Exercise 1, TDT4195 – Mikkel Svagård

1. Theory:

1. The histogram showing frequency of the intensity values of an image will tell you if its low contrast if the frequent values are closely grouped, while it also usually has lacks intensities in the higher and/or lower spectre of the histogram. An example is the following picture:

2. Histogram equalization enhances the contrast to the image. Equalizing an image a second time wont change the image further, as the intensities are already amplified to whatever values they are supposed to be.

0	5	6	3	3
4	7	4	6	4
4	5	3	5	4

3. Histogram Equalization

Old image

N	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
f(n)	1	0	0	3	5	3	2	1
F(r)	1	1	1	4	9	12	14	15
F(r)/15	0.07	0.07	0.07	0.27	0.6	0.88	0.93	1
T(r)	0	0	0	2	4	6	7	7

New image:

0	6	7	2	2
4	7	4	7	4
4	6	2	6	4

4. Correlation is the process of moving a filter mask over the image, as in all spatail filtering, and computing the sum of products at each location. Convolution has a simmilar mechanig, except that the filter is rotated 180 degrees. All symmetrical kernels will have the samme effect for both convolution and correlation. An example is [1,1,1] or [1,2,4,2,1]

5. Spatial Convolution

0	0	5	6	3	3	3
0	**0**	**5**	**6**	**3**	**3**	3
4	**4**	**7**	**4**	**6**	**4**	4
4	**4**	**5**	**3**	**5**	**4**	4
4	4	5	3	5	4	4

0	1	0
1	-4	1
0	1	0

Image, padded with extended edges

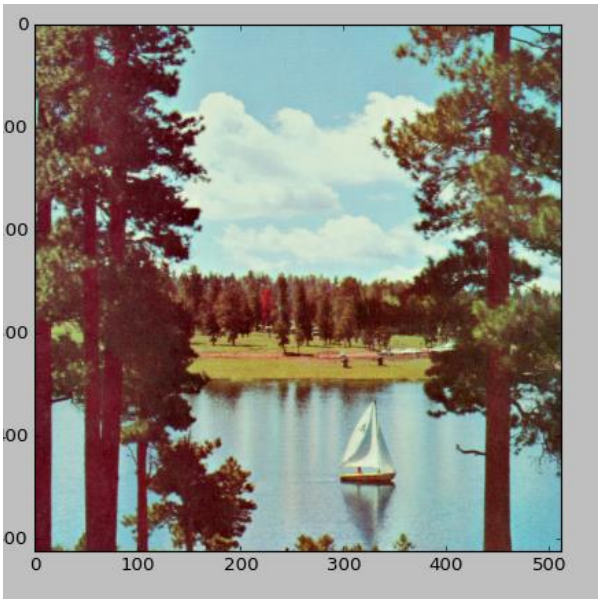
9	**-2**	**-6**	**6**	**1**
-1	**-10**	**6**	**-8**	**1**
1	**-1**	**5**	**-2**	**1**

7	0	0	6	1
0	0	6	0	1
1	0	5	0	1

By applying the filter, we get the new image on the left. As the overflow values are undefined, et -10 and 9, I created a plausible result on the right.

2. Programming

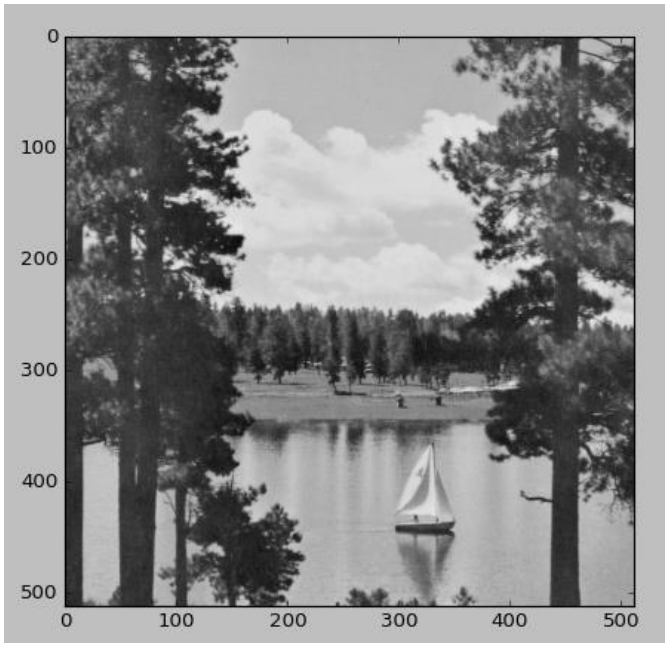
Task 1)

