

Digital Image Processing

數位影像處理

Homework1

**課號 : CO6041
教師 : 唐之瑋
助教 : 詹豐鎧
林邑豪**

**姓名 : 楊中毅
學號 : 110327007**

一、實驗步驟說明

- 1-1. 依照 **bitmap file** 格式，將給定 **.bmp** 檔案的資料 **Header** 段
讀出並顯示。
- 1-2. 將給定 **24-bit bmp** 影像 (**RGB** 影像)，轉為 **8-bit bmp** 影像
(**灰階**影像)。
- 1-3. 計算 1-2. 轉灰階影像結果之累計灰階直方圖，並畫出。
- 1-4. 將 1-2. 轉灰階影像結果按照以下條件進行下取樣。
 - 1-4-1. 沿 **even rows** 下取樣
 - 1-4-2. 沿 **odd rows** 下取樣
 - 1-4-3. 上下方向合併 1-4-1. 與 1-4-2. 兩張圖
- 1-5. 計算 1-4. 結果之累計灰階直方圖，並畫出。

二、學習目的

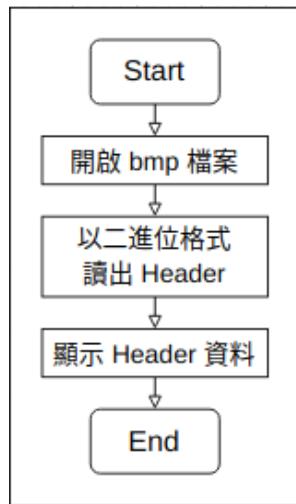
學習基本影像儲存格式Bitmap，了解儲存格式各項欄位定義，並使用 C 語言練習開檔寫檔等動作。

利用上課所學影像原理，驗證基礎影像處理操作，例如將 RGB 三通道的彩色影像轉為灰階影像，並進行下取樣合併等資料操作。

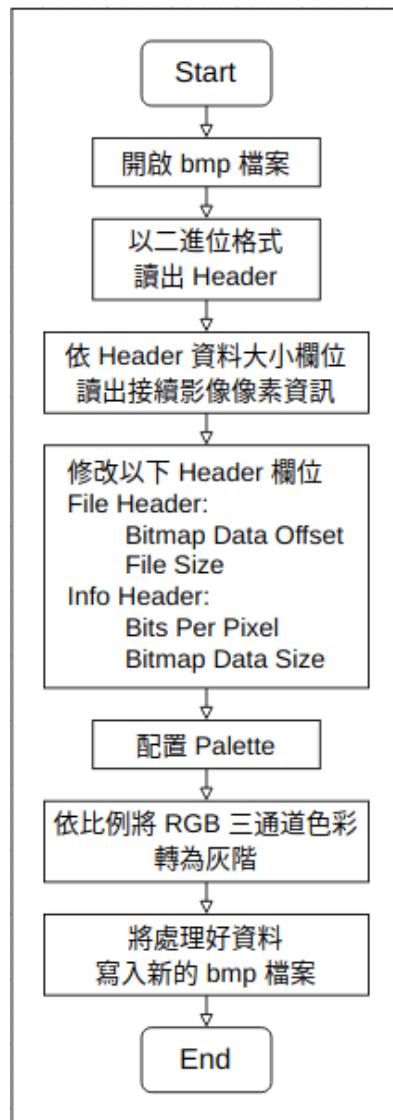
另外額外練習 32 位元影像格式，下取樣，灰階轉換(負片)，旋轉、鏡像、上下顛倒等基本操作。

