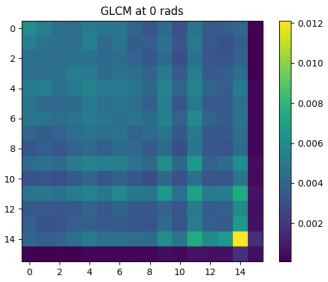


(d) Texture 0

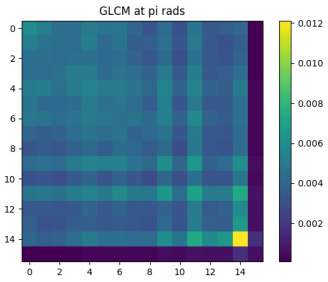
Figur 12: GLCM IMG 0: pi rad

As we can see from the GLCM of the first picture, we got a lot of pixels switching between 0 and 14 in all 3 of the tests. This could indicate that we have a pattern, but it is more likely that any angle would produce this result.

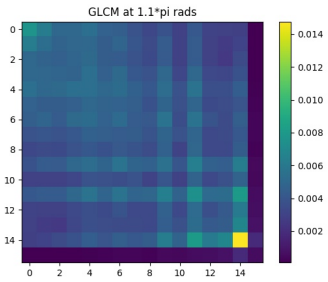
2.2.1 Texture 1



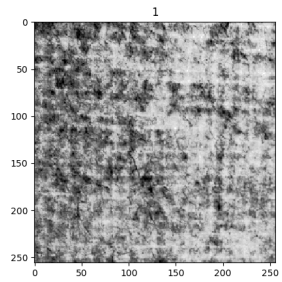
(a) GCLM 1 0



(b) GCLM 1 1



(c) GCLM 1 2

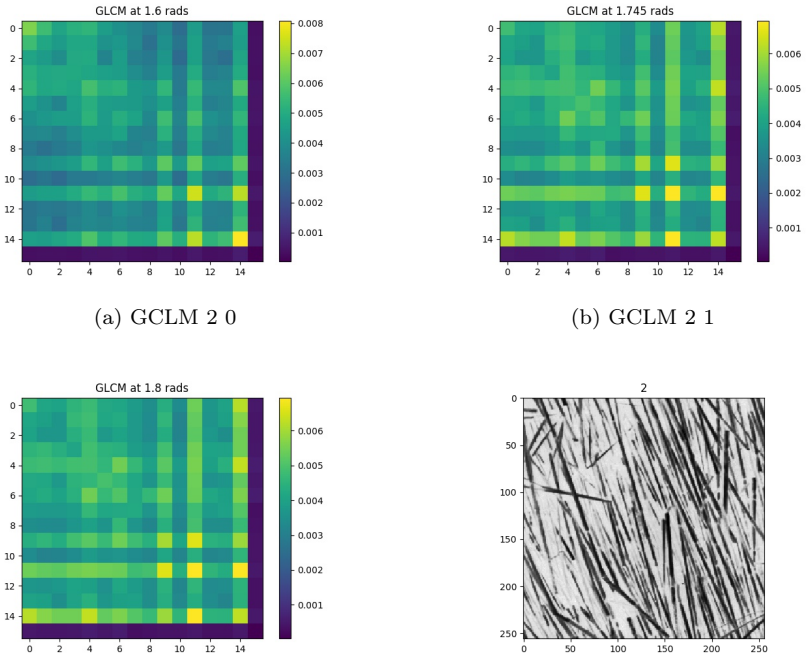


(d) Texture 1

Figur 13: GLCM IMG 1: pi rad

Again we have pretty similar result in all 3 tests. I don't think we can draw any clear results from this test.

2.2.2 Texture 2



(a) GCLM 2 0

(b) GCLM 2 1

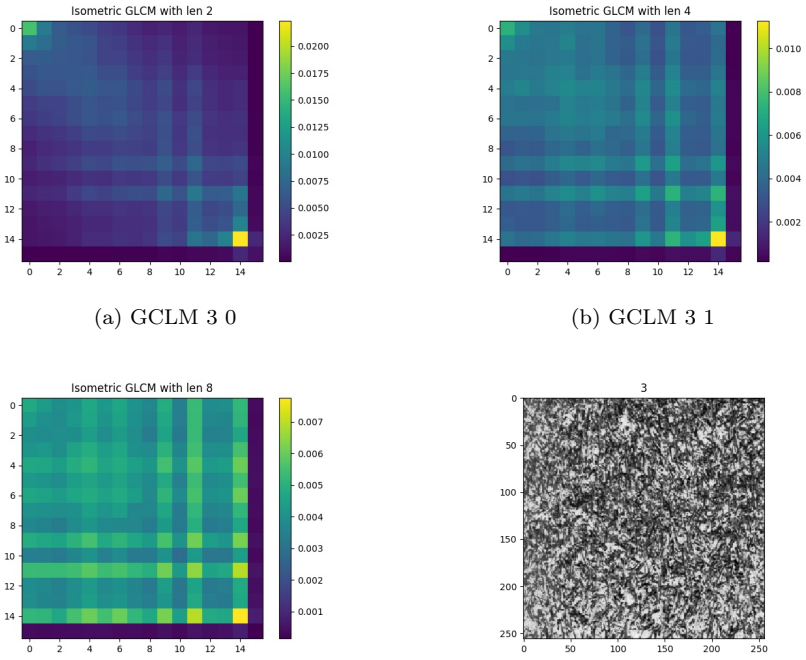
(c) GCLM 2 2

(d) Texture 2

Figur 14: GLCM IMG 2: pi rad

Here we have tried to land on the same line twice. From the tests, it looks like we had some good results at b) and c), since we have some good jumps from high values to other high values.

