YILDIZ TECHNICAL UNIVERSITY
ELECTRICAL- ELECTRONICS
FACULTY
COMPUTER ENGINEERING
DEPARTMENT

Image Processing Lecture
Second Assignment

KAMRAN BALAYEV 17011904

Content Based Image Retrieval

In this assignment, i will explain content based image retrieval application.

- Firstly , i need a normalization function in order to normalize histogram values

```
# min max normalization function
def Normalization(maxNum, minNum, arr):
    arr2 = np.zeros((len(arr)), dtype=np.float32)
    for i in range(len(arr)):
        arr2[i] = (arr[i] - minNum) / (maxNum - minNum)
    return arr2
```

- Secondly, i need to get histogram of H value of HSV color space. I did it with createHueHistogram named function

```
def createHueHistogram(img1):
    hsvImage = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
    Hue_Hist = np.zeros((256), dtype=np.float32)
    for i in hsvImage[:, :, 0]:
        Hue_Hist[i] = Hue_Hist[i] + 1
    maxH = max(Hue_Hist)
    minH = min(Hue_Hist)
    normalHueHistogram = Normalization(maxH, minH, Hue_Hist)
    return normalHueHistogram
```

- After then, i have calculated r,g and b histograms of RBG color space and then i store all 3 histogram values in one list.

```python
#create histogram
def createHistogram(arr):
    hist = np.zeros((256), dtype=np.float32)
    for i in arr:
        hist[i] = hist[i] + 1
    return hist
```

```python
def createRGBhistogram(img):
    histogramList = []
    # Red
    red = img[:, :, 0]
    redHistogram = createHistogram(red)
    # Red Normalization
    maximumRed = max(redHistogram)
    minimumRed = min(redHistogram)
    normalRedHistogram = Normalization(maximumRed, minimumRed, redHistogram)
    # Green
    green = img[:, :, 1]
    greenHistogram = createHistogram(green)
    # Green Normalization
    maximumGreen = max(greenHistogram)
    minimumGreen = min(greenHistogram)
    normalGreenHistogram = Normalization(maximumGreen, minimumGreen, greenHistogram)
    # Blue
    blue = img[:, :, 2]
    blueHistogram = createHistogram(blue)
    # Blue Normalization
    maximumBlue = max(blueHistogram)
    minimumBlue = min(blueHistogram)
    normalBlueHistogram = Normalization(maximumBlue, minimumBlue, blueHistogram)
    # Adding to list
    histogramList.append(normalRedHistogram)
    histogramList.append(normalGreenHistogram)
    histogramList.append(normalBlueHistogram)
    return histogramList
```

- Calculating Euclidean Distance (L2 norm)

```python
#calculate ecludian distance
def Euclidian(arr1, arr2):
    eucArr = 0.0
    for i in range(len(arr1)):
        eucArr = eucArr + pow((arr1[i] - arr2[i]), 2)
    return math.sqrt(eucArr)
```

- After calculating Euclidean Distance between train and test images find the minimum 5 values and return this list

```python
#After calculating euc distances between train and test images find the minimum 5 values with this function
def findMinVals(tmp,minimumTmp,element):
    # define minimum value minimumTmp with high value in order to reach the minimum values in list
    finalList = []
    flag = []
    # find minimum values remove them from tmp list then find other values repeat 5 times
    for x in range(5):
        for y in tmp:
            if (y[element] < minimumTmp):
                minimumTmp = y[element]
                flag = y.copy()
                tmp.remove(y)
        finalList.append(flag)
        minimumTmp = 10000
    return finalList
```

- Store png files in lists, one for train images and other one for test images. Get images from folders with for loop

```
pngfiles = [] #store train files
pngFilesTest = [] #store test files

for file in glob.glob("train/*.jpg"):
    pngfiles.append(file)

for file in glob.glob("test/*.jpg"):
    pngFilesTest.append(file)
```

