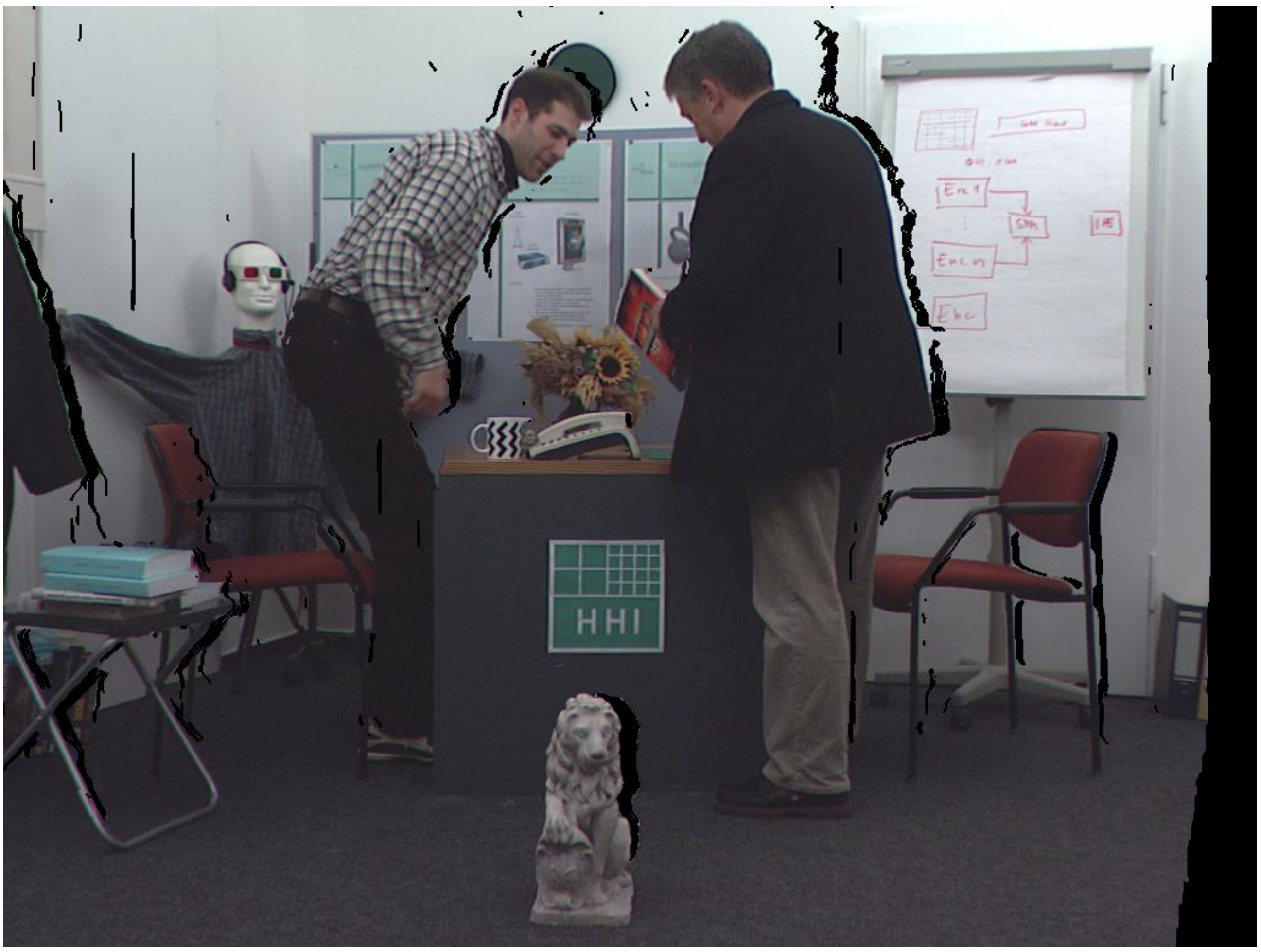
Assignment -2

Report

The aim of this assignment is to make a code so that the computer can vision how the human eye perceives the quality of an image. We have been given 84 images and the subjective score for all of them. The images given are distorted. Visually they have either black patches or are hazy.