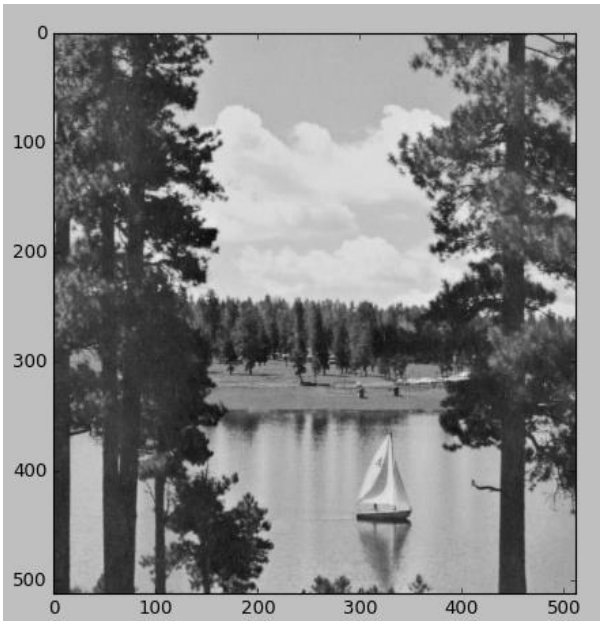


The first picture is the original.

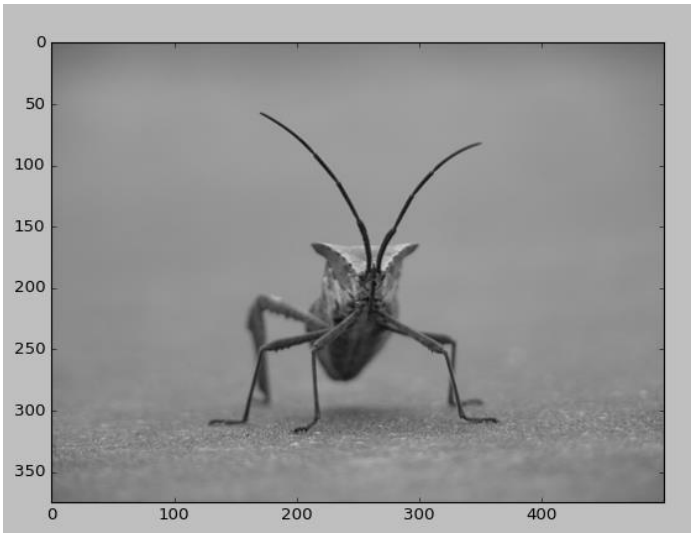
The lower left, picture is the average, while the rightern is the weighted one.

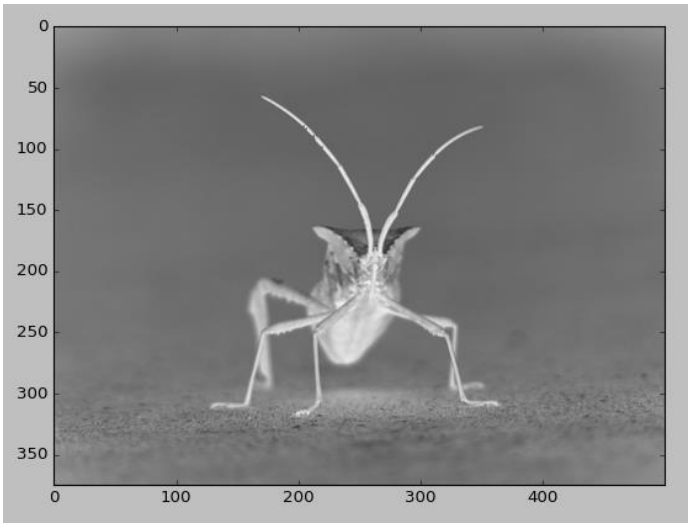
As one might notice, the weighted one has more contrast and feels more darker.





