

1131 數位影像處理 HW 02

資訊三甲 D1109023 楊孟憲

摘要

此次作業實作分為兩個部分；第一部分需要讀取照片後取得顏色的直方圖 (Histogram)，第二部分則是運算獲得直方圖均化 (Histogram Equalization) 以及生成直方圖均化後的結果。

完整程式碼：[Link to github](#)

1 開發環境以及工具

- 開發環境：macOS
- 程式語言：C++
- 編譯版本：clang++ -std=c++17
- 下載 openCV4

2 專案目錄

<code>./images/*</code>	準備好兩組照片
<code>./res/*</code>	圖片輸出結果
<code>app.cpp</code>	主程式
<code>./a.out</code>	執行檔案

3 直方圖 (Histogram)

實作讀取圖片後，分析顏色的 B, G, R 的分佈，並生成直方圖。(0 ~ 255)


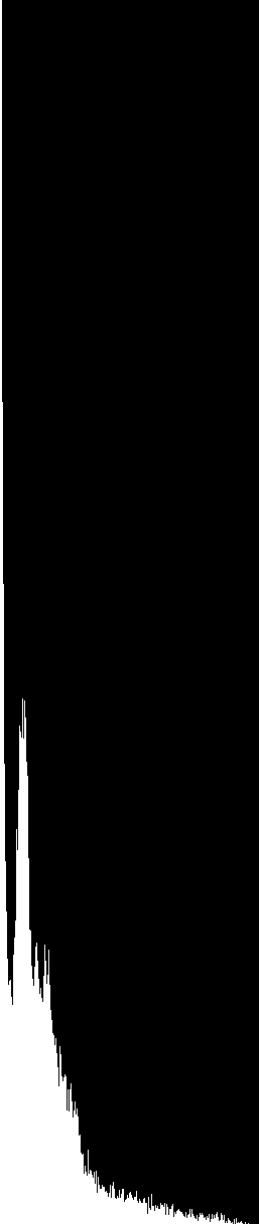
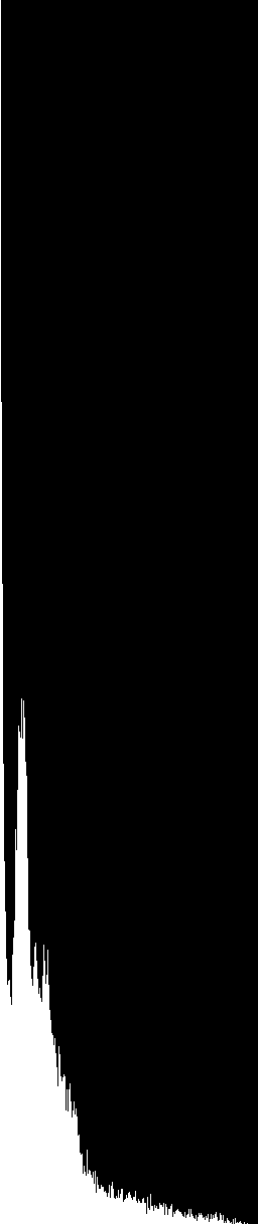
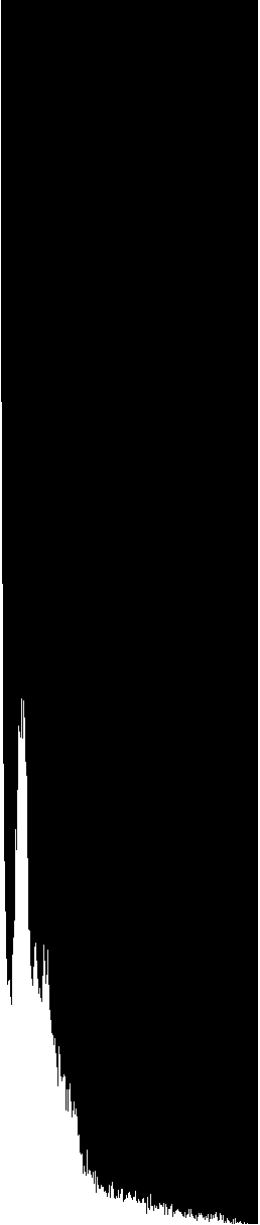

