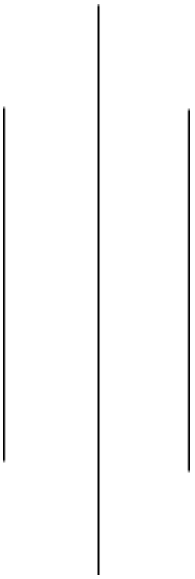
An
Assignment Report
on
Image Analysis and Computer Vision
Spring 2021
(CS 898BA)



Submitted To
Dr. Ajita Rattani
(Course Instructor)

Submitted By
Suresh Pokharel
WSUID: X254W356
Department of Electrical Engineering and Computer Science
Wichita State University

28 February, 2021

Problem:1

Download PASCAL VOC 2012 dataset available at this link: http://host.robots.ox.ac.uk/pascal/VOC/. (50 points)

- ❏ On a subset of 50 images from PASCAL VOC dataset, perform data augmentation by applying affine transformation to the images (translation, rotation, scaling etc..)
- ❏ On a randomly selected image from PASCAL VOC dataset compute (a) image statistics, (b) histogram, (c) histogram equalization and (d) binarize it.

Solution:

Import required libraries

```
import os
import random
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

Define the path and list the images

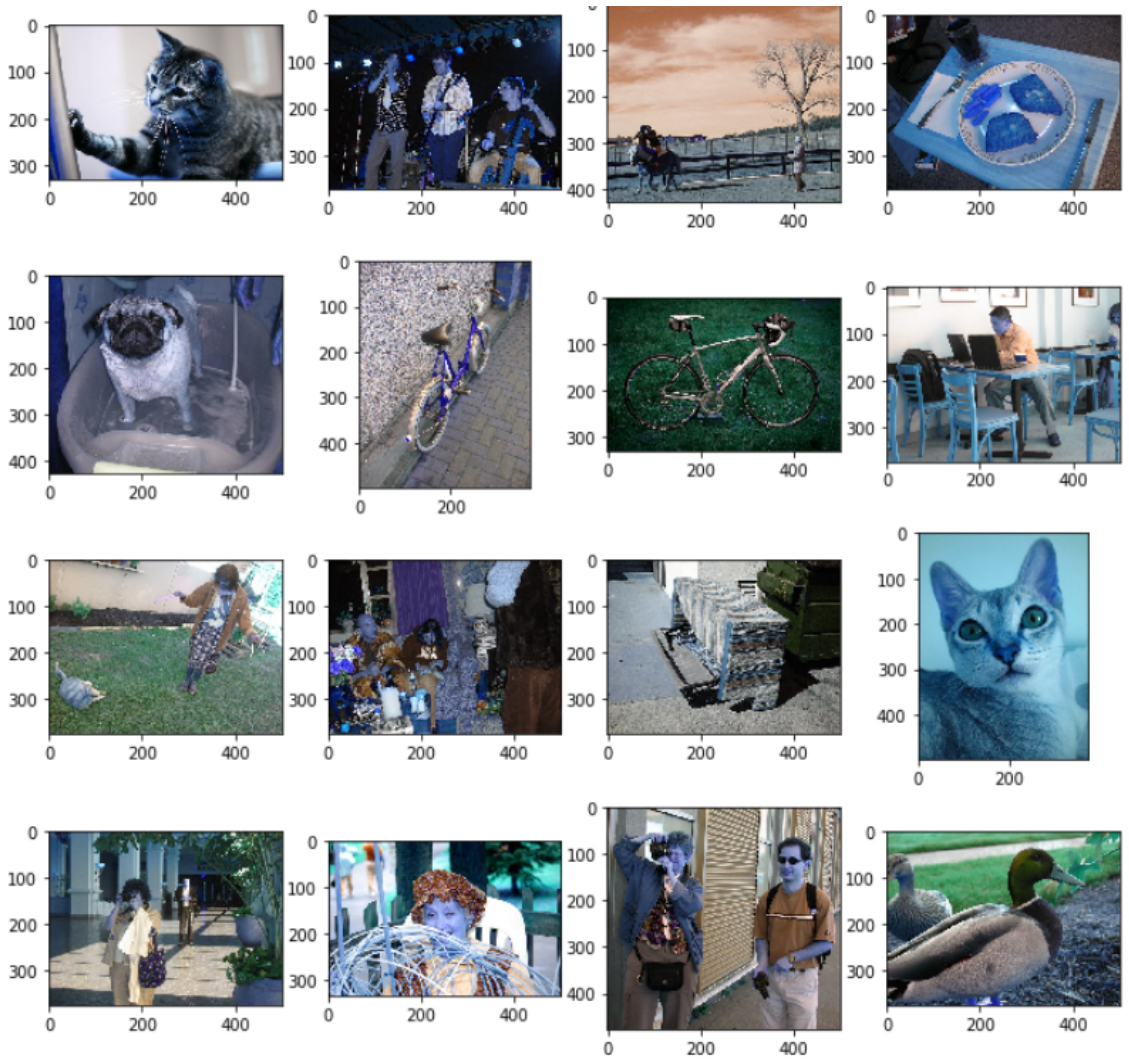
```
# reading files from the given path
data_path = 'VOCdevkit\VOC2012\JPEGImages\\'
files = os.listdir(data_path)
```

Shuffle the image list to pick the random images from VOC2012 dataset of 16135 images and pick first 50 images from the shuffled list