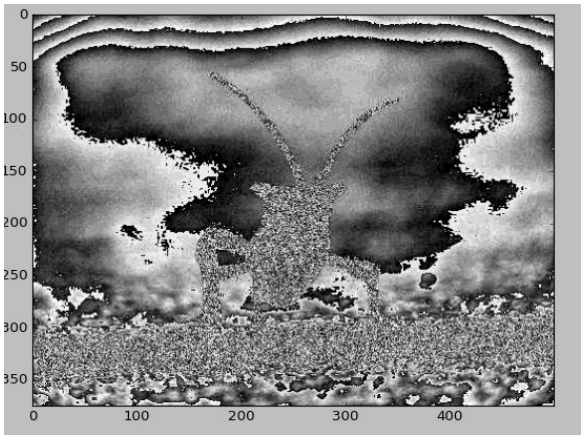
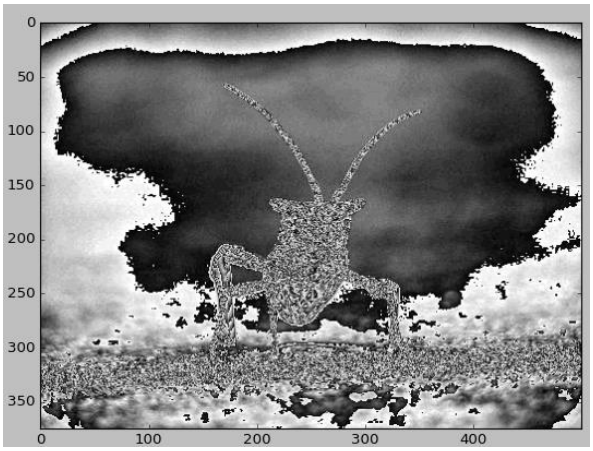
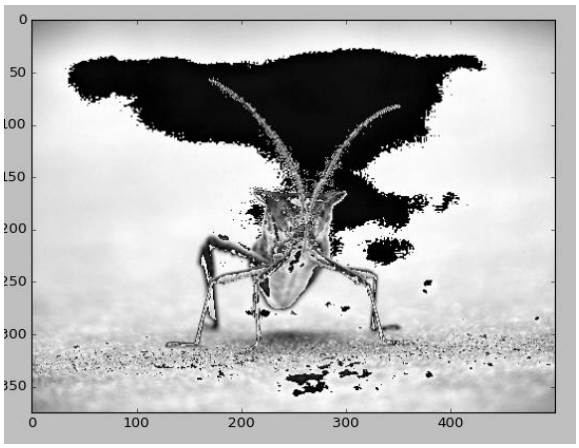
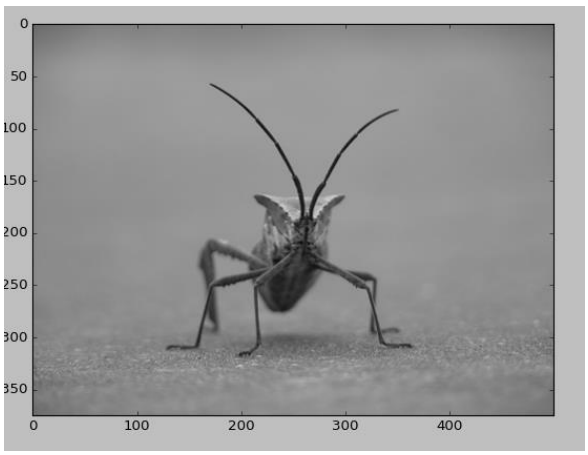
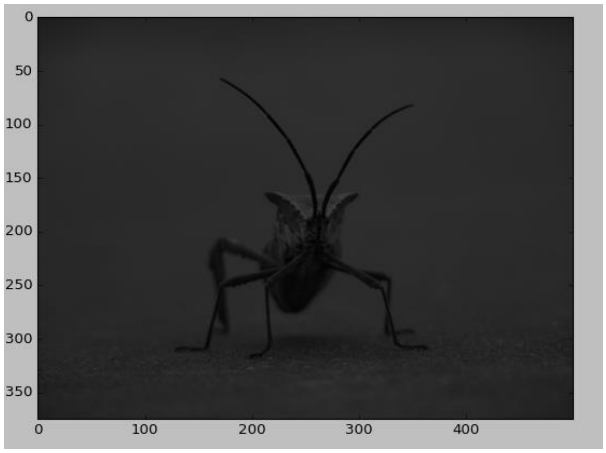
Task 2a)





This transformation can be called “inverting” the picture, and all pixels are (1-p) of its own original intensity.

Task 2b)


Top right to left, then down: gamma values:

0.3 – 0.5 – 0.8 – 1 – 1.2 – 1.5

As you can see, belove 1 is getting superweird. 1 is returning the original picture, while above 1 only makes it darker.