2.2.3 Texture 3



(a) GCLM 3 0

(b) GCLM 3 1

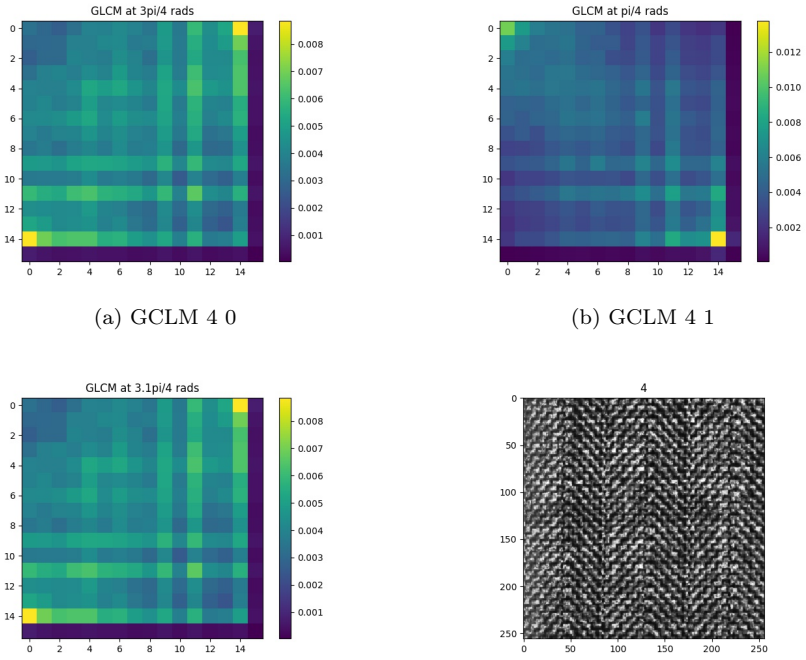
(c) GCLM 3 2

(d) Texture 3

Figur 15: GLCM IMG 3: pi rad

Number 3 was an isometric test, so we are here looking at the length, and not the angle if the GLCM input. (a) gives us the best result, where we can clearly see that we either stay at a high value, or we stay at a low value.

2.2.4 Texture 4



(a) GCLM 4 0

(b) GCLM 4 1

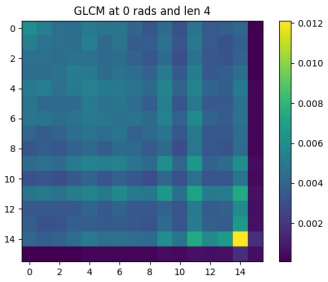
(c) GCLM 4 2

(d) Texture 4

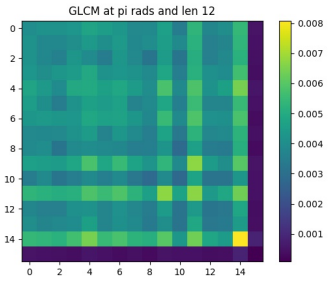
Figur 16: GLCM IMG 4: pi rad

We are here looking at a diagonal GLCM, and especially (b) yields a good result. It looks like we have a texture pattern.

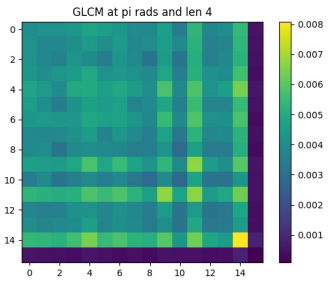
2.2.5 Texture 5



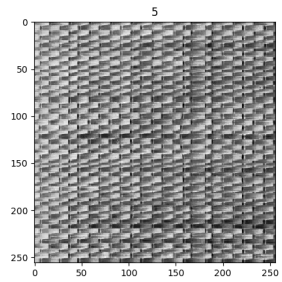
(a) GCLM 5 0



(b) GCLM 5 1



(c) GCLM 5 2

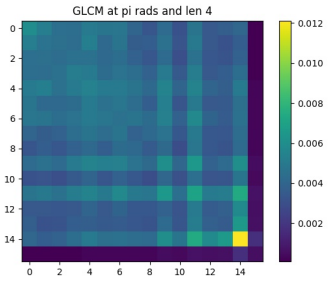


(d) Texture 5

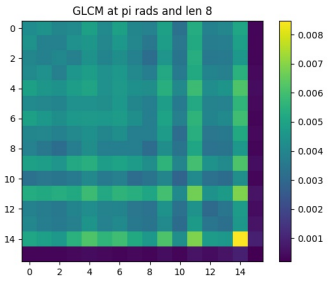
Figur 17: GLCM IMG 5: pi rad

Texture 5 gave us the best result in (a)

