Ankara University Computer Engineering

Com466B-Digital Image Processing

Practical Coursework as Midterm

Exam (2019-2020 Spring Semester)

- Instructor Name Dr. Kurtuluş KÜLLÜ

Student Name: Mehmet KEKEÇ

Student No:15290106

Com466B Practical Coursework as Midterm

IDE used in homework: Visual Studio Code

1. **Convert the image to a grayscale (monochrome) image using a library function.**

```
3   import cv2
4   import numpy as np
5
6   pictures=cv2.imread("picture.png",0) # read and assing real picture from folder
7   cv2.imshow("Midterm photo",pictures) # Display function of new photo on screen
8   cv2.imwrite("picturegray.png",pictures) # write and save new photo to folder
9
10  cv2.waitKey(0)
11  cv2.destroyAllWindows()
```

Method in line 6: cv.imread("picture.png",0) It saves the original photograph(RGB) that it reads from the folder as a single channel to the variable ,so it convert the image to grayscale (monochrome) image.

Method in line 7: cv.imshow("Midterm photo",pictures) This method displays the new version in the variable on the screen.

Method in line 8: cv.imwrite("picturegray.png",pictures) This method saves the new version shown on the screen to the folder with a new name.

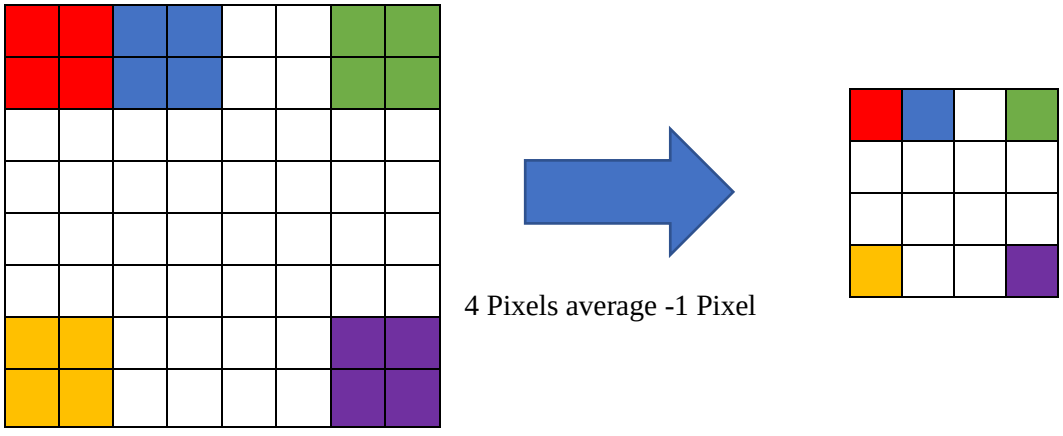
2. By writing your own function called resizeToQuarter, which takes in a grayscale image as input, make the image smaller so that the number of pixels is decreased to 1/4.

```
19  pictures=cv2.imread("picture.png",0)
20  cv2.imshow("Midterm photo",pictures)
21  cv2.imwrite("picturegray.png",pictures)
22
23  resizedpicture=np.zeros((int((pictures.shape[0])/2),int((pictures.shape[1])/2),1),np.uint8)
24  resizeToQuarter(resizedpicture,pictures)
25
26  cv2.waitKey(0)
27  cv2.destroyAllWindows()
```