```
 # shuffle list to select the random images
random.shuffle(files,random.random)

# select first 50 images and place in images
images = files[0:50]
```

show some original images read from dataset

```
# Make a plot of 484 grid and display images
fig = plt.figure(figsize=(12,12))
rows = 4
cols = 4
for i in range(1,rows*cols+1):
    img = cv2.imread(data_path+images[i])
    fig.add_subplot(rows,cols,i)
    plt.imshow(img)
plt.show()
```



Apply affine transformation to each image. The transformation matrix is formed for translation, rotation, and scaling. Then augmented image is created using cv2's warpAffine() method.

To avoid heavy computation for now, translation, rotation, and scaling of 4 images are plotted in the grid. Augmented images can be saved in disk using cv2's imwrite() method if needed.

```
# Affine transformation
for idx,image in enumerate(images):
    img = cv2.imread(data_path+image) # read an image
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    rows,cols,channel = img.shape
    # translation
    M = np.float32([[1,0,100],[0,1,50]])
    img_translation = cv2.warpAffine(img,M,(cols,rows))

    # rotation
    M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
    img_rotation = cv2.warpAffine(img,M,(cols,rows))

    # scaling
    img_scaling = cv2.resize(img,(2*cols, 2*rows), interpolation = cv2.INTER_CUBIC)

    # display results of 4 images only
    # It's heavy to show all here
    if(idx<4):
        fig = plt.figure(figsize=(12,12))
        ax1 = fig.add_subplot(4,4,4*idx+1)
        ax1.title.set_text('Original')
        plt.imshow(img,cmap='gray')
        ax2 = fig.add_subplot(4,4,4*idx+2)
        ax2.title.set_text('Translation')
        plt.imshow(img_translation)
        ax3 = fig.add_subplot(4,4,4*idx+3)
        ax3.title.set_text('Rotation')
        plt.imshow(img_rotation)
        ax4 = fig.add_subplot(4,4,4*idx+4)
        ax4.title.set_text('Scaling')
        plt.imshow(img_scaling)
        plt.show()
```

On a randomly selected image from PASCAL VOC dataset compute: (a) image statistics, (b) histogram, (c) histogram equalization and (d) binarize it.

9]:
```
random_image = images[0] # the list is already shuffled, so taking the first element
image_cv = cv2.imread(data_path + random_image)

