

# 影像處理作業 1

- 系級：資訊工程系一年級
- 姓名：楊啟弘
- 學號：7113056083

本作業由 **C++** 實作出由多張同一場景的 **JPEG** 影像進行 **像素值相加取平均** 與 **中位數計算**，以去除影像中的 **隨機噪點**。

## 目錄結構

```
hw1/  
├─ img/                // 存放測試的 JPEG 檔案  
├─ include/  
│   ├── jpeg_reader.h  
│   └─ save_ppm.h  
├─ src/  
│   ├── avg.cpp        // 平均值合併 (去噪)  
│   ├── median.cpp     // 中位數合併 (去噪)  
│   ├── jpeg_reader.cpp // JPEG 讀取實作  
│   └─ save_ppm.cpp    // PPM 輸出實作  
├─ output/            // 存放結果 PPM 檔案  
├─ static/            // 存放結果 jpeg 檔案  
├─ build_avg.sh        // 編譯 avg  
├─ build_median.sh     // 編譯 median  
└─ ppm_to_jpg.py       // 將 ppm 檔轉成 jpg 檔
```

## 程式碼說明

本專案包含四個主要 **C++** 檔案，用於讀取多張 **JPEG** 圖片、計算像素平均值，並將結果輸出為 **PPM** 圖片：

- `avg.cpp`
- `median.cpp`
- `jpeg_reader.h` / `jpeg_reader.cpp`
- `save_ppm.h` / `save_ppm.cpp`

### avg.cpp

- 主程式，負責：
  1. 解析命令列參數，取得輸入 **JPEG** 檔案列表。

```

if (argc != 2) {
    std::cerr << "Usage: " << argv[0] << " <image_count>\n";
    return 1;
}

int totalImages = 0;
totalImages = std::stoi(argv[1]);

```

2. 呼叫 `read_JPEG_file` 讀取每張圖片，並累加 像素值。

```

int totalImages = argc - 1;

for (int i = 1; i <= totalImages; ++i) {
    int w, h, c;
    unsigned char* buf = read_JPEG_file(argv[i], w, h, c);
    ...
    int pixelCount = width * height * channels;
    for (int j = 0; j < pixelCount; ++j) {
        sumBuffer[j] += buf[j];
    }
    ...
}

```

3. 計算 平均像素值，結果存入 `avgBuf`。

```

std::vector<unsigned char> avgBuf(width * height * channels);

for (size_t i = 0; i < sumBuffer.size(); ++i) {
    int v = int(float(sumBuffer[i]) / validCount + 0.5f);
    avgBuf[i] = static_cast<unsigned char>(std::min(v, 255));
}

```

4. 建立 `output` 資料夾，並呼叫 `save_PPM` 輸出 PPM 檔案。

```

// 確保 output 資料夾存在
mkdir("output", 0755);

// 建立輸出檔名: avg_result_<validCount>.ppm
std::string outPath = "output/avg_result_" +
    std::to_string(validCount) +
    ".ppm";

if (!save_PPM(outPath.c_str(), avgBuf.data(), width, height, channels)) {
    std::cerr << "儲存結果失敗: " << outPath << "\n";
    return 1;
}

std::cout << "完成! 結果存入 " << outPath << std::endl;

```

## median.cpp

- 主程式，改用 中位數 方式去噪：

1. 解析命令列參數，取得 **JPEG** 檔案列表。
2. 呼叫 `read_JPEG_file` 讀取每張圖片，並將同一像素位置的值收集到 `pixelValues` 向量中。

```
int pixelCount = width * height * channels;
for (int j = 0; j < pixelCount; ++j) {
    pixelValues[j].push_back(buf[j]);
}
```

3. 對每個像素位置的值向量排序，取中間位置元素作為 *中位數*（偶數張時取偏右）。

```
// 計算每個像素的中位數
int pixelCount = width * height * channels;
std::vector<unsigned char> medianBuf(pixelCount);
for (int j = 0; j < pixelCount; ++j) {
    auto &vals = pixelValues[j];
    std::sort(vals.begin(), vals.end());
    int mid = validCount / 2;
    int med = vals[mid]; // 偶數時選中間偏右的元素
}
```

4. 將結果寫入 `medianBuf`，並確保不超過 255。

```
medianBuf[j] = static_cast<unsigned char>(std::min(med, 255));
```

5. 建立 `output` 資料夾，呼叫 `save_PPM` 輸出 `median_result_<張數>.ppm`。

## jpeg\_reader.h / jpeg\_reader.cpp

- 使用 **libjpeg API** 讀取 **JPEG** 檔案。
- `read_JPEG_file` 函式簽章：

```
unsigned char* read_JPEG_file(const char* filename,
                              int &width,
                              int &height,
                              int &channels);
```

- 主要流程：
  1. `fopen` 開檔、設定解碼結構。
  2. `jpeg_read_header` → `jpeg_start_decompress`。
  3. 逐行 `jpeg_read_scanlines` 將像素寫入 `buffer`。
  4. 完成後釋放資源並回傳 `buffer`。

## save\_ppm.h / save\_ppm.cpp

- 將 Raw 像素資料寫出為 **PPM/PGM** 格式。

- `save_PPM` 函式簽章：

```
bool save_PPM(const char* filename,
              const unsigned char* data,
              int width,
              int height,
              int channels);
```

- **PPM 標頭：**
  - P2 (灰階) 或 P3 (彩色)
  - width height
  - 255
- 寫入像素值並自動換行。

## 結果

---

以下展示部分輸入與輸出圖片：


### 輸入範例

放大明顯看會發現有許多噪點



### 去噪結果比較

平均值去噪

樣本數	範例
10 張	
140 張	

中位數去噪

樣本數	範例

10 張



140  
張



### 討論

- 10 張：經過處理後，放大觀看噪點已 明顯減少
- 140 張：與 10 張對比，幾乎看不出噪點