Cover Page

影像處理作業1 - Histogram equalization

409410019 資工四 王郁誠

HW due : 4/17 00:00

HW handed in : 4/16

前言

由於圖片太多，所以我將Lena.bmp和Peppers.bmp的執行結果分開顯示，但因為code基本只改input，所以沒有寫成2份。因此，助教可以根據 `hw1.py` 中第92、93行code來決定此次input image是Lena.bmp還是Peppers.bmp。

程式執行流程：

執行 `python hw1.py` 後，會以Lena.bmp作為input，並跳出7個視窗，包含：

1. result：原圖、global、local histogram equalization的比較結果
2. img_list：原圖切成16等份的subimg
3. img_list_histogram：該16等份subimg的histogram
4. img_list_cdf：該16等份subimg的cdf
5. new_img_list：16份subimg各自進行histogram equalization的結果
6. new_hist_list：subimg經過histogram equalization後的histogram
7. new_cdf_list：subimg經過histogram equalization後的cdf

接著將93行取消註解，重新執行一次，會改為以Peppers.bmp作為input，並顯示對應結果。

Technical description

實作計算Histogram、CDF

```python
def calc_hist(img: np.ndarray):
    height, width = img.shape
    hist = np.zeros(256, dtype = np.float32)
    for i in range(height):
        for j in range(width):
            hist[img[i,j]] += 1
    hist = hist / (height*width)

    cdf = np.zeros(256, dtype = np.float32)
    for i in range(hist.size):
        if i == 0:
            cdf[i] = hist[i]
        else:
            cdf[i] = hist[i] + cdf[i-1]
    return hist, cdf
```

將input img遍歷，把掃到的pixel值存進hist陣列，e.g.該pixel值為5，則hist[5]+=1，遍歷後再將其標準化得到histogram陣列，也是該img的PMF，最後再根據PMF得到其CDF。

實作histogram equalization

```
def histogram_equalize(img: np.ndarray, cdf: np.ndarray):
    height, width = img.shape
    equ_img = np.zeros((height, width), dtype=np.uint8)
    for i in range(height):
        for j in range(width):
            equ_img[i,j] = 255 * (cdf[img[i,j]] - min(cdf)) / (max(cdf) - min(cdf))
    return equ_img
```

$$h(y) = (L-1)\left\{\frac{F(y)-F_{min}}{F_{max}-F_{min}}\right\}$$

$$g(x,y) = h(f(x,y))$$

1. **L**：灰階的深度個數**255**(以8位元深度為例)。
2. **F(y)**：直方圖所產生的累積分部函數(CDF)，這代表灰階值小於等於y的像素的總和機率。
3. **Fmin**：最小灰階值的累積分布函數值，對應於圖片中最暗的像素。
4. **Fmax**：最大灰階值的累積分布函數值，對應於圖片中最亮的像素。
5. **h(y)**：新的輸出灰階值，原始的灰階值y會被轉換到h(y)。
6. **f(x,y)**：原始影像。
7. **g(x,y)**：經過直方圖均衡化的輸出影像。

透過上圖之數學公式進行實作。

實作切割&合併subimg

```
def cut_and_combine_img(img: np.ndarray):
    height, width = img.shape
    cut_h, cut_w = 4, 4
    sub_img_h, sub_img_w = int(height/cut_h), int(width/cut_w)
    img_list = np.zeros((cut_h, cut_w, sub_img_h, sub_img_w), dtype = np.uint8)
    new_img_list = np.zeros((cut_h, cut_w, sub_img_h, sub_img_w), dtype = np.uint8)
    hist_list = np.zeros((cut_h, cut_w, 256), dtype = np.float32)
    cdf_list = np.zeros((cut_h, cut_w, 256), dtype = np.float32)
    new_hist_list = np.zeros((cut_h, cut_w, 256), dtype = np.float32)
    new_cdf_list = np.zeros((cut_h, cut_w, 256), dtype = np.float32)
    combine_img = np.zeros((height, width), dtype=np.uint8)

    for i in range(cut_h):
        for j in range(cut_w):
            h_start, h_end = i*sub_img_h, (i+1)*sub_img_h
            w_start, w_end = j*sub_img_w, (j+1)*sub_img_w
            img_list[i,j] = img[h_start:h_end, w_start:w_end]
            hist_list[i,j], cdf_list[i,j] = calc_hist(img_list[i,j])
            new_img_list[i,j] = histogram_equalize(img_list[i,j], cdf_list[i,j])
            new_hist_list[i,j], new_cdf_list[i,j] = calc_hist(new_img_list[i,j])
            combine_img[h_start:h_end, w_start:w_end] = new_img_list[i,j]
```