Task 3)

a)

```
def convolution(H,F): #The function that invertes the picture
  V = 0;
  for i in F:
    for v in i:
      V += v
  h = copy.copy(H) #makes aa copy of the image to write over
  o = int(float(len(F)/2)) #The offsett of the filter
  l = len(F)
  for x in range (o,len(H)-o):
    for y in range(o,len(H[x])-o):
      h[x][y]=0 #Sets the current color at current pixel to zero
      for i in range (0,len(F)):
        for j in range(0,len(F[i])):
            #adds the filters multiplier to the neighboorhood and adds it to the color
            h[x][y] += H[x-o+i][y-o+j]*F[i][j]/V
  return h
```

Implemented to function above, along with a bunch of other stuff. As you can see, the algorithm only targets pixels one or two pixels form the edges, as we weren't supposed to take in account for the padding. The rest of the code is in the zip-file

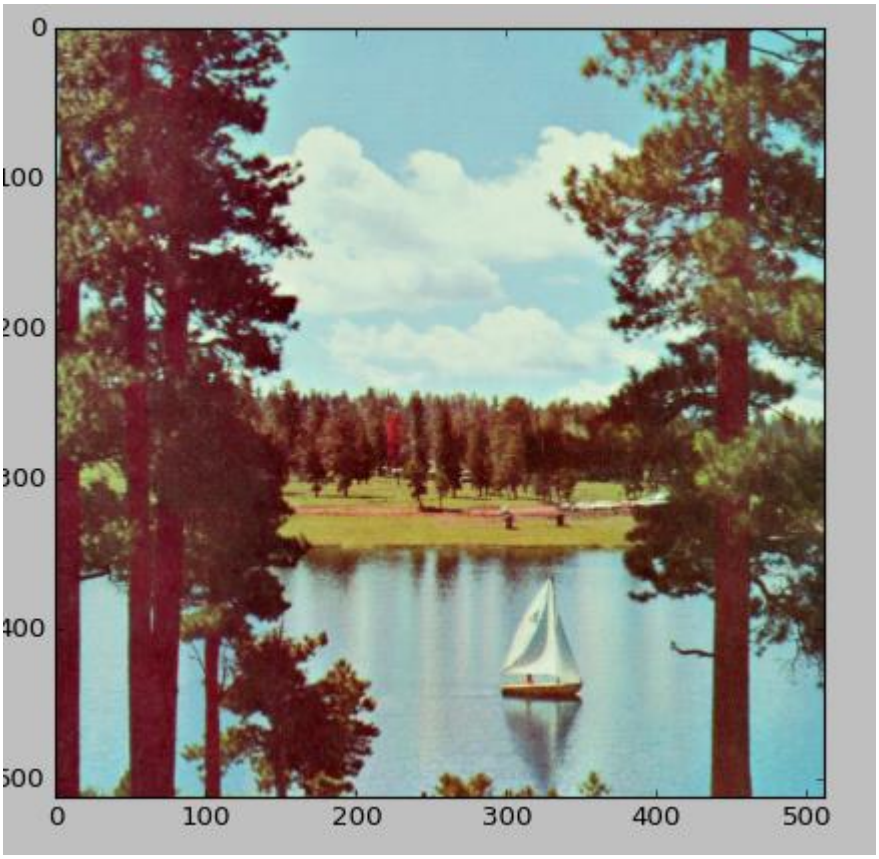
b

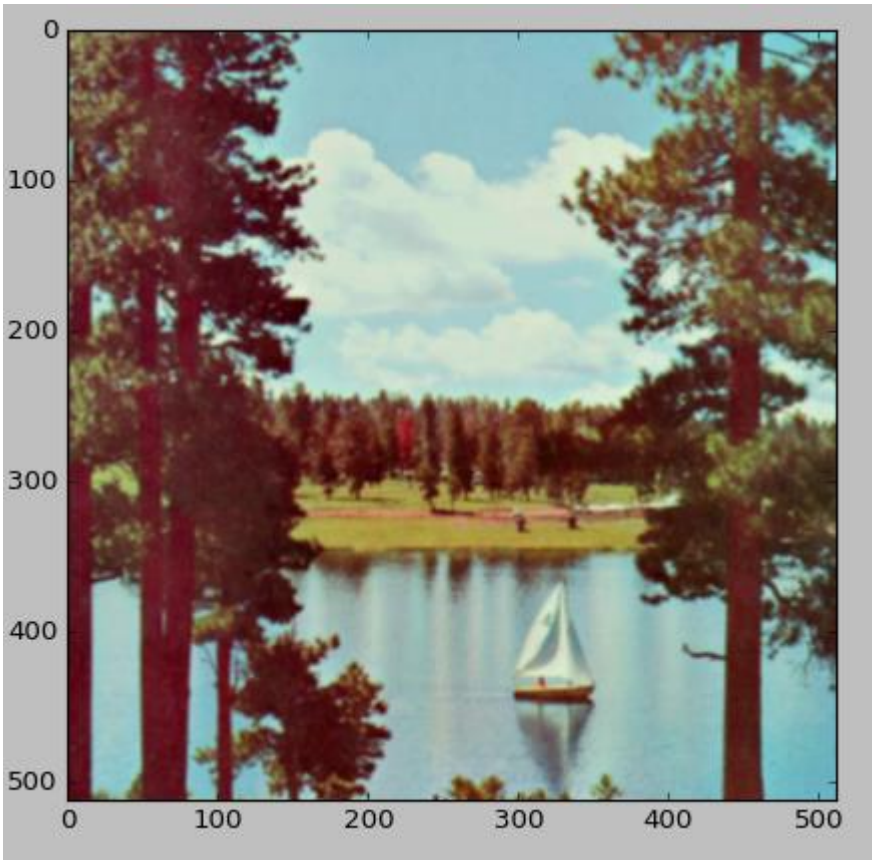
```
def convolution(H,F): #The function that invertes the picture
  V = 0;
  for i in F:
    for v in i:
      V += v
  h = copy.copy(H) #makes aa copy of the image to write over
  o = int(float(len(F)/2)) #The offsett of the filter
  l = len(F)
  for x in range (o,len(H)-o):
    for y in range(o,len(H[x])-o):
      for c in range(0,3):
        h[x][y][c]=0 #Sets the current color at current pixel to zero
        for i in range (0,len(F)):
          for j in range(0,len(F[i])):
              #adds the filters multiplier to the neighboorhood and adds it to the color
              h[x][y][c] += H[x-o+i][y-o+j][c]*F[i][j]/V
  return h
```