# convert BGR to RGB channels
image_cv = cv2.cvtColor(image_cv, cv2.COLOR_BGR2RGB)
```

Calculate Mean, SD and Median of each channel(RGB) of the selected image using cv2's meanStdDev() method

1]:
```
# image statistics: it returns mean and sd
stat = cv2.meanStdDev(image_cv)
mean = stat[0].flatten() #flattened to convert into 1d array
sd = stat[1].flatten()
print("The means for RGB channel is: ", mean)
print("The sd for RGB channel is: ", sd)
```

```
The means for RGB channel is:  [109.24736193  94.80538338  82.70827882]
The sd for RGB channel is:  [52.43141106 48.84321913 50.63719043]
```

Calculate Median of each channel(RGB) of the selected image using median function from statistics module.

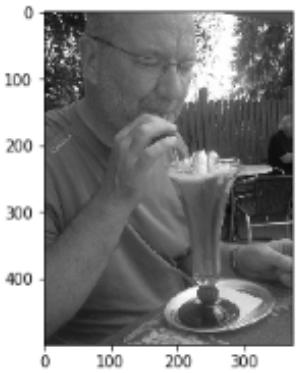
```
#calculate median
from statistics import median

# splitting the RGB component
r,g,b = cv2.split(image_cv)

# flatten the 2d array into 1D and find the median of individual channel
med = [median(r.flatten()), median(g.flatten()), median(b.flatten())]

print("The median for RGB channel is: ", med)
```

```
The median for RGB channel is:  [102.0, 84.0, 71.0]
```

Convert the image into grayscale and create histogram

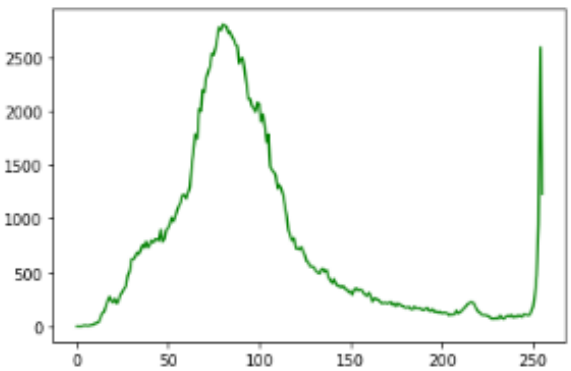
```
# convert the image to grayscale so that we can plot the histogram
image_cv_bw = cv2.cvtColor(image_cv,cv2.COLOR_RGB2GRAY)
plt.imshow(image_cv_bw, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x23154a6dac8>
```



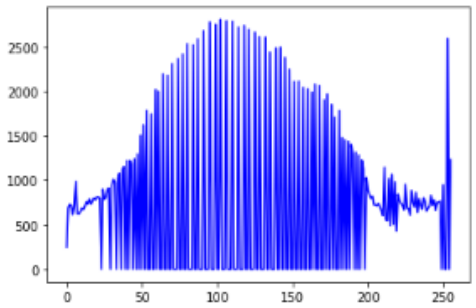

```
hist = cv2.calcHist([image_cv_bw],[0],None,[256],[0,256])
# plot in green
plt.plot(hist,'g')
```

```
[<matplotlib.lines.Line2D at 0x23154694988>]
```



Perform histogram equilization operation using cv2's equalizeHist method. Histogram equilization is a contrast stretching algorithm which is used to adjust the contrast by altering the intensity distribution of the histogram.

```
# histogram equilization
image_after_equilization = cv2.equalizeHist(image_cv_bw)
hist = cv2.calcHist([image_after_equilization],[0],None,[256],[0,256])
plt.plot(hist,'b')
plt.show()
```




