HW2 報告

D1149576 曹世杰

程式碼：

灰階：

```python
import cv2 as cv;
import numpy as np;

import matplotlib.pyplot as plt

def makePicShow(name, img):
    cv.imshow(name,img);
    cv.waitKey();

def makeHistogramArray(img):
    (x,y) = img.shape;
    pixelArray = np.zeros(256, dtype=int);

    for i in img:
        pixelArray[i]+=1;

    return pixelArray;

def findMax(pixelArray):
    Max = pixelArray[0];

    for i in pixelArray:
        if(i>Max):
            Max = i;

    return Max;

def makeHistogramPictrue(pixelArray):
    count=0;
    Max = findMax(pixelArray);
    pic = np.zeros((Max,256),np.uint8);

    for i in range(Max):
        for j in range(256):
            pic[i][j] = 255;

    for i in pixelArray:
        for j in range(i):
            pic[Max-j-1][count]=0;
        count+=1;


    return pic;

def makeEqualization(pixelArray):
    total=0;
    temp=0;
    Max = findMax(pixelArray);
    newArray = np.zeros(256, dtype=int);
    cdf = np.zeros(256, dtype=int);

    for i in pixelArray:
        total+=i;

    for i in range(len(pixelArray)):
        temp+=pixelArray[i];
        pos = temp*255//total;
        newArray[pos]=pixelArray[i];
        cdf[i]=pos;

    equalizationArray = makeHistogramPictrue(newArray);
    makePicShow("equalizationArray",equalizationArray);

    return cdf;




img = cv.imread("./pic/123.jpg");  #讀入原圖
(x, y) = img.shape;
pixelArray = makeHistogramArray(img);
makePicShow("Ori",img);

histogram = makeHistogramPictrue(pixelArray);
makePicShow("histogram",histogram);

# hist = cv.calcHist([img], [0], None, [256], [0,256]);
# plt.bar(range(256), hist.flatten())
# plt.show()

cdf = makeEqualization(pixelArray);
equalization = np.zeros((x,y),np.uint8)
for i in range(x):
    for j in range(y):
        equalization[i][j] = cdf[img[i][j]];

makePicShow("equalization",equalization);
```

彩色：

```python
import cv2 as cv;
import numpy as np;

def makePicShow(name, img):
    cv.imshow(name,img);
    cv.waitKey();

def makeHistogramArray(img):
    (x, y) = img.shape;
    pixelArray = np.zeros(256, dtype=int);

    for i in range(x):
        for j in range(y):
            pixelArray[img[i, j]] += 1

    return pixelArray;

def findMax(pixelArray):
    Max = pixelArray[0];

    for i in pixelArray:
        if(i>Max):
            Max = i;

    return Max;

def makeHistogramPictrue(pixelArray):
    count=0;
    Max = findMax(pixelArray);
    pic = np.zeros((Max,256),np.uint8);

    for i in range(Max):
        for j in range(256):
            pic[i][j] = 255;

    for i in pixelArray:
        for j in range(i):
            pic[Max-j-1][count]=0;
        count+=1;


    return pic;

def makeEqualization(pixelArray):
    total=0;
    temp=0;
    Max = findMax(pixelArray);
    newArray = np.zeros(256, dtype=int);
    cdf = np.zeros(256, dtype=int);

    for i in pixelArray:
        total+=i;

    for i in range(len(pixelArray)):
        temp+=pixelArray[i];
        pos = temp*255//total;
        newArray[pos]=pixelArray[i];
        cdf[i]=pos;

    equalizationArray = makeHistogramPictrue(newArray);
    makePicShow("equalizationArray",equalizationArray);

    return cdf;


img = cv.imread("./pic/dog.jpg");  #讀入原圖
(x, y, z) = img.shape;
makePicShow("ori",img);

blue = img[:, :, 0];
green = img[:, :, 1];
red = img[:, :, 2];

pixelArray = makeHistogramArray(blue);
blueHistogram = makeHistogramPictrue(pixelArray);
makePicShow("Blue",blueHistogram);
blueCdf = makeEqualization(pixelArray);

pixelArray = makeHistogramArray(green);
greenHistogram = makeHistogramPictrue(pixelArray);
makePicShow("Green",greenHistogram);
greenCdf = makeEqualization(pixelArray);


pixelArray = makeHistogramArray(red);
redHistogram = makeHistogramPictrue(pixelArray);
makePicShow("Red",redHistogram);
redCdf = makeEqualization(pixelArray);

equalization = np.zeros((x,y,z),np.uint8)
for k in range(z):
    for i in range(x):
        for j in range(y):
            if(k==0):
                equalization[i][j][k] = blueCdf[img[i][j][k]];
            if(k==1):
                equalization[i][j][k] = greenCdf[img[i][j][k]];
            if(k==2):
                equalization[i][j][k] = redCdf[img[i][j][k]];

makePicShow("equalization",equalization);
```

