COMPUTER VISION

ISTANBUL TECHNICAL UNIVERSITY

HOMEWORK-1 REPORT

OCTOBER 11, 2017

Name: EMRE KÖSE
ID: 150130037

Computer Vision Homework-0
Report

NOTE: Codes of first and second questions were implemented in one file(“Q1Q2.py”) file

1) Histogram Equalization

In this question some probability issues were handled. These are generally listed below:

-Mean
-Variance
-Histogram
- Probability density function
- Cumulative distribution function

There are code which are implemented according to the formulas for mean and variance variables. And these are shown below:

```
def meanAndDeviation(image):
  mean = [0, 0, 0]
  variance = [0, 0, 0]
  std = [0, 0, 0]
  img = np.array(image)
  row = np.size(img, 0)   #number of row in image
  column = np.size(img, 1) #number of column in image
  cF = 1/(row*column)  #cardinality factor
  for i in range(3):
    for r in range(row):
      for c in range(column):
        temp = img[r, c, i]
        variance += temp ** 2
        mean[i] += temp
    mean[i] *= cF
    variance[i] *= cF
    variance[i] = variance[i] - (mean[i] ** 2)
    std[i] = sqrt(variance[i])
  return mean[0],mean[1],mean[2],std[0],std[1],std[2]
```

And then histogram of image was provided with counting of every pixels value’s number in image.

```
def histFunc(image):
  img = np.array(image)
  row = np.size(img, 0)
  column = np.size(img, 1)
  histogram = np.zeros((256, 1, 3), dtype=np.int)
  for k in range(3):
    for i in range(row):
      for j in range(column):
        histogram[img[i, j, k], 0, k] += 1
  return np.array(histogram)
```

And then pdf and cdf values are found with the usage of this image’s histogram values, lastIy the output values for image’s intensity values were found with the product cdf values and L-1 (L represents the range of intensity values)

```
def calculator(image):
  img = np.array(image)
  y = histFunc(image)
  row = np.size(img, 0)
  column = np.size(img, 1)
  pdf = np.zeros((256, 1, 3), dtype=np.float32)
  for i in range(3):
    for j in range(len(y)):
      pdf[j, 0, i] = (y[j, 0, i]) / (row * column)
  # part c => cdf
  L = 255
  cdf = np.zeros((256, 1, 3), dtype=np.float32)
  out = np.zeros((256, 1, 3), dtype=np.int)
  cdf[:, 0, 0] = np.cumsum(pdf[:, :, 0])
  cdf[:, 0, 1] = np.cumsum(pdf[:, :, 1])
  cdf[:, 0, 2] = np.cumsum(pdf[:, :, 2])
  for i in range(3):
    for j in range(len(cdf)):
      out[j, 0, i] = round(cdf[j, 0, i] * L)
  return np.array(out), cdf , pdf
```

After this operations histogram equalization was applied to original image :

```
def HistEqFunc(image):
  img = np.array(image)
  out,cdf,pdf= calculator(image)
  row = np.size(img, 0)
  column = np.size(img, 1)
  # newImg = np.zeros((row, column, 3), dtype=np.int)
  newImg = np.zeros_like(image)
  for i in range(3):
    for j in range(row):
      for k in range(column):
        newImg[j, k, i] = out[img[j, k, i], 0, i]
  DerivatedImg = Image.fromarray(newImg)
  return DerivatedImg
```

At last we can say intensity values of the image turned more balanced situation and according to the the values the distributions of values which are derived from histogram values were more balanced with the increasing of low values and decreasing of high intensity values.**(part e)**

If we want to give an intuitive example to histogram equalization in gray level:
Lets assume that our image is so dark, and then values of the image will be near to intensity values like zero, if we apply equalization to this image the intensity values of this image will increase and our image will be more bright, or if our image is so bright, and then values of the image will be near to intensity values like 255, then applying of histogram equalization to this image the intensity values of this image will decrease and our image will be more dark. **(part f)**

2) Histogram Matching

In this part, histogram and probability density function were provided with using of functions which are implemented for first question and then according to the cdf and output(derived from cdf matrix, name of the array is “out” in code) values of both image, **Look up Table** was created with the comparing of the two images output matrices.

```
out1,cdf1, pdf1 = calculator(im1)
out2,cdf2, pdf1 = calculator(im2)
lut = np.zeros((256, 1, 3), dtype=np.int)
for i in range(3):
  for j in range(len(cdf1)):
    dif = abs(out1[j,0,i] - out2[:,:,i])
    index_min = np.argmin(dif)
    lut[j,0,i] = index_min
```

Then with the using of these Look Up Table, the first image which the histogram values should be similar to second image's histogram values modified and at last derived image was created.

```
newImg = img1
row = np.size(img1, 0)
column = np.size(img1, 1)
for i in range(3):
  for j in range(row):
    for k in range(column):
      newImg[j, k, i] = lut[img1[j, k, i], 0, i]
DerivedImg = Image.fromarray(newImg)
```

3) Lack and Unavailable implementation

4) Rotation

Center Rotation

In this part I tried the implementation of rotation of image, firstly the coordinates of image hold in two matrices:

```
points = np.mgrid[0:row, 0:column].reshape((2, row * column))
new_points = np.mgrid[0:row, 0:column].reshape((2, row * column))
```

Then the rotation of coordinates are implemented with the trigonometric function values:

```
  for i in range(c1):
      new_points[0, i] = round(points[0, i] * 0.87 + points[1, i] * -0.5)
       new_points[1, i] = round(points[0, i] * 0.5 + points[1, i] * 0.87)
```

Because of the rotation, there were some negative indices in new_points, to handle this problem firstly the most negative number was found and then points were slipped according to the this negative value:

```
for i in range(c1):
    if new_points[0, i] < xmin:
      xmin = new_points[0, I]

for i in range(c1):
  new_points[0, i] = new_points[0, i] - xmin
```

After the determining of destination of coordinates, the values were sent from source image to destination image:

```
 for ch in range(3):
     for i in range(c1):
          newImg[new_points[0, i], new_points[1, i], ch] = img[points[0, i],
          points[1, i], ch]
```

Corner Rotation

In the corner rotation part, firstly the image transported to another images’ right -bottom corner, after that the rotation transformation was applied to this image with the same logic in above:

5) Lack and unavailable implementation