三、實驗步驟流程圖

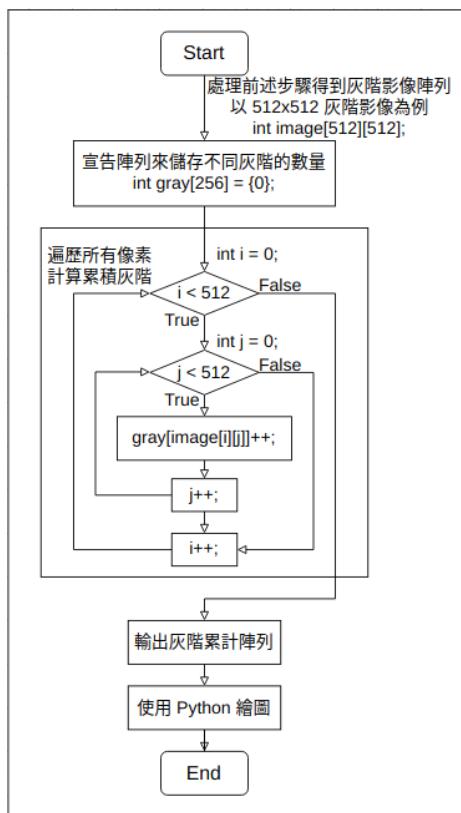
1-1.



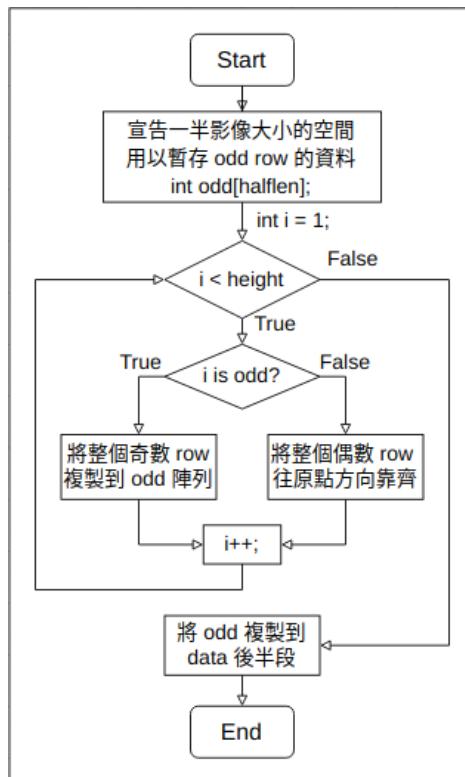
1-2.



1-3.



1-4.



1-5. 步驟同 1-3.

四、實驗結果

1-1. 依照 bitmap file 格式，將給定 .bmp 檔案的資料 Header 段 讀出並顯示。

依 bitmap 格式封裝 header 各欄位，開檔時以給定格式解析各欄位讀出的值，並顯示在終端機上。

成功解讀出 header 重要資訊，如 datasize, width, height, bit_per_pixel ...等等。

```
#pragma pack(1)
typedef struct {
    uint16_t ID;
    uint32_t file_size;
    uint32_t reserved;
    uint32_t offset;
} BMP_FILE_HEADER_t;
#pragma pack()

#pragma pack(1)
typedef struct {
    uint32_t hdr_size;
    uint32_t width;
    uint32_t height;
    uint16_t planes;
    uint16_t bits_per_pixel;
    uint32_t compression;
    uint32_t data_size;
    uint32_t H_res;
    uint32_t V_res;
    uint32_t used_colors;
    uint32_t important_colors;
} BMP_INFO_HEADER_t;
#pragma pack()

typedef struct {
    BMP_FILE_HEADER_t file_header;
    BMP_INFO_HEADER_t info_header;
    uint8_t palette[PALETTE_SIZE];
    uint8_t *data;
} BMP_t;
```

```
cy023@cy023-e14:~/project/DIP-Fall2021/ImageFormat/BMP$ ./hw1.out
Origin Image.

===== BMP FILE HEADER =====
| ID          : 19778 |
| file_size   : 786486 |
| offset      : 54     |
=====
===== BMP INFO HEADER =====
| hdr_size    : 40    |
| width       : 512   |
| height      : 512   |
| planes      : 1     |
| bit_per_pixel : 24   |
| compression  : 0     |
| data_size   : 786432 |
| H_res       : 11811 |
| V_res       : 11811 |
| used_colors : 0     |
| important_colors : 0 |
=====
```

1-2. 將給定 24-bit bmp 影像 (RGB 影像), 轉為 8-bit bmp 影像 (灰階影像)。

灰階圖片須加入調色盤, Data offset 會增加 1024 Bytes.

```
for (i = 0; i < 256; i++) {
    bmp->palette[(4 * i)] = i; // B
    bmp->palette[(4 * i) + 1] = i; // G
    bmp->palette[(4 * i) + 2] = i; // R
    bmp->palette[(4 * i) + 3] = 0;
}
```