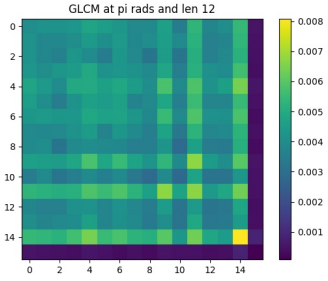
2.2.6 Texture 6



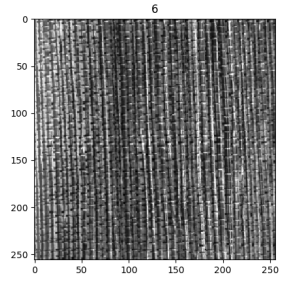
(a) GCLM 6 0



(b) GCLM 6 1



(c) GCLM 6 2

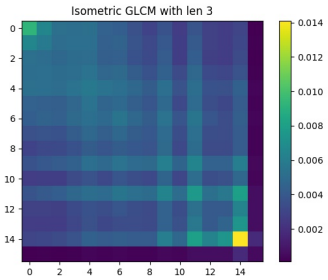


(d) Texture 6

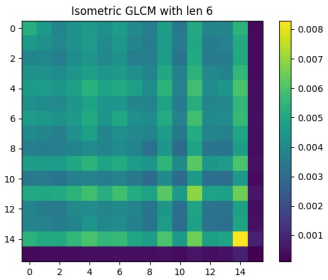
Figur 18: GLCM IMG 6: pi rad

At texture 6 we got the best result at (a), this is because of the short length. It is less likely to swey off target compared to the other length.

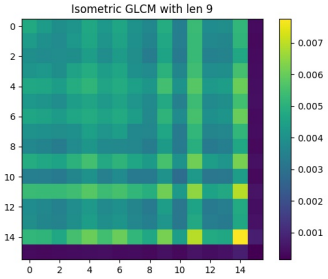
2.2.7 Texture 7



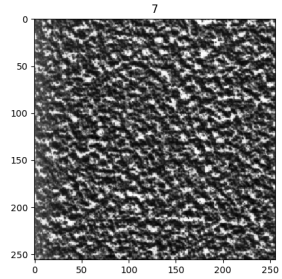
(a) GCLM 7 0



(b) GCLM 7 1



(c) GCLM 7 2



(d) Texture 7

Figur 19: GLCM IMG 7: pi rad

Texture 7 is another isometric GLCM, and the shortest length gave us the best result.

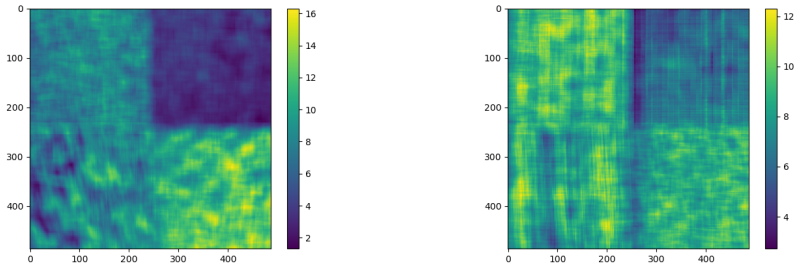
3 Computing features

Now that we have the have the GLCM matrices, we can now try to find:

- GLCM homogeneity
- GLCM inertia
- GLCM cluster shade

The formulas for the sliding window is in the assignment, and the code for the sliding window is found in appendix A.

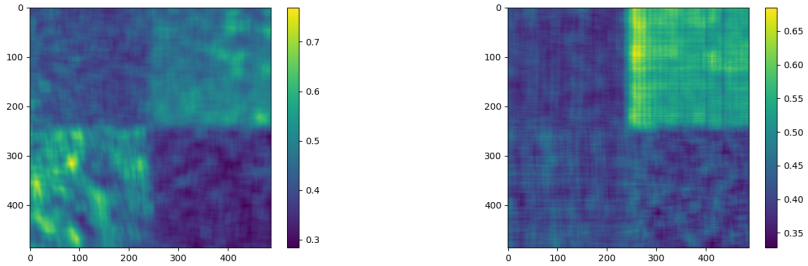
3.1 Computing features result



(a) First 4 inertia images (b) Last 4 inertia images

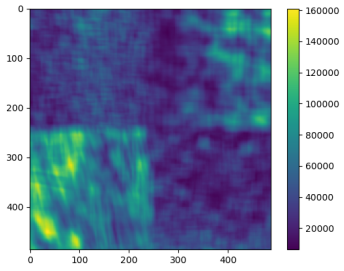
Figur 20: Global inertia with sliding window with size 25

Here we can see the global inertia, with a windowsize of 25. I've used the value 25 for the window size for all 3 of the calculations. This is because it is big enough to get multiple texture elements, and also small enough to not take multiple textures at once.