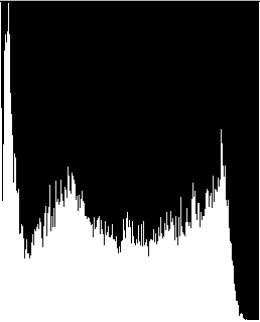
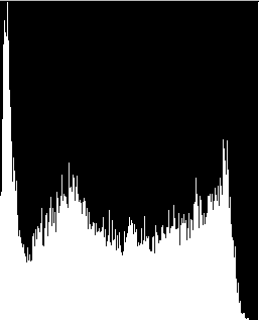
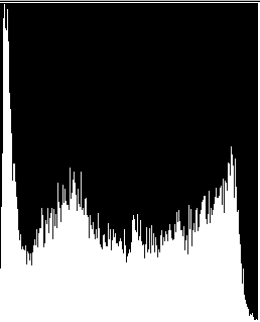
這邊實作是以預設為三個通道的方式來做實作。以下講解彩色、灰階以及黑白的不同。


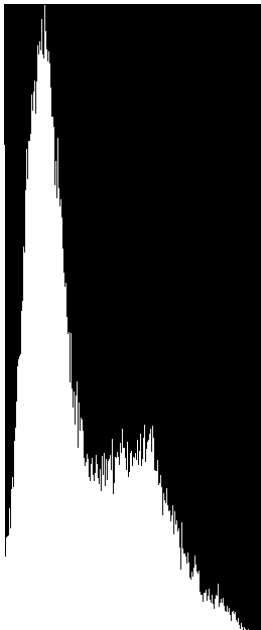
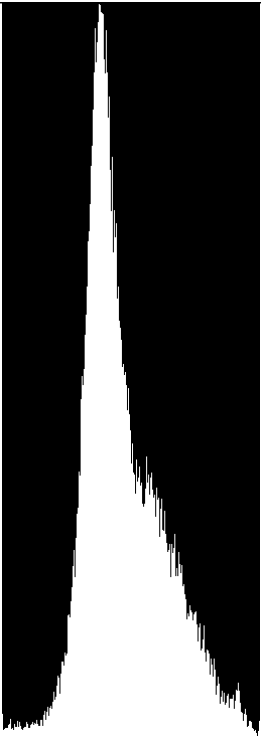


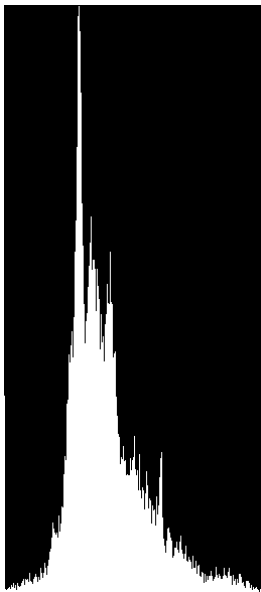

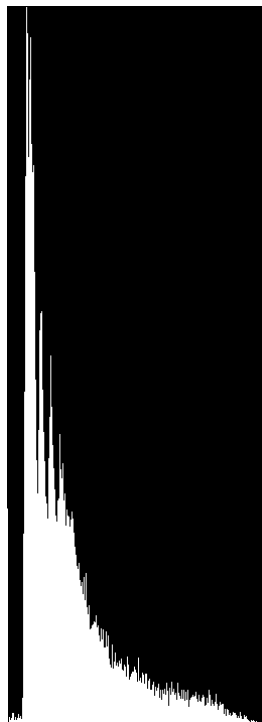
- 彩色：B, G, R 通道值可能不一樣 (介於 0 255 之間)。
- 灰階：B, G, R 通道的值必須一樣 (介於 0 255 之間)。
- 黑白：B, G, R 通道的值必須一樣 (只能為兩個數值，通常設定為 0 和 255)