- Create an image dictionary in order to store all histogram values

```
dictionaryList = []
imageDictionary = {
    "Image1": "img1",
    "Image2": "img2",
    "R_Dist": 0,
    "G_Dist": 0,
    "B_Dist": 0,
    "H_Dist": 0
}
```

- Get train and test photos from lists, and calculate distances between train and test images. After then, save results to the dictionary and then append dictionary to the list for creating dictionary list.

```
# read train photo from folders
for i in range(len(pngfiles) - 1):
    img1 = cv2.imread(pngfiles[i])
    img1_hue_hist = createHueHistogram(img1)
    #read test photo from folder
    for a in range(i, len(pngFilesTest)):
        # read test photo
        img2 = cv2.imread(pngFilesTest[a])
        img2_hue_hist = createHueHistogram(img2) #create hue histogram of photo
        hueEucDist = calculateImgEuc(img1_hue_hist, img2_hue_hist)
        rgbList = rbgEucCalc(img1, img2)
        imageDictionary["Image1"] = pngfiles[i]
        imageDictionary["Image2"] = pngFilesTest[a]
        imageDictionary["R_Dist"] = rgbList[0]
        imageDictionary["G_Dist"] = rgbList[1]
        imageDictionary["B_Dist"] = rgbList[2]
        imageDictionary["H_Dist"] = hueEucDist
        dictionaryList.append(imageDictionary.copy())
```

- Find 5 minimum values of all histogram values and save results to the list

```
# CREATE LIST FOR R
RfinalList=findMinVals(dictionaryList.copy(), 5000, "R_Dist")

# CREATE LIST FOR G
GfinalList = findMinVals(dictionaryList.copy(), 5000, "G_Dist")

# CREATE LIST FOR B

BfinalList = findMinVals(dictionaryList.copy(), 5000, "B_Dist")

# CREATE LIST FOR H
HfinalList = findMinVals(dictionaryList.copy(), 5000, "H_Dist")
```

➢ Print result

```
print("R")
for list in RfinalList:
    print(list)
    print('\n')

print("G")
for list in GfinalList:
    print(list)
    print('\n')

print("B")
for list in BfinalList:
    print(list)
    print('\n')

print("H")
for list in HfinalList:
    print(list)
    print('\n')
```

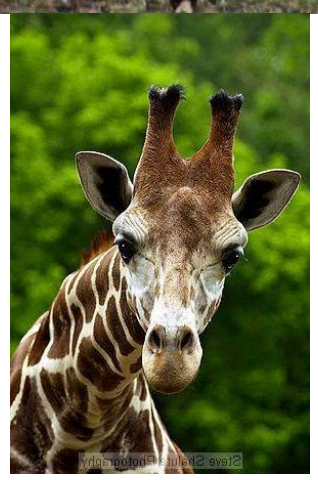
Output of Program

- Success percentage of R : 0/5 = 0
- Success percentage of G : 3/5 = 0.6
- Success percentage of B : 0/5 = 0
- Success percentage of H : 2/5 = 0.4

Succes rate of this method is low. 3 separate histograms are created for RGB components. This method is not a very effective way.