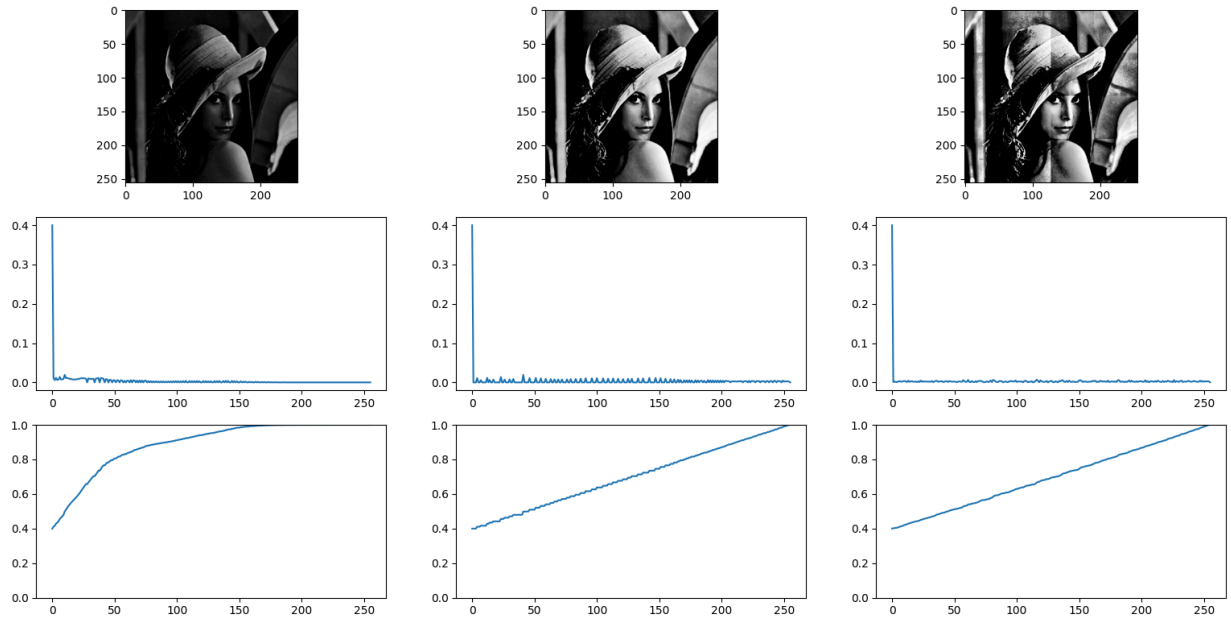
有了要切的份數後(此次作業為16等份)，計算subimg的高、寬，接著用雙層for迴圈，訂好每個subimg的起始、結束位置，並依序將其存入img_list裡面。(此次作業中，img_list size為4*4*64*64)

接著再調用上面的副函式，逐一對這些subimg進行histogram equalization(存入new_img_list)，最後再根據每個subimg的起始、結束位置，依序將他們填入combine_img裡面，即完成切割->處理->合併的流程。

Experimental results

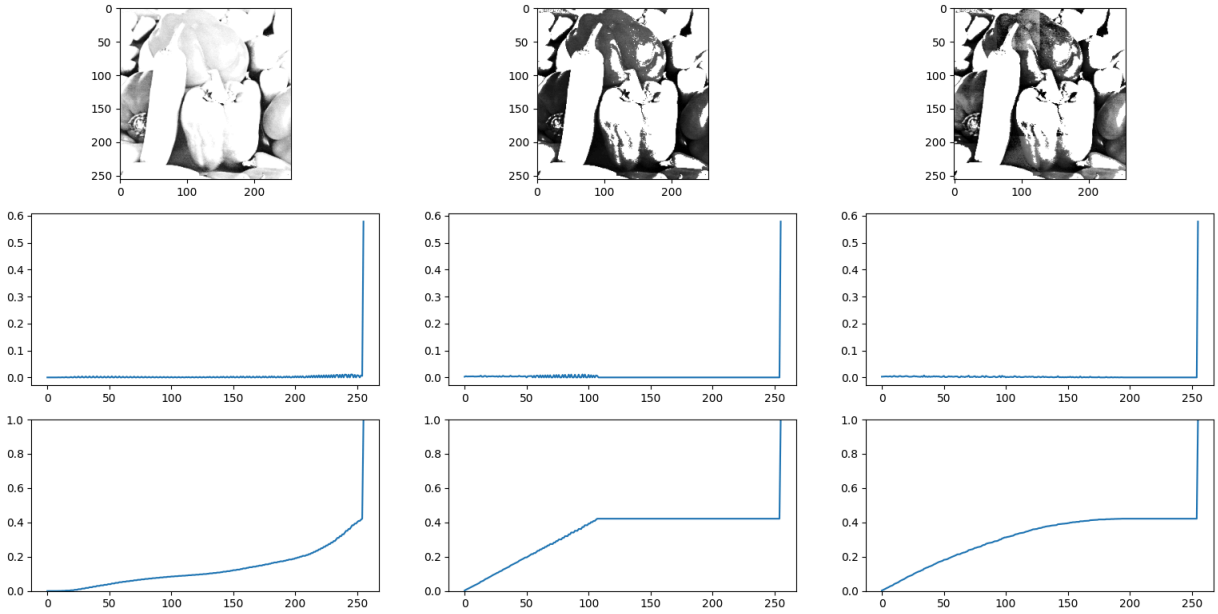
這裡只show出整體比較圖，切割之subimg的histogram、cdf放在Appendix。

Lena.bmp



此圖有很多pixel value為0的pixel，經過histogram equalization後，明顯變清楚許多。而各自進行等化並拼貼的case中，拼貼的邊緣會比較明顯，但因為分成local block各自histogram equalization，所以block內部有些細節會比用global histogram equalization還明顯。

Peppers.bmp



此圖和上圖相反，有很多pixel value為255的pixel，同樣經過histogram equalization後比原圖清楚很多。在拼貼case的結論也差不多，拼貼各自等化後合併會使得邊緣變得明顯一些，但能增加各自block內的對比，有些細節會比用global方法還明顯。

