Digital Image Processing

Assignment 1 - Histogram Equalization

1. Code

- Language: python
- Libraries:
  - matplotlib: read pictures, display pictures and histograms
  - numpy: data processing
  - math: computation support
- Implementation:
  - Global histogram equalization
  - Local histogram equalization (with 27*27 windows)

HistogramEqualization.py

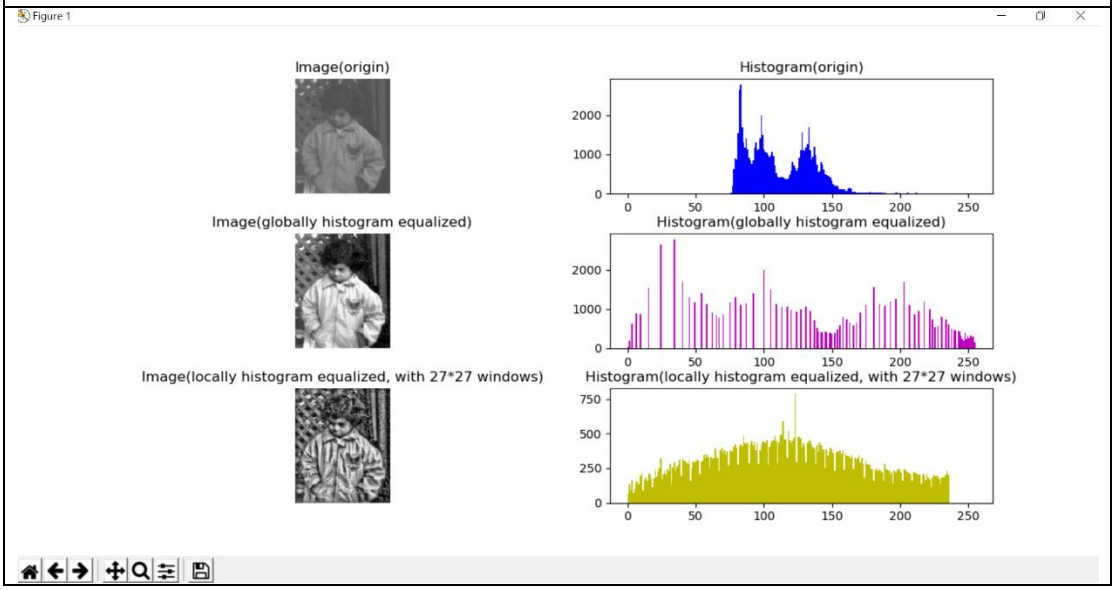
```
import matplotlib.pyplot as plt   # plt is used to display pictures and histograms
import matplotlib.image as mpimg   # mpimg is used to load pictures
import numpy as np
import math

picName = ['TestImg/child.tif', 'TestImg/lena.tif',
           'TestImg/Stone.tif', 'TestImg/university.tif']
picNumber = 3   # select a picture in the picture list


def globalHE(img):   # implement global histogram equalization
    # adjust the img to 1D array
    imgRavel = img.ravel()

    # build a pixel table
    pTable = []
    for i in range(0, 256):
        pTable.append(0)

    # count the probability distribution
    for i in imgRavel:
        pTable[i] = pTable[i] + 1
    for i in range(0, 256):
        pTable[i] = pTable[i] / img.size

    # transform to cumulative distribution
    for i in range(0, 255):
        pTable[i+1] = pTable[i] + pTable[i+1]

    # round to integer
    for i in range(0, 256):
        pTable[i] = round(pTable[i] * 255)

    # construct new image
    newImg = img
    row, col = newImg.shape

    # set every pixel to new value
    for i in range(0, row):
        for j in range(0, col):
```

```
            newImg[i][j] = pTable[newImg[i][j]]

    return newImg


def localHE(img, winSize):   # implement local histogram equalization
    newImg = img
    row, col = img.shape
    for i in range(0, row):
        for j in range(0, col):
            # build a pixel table
            hist = []
            for k in range(0, 256):
                hist.append(0)

            # for every pixel in the window, add it to the probability distribution
            for x in range(i-math.floor((winSize-1)/2), i+math.floor((winSize-1)/2)):
                for y in range(j-math.floor((winSize-1)/2), j+math.floor((winSize-1)/2)):
                    if x < 0 or x >= row or y < 0 or y >= col:   # check if the position is in the window
                        continue
                    hist[img[x][y]] = hist[img[x][y]] + 1

            # transform to cumulative distribution(but only for the central point of the window)
            cum = 0
            for k in range(0, img[i][j]):
                cum = cum + hist[k]
            cum = cum*255/(winSize*winSize)

            # set the new image to new value
            newImg[i][j] = round(cum)

    return newImg


# read source image
img = np.array(mpimg.imread(picName[picNumber]))

# construct a canvas to display the results
plt.figure(figsize=(12, 6))

# set new pictures and histograms
plt.subplot(3, 2, 1)
plt.title("Image(origin)")
plt.axis('off')
plt.imshow(img, cmap='gray', vmin=0, vmax=255)

plt.subplot(3, 2, 2)
plt.title("Histogram(origin)")
plt.hist(img.ravel(), bins=256,   range=(0, 255), color='b')

transformedImg = globalHE(img)

plt.subplot(3, 2, 3)
plt.title("Image(globally histogram equalized)")
plt.axis('off')
plt.imshow(transformedImg, cmap='gray', vmin=0, vmax=255)

plt.subplot(3, 2, 4)
plt.title("Histogram(globally histogram equalized)")
plt.hist(transformedImg.ravel(), bins=256,   range=(0, 255), color='m')

transformedImg2 = localHE(img, 27)

plt.subplot(3, 2, 5)
plt.title("Image(locally histogram equalized, with 27*27 windows)")
plt.axis('off')
plt.imshow(transformedImg2, cmap='gray', vmin=0, vmax=255)

plt.subplot(3, 2, 6)
```