首先修改 Header 內需要改動欄位的值。
bit per pixels 改為 8bit. datasize, filesize 皆需調整大小。

```
bmp->file_header.offset = HEADER_SIZE + PALETTE_SIZE;
bmp->info_header.bits_per_pixel = 8;
pixelSize = bmp->info_header.width * bmp->info_header.height;
bmp->info_header.data_size = pixelSize * bmp->info_header.bits_per_pixel/8;
bmp->file_header.file_size = bmp->file_header.offset + \
                                bmp->info_header.data_size;
```

最後將 R G B 三通道數值按照比例分配為灰階

```
for (i = 0; i < bmp->info_header.height; i++) {
    for (j = 0; j < bmp->info_header.width; j++) {
        bmp->data[i * bmp->info_header.width + j] =
            (bmp->data[3 * (i * bmp->info_header.width + j)] * 0.114) +
            (bmp->data[3 * (i * bmp->info_header.width + j) + 1] * 0.587) +
            (bmp->data[3 * (i * bmp->info_header.width + j) + 2] * 0.299);
    }
}
```

可以將多餘空間釋放

```
bmp->data = (uint8_t *)realloc(bmp->data, bmp->info_header.data_size);
if (!bmp->data) {
    printf("[ERROR] : realloc failed.\n");
    exit(1);
}
```



1-3. 計算 1-2. 轉灰階影像結果之累計灰階直方圖，並畫出。

計算累積灰階 pixel 數，開一 256 大小的陣列 buff 用以儲存累積值，遍歷所有 pixel，data[i] 即為將每個像素對應的灰階值(落於 0 ~ 255)，將 buff[data[i]]++; 最終的結果就是我們要的累積灰階數。(buff[0] 的值就是所有灰階值為 0 的 pixel 數量)

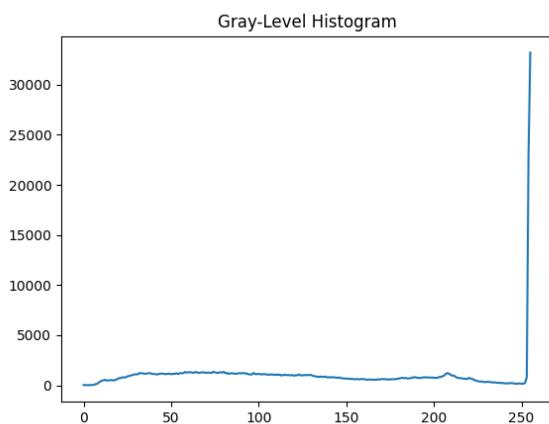
```
uint32_t buff[256] = {0};  
uint32_t i;  
  
for (i = 0; i < dataLength; i++) {  
    buff[data[i]]++;  
}
```

將 buff 陣列寫到一 txt 文字檔

```
FILE *fd;  
fd = fopen(path, "w");  
if (!fd) {  
    printf("[ERROR] : open %s bmp failed.\n", path);  
    exit(1);  
}  
  
for (i = 0; i < 256; i++) {  
    fprintf(fd, "%d ", buff[i]);  
}  
fclose(fd);
```

用 python 讀出文字檔內數值並畫圖

```
import matplotlib.pyplot as plt  
  
with open("./script/gray.txt") as f:  
    data1 = f.read()  
data1 = [int(i) for i in data1.strip().split(" ")]  
  
plt.plot(data1)  
plt.title("Gray-Level Histogram")  
plt.savefig('./script/Gray-Level_Histogram.png')  
plt.show()
```