Discussions

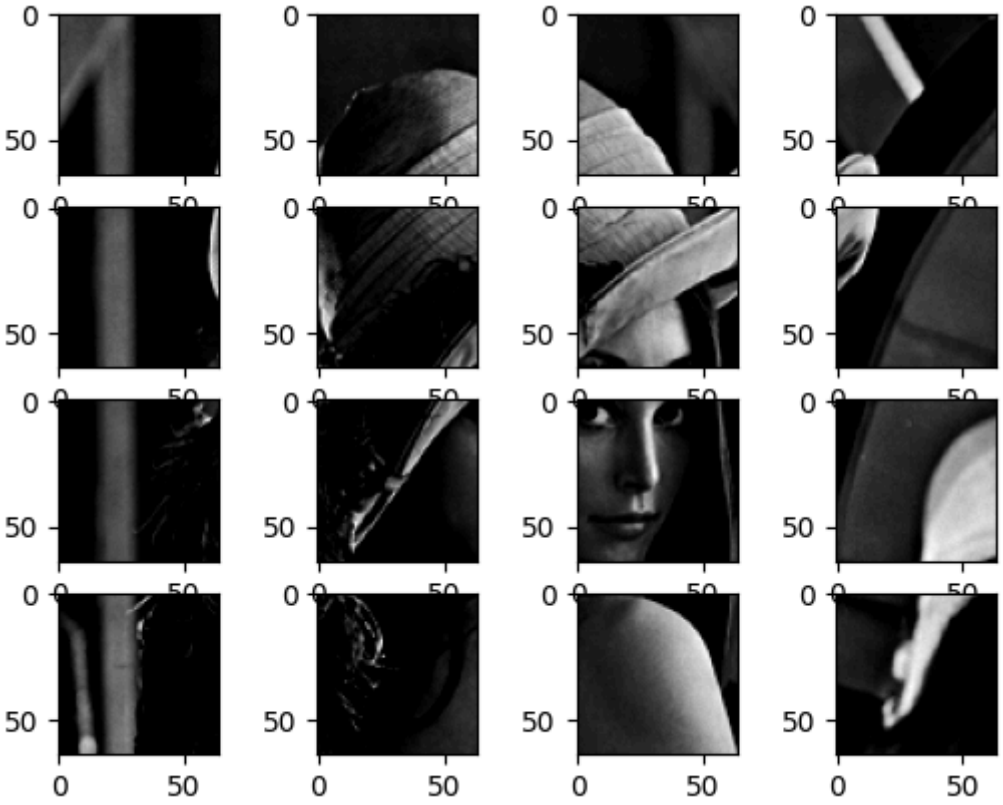
這次作業我遇到了兩個挫折：
第一個是這兩張圖片的灰階分布太奇怪，太多極端值(0、255)，導致histogram總有一條很長的卡在那邊，讓我以為我做錯，直到我拿其他圖測試才確信我的做法沒問題。
第二個挫折是histogram equalization的實作，PPT是使用微積分的寫法，但我微積分太爛所以沒有很理解原理，而我也不知道怎麼將微積分公式寫成code。因此我最後上網找了兩個公式，並挑出我認為效果較好的來用，因此結果可能會跟用微積分做的結果有些小差距。
這次是影像處理第一次作業，一開始聽到不能call function著實捏了把冷汗，以為自己完蛋了，但真正去寫後才發現沒有想的那麼難，完全理解概念的話就做得出來，難的反而是理解概念。希望在寫後面作業時，我也能理解該題目的概念，並成功寫出來。