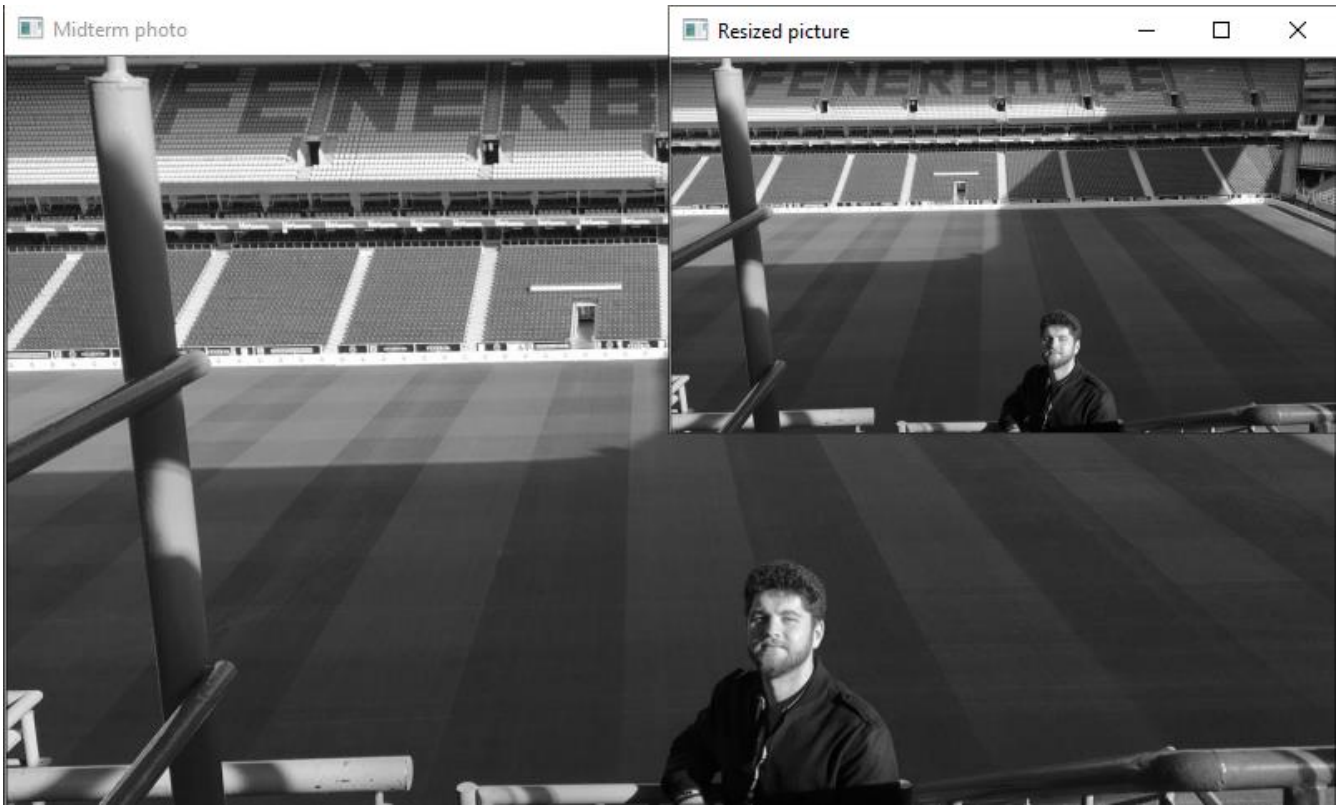
```
4   def resizeToQuarter(resizedpicture,pictures):
5       i=-1
6       j=-1
7       for num0 in range (0,int((pictures.shape[0]))-1,2):
8           j=-1
9           i=i+1
10          for num1 in range(0,int((pictures.shape[1]))-1,2):
11              j=j+1
12              resizedpicture[(i,j)]=(pictures[(num0,num1)])/4+\
13                  (pictures[(num0+1,num1)])/4+(pictures[(num0,num1+1)])/4+(pictures[(x+1,y+1)])/4
14      cv2.imshow("Resized picture", resizedpicture)
15      cv2.imwrite("Resized picture.png",resizedpicture)
```

ALGORİTHM

First of all, the width and length of gray photo were calculated (with pictures.shape[0], pictures.shape[0]). Then a new empty photo named x with half of both dimensions was created with np.zeros((x,y,1),np.unit8) methods from numpy library. Next Step is to send the empty photo and the gray photo to the created method named resizeToQuarter. This function takes the average of every 4 pixels of the gray photo and turns it into a pixel of the empty photo.



RESULT




```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
PS C:\Users\Mehmet KEKEÇ\Desktop\Görüntü> & D:/Anaconda3/envs/opencv/python.exe "c:/Users/Mehmet KEKEÇ/Desktop/Görüntü/Midterm1-2.py"
Pictures 450 * 800 = 360000 Pixels
Resized picture 225 * 400 = 90000 Pixels
PS C:\Users\Mehmet KEKEÇ\Desktop\Görüntü>
```

3. Write a function/method called mySharpening, which applies the Laplacian filter given below to its input image.

Given Filter:

-1	**-1**	**-1**
-1	**9**	**-1**
-1	**-1**	**-1**

$$Laplace(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

ALGORITHM

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	X	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	-1	-1	-1	x	x	-1	-1	-1	x	x	x	x	x	x	x	x	x	x
x	x	-1	9	-1	…	…	-1	9	-1	x	x	x	x	x	x	x	x	x	x
x	x	-1	-1	-1	x	x	-1	-1	-1	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Note: Each x value is different from each other and is in the range of [0.255].