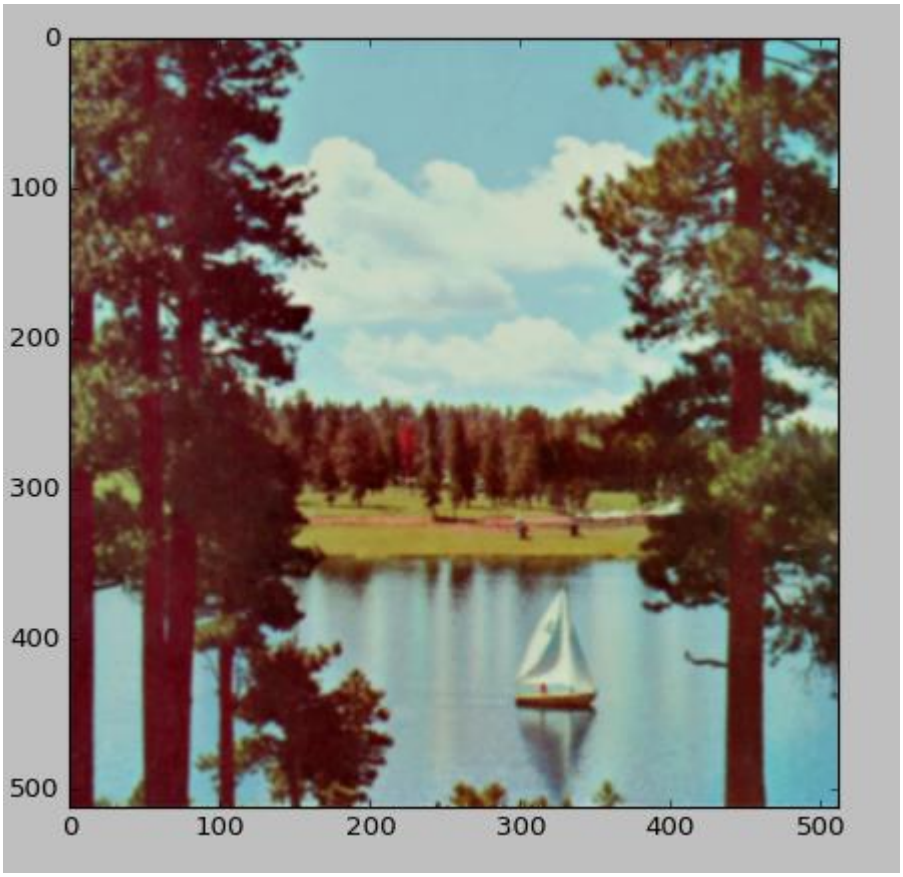
Fairly similar to the one from a, just added a loop to handle the different colors.

The following pictures are from running the algorithm: original, 3x3 filter and 5x5 filter respectively

Its hard to notice, but by looking at the trees one can easily identify the smoothening.