Appendix

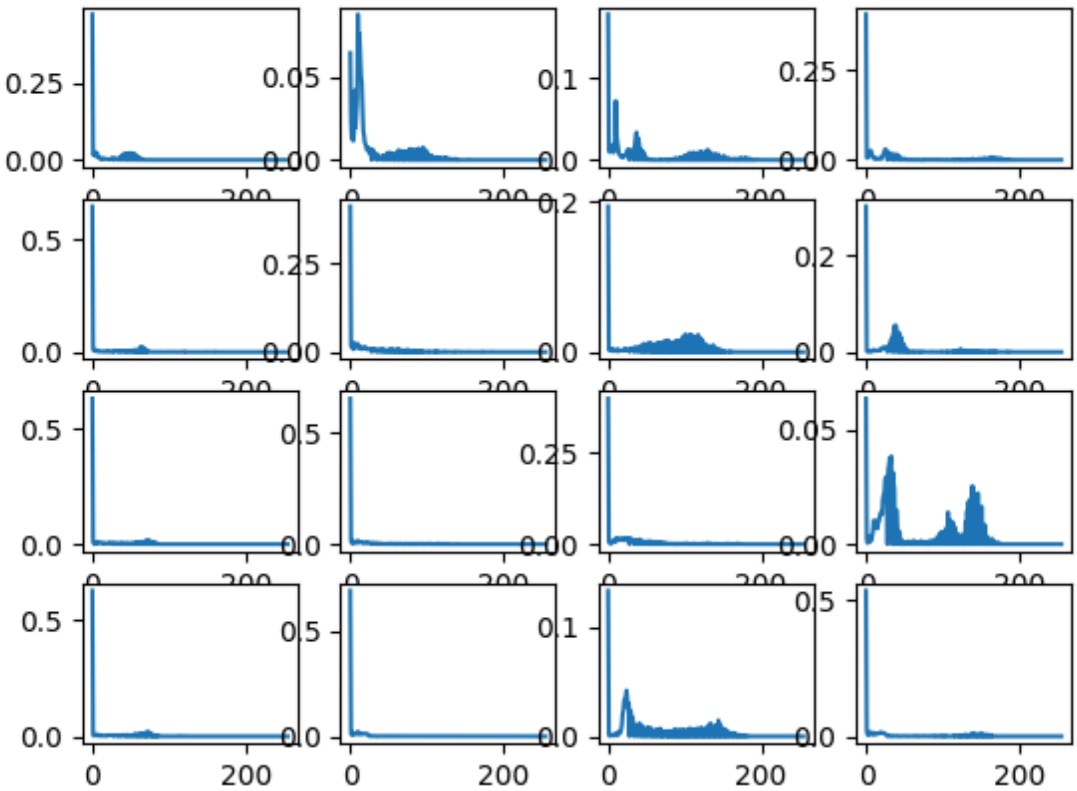
Lena.bmp

- subimg處理前

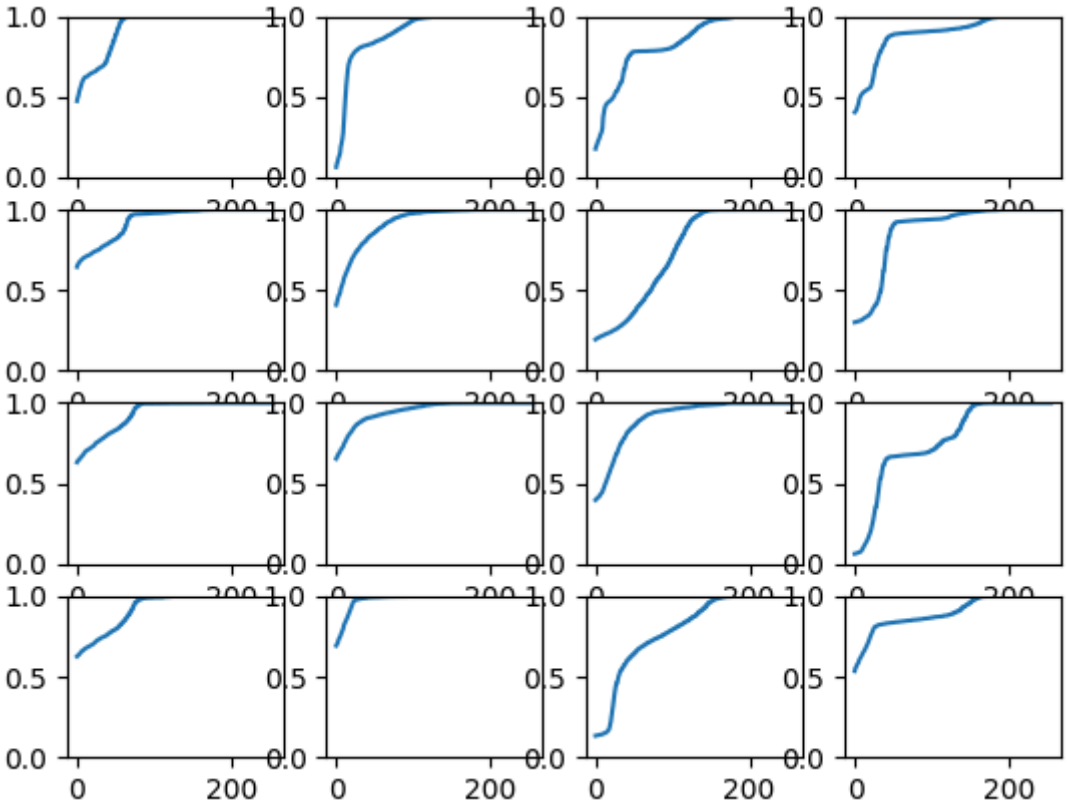
- image



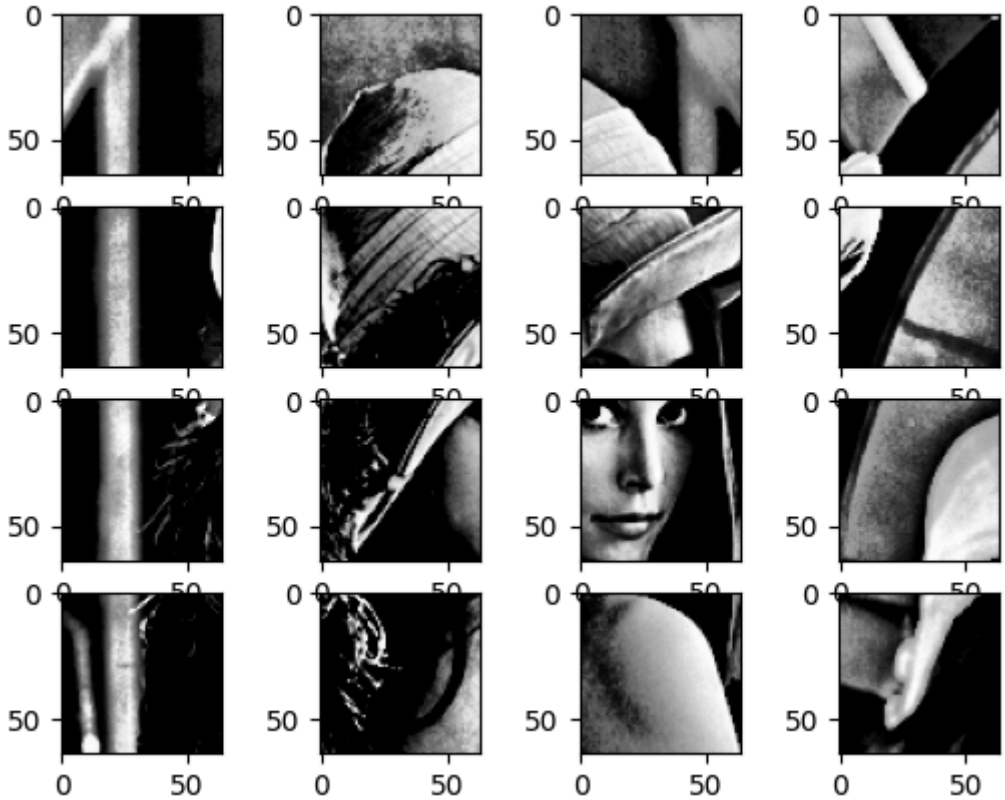
- histogram



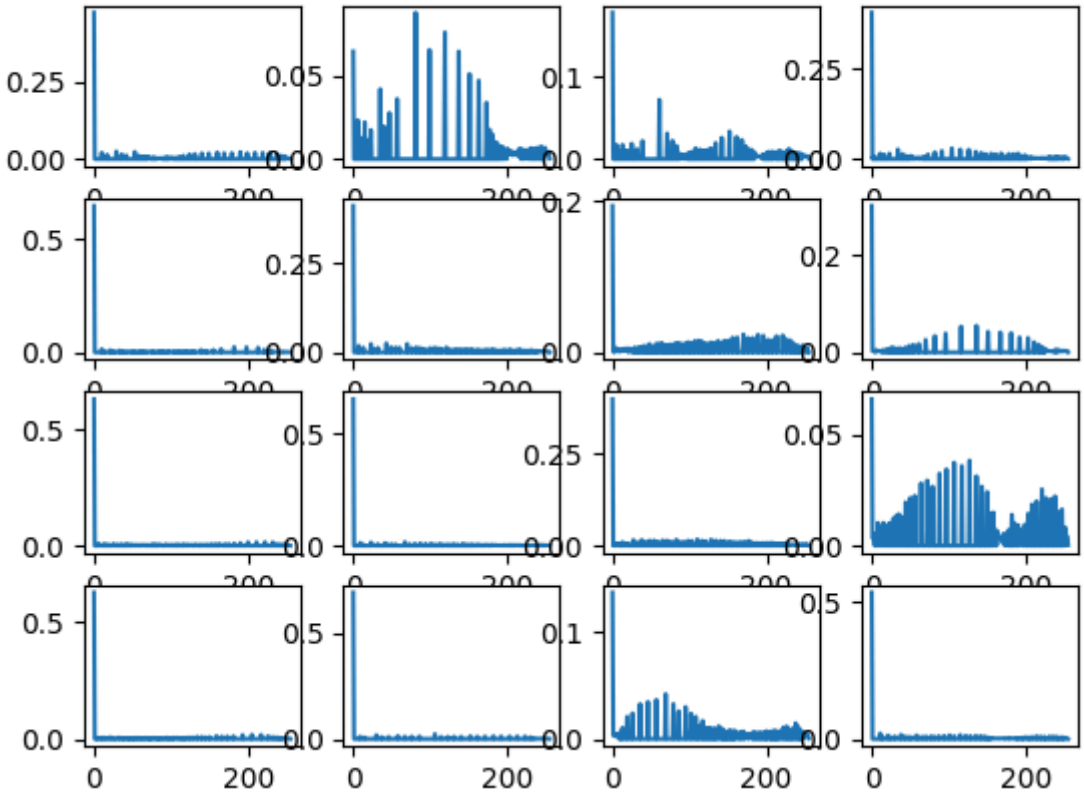
- cdf



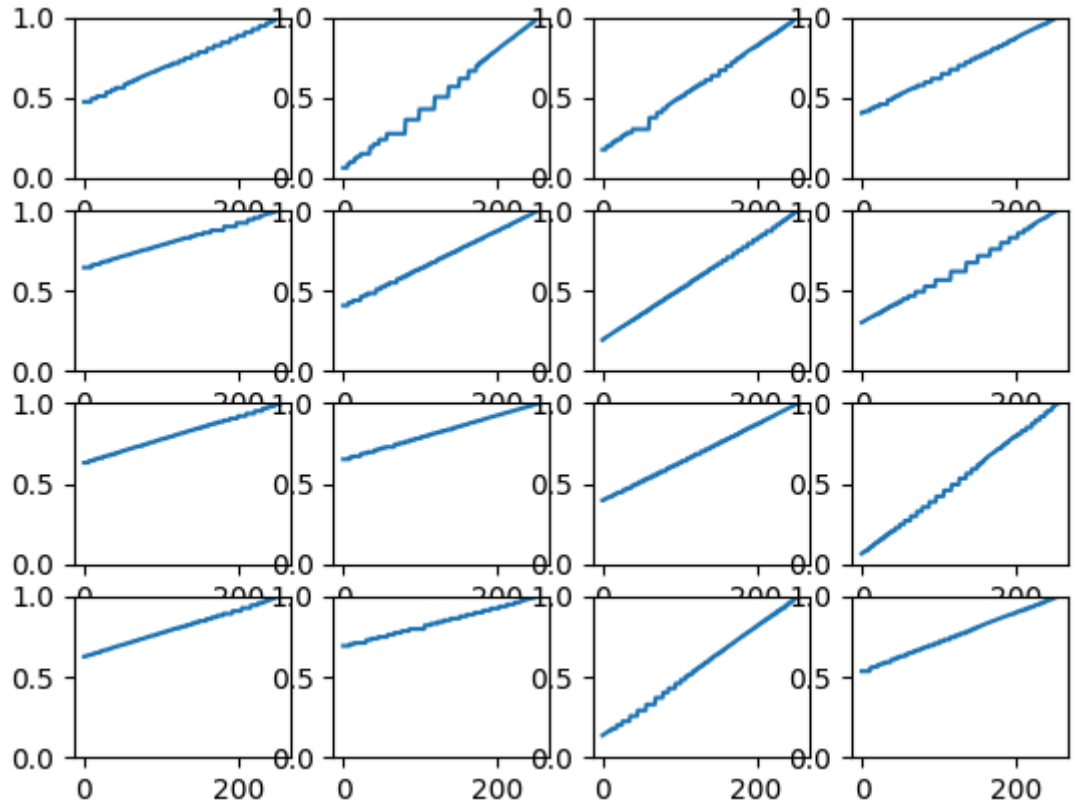
- subimg處理後
  - image