After applying the filter for each pixel, the filter is moved in the direction of the arrow.

CODE:

```
31    picture = cv2.imread("picture.png",0)
32    cv2.imshow("First Version",picture)
33    mySharpening(picture)
34    cv2.waitKey()
```

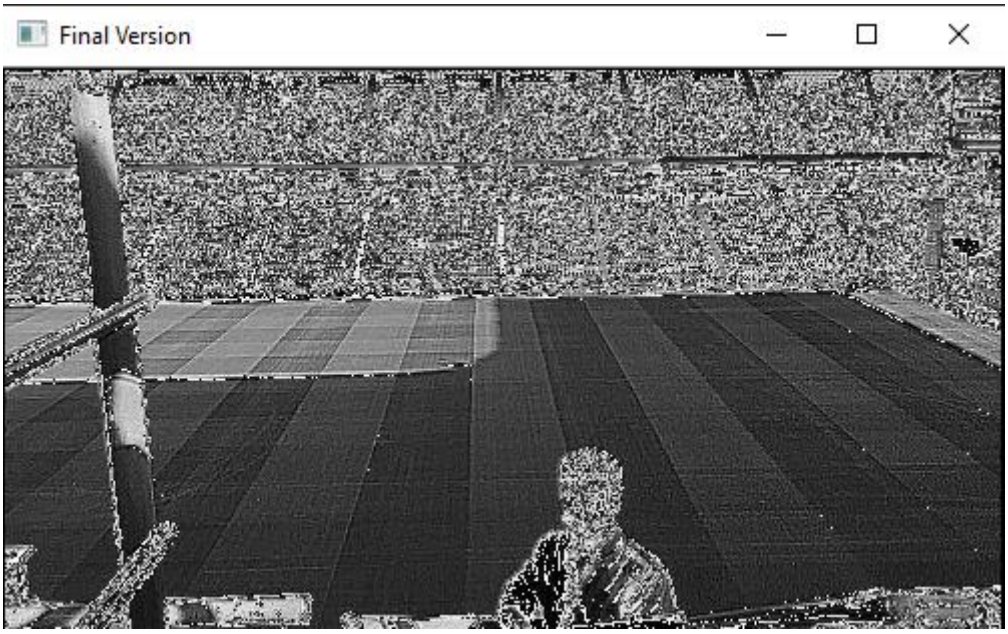
```
3   def mySharpening(picture):
4
5       laplacian =[[-1,-1, -1], [-1, 9, -1], [-1, -1, -1]]
6       lfilter=np.zeros((int(picture.shape[0]),int(picture.shape[1]),1),np.uint8)
7       filtered=np.zeros((int(picture.shape[0])*int(picture.shape[1]),1),'int32')
8       counter=0
9       maks=0
10      mini=0
11      x=int(picture.shape[0])
12      y=int(picture.shape[1])
13      for i in range(1,x-2,1):
14          for j in range(1,y-2,1):
15              filtered[counter]=laplacian[0][0]*picture[i-1][j-1] + laplacian[0][1]*picture[i-1][j+0] + \
16                  laplacian[0][2]*picture[i-1][j+1] + laplacian[1][0]*picture[i+0][j-1] + \
17                      laplacian[1][1]*picture[i+0][j+0] + laplacian[1][2]*picture[i+0][j+1] + \
18                          laplacian[2][0]*picture[i+1][j-1] + laplacian[2][1]*picture[i+1][j+0] + \
19                              laplacian[2][2]*picture[i+1][j+1]
20
21              if (filtered[counter] > 255):
22                  maks = filtered[counter]
23                  filtered[counter]=255
24              if (filtered[counter] < 0):
25                  mini = filtered[counter]
26                  filtered[counter]=0
27              lfilter[i][j]=filtered[counter]
28              counter=counter+1
29      print(maks, mini)
30      cv2.imwrite("deneme.png",lfilter)
31      cv2.imshow('Final Version', lfilter)
```

As can be seen from the code, after applying the filter, the values corresponding to each pixel are compressed into the range [0.255]. The reason for this is to prevent the excessive tips (-8 * 255 or 9 * 255) that may occur as a result of the process from distorting the sharpness in the photograph. For this reason, all shades smaller than 0 are synchronized to 0. All shades larger than 255 are equalized to 255. If we did not compress it to this range ([0.255]), the photos could have been more gray in order to avoid this, noise removal may have to be done in the image.