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500

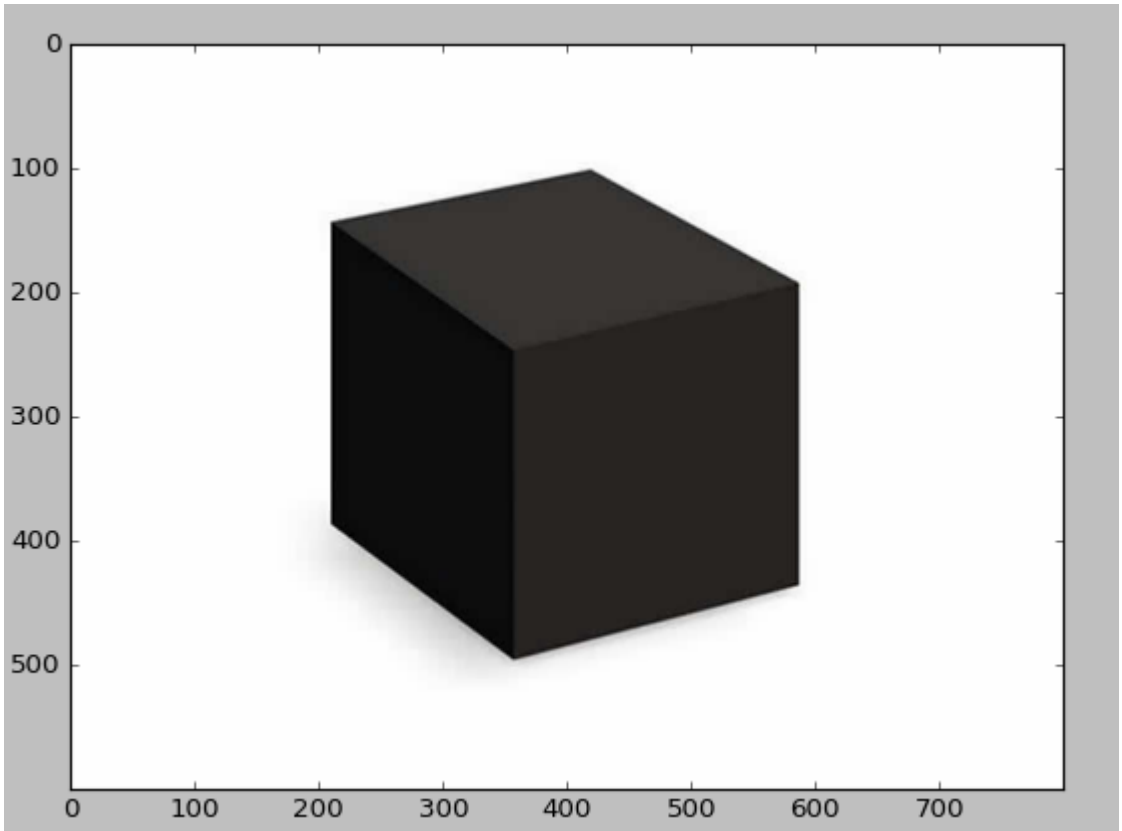


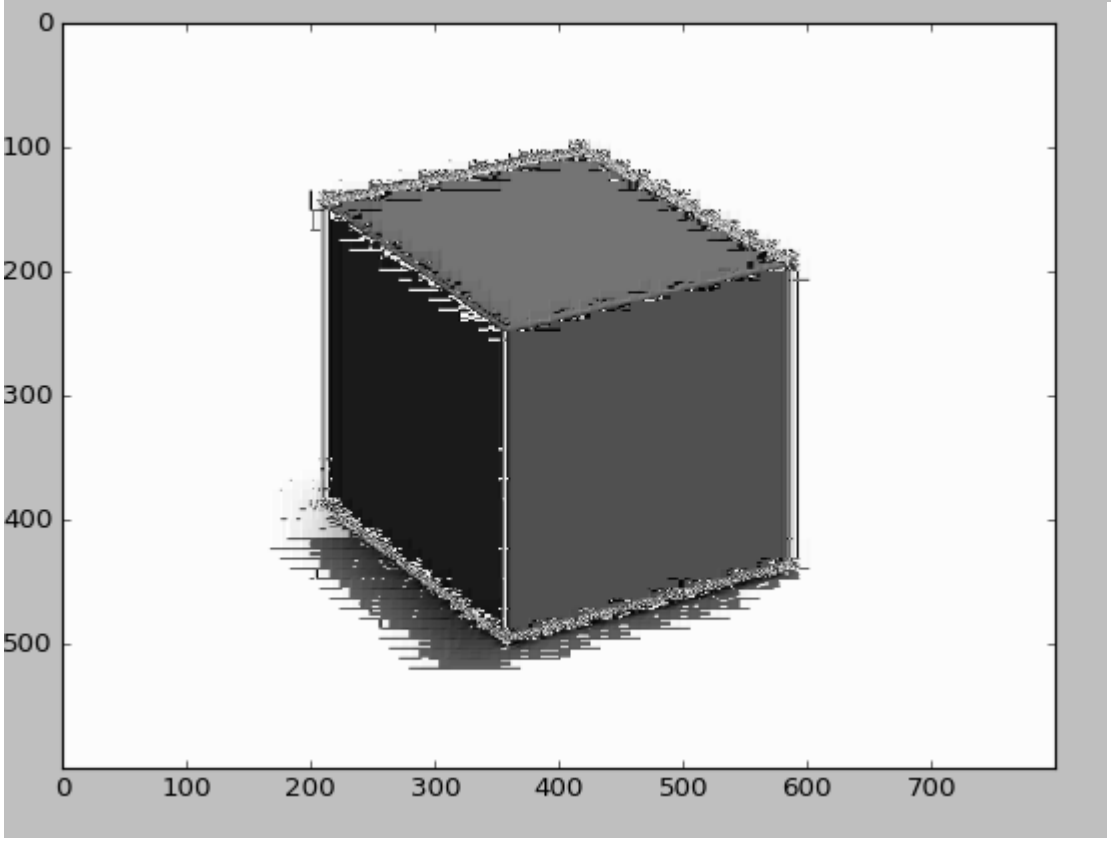
c)