程式碼說明：

makePicShow：顯示圖片

```
def makePicShow(name, img):
    cv.imshow(name,img);
    cv.waitKey();
```

makeHistogramArray：創建一個 256 的陣列來存放各分布數量，並回傳陣列

```
def makeHistogramArray(img):
    (x, y) = img.shape;
    pixelArray = np.zeros(256, dtype=int);

    for i in range(x):
        for j in range(y):
            pixelArray[img[i, j]] += 1

    return pixelArray;
```

findMax：找到 0~255 中最大的數值

```
def findMax(pixelArray):
    Max = pixelArray[0];

    for i in pixelArray:
        if(i>Max):
            Max = i;

    return Max;
```

makeHistogramPictrue：將直方圖的數據做成圖片

```
def makeHistogramPictrue(pixelArray):
    count=0;
    Max = findMax(pixelArray);
    pic = np.zeros((Max,256),np.uint8);

    for i in range(Max):
        for j in range(256):
            pic[i][j] = 255;

    for i in pixelArray:
        for j in range(i):
            pic[Max-j-1][count]=0;
        count+=1;


    return pic;
```

makeEqualization：將數據做均化，並回傳一個對應位移位置的陣列

```
def makeEqualization(pixelArray):
    total=0;
    temp=0;
    Max = findMax(pixelArray);
    newArray = np.zeros(256, dtype=int);
    cdf = np.zeros(256, dtype=int);

    for i in pixelArray:
        total+=i;

    for i in range(len(pixelArray)):
        temp+=pixelArray[i];
        pos = temp*255//total;
        newArray[pos]=pixelArray[i];
        cdf[i]=pos;

    equalizationArray = makeHistogramPictrue(newArray);
    makePicShow("equalizationArray",equalizationArray);

    return cdf;
```

剩下主程式：將 bgr 個別分離出來做處理再合併

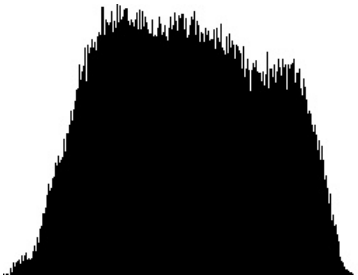
```
img = cv.imread("./pic/dog.jpg");  #讀入原圖
(x, y, z) = img.shape;
makePicShow("ori",img);

blue = img[:, :, 0];
green = img[:, :, 1];
red = img[:, :, 2];

pixelArray = makeHistogramArray(blue);
blueHistogram = makeHistogramPictrue(pixelArray);
makePicShow("Blue",blueHistogram);
blueCdf = makeEqualization(pixelArray);

pixelArray = makeHistogramArray(green);
greenHistogram = makeHistogramPictrue(pixelArray);
makePicShow("Green",greenHistogram);
greenCdf = makeEqualization(pixelArray);


pixelArray = makeHistogramArray(red);
redHistogram = makeHistogramPictrue(pixelArray);
makePicShow("Red",redHistogram);
redCdf = makeEqualization(pixelArray);

equalization = np.zeros((x,y,z),np.uint8)
for k in range(z):
    for i in range(x):
        for j in range(y):
            if(k==0):
                equalization[i][j][k] = blueCdf[img[i][j][k]];
            if(k==1):
                equalization[i][j][k] = greenCdf[img[i][j][k]];
            if(k==2):
                equalization[i][j][k] = redCdf[img[i][j][k]];

makePicShow("equalization",equalization);
```