RESULT

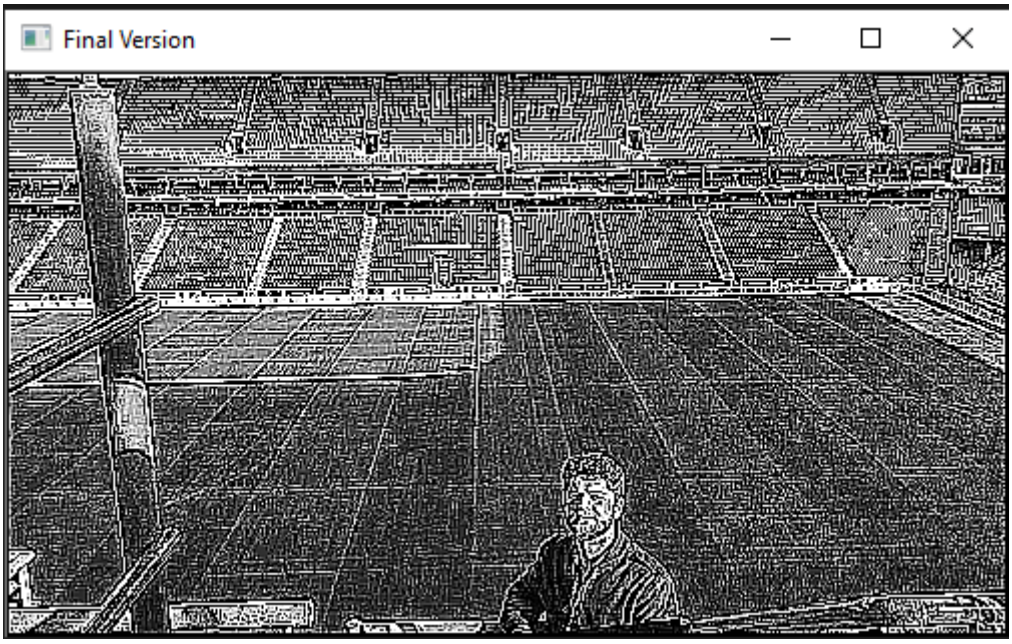

```
PS C:\Users\Mehmet KEKEÇ\Desktop\Görüntü> & D:/Anaconda3/envs/opencv/python.exe "c:/Users/Mehmet KEKEÇ/Desktop/Görüntü/deneme.py"
[255] [0]
```

What would happen if the values were not compressed?




```
PS C:\Users\Mehmet KEKEÇ\Desktop\Görüntü> & D:/Anaconda3/envs/opencv/python.exe "c:/Users/Mehmet KEKEÇ/Desktop/Görüntü/deneme.py"
[1419] [-980]
```

What would happen if laplacian was applied twice?



4. You are given a noisy grayscale image together with this document. Perform noise removal to improve this image.

First of all, the photo given was reduced by the ready-made re-size function.

```
import cv2
image=cv2.imread("Task.jpg")
image=cv2.resize(image,(500,282))
cv2.imwrite("Task.png",image)
cv2.imshow("Midterm photo",image)
cv2.waitKey(0)
```

Secondly, The exam photo was processed with the median blur function. The function smoothes an image using the median filter with the ksize*ksize aperture. Each channel of a multi-channel image is processed independently.