PS1. 因為避免在像素很多的時候有顏色偏差導致直方圖某些值變太高，不易展示和閱覽，所以會將圖片大小控制在一定的範圍中。我會先手動將圖片寬度訂在 200px。

PS1. 不管彩色或是黑白，輸出的直方圖都是三張，如果是黑白，就是輸出三張長得一樣的圖直方圖。

3.1 結果展示

Channel/Image	Origin	B	G	R
Tiger				
Woman				

Channel/Image	Origin	B	G	R
Dog				
Canada				

3.2 實作代碼及註解

實作一個 struct，其中包含了原始圖片、均化後的圖片、均化前的直方圖 (B, G, R)、均化後的直方圖 (B, G, R)，以及圖片的寬高。另外使用 C++ 內建的 STL 工具，創建直方圖 (均前 /

後) 的 mapping 關係，key 代表顏色強度；value 代表出現的個數。

另外創建一個 vector 用來記錄前綴和。

Listing 1: Image struct and Histogram generation

```
1 struct Image {
2     string imageName;
3     Mat originImage, imageAfterProcess;
4     Mat rHisImage, bHisImage, gHisImage;
5     Mat rEquaImage, bEquaImage, gEquaImage;
6     int height, width;
7     map<int, int> rHis, gHis, bHis;
8     map<int, int> rEqua, gEqua, bEqua;
9     vector<int> subSum;
10
11
12     Image(const string imagePath) {
13         this -> originImage = imread(imagePath);
14         this -> height = (this -> originImage).rows;
15         this -> width = (this -> originImage).cols;
16         this -> imageName = getImageName(imagePath);
17
18         // 初始化 ( 可做可不做 )
19         for(int i = 0; i < 256; i++) {
20             rHis[i] = gHis[i] = bHis[i] = 0;
21         }
22     }
23
24     // 取得檔名
25     string getImageName(string imagePath) {
26         size_t lastSlash = imagePath.find_last_of("/");
27         string fileName = (lastSlash == string::npos) ?
28             imagePath :
29             imagePath.substr(lastSlash + 1);
30
31         // 移除副檔名
32         size_t lastDot = fileName.find_last_of(".");
```

```

33     if (lastDot != string::npos) {
34         return fileName.substr(0, lastDot);
35     }
36     return fileName;
37 }
38
39 // 計算 R G B Value 出現的次數
40 void calculateHis() {
41     for(int i = 0; i < height; i++) {
42         for(int j = 0; j < width; j++) {
43             Vec3b &color = originImage.at<Vec3b>(i, j);
44             bHis[color[0]]++; // 出現次數加一
45             gHis[color[1]]++; // ..
46             rHis[color[2]]++; // ..
47         }
48     }
49
50     rHisImage = generateHis(rHis);
51     gHisImage = generateHis(gHis);
52     bHisImage = generateHis(bHis);
53
54     return;
55 }
56
57 // 根據直方圖的 map 生成直方圖
58 Mat generateHis(map<int, int>& color) {
59     // 先尋找直方圖的最大高度
60     int maxHeight = -INF;
61     for(auto it : color) {
62         maxHeight = max(maxHeight, it.second);
63     }
64
65     maxHeight += 1;
66
67     Mat image(maxHeight, 256, CV_8UC3, Scalar(0, 0, 0));
68

```

```

69 // j 就是 x，也就是 0 ~ 255
70 for(int j = 0; j < 256; j++) {
71     int amountOfColor = color[j];
72     for(int i = 0; i < maxHeight; i++) {
73         // 如果當前 y 軸的值比 color[i] 來得小那就標記白色(255)
74         if ( maxHeight - i <= amountOfColor ) {
75             Vec3b &curColor = image.at<Vec3b>(i, j);
76             for(int k = 0; k < 3; k++) curColor[k] = 255;
77         }
78     }
79 }
80 return image;
81 }
82 }