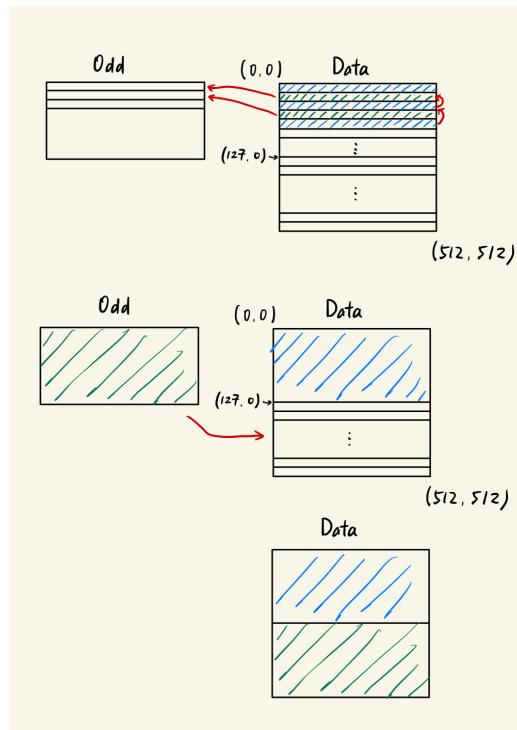
1-4. 將 1-2. 轉灰階影像結果按照以下條件進行下取樣。

1-4-1. 沿 even rows 下取樣

1-4-2. 沿 odd rows 下取樣

1-4-3. 上下方向合併 1-4-1. 與 1-4-2. 兩張圖

步驟如圖，將偶數 row 往原空間原點方向靠齊，奇數 row 存至另外空間 odd 陣列，最後將 odd 陣列寫回原空間後半段。



```
uint32_t i, j;
uint32_t halflen = bmp->info_header.data_size / 2;
uint32_t rowsize = bmp->info_header.width * (bmp->info_header.bits_per_pixel / 8);

uint8_t *odd = (uint8_t *)malloc(halflen * sizeof(uint8_t));

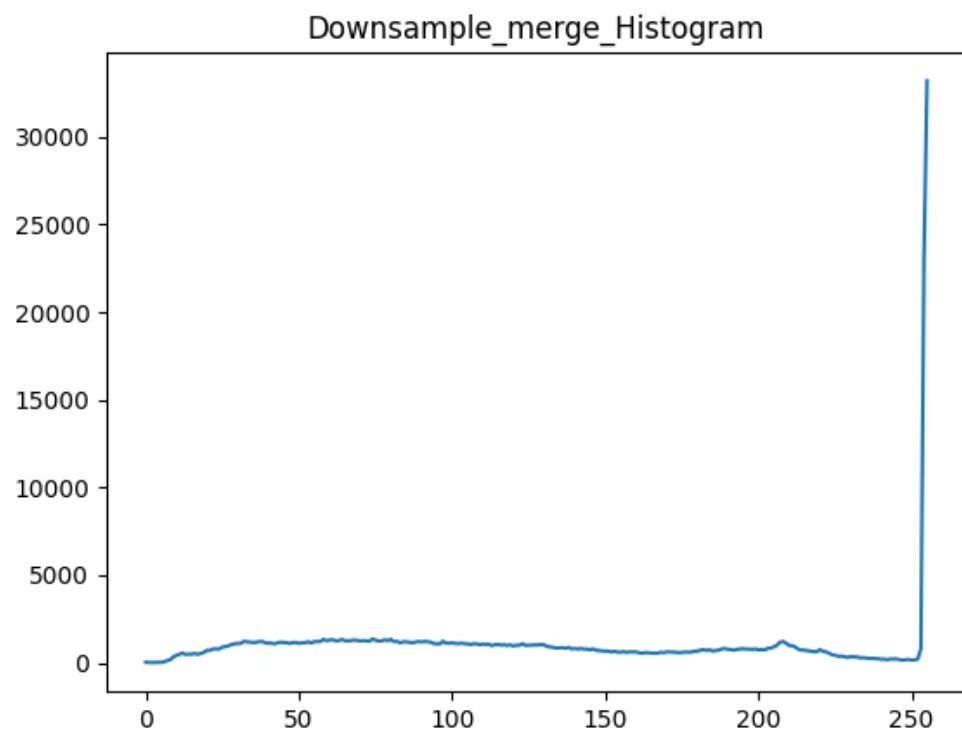
for (i = 1; i < bmp->info_header.height; i++) {
    if ((i % 2)) {
        memcpy(&(odd[((i-1)/2) * rowsize]), &(bmp->data[i * rowsize]), rowsize);
    } else {
        memcpy(&(bmp->data[(i/2) * rowsize]), &(bmp->data[i * rowsize]), rowsize);
    }
}
memcpy(&(bmp->data[halflen]), odd, halflen);
free(odd);
```



1-5. 計算 1-4. 結果之累計灰階直方圖，並畫出。

計算方式同 1-3.