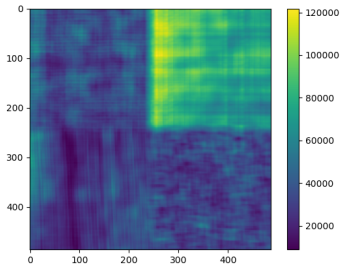


(a) First 4 homogeneity images (b) Last 4 homogeneity images

Figur 21: Global homogeneity with sliding window with size 25



(a) First 4 cluster shade images



(b) Last 4 cluster shade images

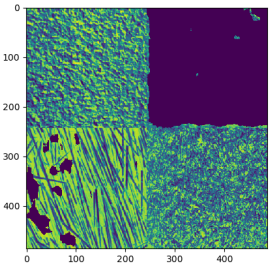
Figur 22: Global cluster shade with sliding window with size 25

4 The GLCM Global thresholding

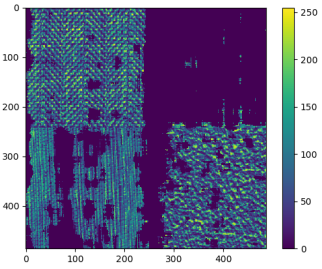
Now that we have the features from assignment C, we can now use the result to try to globally threshold:

4.1 Inertia

From 20, we can assume that the first image has a threshold of 5 and the second 8.



(a) First 4 inertia images



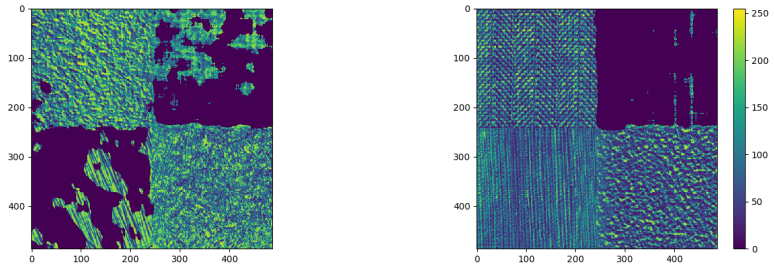
(b) Last 4 inertia images

Figur 23: filter of the inertia

From this result we managed to filter out one of the textures from each picture.

5 Homogeneity

From 21, we can assume that the first image has a threshold of 0.45 and the second image 0.5



(a) First 4 homogeneity images

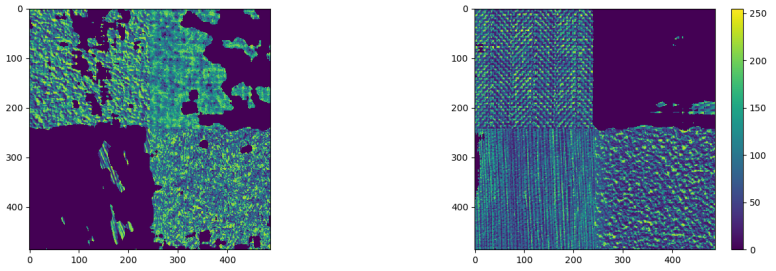
(b) Last 4 homogeneity images

Figur 24: filter of the homogeneity

From this result we managed to filter out one of the textures from each picture. from homogeneity we can se tht the first img has threshold 0.45 and the second 0.5
45000 and 60000

5.1 Cluster shade

From 22, we can assume that the first image has a threshold of 450000 and the second image 60000



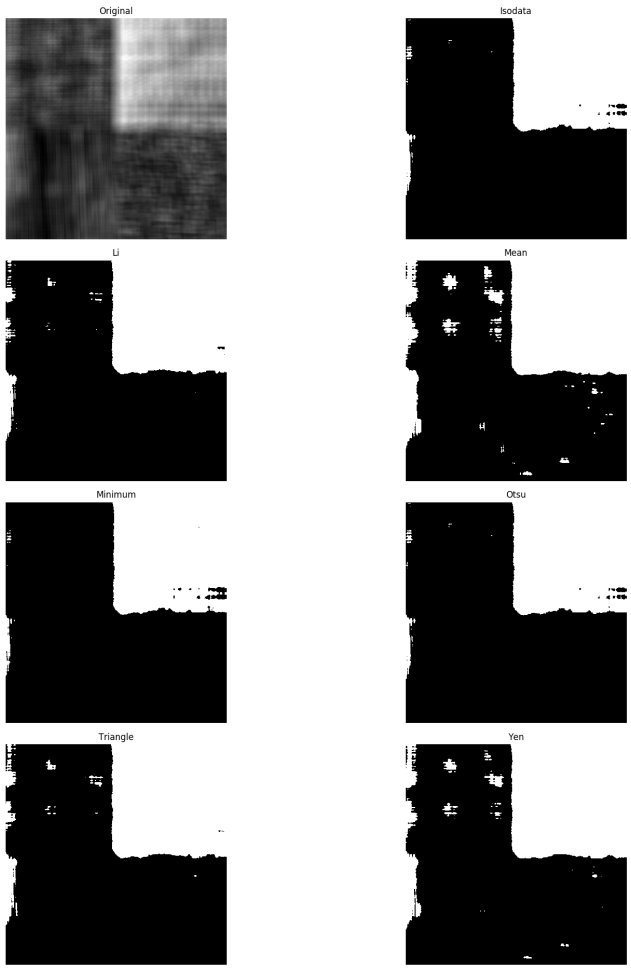
(a) First 4 cluster shade images

(b) Last 4 cluster shade images

Figur 25: filter of the cluster shade

5.2 Comment on thresholding

In this last assignment, we used the most primitive way of thresholding: trail and error.
Another way is to use one of the predetermined algorithms from the course: This is in the case that you need an automatic way to find the threshold.