```


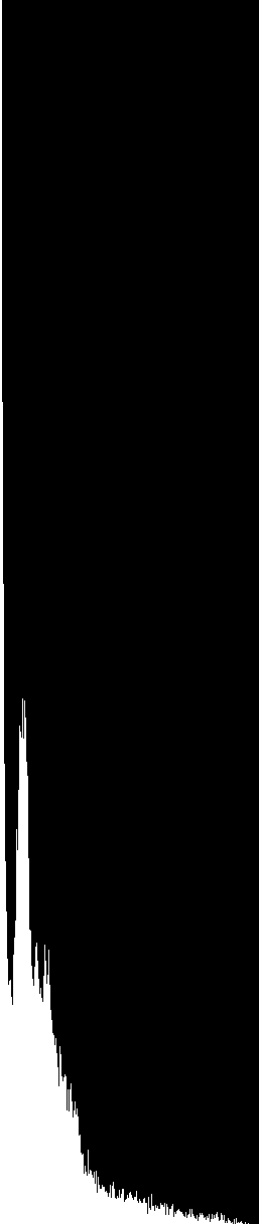
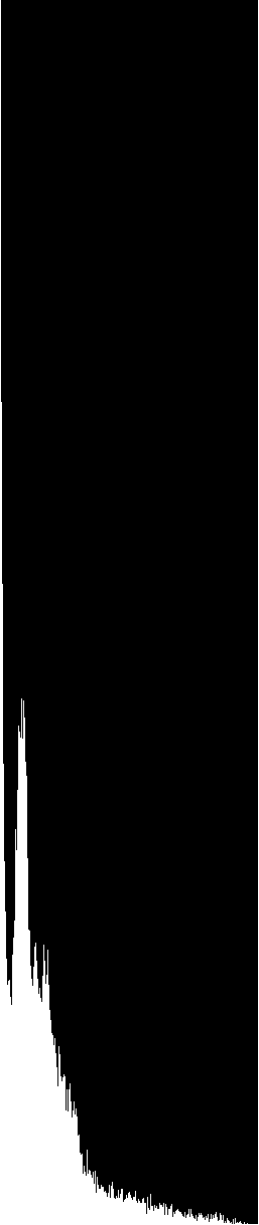
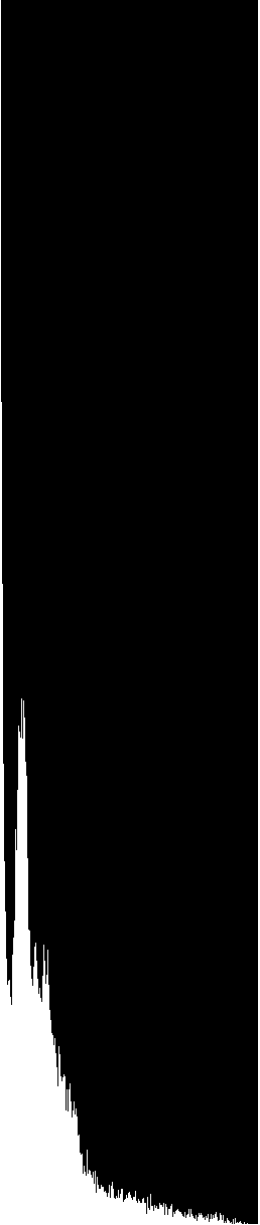

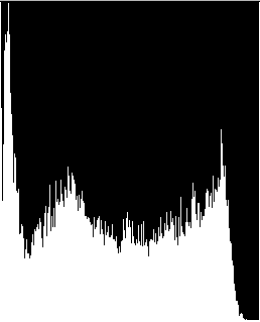
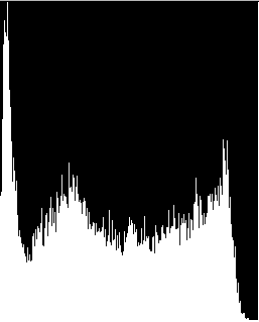
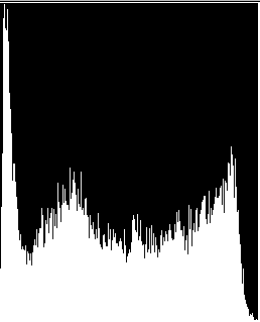
4 直方圖均化 (Histogram Equalization)