因為只是同一張影像，像素排列位置不同，不影響灰階直方圖的計算結果。



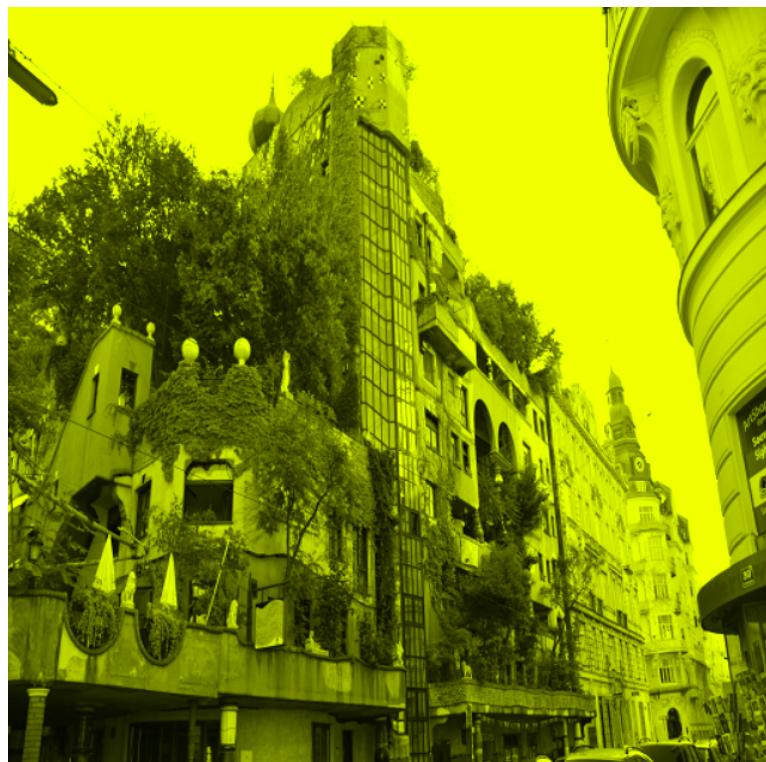
五、分析

在使用 struct 包裝 header 時，需要注意記憶體對齊，若結構大小為奇數，在實際配置記憶體時可能被補齊為二的次方倍大小，在此以 #pragma pack(1) 語法，讓編譯器以 1 byte 對齊，以免後續使用時，應是連續空間的配置中間被穿插 padding。

注意當寬度不是四的倍數的時候，會存在 padding 的空間，需要小心處理。當高度設為負數，可以發現圖片上下顛倒。

調色盤的使用為色光三原色加上光強度的混色，例如將調色盤的藍色成份關閉，保留紅綠兩色光，最後圖片會呈黃色。

```
for (i = 0; i < 256; i++) {
    bmp->palette[(4 * i)] = 0; // B
    bmp->palette[(4 * i) + 1] = i; // G
    bmp->palette[(4 * i) + 2] = i; // R
    bmp->palette[(4 * i) + 3] = 0;
}
```



RGB 轉灰階的公式是來自 RGB 空間 \leftrightarrow YUV 空間的轉換，灰階只取 Y 成份 (Luminance 明亮度)。

灰階轉換有許多轉換方式，練習將灰階圖片轉負片。

```
for (i = 0; i < bmp->info_header.height; i++) {  
    for (j = 0; j < bmp->info_header.width; j++) {  
        bmp->data[i * bmp->info_header.width + j] =  
            ~bmp->data[i * bmp->info_header.width + j];  
    }  
}
```



影像處理涉及到大筆資料的計算搬移，在設計演算法時需要特別考慮時間複雜度與空間複雜度方可更有效率的解決問題。

另額外練習其他基礎影像操作、旋轉、上下顛倒、左右鏡像 ...



多次下取樣



以上程式皆在附錄程式碼內

六、Reference

- <https://crazycat1130.pixnet.net/blog/post/1345538#mark-11>
- https://en.wikipedia.org/wiki/Claude_Shannon