Figur 26: Different algorithms for testing thresholding

6 Appendix A

This is the main program, that runs though all the assignments.

```
1 import cv2
```

```
import numpy as np
import matplotlib.pyplot as plt
import skimage.feature as skif
import skimage as ski
from matplotlib import animation
import sys
from numpy import pi
from numba import jit
from funcs import *
#import seaborn
plot1=False
plot2=False

img0 = cv2.imread('mosaic1.png',0)
img1 = cv2.imread('mosaic2.png',0)
#img1 = cv2.imread('zebra_3.tif',0)
img0shape=img0.shape
img1shape=img1.shape



#splitting the images
#assuming that the image is L/2 in with and height
subimg00=img0[0:256 , 0:256 ]
subimg01=img0[0:256 , 256:512 ]
subimg02=img0[256:512 , 0:256 ]
subimg03=img0[256:512 , 256:512 ]
subimg10=img1[0:256 , 0:256 ]
subimg11=img1[0:256 , 256:512 ]
subimg12=img1[256:512 , 0:256 ]
subimg13=img1[256:512 , 256:512 ]
original_img=[subimg00,subimg01,subimg02,subimg03,subimg10,subimg11
    ,subimg12,subimg13,img0,img1]
#original_img is now on the sape over


#from this point, every transformation will be in lists:
"""
element in list is described under
 img0        img1       img0      img1
|---|---| |---|---| |-------| |-------|
| 0 | 1 | | 4 | 5 | |       | |       |
|---|---| |---|---| |   8   | |   9   |
| 2 | 3 | | 6 | 7 | |       | |       |
|---|---| |---|---| |-------| |-------|


"""




####################
######PART 1#######
####################
```

```
58
59 def oppg1():
60     if plot1:
61         plt.subplot(121),plt.imshow(img0,cmap = 'gray')
62         plt.title('Original Image1'), plt.xticks([]), plt.yticks
       ([])
63         plt.subplot(122),plt.imshow(img1,cmap = 'gray')
64         plt.title('Original Image2'), plt.xticks([]), plt.yticks
       ([])
65         plt.show()
66     for i in enumerate(original_img):
67         plt.imshow(i[1],cmap='gray')
68         a="%s"%(i[0])
69         plt.title(a,)
70         plt.savefig('report/%s.png'%(i[0]))
71         plt.clf()
72
73         plt.imshow(ski.feature.canny(i[1]),cmap='gray')
74         a="canny%s"%(i[0])
75         plt.title(a,)
76         plt.savefig('report/canny%s.png'%(i[0]))
77         plt.clf()
78
79         plt.hist(i[1].ravel(), bins=256, histtype='step', color='
       black')
80         a="hist%s"%(i[0])
81         plt.title(a,)
82         plt.savefig('report/hist%s.png'%(i[0]))
83         plt.clf()
84
85
86
87
88
89
90 ######################
91 ######PART 2########
92 ######################
93
94 def oppg2():
95     glcm_img=[]
96
97
98     #image 0 as described in the report
99     equalized0=cv2.equalizeHist(original_img[0])
100    equalized0=(ski.exposure.rescale_intensity(equalized0,
       out_range=(0, 15)))
101    #from the report, we have the angle=3pi/4 and len=4
102    #using a small delta on each side
103    ang=[2.8*pi/4,3*pi/4,3.2*pi/4]
104    glcm_img.append(GLCM(equalized0, [4], ang ,normed1=True,
       symmetric1=True))
105
106    plt.title("GLCM at 2.8*pi/4 rads")
107    plt.imshow(glcm_img[0][:,:,0,0])
108    plt.colorbar()
109    plt.savefig("report/GCLM_IMG_0_0.jpg")
```

```
    plt.clf()
    plt.title("GLCM at 3*pi/4 rads")
    plt.imshow(glcm_img[0][:,:,0,1])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_0_1.jpg")
    plt.clf()
    plt.title("GLCM at 3.2*pi/4 rads")
    plt.imshow(glcm_img[0][:,:,0,2])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_0_2.jpg")
    plt.clf()



    #image 1 as described in the report
    equalized1=cv2.equalizeHist(original_img[1])
    equalized1=(ski.exposure.rescale_intensity(equalized0,
    out_range=(0, 15)))
    #from the report, we have the angle=3pi/4 and len=4
    #using a small delta on each side
    ang=[0,pi,1.1*pi]
    glcm_img.append(GLCM(equalized0, [4], ang ,normed1=True,
    symmetric1=True))

    plt.title("GLCM at 0 rads")
    plt.imshow(glcm_img[1][:,:,0,0])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_1_0.jpg")
    plt.clf()
    plt.title("GLCM at pi rads")