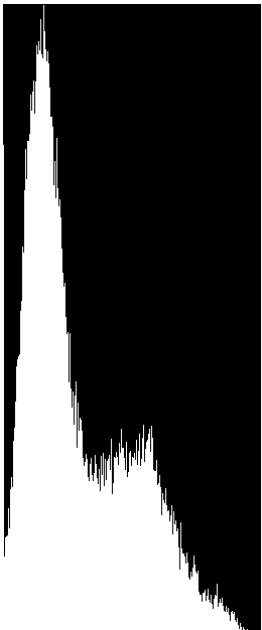
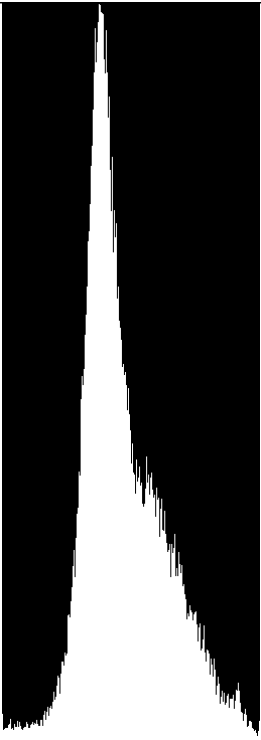
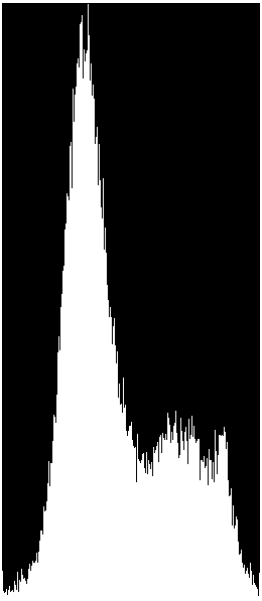

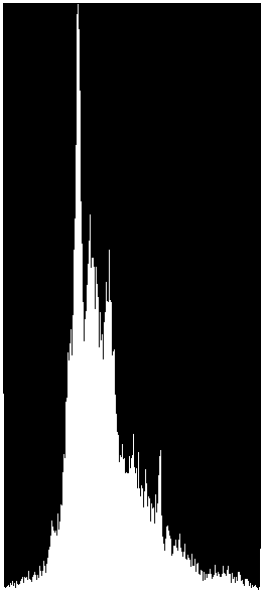

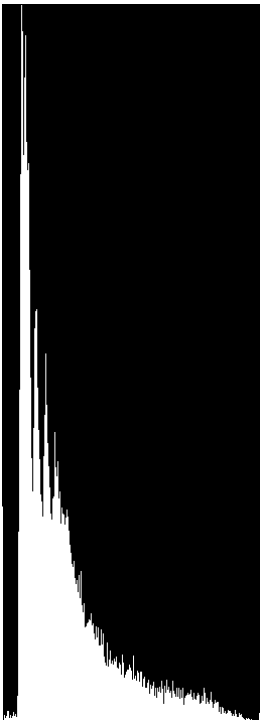
將直方圖均化的意義就是避免差值過大，所以用分佈的概念來均攤個強度出現的次數。實作方法很簡單，使用前綴和計算當下狀態的分佈狀況後，再重新分布。

為了保持程式結構清晰，將均化前後的映射表（map）分開宣告，這樣可以直接進行像素值的轉換，提高程式的可讀性和效能。

計算公式如下：

$$newValue = value \times \frac{subSum}{totalSum}$$

Channel/Image	Origin	B	G	R
Tiger				
Woman				

Channel/Image	Origin	B	G	R
Dog				
Canada				

4.1 實作代碼及註解

Listing 2: Histogram Equalization

```

1 struct Image {
2     string imageName;
3     Mat originImage, imageAfterProcess;
4     Mat rHisImage, bHisImage, gHisImage;
5     Mat rEquaImage, bEquaImage, gEquaImage;
6     int height, width;
7     map<int, int> rHis, gHis, bHis;
8     map<int, int> rEqua, gEqua, bEqua;
9     vector<int> subSum;
10
11
12     Image(const string imagePath) {
13         this -> originImage = imread(imagePath);
14         this -> height = (this -> originImage).rows;
15         this -> width = (this -> originImage).cols;
16         this -> imageName = getImageName(imagePath);
17
18         for(int i = 0; i < 256; i++) {
19             rHis[i] = gHis[i] = bHis[i] = 0;
20         }
21     }
22
23
24     void calculateHis() {
25         for(int i = 0; i < height; i++) {
26             for(int j = 0; j < width; j++) {
27                 Vec3b &color = originImage.at<Vec3b>(i, j);
28                 bHis[color[0]]++;
29                 gHis[color[1]]++;
30                 rHis[color[2]]++;
31             }
32         }
33
34
35         rEquaImage = generateEquaHis(rHis, rEqua);
36         gEquaImage = generateEquaHis(gHis, gEqua);

