Homework 2 Report

1. Method
   * Histogram Equalization   
     我定義了以下的function來實現Histogram Equalization：

|  |
| --- |
| def get\_hist(img\_one\_channel):      hist = np.zeros(256, dtype=np.float32)        for y in range(img\_one\_channel.shape[0]):          for x in range(img\_one\_channel.shape[1]):              hist[img\_one\_channel[y,x]] += 1      for i in range(1, 256):          hist[i] += hist[i-1]        max\_val = img\_one\_channel.shape[0] \* img\_one\_channel.shape[1]        result = np.zeros(256, dtype=np.uint8)      for i in range(256):          result[i] = np.round(255.0 \* (hist[i] / max\_val)).astype(np.uint8)        return result  def hist\_equalization(img\_one\_channel):      result = np.zeros\_like(img\_one\_channel)      hist = get\_hist(img\_one\_channel)        for y in range(img\_one\_channel.shape[0]):          for x in range(img\_one\_channel.shape[1]):              result[y,x] = hist[img\_one\_channel[y,x]]        return result  img = cv2.imread('Q1.jpg', cv2.IMREAD\_GRAYSCALE)  res\_img = hist\_equalization(img)  cv2.imwrite('Q1\_processed.jpg', res\_img) |

hist\_equalization 的用途是取得圖片的histogram，然後再將原圖每個pixel的值換成{histogram之CDF對應的機率}\*255。至於乘255的部分已經在get\_hist內做過了。

* + Histogram Specification  
    這部分也是定義一個function來實現：

|  |
| --- |
| def hist\_specification(src\_img, ref\_img):      T = get\_hist(src\_img)      G = get\_hist(ref\_img)      # histogram specification      H = np.zeros(256, dtype=np.uint8)      j = 1        for a in range(256):          while T[a] > G[j]:              j += 1            if not T[a] == G[j]:              j -= 1          H[a] = j        # apply histogram specification      dst\_img = np.zeros\_like(src\_img)      for i in range(src\_img.shape[0]):          for j in range(src\_img.shape[1]):              dst\_img[i,j] = H[src\_img[i,j]]        return dst\_img  src\_img = cv2.imread('Q2\_source.jpg', cv2.IMREAD\_GRAYSCALE)  ref\_img = cv2.imread('Q2\_reference.jpg', cv2.IMREAD\_GRAYSCALE)  res\_img = hist\_specification(src\_img, ref\_img)  cv2.imwrite('Q2\_processed.jpg', res\_img) |

這個function內做的事也不多，只要將source跟reference圖片的histogram找到之後，將z = T(a)的值轉換到對應G(j)的index，如果在G裡面沒有找到對應的index，就找小於且最接近z的G(j)來近似；如果z對應到很多個G(j)則是取最小的index。

1. Result  
   執行完程式之後可以看到結果如下：
   * Histogram Equalization

|  |  |
| --- | --- |
|  |  |
| 原圖 | 處理後 |

原圖會偏白是因為pixel的值分佈都在較高的地方，把處理前後的histogram畫出來看就可以知道原因了。處理後的值會分佈得比較散，但因為原圖的分佈就已經非常偏了，所以經過處理的圖片多少還是會有一樣的情形。

|  |  |
| --- | --- |
|  |  |
| 原圖 | 處理後 |

* + Histogram Specification

|  |  |
| --- | --- |
|  |  |
| source | reference |
|  | |
| result | |

經過處理的圖片色調會更接近reference，就是經過histogram specification得到的結果。

|  |  |  |
| --- | --- | --- |
|  |  |  |
| source | reference | result |

把對應的CDF畫出來，可以得知histogram specification能讓原圖上pixel的機率分佈轉換到在reference上的分佈。

1. Feedback  
   這次作業將課本上的公式轉為程式，並實際跑過一遍得到結果，讓我對使用histogram來處理圖片的幾種方式更為熟悉，也對這些影像處理方式更有概念。