    plt.imshow(glcm_img[1][:,:,0,1])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_1_1.jpg")
    plt.clf()
    plt.title("GLCM at 1.1*pi rads")
    plt.imshow(glcm_img[1][:,:,0,2])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_1_2.jpg")
    plt.clf()



    #image 2 as described in the report
    equalized1=cv2.equalizeHist(original_img[2])
    equalized1=(ski.exposure.rescale_intensity(equalized0,
    out_range=(0, 15)))
    #from the report, we have the angle=3pi/4 and len=4
    #using a small delta on each side
    ang=[1.6,1.745,1.8]
    glcm_img.append(GLCM(equalized0, [4], ang ,normed1=True,
    symmetric1=True))

    plt.title("GLCM at 1.6 rads")
    plt.imshow(glcm_img[2][:,:,0,0])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_2_0.jpg")
```

```
    plt.clf()
    plt.title("GLCM at 1.745 rads")
    plt.imshow(glcm_img[2][:,:,0,1])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_2_1.jpg")
    plt.clf()
    plt.title("GLCM at 1.8 rads")
    plt.imshow(glcm_img[2][:,:,0,2])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_2_2.jpg")
    plt.clf()



    #image 3 as described in the report
    equalized1=cv2.equalizeHist(original_img[3])
    equalized1=(ski.exposure.rescale_intensity(equalized0,
    out_range=(0, 15)))
    #from the report, we have the angle=3pi/4 and len=4
    #using a small delta on each side
    ang=np.linspace(0,2*pi,4)
    length=[2,4,8]
    glcm_img.append(GLCM(equalized0, length, ang ,normed1=True,
    symmetric1=True))

    plt.title("Isometric GLCM with len 2")
    iso=0
    for i in range(len(ang)):
        iso=np.add(glcm_img[3][:,:,0,i], iso)

    plt.imshow(np.divide(iso,len(ang)))
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_3_0.jpg")
    plt.clf()

    plt.title("Isometric GLCM with len 4")
    iso=0
    for i in range(len(ang)):
        iso=np.add(glcm_img[3][:,:,1,i], iso)

    plt.imshow(np.divide(iso,len(ang)))
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_3_1.jpg")
    plt.clf()

    plt.title("Isometric GLCM with len 8")
    iso=0
    for i in range(len(ang)):
        iso=np.add(glcm_img[3][:,:,2,i], iso)

    plt.imshow(np.divide(iso,len(ang)))
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_3_2.jpg")
    plt.clf()
```

```
218
219     #image 4 as described in the report
220     equalized1=cv2.equalizeHist(original_img[4])
221     equalized1=(ski.exposure.rescale_intensity(equalized0,
        out_range=(0, 15)))
222     #from the report, we have the angle=3pi/4 and len=4
223     #using a small delta on each side
224     ang=[3*pi/4,pi/4,3.1*pi/4]
225     glcm_img.append(GLCM(equalized0, [4], ang ,normed1=True,
        symmetric1=True))
226
227     plt.title("GLCM at 3pi/4 rads")
228     plt.imshow(glcm_img[4][:,:,0,0])
229     plt.colorbar()
230     plt.savefig("report/GCLM_IMG_4_0.jpg")
231     plt.clf()
232     plt.title("GLCM at pi/4 rads")
233     plt.imshow(glcm_img[4][:,:,0,1])
234     plt.colorbar()
235     plt.savefig("report/GCLM_IMG_4_1.jpg")
236     plt.clf()
237     plt.title("GLCM at 3.1pi/4 rads")
238     plt.imshow(glcm_img[4][:,:,0,2])
239     plt.colorbar()
240     plt.savefig("report/GCLM_IMG_4_2.jpg")
241     plt.clf()
242
243
244
245     #image 5 as described in the report
246     equalized1=cv2.equalizeHist(original_img[5])
247     equalized1=(ski.exposure.rescale_intensity(equalized0,
        out_range=(0, 15)))
248     #from the report, we have the angle=3pi/4 and len=4
249     #using a small delta on each side
250     ang=[0,pi]
251     glcm_img.append(GLCM(equalized0, [4,12], ang ,normed1=True,
        symmetric1=True))
252
253     plt.title("GLCM at 0 rads and len 4")
254     plt.imshow(glcm_img[5][:,:,0,0])
255     plt.colorbar()
256     plt.savefig("report/GCLM_IMG_5_0.jpg")
257     plt.clf()
258     plt.title("GLCM at pi rads and len 12")
259     plt.imshow(glcm_img[5][:,:,1,1])
260     plt.colorbar()
261     plt.savefig("report/GCLM_IMG_5_1.jpg")