執行結果：

黑白影像：

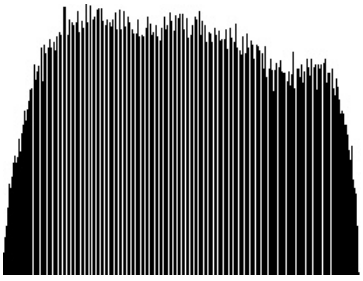
1. 原圖：

1. 直方圖：



1. 均化後：

1. 均化後直方圖：

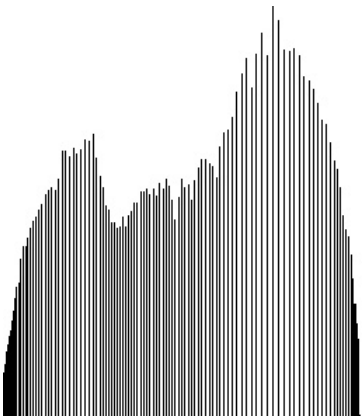


2. 原圖：

2. 直方圖：

2. 均化後：

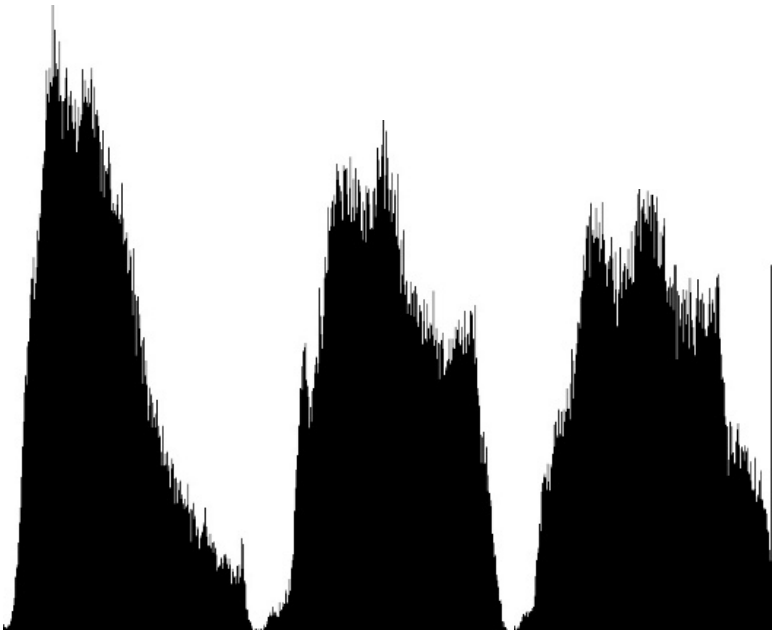
2. 均化直方圖：



彩色影像：

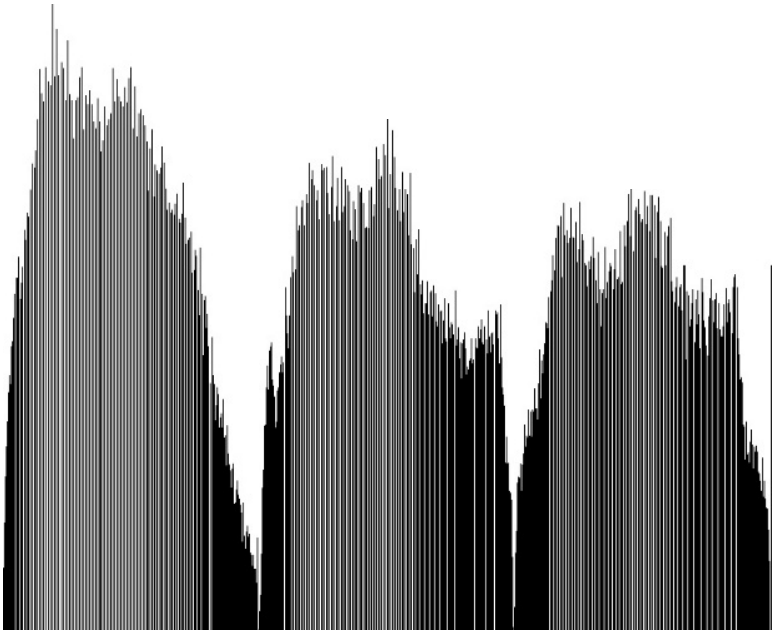
1. 原圖：

1. 直方圖（由左到右為藍綠紅）:



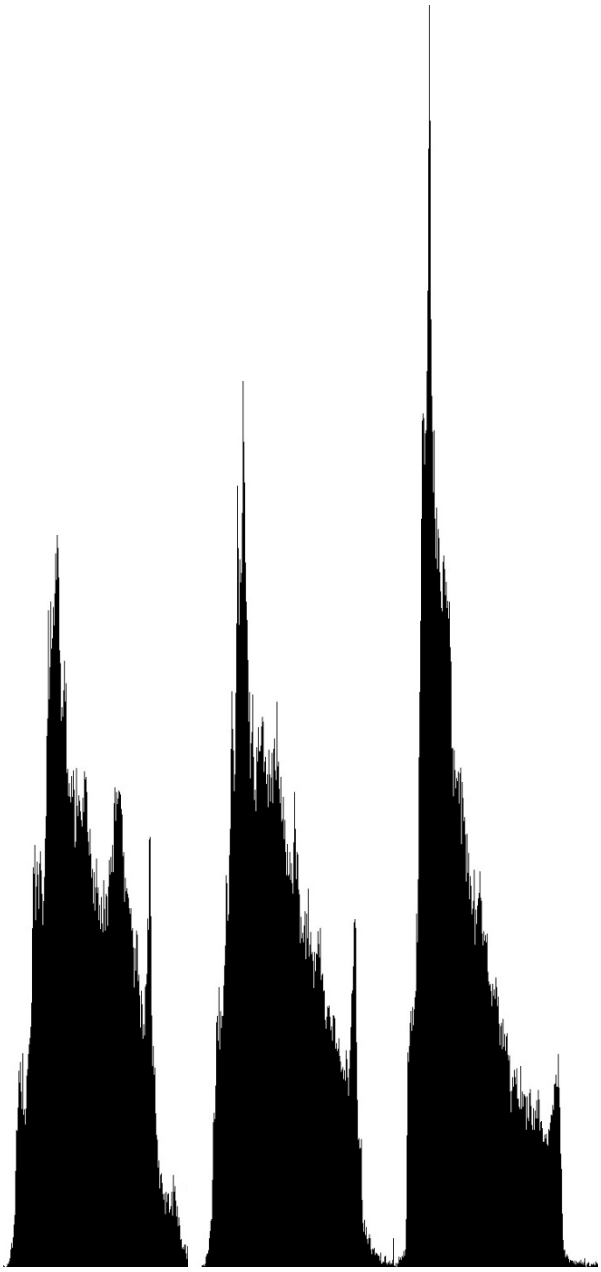
1. 均化後：

1. 均化直方圖（由左到右為藍綠紅）:



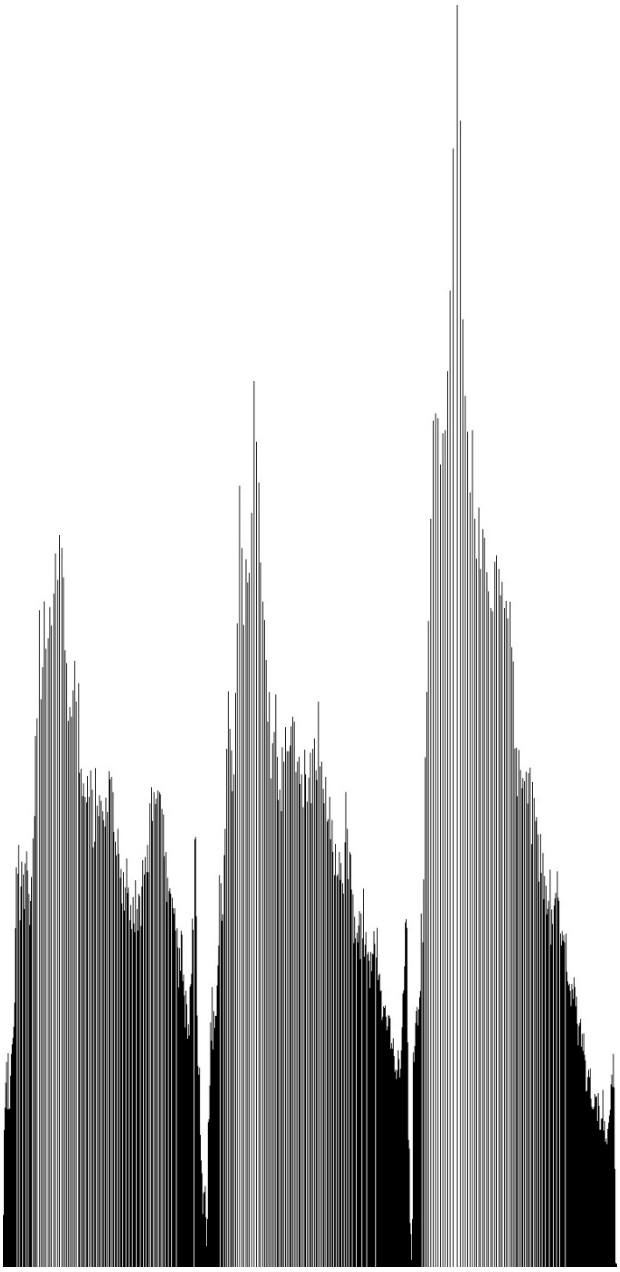
2. 原圖：

2. 直方圖（由左到右為藍綠紅）:



2. 均化後：

2. 均化後直方圖（由左到右為藍綠紅）:



心得：

比想像中的簡單一些，不過跟 calcHist 比對起來數據上還是有些差距，可能計算分布的方式不夠準確。