```
fig = plt.figure(figsize=(12,12))
ax1 = fig.add_subplot(1,2, 1)
ax1.title.set_text("Before Histogram Equilization")
plt.imshow(image_cv_bw,cmap='gray')
ax2 = fig.add_subplot(1,2, 2)
ax2.title.set_text("After Histogram Equilization")
plt.imshow(image_after_equilization, cmap='gray')
plt.show()
```

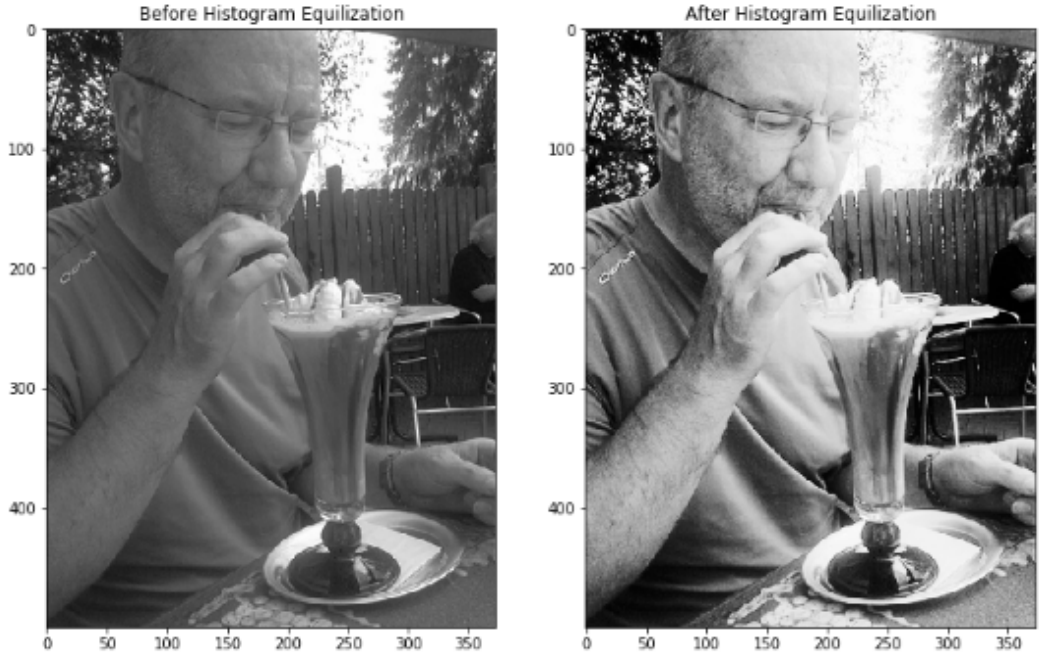


Image Binarization

```
# Image Binarization
ax,thresh = cv2.threshold(image_after_equilization,127,255,cv2.THRESH_BINARY)
```

```
fig = plt.figure(figsize=(12,12))
fig.add_subplot(1,2, 1)
plt.imshow(thresh,'gray',vmin=0,vmax=255)
```

```
182]: <matplotlib.image.AxesImage at 0x23157c47e88>
```

Problem:2

Apply inpainting operation based on the algorithm Fast Marching to remove glare from the below image. The corresponding mask is also given. (50 points)

Notes: Glare is a visual sensation caused by excessive and uncontrolled brightness. It can be disabling or simply uncomfortable.

Solution:

Image inpainting is a process of removing unwanted bad marks in the image by replacing them with neighbouring pixels. The Fast Marching Method is one of the image inpainting algorithms.

The algorithm begins from the boundary of the region to be inpainted and fills the boundary pixels first. After that, It considers a small neighbourhood around the pixel to be inpainted and that is replaced by a weighted sum of the neighbouring pixels.

```
# Image Inpainting
img = cv2.imread('Image_with_glare.JPEG') # read the image
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
mask = cv2.imread('Mask.JPEG',0) # read the mask

result = cv2.inpaint(img,mask,3,cv2.INPAINT_TELEA) # apply the INPAINT_TELEA algorithm from opencv

# plot the images
fig = plt.figure(figsize=(12,12))
ax1 = fig.add_subplot(221)
ax1.set_title("Before Inpainting")
plt.imshow(img)
ax2 = fig.add_subplot(222)
ax2.set_title("After Inpainting")
plt.imshow(result)
plt.show()
```