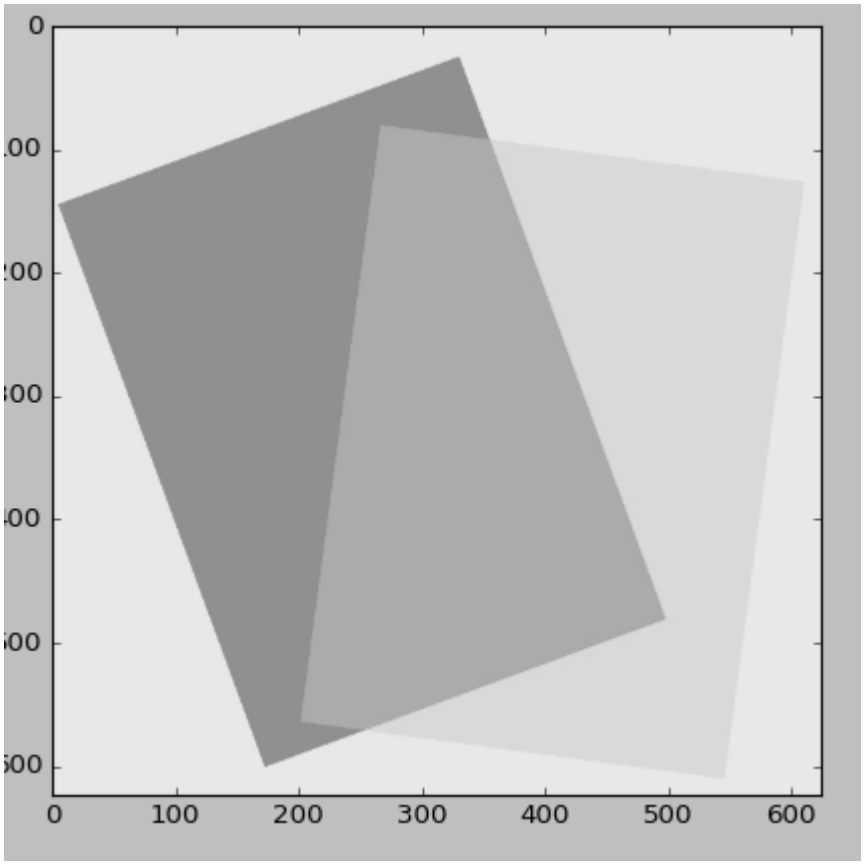
So I computed the following algorithm. Not sure if I understood the adding of the filters correctly, but here it is. The result can be seen above. The results show me clearly where the edges of the object in the pictures are, as they work as a vector for which intensity the pixels move towards, from either both top to bottom and left to right.