- histogram



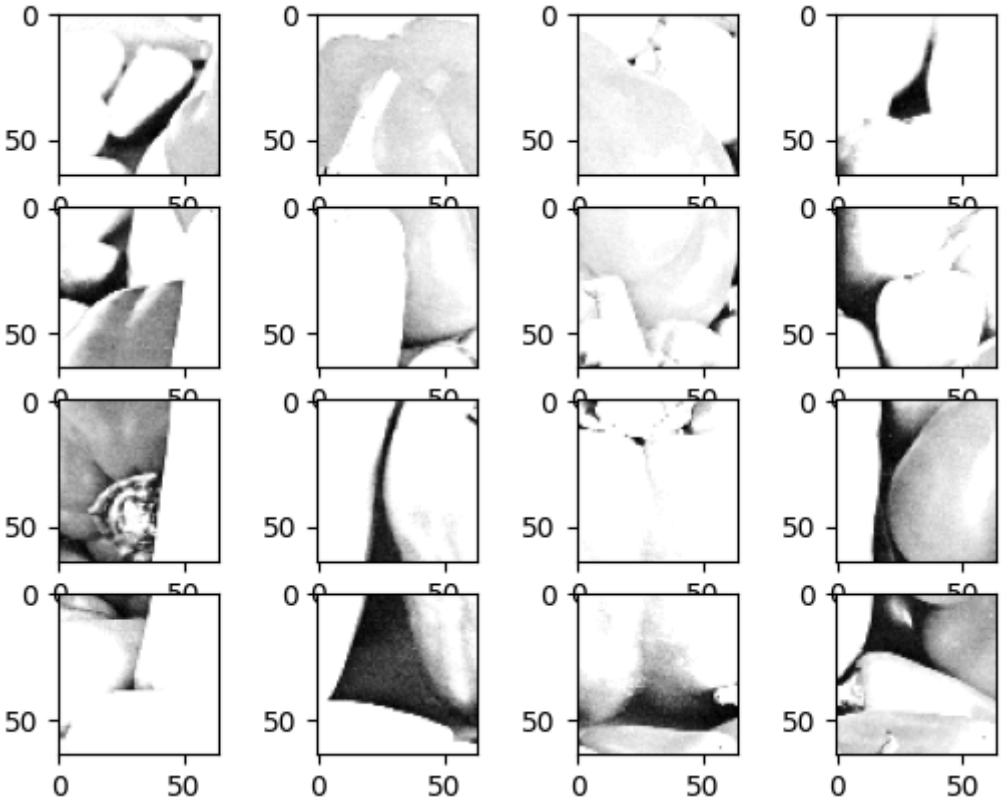
- cdf



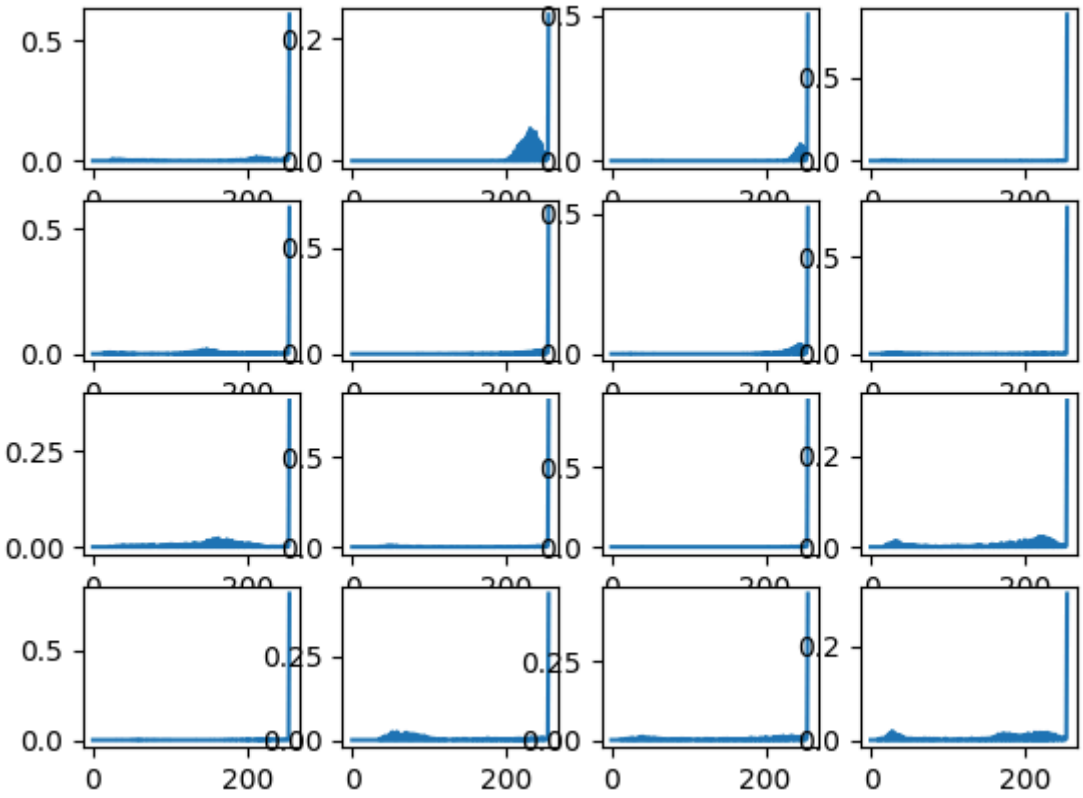
Peppers.bmp

- subimg處理前

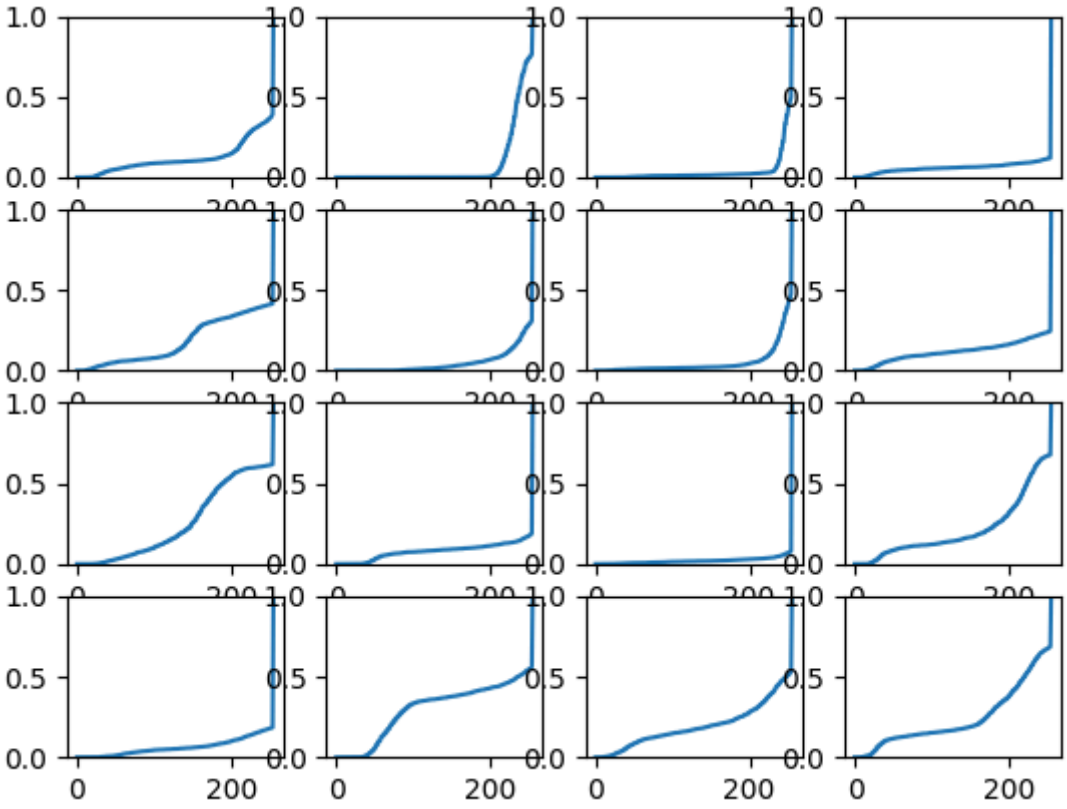
- image



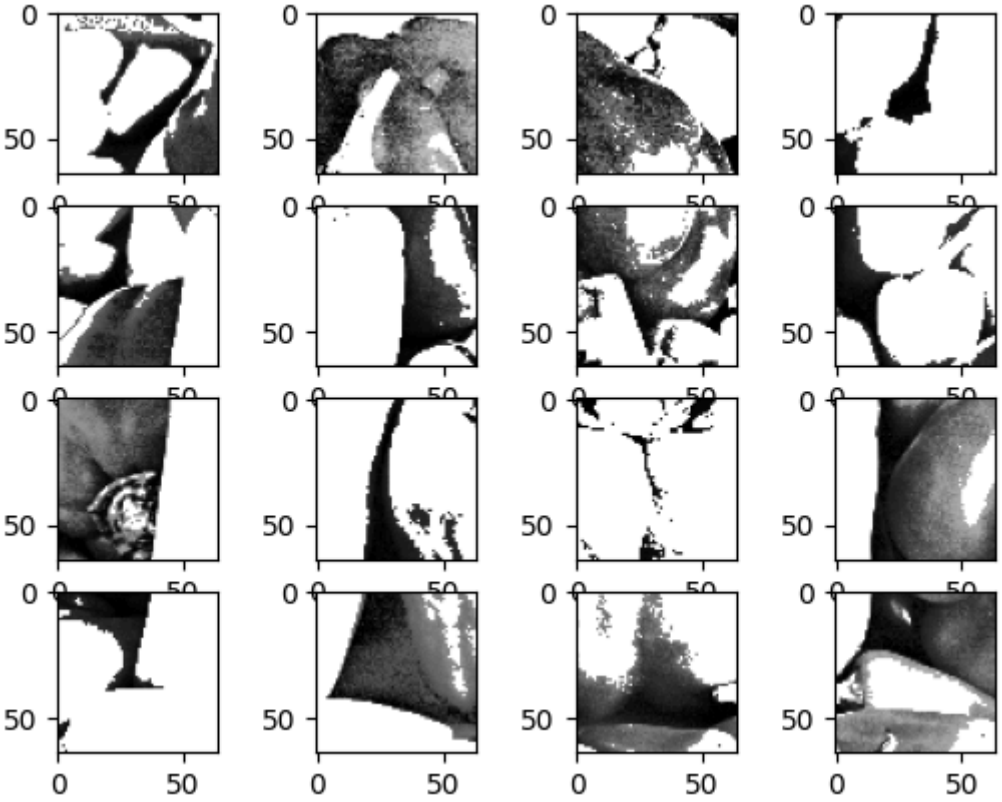
- histogram



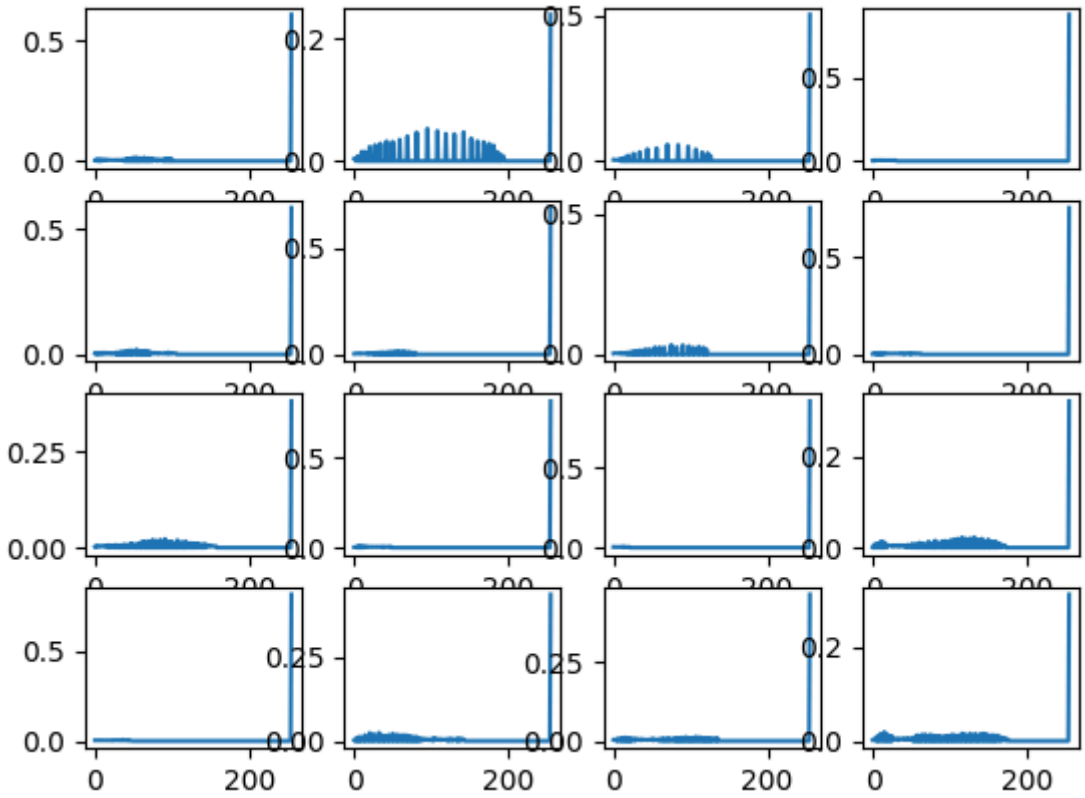
- cdf