Succesful Result

Unsuccessfull Result



```
R
{'Image1': 'train/105_0007.jpg', 'Image2': 'test/084_0080.jpg', 'R_Dist': 1.54278009774647 43, 'G_Dist': 2.707782715288 45, 'B_Dist': 2.591763713913779, 'H_Dist': 5.718114480768 25}

{'Image1': 'train/089_0025.jpg', 'Image2': 'test/056_0098.jpg', 'R_Dist': 1.8006349034326414, 'G_Dist': 2.8617810574511577, 'B_Dist': 3.6540064248096207, 'H_Dist': 4.536939492963892}

{'Image1': 'train/028_0032.jpg', 'Image2': 'test/056_0099.jpg', 'R_Dist': 1.8679434165176967, 'G_Dist': 3.3282723496997666, 'B_Dist': 4.965970818809444, 'H_Dist': 5.3117886482479 75}

{'Image1': 'train/089_0020.jpg', 'Image2': 'test/056_0100.jpg', 'R_Dist': 1.9345363209911555, 'G_Dist': 5.395927613139046, 'B_Dist': 5.5224796798444 89, 'H_Dist': 3.674254898721961}

{'Image1': 'train/028_0030.jpg', 'Image2': 'test/084_0080.jpg', 'R_Dist': 2.1498012285209342, 'G_Dist': 5.151655037061805, 'B_Dist': 4.5482491572687 53, 'H_Dist': 6.92972166050639}

G
{'Image1': 'train/089_0010.jpg', 'Image2': 'test/089_0110.jpg', 'R_Dist': 4.4246537477299 26, 'G_Dist': 1.9870223665512872, 'B_Dist': 1.9327688924611905, 'H_Dist': 8.142174045330513}

{'Image1': 'train/056_0006.jpg', 'Image2': 'test/056_0099.jpg', 'R_Dist': 3.0777753593936 77, 'G_Dist': 2.0567663130335903, 'B_Dist': 5.131625696589952, 'H_Dist': 4.700395913149043}

{'Image1': 'train/057_0025.jpg', 'Image2': 'test/089_0110.jpg', 'R_Dist': 5.1448149030937 92, 'G_Dist': 2.2516103436044386, 'B_Dist': 2.4467361830831 87, 'H_Dist': 5.326064398058834}

{'Image1': 'train/028_0030.jpg', 'Image2': 'test/084_0081.jpg', 'R_Dist': 7.2346412901895 86, 'G_Dist': 2.548352113131755, 'B_Dist': 1.708375231342 96, 'H_Dist': 6.420309768571687}

{'Image1': 'train/028_0027.jpg', 'Image2': 'test/028_0108.jpg', 'R_Dist': 6.6626788251471 11, 'G_Dist': 2.6358734530304613, 'B_Dist': 4.5197119880237 21, 'H_Dist': 3.7363693975785943}

B
{'Image1': 'train/028_0030.jpg', 'Image2': 'test/084_0081.jpg', 'R_Dist': 7.2346412901895 86, 'G_Dist': 2.548352113131755, 'B_Dist': 1.708375231342 96, 'H_Dist': 6.420309768571687}

{'Image1': 'train/057_0015.jpg', 'Image2': 'test/084_0081.jpg', 'R_Dist': 9.0202862687164 08, 'G_Dist': 7.10554736752367, 'B_Dist': 2.040140908800692, 'H_Dist': 5.507756797262798}

{'Image1': 'train/105_0012.jpg', 'Image2': 'test/028_0108.jpg', 'R_Dist': 3.0842701351396 52, 'G_Dist': 4.8296486593368195, 'B_Dist': 2.065863353237996, 'H_Dist': 3.47787243032492}

{'Image1': 'train/057_0025.jpg', 'Image2': 'test/089_0110.jpg', 'R_Dist': 5.1448149030937 92, 'G_Dist': 2.2516103436044386, 'B_Dist': 2.4467361830831 87, 'H_Dist': 5.326064398058834}

{'Image1': 'train/105_0012.jpg', 'Image2': 'test/089_0110.jpg', 'R_Dist': 3.8531775378160 26, 'G_Dist': 4.531071889856677, 'B_Dist': 2.6419406190859833, 'H_Dist': 5.195801969809563}

H
{'Image1': 'train/057_0015.jpg', 'Image2': 'test/057_0102.jpg', 'R_Dist': 6.3071228096136 02, 'G_Dist': 4.178717846731541, 'B_Dist': 5.509584627298544, 'H_Dist': 1.207650917854032}
```

{'Image1': 'train/028_0030.jpg', 'Image2': 'test/084_0080.jpg', 'R_Dist': 2.1498012285209342, 'G_Dist': 5.151655037061805, 'B_Dist': 4.548249157268753, 'H_Dist': 6.92972166050639}