262     plt.clf()
263     plt.title("GLCM at pi rads and len 4")
264     plt.imshow(glcm_img[5][:,:,1,1])
265     plt.colorbar()
266     plt.savefig("report/GCLM_IMG_5_2.jpg")
267     plt.clf()
268
269
270
```

```
    #image 6 as described in the report
    equalized1=cv2.equalizeHist(original_img[6])
    equalized1=(ski.exposure.rescale_intensity(equalized0,
    out_range=(0, 15)))
    #from the report, we have the angle=3pi/4 and len=4
    #using a small delta on each side
    ang=[pi]
    glcm_img.append(GLCM(equalized0, [4,8,12], ang ,normed1=True,
    symmetric1=True))

    plt.title("GLCM at pi rads and len 4")
    plt.imshow(glcm_img[6][:,:,0,0])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_6_0.jpg")
    plt.clf()
    plt.title("GLCM at pi rads and len 8")
    plt.imshow(glcm_img[6][:,:,1,0])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_6_1.jpg")
    plt.clf()
    plt.title("GLCM at pi rads and len 12")
    plt.imshow(glcm_img[6][:,:,2,0])
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_6_2.jpg")
    plt.clf()


    #image 7 as described in the report
    equalized1=cv2.equalizeHist(original_img[7])
    equalized1=(ski.exposure.rescale_intensity(equalized0,
    out_range=(0, 15)))
    #from the report, we have the angle=3pi/4 and len=4
    #using a small delta on each side
    ang=np.linspace(0,2*pi,4)
    length=[3,6,9]
    glcm_img.append(GLCM(equalized0, length, ang ,normed1=True,
    symmetric1=True))

    plt.title("Isometric GLCM with len 3")
    iso=0
    for i in range(len(ang)):
        iso=np.add(glcm_img[7][:,:,0,i], iso)

    plt.imshow(np.divide(iso,len(ang)))
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_7_0.jpg")
    plt.clf()

    plt.title("Isometric GLCM with len 6")
    iso=0
    for i in range(len(ang)):
        iso=np.add(glcm_img[7][:,:,1,i], iso)

    plt.imshow(np.divide(iso,len(ang)))
    plt.colorbar()
    plt.savefig("report/GCLM_IMG_7_1.jpg")
    plt.clf()
```

```
324
325     plt.title("Isometric GLCM with len 9")
326     iso=0
327     for i in range(len(ang)):
328         iso=np.add(glcm_img[3][:,:,2,i], iso)
329
330     plt.imshow(np.divide(iso,len(ang)))
331     plt.colorbar()
332     plt.savefig("report/GCLM_IMG_7_2.jpg")
333     plt.clf()
334
335
336
337 ######################
338 ######PART 3########
339 ######################
340
341 def oppg3():
342     inertia_attr=       [25,25]
343     homogenity_attr=    [25,25]
344     cluster_shade_attr=[25,25]
345     inertia_img,homogenity_img,cluster_shade_img=save_features(
        original_img[8:], inertia_attr, homogenity_attr,
346                 cluster_shade_attr)
347
348     return inertia_img,homogenity_img,cluster_shade_img
349 ######################
350 ######PART 4########
351 ######################
352
353 def oppg4():
354     inertia_img=list(np.load('cluster_file_25_1.npy'))
355
356
357     #ski.filters.try_all_threshold(inertia_img[0],figsize=(19,19),
        verbose=True)
358     #ski.filters.try_all_threshold(inertia_img[1],figsize=(19,19),
        verbose=False)
359     plt.imshow(inertia_img[0])
360     plt.colorbar()
361     plt.savefig("report/oppg4cluster0.png")
362     plt.show()
363     plt.imshow(inertia_img[1])
364     plt.colorbar()
365     plt.savefig("report/oppg4cluster1.png")
366     plt.show()
367
368
369     tresh=45000
370     eq=eq_img(original_img)
371     img0=inertia_img[0]<=tresh
372     tresh=60000
373     img1=inertia_img[1]<=tresh
374
375     plt.imshow(np.multiply(img0,original_img[8][12:-13,12:-13]))
376     plt.savefig("report/oppg4cluster0filter.png")
377     plt.show()
```

```
    plt.imshow(np.multiply(img1,original_img[9][12:-13,12:-13]))
    plt.colorbar()
    plt.savefig("report/oppg4cluster1filter.png")
    plt.show()



#oppg2()
#inertia_img,homogenity_img,cluster_shade_img=oppg3()
#np.save("inertia_file_25_1", inertia_img)
#np.save("homo_file_25_1", homogenity_img)
#np.save("cluster_file_25_1", cluster_shade_img)
oppg4()
```