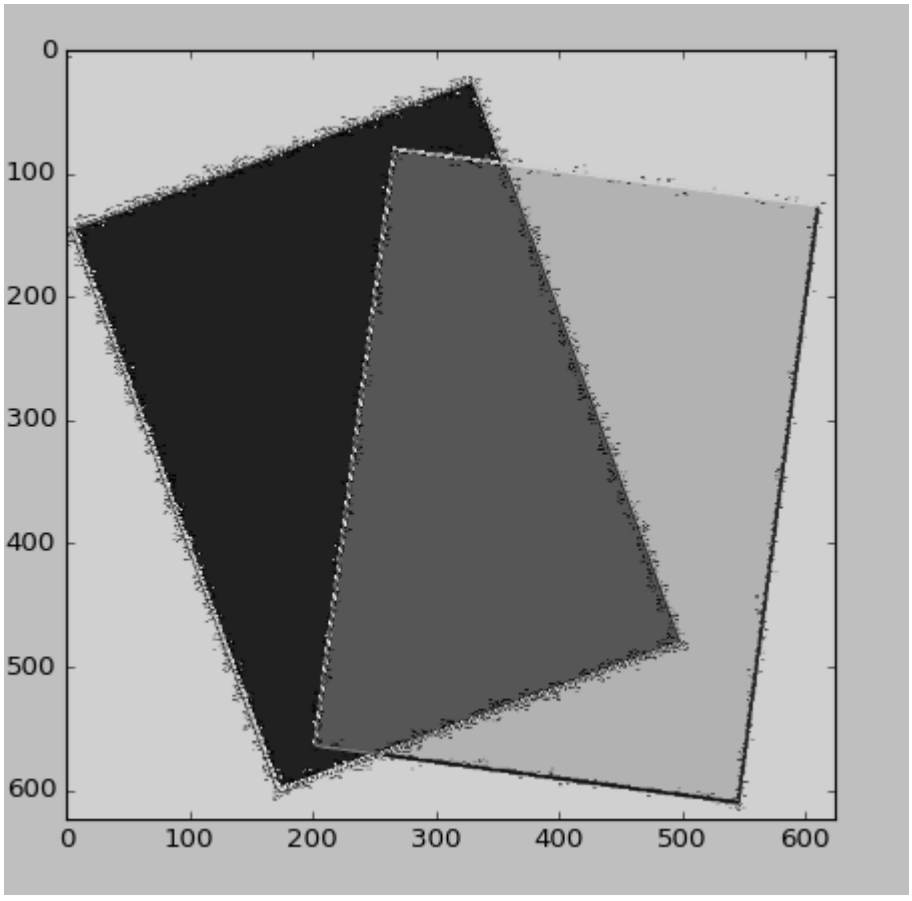
```
def convolutionMagnitude(H,Fx,Fy): #The function that invertes the picture

  hx = copy.copy(H) #makes aa copy of the image to write over
  hy = copy.copy(H) #makes aa copy of the image to write over
  h = copy.copy(H) #makes aa copy of the image to write over
  print(hx[100][100])
  o = int(float(len(Fx)/2)) #The offsett of the filter
  print(o)
  for x in range (o,len(H)-o):
    for y in range(o,len(H[x])-o):
      hx[x][y]=0 #Sets the current color at current pixel to zero
      hy[x][y]=0
      for i in range (0,len(Fx)):
        for j in range(0,len(Fx[i])):
            #adds the filters multiplier to the neighboorhood and adds it to the color
            hx[x][y][0] += H[x-o+i][y-o+j][0]*Fx[i][j]
            hy[x][y][0] += H[x-o+i][y-o+j][0]*Fy[i][j]
      #Calculates the new values
      vx = math.pow(hx[x][y][0],2)
      vy = math.pow(hy[x][y][0],2)
      v = vx + vy
      m=float(math.sqrt(v))#adds them into the return matrix
      for k in range(3):
        h[x][y][k]=m
  return h
```


0
100
200
300
400
500
0
100
200
300
400
500
600
700


0
100
200
300
400
500
0
100
200
300
400
500
600
700


0
100
200
300
400
500
600


0
100
200
300
400
500
600

Task 3 Exam Question:

Question: Pad the image two different ways and perform spatial correlation with kernel on the right. Compare the two results

0	**0**	**2**
0	**1**	**1**
3	**5**	**7**

0	-1	0
-1	4	-1
0	-1	0

Solution:

0	0	0	0	0
0	**0**	**0**	**2**	0
0	**0**	**1**	**1**	0
0	**3**	**5**	**7**	0
0	0	0	0	0

Padded with zeroes

0	-3	6
-4	-2	-6
7	9	22

0	0	0	2	2
0	**0**	**0**	**2**	2
0	**0**	**1**	**1**	1
3	**3**	**5**	**7**	7
3	3	5	7	7

0	-3	3
-4	-2	-7
1	4	8

Padded with extended edges

As we can see, there are big differences between the two ways of padding. Padding with zeros gives a higher overall intensity compared to padding my extending the edges.