G
{'Image1': 'train/089_0010.jpg', 'Image2': 'test/089_0110.jpg', 'R_Dist': 4.4246537747729926, 'G_Dist': 1.9870223665512872, 'B_Dist': 1.9327688924611905, 'H_Dist': 8.142174045330513}

{'Image1': 'train/056_0006.jpg', 'Image2': 'test/056_0099.jpg', 'R_Dist': 3.0777753593936777, 'G_Dist': 2.0567663130335903, 'B_Dist': 5.131625696589952, 'H_Dist': 4.700395913149043}

{'Image1': 'train/057_0025.jpg', 'Image2': 'test/089_0110.jpg', 'R_Dist': 5.144814903093792, 'G_Dist': 2.2516103436044386, 'B_Dist': 2.446736183083187, 'H_Dist': 5.326064398058834}

{'Image1': 'train/028_0030.jpg', 'Image2': 'test/084_0081.jpg', 'R_Dist': 7.234641290189586, 'G_Dist': 2.548352113131755, 'B_Dist': 1.70837523134296, 'H_Dist': 6.420309768571687}

{'Image1': 'train/028_0027.jpg', 'Image2': 'test/028_0108.jpg', 'R_Dist': 6.662678825147111, 'G_Dist': 2.6358734530304613, 'B_Dist': 4.519711988023721, 'H_Dist': 3.7363693975785943}

B
{'Image1': 'train/028_0030.jpg', 'Image2': 'test/084_0081.jpg', 'R_Dist': 7.234641290189586, 'G_Dist': 2.548352113131755, 'B_Dist': 1.70837523134296, 'H_Dist': 6.420309768571687}

{'Image1': 'train/057_0015.jpg', 'Image2': 'test/084_0081.jpg', 'R_Dist': 9.020286268716408, 'G_Dist': 7.10554736752367, 'B_Dist': 2.040140908800692, 'H_Dist': 5.507756797262798}

{'Image1': 'train/105_0012.jpg', 'Image2': 'test/028_0108.jpg', 'R_Dist': 3.084270135139652, 'G_Dist': 4.8296486593368195, 'B_Dist': 2.065863353237996, 'H_Dist': 3.477872430324492}

{'Image1': 'train/057_0025.jpg', 'Image2': 'test/089_0110.jpg', 'R_Dist': 5.144814903093792, 'G_Dist': 2.2516103436044386, 'B_Dist': 2.446736183083187, 'H_Dist': 5.326064398058834}

{'Image1': 'train/105_0012.jpg', 'Image2': 'test/089_0110.jpg', 'R_Dist': 3.853177537816026, 'G_Dist': 4.531071889856677, 'B_Dist': 2.6419406190859833, 'H_Dist': 5.195801969809563}

H
{'Image1': 'train/057_0015.jpg', 'Image2': 'test/057_0102.jpg', 'R_Dist': 6.307122809613602, 'G_Dist': 4.1787178467315 41, 'B_Dist': 5.509584627298544, 'H_Dist': 1.207650917854032}

{'Image1': 'train/105_0025.jpg', 'Image2': 'test/089_0110.jpg', 'R_Dist': 4.1899695634174 41, 'G_Dist': 3.5662491552086117, 'B_Dist': 3.055374116402207, 'H_Dist': 1.9382505549682267}

{'Image1': 'train/028_0032.jpg', 'Image2': 'test/028_0108.jpg', 'R_Dist': 4.84920525947414, 'G_Dist': 4.475557980763157, 'B_Dist': 3.428358840877844, 'H_Dist': 2.1735892608481167}

{'Image1': 'train/028_0027.jpg', 'Image2': 'test/105_0263.jpg', 'R_Dist': 8.29567536572985, 'G_Dist': 7.130639010131598, 'B_Dist': 5.900898071716538, 'H_Dist': 2.193351241200157}

{'Image1': 'train/028_0030.jpg', 'Image2': 'test/057_0104.jpg', 'R_Dist': 4.500531729196143, 'G_Dist': 5.1701337292128215, 'B_Dist': 5.346229622358285, 'H_Dist': 2.3075086136326766}