7 Appendix B

This is all the functions used by the main program

```
#DIV Funcs
import cv2
import numpy as np
import matplotlib.pyplot as plt
import skimage.feature as skif
import skimage as ski
from matplotlib import animation
import sys
from numpy import pi
from numba import jit

def make_gif(image,num_points,length):
    points=np.linspace(0,np.pi,num_points)
    glcm_img=GLCM(image, [length], points)
    for i in range(len(points)):
        plt.imshow(glcm_img[:,:,0,i])
        a="%f"%(i)
        plt.title(a,)
        plt.savefig('img%.2d.png'%(i))
        plt.clf()
    import glob
    filenames=glob.glob('img*')
    f=sorted(filenames)
    g=f[:-2]
    filenames=g
    import imageio
    with imageio.get_writer('movie.gif', mode='I') as writer:
        for filename in filenames:
            image = imageio.imread(filename)
            writer.append_data(image)
    import glob, os
    for f in glob.glob("img*.png"):
        os.remove(f)
```

```
38 def GLCM(image,distances,angles,levels1=16, symmetric1=False,
       normed1=False):
39     #calculates the grey level co-ocurtance matrix
40     return skif.greycomatrix(image, distances, angles, levels=
       levels1, symmetric=symmetric1, normed=normed1)
41
42 def plot_imglist(img,sub=False):
43     if sub:
44         plt.subplot(221),plt.imshow(img[0],cmap = 'gray')
45         plt.title('Img1'), plt.xticks([]), plt.yticks([])
46         plt.subplot(222),plt.imshow(img[1],cmap = 'gray')
47         plt.title('Img2'), plt.xticks([]), plt.yticks([])
48         plt.subplot(223),plt.imshow(img[2],cmap = 'gray')
49         plt.title('Img3'), plt.xticks([]), plt.yticks([])
50         plt.subplot(224),plt.imshow(img[3],cmap = 'gray')
51         plt.title('Img4'), plt.xticks([]), plt.yticks([])
52         plt.show()
53         plt.subplot(221),plt.imshow(img[4],cmap = 'gray')
54         plt.title('Img5'), plt.xticks([]), plt.yticks([])
55         plt.subplot(222),plt.imshow(img[5],cmap = 'gray')
56         plt.title('Img6'), plt.xticks([]), plt.yticks([])
57         plt.subplot(223),plt.imshow(img[6],cmap = 'gray')
58         plt.title('Img7'), plt.xticks([]), plt.yticks([])
59         plt.subplot(224),plt.imshow(img[7],cmap = 'gray')
60         plt.title('Img8'), plt.xticks([]), plt.yticks([])
61         plt.show()
62     else:
63         plt.subplot(121),plt.imshow(img[8],cmap = 'gray')
64         plt.title('Img1'), plt.xticks([]), plt.yticks([])
65         plt.subplot(122),plt.imshow(img[9],cmap = 'gray')
66         plt.title('Img2'), plt.xticks([]), plt.yticks([])
67         plt.show()
68
69
70 def save_features(original_img,inertia_attr,homogenity_attr,
       cluster_shade_attr):
71     inertia_img=[]
72     homogenity_img=[]
73     cluster_shade_img=[]
74     import time
75     for i in enumerate(original_img):
76         start_time = time.time()
77         a=sliding_window(i[1],inertia_attr[i[0]],inertia)
78         b=sliding_window(i[1],homogenity_attr[i[0]],homogenity)
79         c=sliding_window(i[1],cluster_shade_attr[i[0]],
       cluster_shade)
80         inertia_img.append(a)
81         homogenity_img.append(b)
82         cluster_shade_img.append(c)
83
84         print("--- %s seconds ---" % (time.time() - start_time))
85
86         plt.imshow(a)
87         plt.savefig('report/inertia%s.png'%(i[0]))
88         plt.clf()
89         plt.imshow(b)
90         plt.savefig('report/homogenity%s.png'%(i[0]))
```

```
91          plt.clf()
92          plt.imshow(c)
93          plt.savefig('report/cluster_shade%s.png'%(i[0]))
94          plt.clf()
95
96      return inertia_img,homogenity_img,cluster_shade_img
97
98
99
100 def eq_img(original_img,plotting=False):
101     #equalized img with 16 greysacles
102     equalized_img=[]
103     for i in original_img:
104             equalized_img.append(ski.exposure.rescale_intensity(cv2
    .equalizeHist(i), out_range=(0, 15)))
105
106     if plotting:
107         plot_imglist(equalized_img,sub=True)
108     return equalized_img
109
110
111 def homogenity(p):
112         rows,cols=p.shape
113         a=np.fromfunction(lambda x, y: 1/(1+(x-y)**2) , (rows, cols
    ), dtype=int)
114         return np.sum(np.multiply(a,p))
115
116
117 def inertia(p):
118     rows,cols=p.shape
119     a=np.fromfunction(lambda x, y:(x-y)**2 , (rows, cols), dtype=
    int)
120     return np.sum(np.multiply(a,p))
121
122 def cluster_shade(p):
123     rows,cols=p.shape
124
125     def u_x(rows,cols,p):
126
127         return np.fromfunction(lambda  i,j : i*np.sum(p[i,:]) , (
    rows,1), dtype=int)
128
129     def u_y(rows,cols,p):
130         return np.fromfunction(lambda  j,i : j*np.sum(p[:,j]) , (
    cols,1), dtype=int)
131
132     a=np.fromfunction(lambda x, y:x+y , (rows, cols), dtype=int)
133     b=u_x(rows,cols,p)
134     c=u_y(rows,cols,p)
135     d=np.add(a, b)
136     d=np.add(d,c)
137     d=np.power(d, 3)
138     return np.sum(np.multiply(d,p))
139
140 def sliding_window(inarray,size,f):
141     equalized0=cv2.equalizeHist(inarray)
```

```
    inarray_16=(ski.exposure.rescale_intensity(equalized0,
    out_range=(0, 15)))
    s=int((size-1)/2)
    newarray=np.zeros(inarray.shape)
    A,B=np.shape(inarray)


    for i in enumerate(inarray):
        if i[0]+s >= A or i[0]-s <= 0:
            continue
        for j in enumerate(i[1]):
            if j[0]+s >= B or j[0]-s <= 0:
                continue


            w=[i[0]-s,i[0]+s,j[0]-s,j[0]+s]
            p=GLCM(inarray_16[w[0]:w[1],w[2]:w[3]], [1], [0,pi/2,pi
    ,3*pi/2], symmetric1=True, normed1=True)
            iso=0
            for k in range(4):
                iso=np.add(p[:,:,0,k],iso) #gets only the first
    gclm
            iso=np.divide(iso, 4)

            newarray[i[0]][j[0]]=f(iso)

    return newarray[0+s+1:A-s,0+s+1:B-s]
```