```

```

37     bEquaImage = generateEquaHis(bHis, bEqua);
38
39
40     return;
41 }
42
43 // 計算均化後的直方圖表
44 Mat generateEquaHis(map<int, int>& color, map<int, int>& equaColor) {
45     // 兩個參數對應的是均化前和後
46     // 計算前綴和
47     int sum = 0;
48     for(int i = 0; i < 256; i++) {
49         sum += color[i];
50         subSum.pb(sum);
51     }
52
53     // 計算新的 key 值，參考以上公式
54     for(int i = 0; i < 256; i++) {
55         int curKey = 255.0 * ((double) subSum[i] / (double) subSum.rbegin
56             ());
57         equaColor[curKey] += color[i];
58     }
59
60     return generateHis(equaColor);
61 }
62
63 // 根據直方圖的 map 生成直方圖
64 Mat generateHis(map<int, int>& color) {
65     // 先尋找直方圖的最大高度
66     int maxHeight = -INF;
67     for(auto it : color) {
68         maxHeight = max(maxHeight, it.second);
69     }
70
71     maxHeight += 1;

```

```

72     Mat image(maxHeight, 256, CV_8UC3, Scalar(0, 0, 0));
73
74     // j 就是 x，也就是 0 ~ 255
75     for(int j = 0; j < 256; j++) {
76         int amountOfColor = color[j];
77         for(int i = 0; i < maxHeight; i++) {
78             // 如果當前 y 軸的值比 color[i] 來得小那就標記白色(255)
79             if ( maxHeight - i <= amountOfColor ) {
80                 Vec3b &curColor = image.at<Vec3b>(i, j);
81                 for(int k = 0; k < 3; k++) curColor[k] = 255;
82             }
83         }
84     }
85     return image;
86 }
87
88 // 生成均化後的圖片
89 void transformImage() {
90     Mat background(height, width, CV_8UC3, Scalar(0, 0, 0));
91     imageAfterProcess = background;
92
93     for(int i = 0; i < height; i++) {
94         for(int j = 0; j < width; j++) {
95             Vec3b originColor = originImage.at<Vec3b>(i, j);
96             Vec3b &curColor = imageAfterProcess.at<Vec3b>(i, j);
97
98             // 根據 b, g, r 計算新的數值並替換
99             int bColorOfOrigin = originColor[0];
100             int bColorOfProcess = 255 * subSum[bColorOfOrigin] / *subSum.
                rbegin();
101             curColor[0] = bColorOfProcess;
102
103             int gColorOfOrigin = originColor[1];
104             int gColorOfProcess = 255 * subSum[gColorOfOrigin] / *subSum.
                rbegin();
105             curColor[1] = gColorOfProcess;









```

```

106
107
108     int rColorOfOrigin  = originColor[2];
109     int rColorOfProcess = 255 * subSum[rColorOfOrigin] / *subSum.
        rbegin();
110     curColor[2] = rColorOfProcess;
111 }
112 }
113 return;
114 }
115
116 };

```

5 前後對比圖

Compare/Image	Origin	Process
Tiger		
Woman		
Dog		
Canada		

6 心得

這次的直方圖均化作業讓我深入理解了數位影像處理中的幾個重要概念：

- 掌握了影像直方圖的分析方法，包括 RGB 三個色彩通道的分布特性
- 理解了直方圖均化的原理，學會運用累積分布函數（CDF）來改善影像對比度
- 實作過程中對 OpenCV 的操作更加熟悉，特別是在矩陣運算和色彩空間轉換方面
- 透過觀察均化前後的結果，更加理解影像增強技術對視覺效果的影響

在實作過程中，雖然遇到了一些技術細節需要克服，像是圖片資訊過大導致直方圖難以閱讀以及如確保均化後的像素值正確映射到 $[0,255]$ 範圍，但通過查詢資料和反覆驗證，最終都順利解決了這些問題。這次作業不僅強化了程式實作能力，也加深了對影像處理理論的認識。