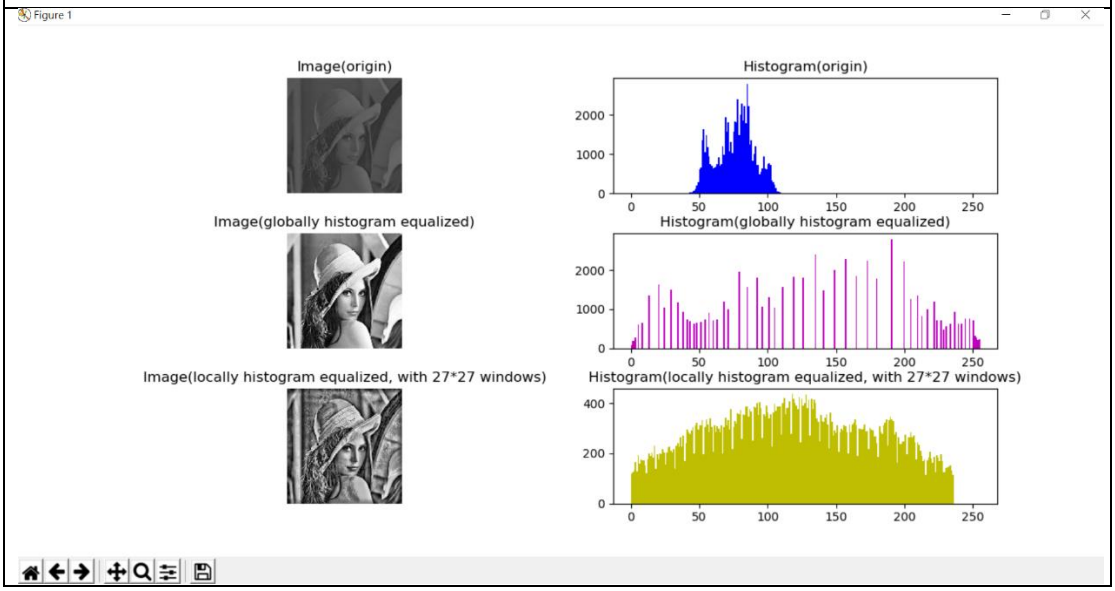
```
plt.title("Histogram(locally histogram equalized, with 27*27 windows)")
plt.hist(transformedImg2.ravel(), bins=256,   range=(0, 255), color='y')

# adjust layout of canvas
plt.subplots_adjust(left=0.125,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.2,
                    hspace=0.35)

# show the canvas
plt.show()
```

2. Result

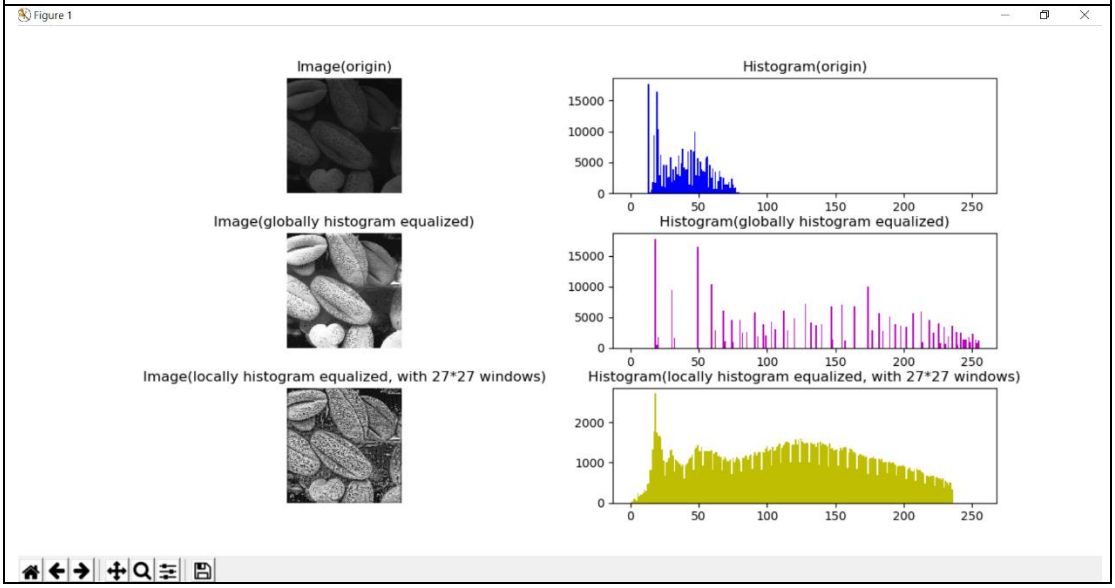
child.tif



lena.tif



Stone.tif



university.tif