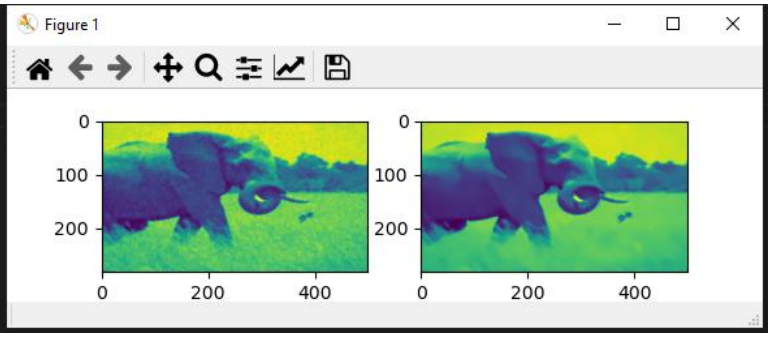
```
import cv2

img = cv2.imread('Task.png',0) # Load image
img_median = cv2.medianBlur(img, 5) # Add median filter to image

cv2.imshow('img', img_median) # Display img with median filter
cv2.imwrite("median.png",img_median)
cv2.waitKey(0)          # Wait for a key press to
cv2.destroyAllWindows # close the img window.
```

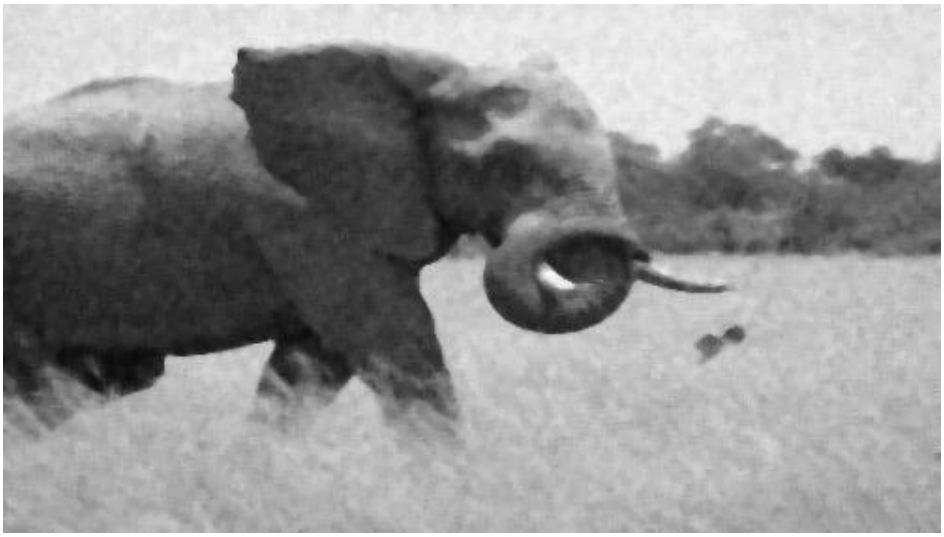
Thirdly, The fastNlMeansDenoising () method was used to further reduce the noise.

```
secondnoisy.py > ...
1   import numpy as np
2   import cv2
3   from matplotlib import pyplot as plt
4   img = cv2.imread('median.png',0)
5   dst = cv2.fastNlMeansDenoising(img,None,9,13)
6   cv2.imshow("ilk",img)
7   cv2.imshow("son",dst)
8   cv2.imwrite("noisy1.png",dst)
9   plt.subplot(121), plt.imshow(img)
10  plt.subplot(122), plt.imshow(dst)
11  plt.show()
12  cv2.waitKey()
```



This method did not work very functional and effective for this picture.

To be seen more clearly;



In the next step, the median method was reapplied and the photos were compared.

```python
import cv2
import numpy as np
img = cv2.imread('noisy1.png',0)
median = cv2.medianBlur(img, 5)
compare = np.concatenate((img, median), axis=1) #side by side comparison
cv2.imshow('img', compare)
cv2.waitKey(0)
cv2.destroyAllWindows
```

Again, it produced a successful result.

As a result,the most important conclusion after using these filters is still producing successful results even when the median blur function is used more than once. Another result is that the median blur function produces less successful results than the fastNlMeansDenoising method. At this point, the situation that should not be forgotten and should be noted is that; The parameter values of the fastNlMeansDenoising () method must be set very well. Otherwise, the desired results cannot be obtained, and in some cases the filter does not make any changes.