Firstly, we will count the number of black pixels in the black patches.



We break the image into R, G and B layers and apply the following individually:

The above image is 3.png. we can see the black patches and distortions. To obtain this we will use median filter to blur the image as I am taking the black patches as dead pixels. Now the resultant blurred image is subtracted from the original image. To remove the noise, we apply the closing morphological operations. The resultant image is as given below:

The black patches

```
In [1]: import cv2
import numpy as np
import xlrd
import scipy
from scipy.stats.stats import pearsonr
from scipy.optimize import curve_fit
In [2]: ObjectiveScore = []
```

```
SubjectiveScore=[]
In [3]: for z in range (0,84):
img = cv2.imread(f"Assignment2/DIBR_Data/{z+1}.png",0);
length = img.shape[0]
width = img.shape[1]
count = 0;
for i in range(0,length):
for j in range(0,width):
if img[i][j] < 16:
count = count + 1
ObjectiveScore.append(count)
In [4]: loc = ("Assignment2/DIBR_Data/DMOS_DIBR.xlsx")
wb = xlrd.open_workbook(loc)
sheet = wb.sheet_by_index(0)
for i in range(sheet.nrows):
SubjectiveScore.append(sheet.cell_value(i, 0))
In [5]: PLCC = pearsonr(SubjectiveScore,ObjectiveScore)
print(PLCC)
(-0.6355511439245668, 8.328372635432614e-11)
In [6]: def GaussianFilter(size):
output = np.zeros([512 - size + 1, 512 - size + 1])
for i in range(0,512-size+1):
1
for j in range(0,512-size+1):
sum=0
for k in range(i,i+size):
for l in range(j,j+size):
output[i][j]=output[i][j] + img[k][l]*math.exp(-1*((k-i-(size//2))*(sum+=math.exp(-
1*((k-i-(size//2))*(k-i-(size//2)) + (l-j-(size//2))*(output[i][j]=output[i][j]/sum
return output
In [7]: def BilateralFilter(size):
output = np.zeros([512 - size + 1, 512 - size + 1])
for i in range(0,512-size+1):
for j in range(0,512-size+1):
sum=0
for k in range(i,i+size):
for l in range(j,j+size):
output[i][j]=output[i][j] + img[k][l]*math.exp(-1*((k-i-(size//2))*(sum+=math.exp(-
1*((k-i-(size//2))*(k-i-(size//2)) + (l-j-(size//2))*(output[i][j]=output[i][j]/sum
return output
In [8]: def func(x, a, b, c):
return a * np.exp(-b * x) + c
In [ ]:
```

```
In [9]: for i in range(0,84):
ObjectiveScore[i]= ObjectiveScore[i]/5000
In [10]: popt, pcov = curve_fit(func, ObjectiveScore, SubjectiveScore)
In [11]: popt
Out[11]: array([2.23916942, 0.09116123, 1.29475961])
In [12]: pcov
Out[12]: array([[ 0.90966384, -0.05361686, -0.91930023],
[-0.05361686, 0.00347424, 0.05499509],
[-0.91930023, 0.05499509, 0.93425164]])
In [34]: NewObjectiveScore = []
for i in range (0,84):
NewObjectiveScore.append(func(ObjectiveScore[i],*popt))
In [41]: # for i in range (0,84):
# NewObjectiveScore[i] = NewObjectiveScore[i] + 0.4
2
In [42]: PLCC = pearsonr(SubjectiveScore,NewObjectiveScore)
print(PLCC)
(0.737560418368689, 1.2286194889387519e-15)
```

Taking the threshold value of 16 we will count the number of black pixels in the above image and obtain a score.

Secondly to see how much hazy an image is we will use Laplacian function. The variance of the Laplacian on an image determines the amount of blur.

We will now use curve fitting. We have used SciPy. Curve_fit and taken e^x function to fit the curve.

Result: We got the PLCC value of 0.73 value of PLCC after the above method