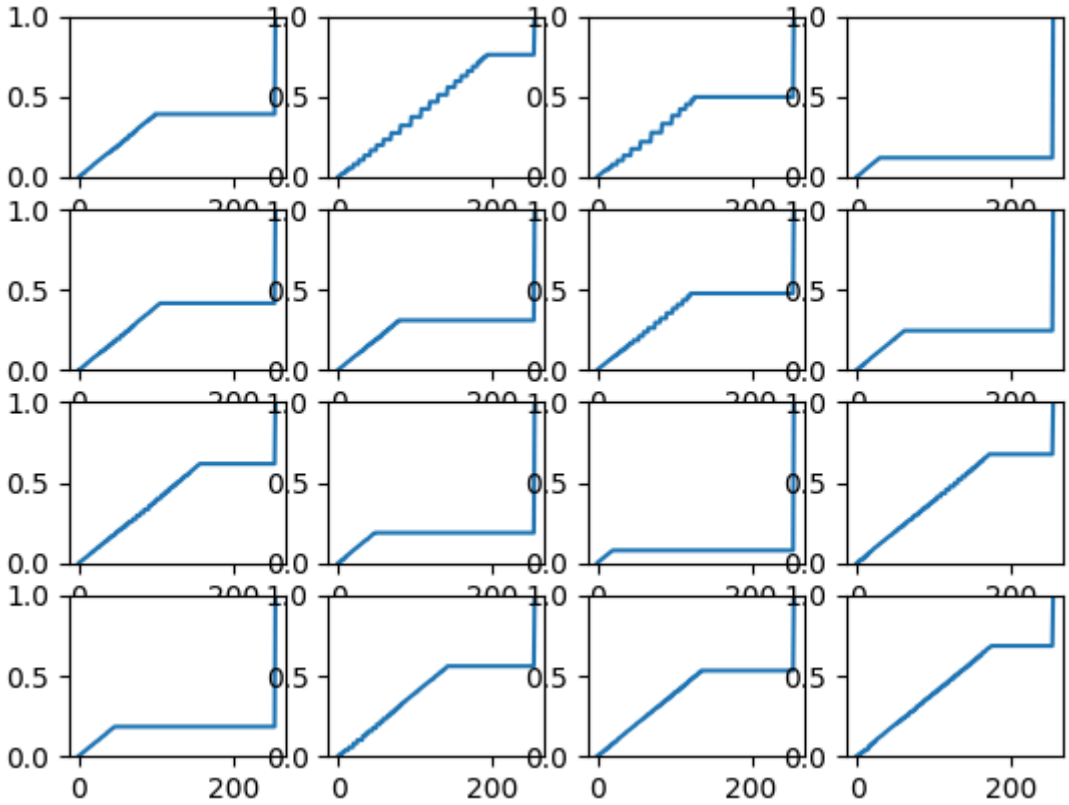
- subimg處理後
  - image



- histogram



- cdf



References

- https://www.youtube.com/watch?v=tn2kmbUVK50&t (https://www.youtube.com/watch?v=tn2kmbUVK50&t)

- https://www.youtube.com/watch?v=ij18XyV4yBQ&list=PLI6pJZaOCtF3s0_Mcwr2AT75DX3Q_SfDh&index=9
  (https://www.youtube.com/watch?v=ij18XyV4yBQ&list=PLI6pJZaOCtF3s0_Mcwr2AT75DX3Q_SfDh&index=9)
- https://www.youtube.com/watch?v=l8E6SVW0bRA&list=PLI6pJZaOCtF3s0_Mcwr2AT75DX3Q_SfDh&index=10
  (https://www.youtube.com/watch?v=l8E6SVW0bRA&list=PLI6pJZaOCtF3s0_Mcwr2AT75DX3Q_SfDh&index=10)
- https://www.youtube.com/watch?v=a7WAMC8qvI4&list=PLI6pJZaOCtF3s0_Mcwr2AT75DX3Q_SfDh&index=11
  (https://www.youtube.com/watch?v=a7WAMC8qvI4&list=PLI6pJZaOCtF3s0_Mcwr2AT75DX3Q_SfDh&index=11)
- https://ithelp.ithome.com.tw/m/articles/10323301
  (https://ithelp.ithome.com.tw/m/articles/10323301)