Some other filter results;

APPENDICES- CODE

1-)

```
import cv2
import numpy as np

pictures=cv2.imread("picture.png",0) # read and assing real picture from folde
r
cv2.imshow("Midterm photo",pictures) # Display function of new photo on screen
cv2.imwrite("picturegray.png",pictures) # write and save new photo to folder

cv2.waitKey(0)
cv2.destroyAllWindows()
```

2-)

```
import cv2

import numpy as np

def resizeToQuarter(resizedpicture,pictures):
    i=-1
    j=-1
    for num0 in range (0,int((pictures.shape[0]))-1,2):
        j=-1
        i=i+1
        for num1 in range(0,int((pictures.shape[1]))-1,2):
            j=j+1
            resizedpicture[(i,j)]=(pictures[(num0,num1)])/4+\
                (pictures[(num0+1,num1)])/4+(pictures[(num0,num1+1)])/4+(pictu
res[(num0+1,num1+1)])/4
    cv2.imshow("Resized picture", resizedpicture)
    cv2.imwrite("Resized picture.png",resizedpicture)
    print("Pictures",pictures.shape[0],"*",pictures.shape[1],"=",pictures.shap
e[0]*pictures.shape[1],'Pixels')
    print("Resized picture",resizedpicture.shape[0],'*',resizedpicture.shape[1
],"=",resizedpicture.shape[0]*resizedpicture.shape[1],'Pixels')

pictures=cv2.imread("picture.png",0)
cv2.imshow("Midterm photo",pictures)
cv2.imwrite("picturegray.png",pictures)

resizedpicture=np.zeros((int((pictures.shape[0])/2),int((pictures.shape[1])/2)
,1),np.uint8)
resizeToQuarter(resizedpicture,pictures)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

3-)

```
import cv2
import numpy as np
def mySharpening(picture):

    laplacian =[[-1,-1, -1], [-1, 9, -1], [-1, -1, -1]]
    lfilter=np.zeros((int(picture.shape[0]),int(picture.shape[1]),1),np.uint8)
    filtered=np.zeros((int(picture.shape[0])*int(picture.shape[1]),1),'int32')
    counter=0
    maks=0
    mini=0
    x=int(picture.shape[0])
    y=int(picture.shape[1])
    for i in range(1,x-2,1):
        for j in range(1,y-2,1):
            filtered[counter]=laplacian[0][0]*picture[i-1][j-
1] + laplacian[0][1]*picture[i-1][j+0] + \
                laplacian[0][2]*picture[i-
1][j+1] + laplacian[1][0]*picture[i+0][j-1] + \
                    laplacian[1][1]*picture[i+0][j+0] + laplacian[1][2]*pict
ure[i+0][j+1] + \
                            laplacian[2][0]*picture[i+1][j-
1] + laplacian[2][1]*picture[i+1][j+0] + \
                                laplacian[2][2]*picture[i+1][j+1]

            if (filtered[counter] > 255):
                maks = filtered[counter]
                filtered[counter]=255
            if (filtered[counter] < 0):
                mini = filtered[counter]
                filtered[counter]=0
            lfilter[i][j]=filtered[counter]
            counter=counter+1
    print(maks, mini)
    cv2.imwrite("deneme.png",lfilter)
    cv2.imshow('Final Version', lfilter)
    return lfilter

picture = cv2.imread("picture.png",0)
cv2.imshow("First Version",picture)
img=cv2.imread("denem.png",0)
mySharpening(picture)
mySharpening(img)
cv2.waitKey()
```

4-)

Resize Step

```
import cv2
image=cv2.imread("Task.jpg")
image=cv2.resize(image,(500,282))
cv2.imwrite("Task.png",image)
cv2.imshow("Midterm photo",image)
cv2.waitKey(0)
```

First Step Noising

```
import cv2
img = cv2.imread('Task.png',0) # Load image
img_median = cv2.medianBlur(img, 5) # Add median filter to image
cv2.imshow('img', img_median) # Display img with median filter
cv2.imwrite("median.png",img_median)
cv2.waitKey(0)         # Wait for a key press to
cv2.destroyAllWindows # close the img window.
```

Second Step Noising

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
img = cv2.imread('median.png',0)
dst = cv2.fastNlMeansDenoising(img,None,9,13)
cv2.imshow("ilk",img)
cv2.imshow("son",dst)
cv2.imwrite("noisy1.png",dst)
plt.subplot(121), plt.imshow(img)
plt.subplot(122), plt.imshow(dst)
plt.show()
cv2.waitKey()
```

Third Step Noising

```
import cv2
import numpy as np
img = cv2.imread('noisy1.png',0)
median = cv2.medianBlur(img, 5)
compare = np.concatenate((img, median), axis=1) #side by side comparison
cv2.imshow('img', compare)
cv2.waitKey(0)
cv2.destroyAllWindows
```