DigiVFX
HW2 Image Stitching

B02901072 楊喬諳
B02901065 李洺曦

資料夾內包含

```
+- HW2/
  +- report.pdf
  +- README.txt
  +- CMakeList.txt

  +- include/
    +- stitch.h   由許多個cpp檔定義完成，見下方.cpp
    +- util.h     一些parsing, 讀取檔案, 或其他不是很重要的通用函式
    +- tm_usage.h CSIE網路上有提供，可以計算程式所花的時間的library

  +- src/
    +- main.cpp      主程式
    +- moravec.cpp   定義Moravec corner演算法相關的函式
    +- harris.cpp    定義Harris corner演算法相關的函式
    +- sift.cpp      定義SIFT演算法相關的函式
    +- warp.cpp      定義圓柱投影相關的函式
    +- blend.cpp     定義不同Blending演算法相關的函式
    +- feature.cpp   其他跟image stitching會用到的相關函式
    +- tm_usage.cpp  見上方.h
    +- util.cpp      見上方.h

  +- input/
    +- [where_mostly_the_input_images_are]

  +- output/
    +- [where_mostly_the_result_paranoma_are_saved]

  +- build/
    +- main (the excitable file)
    +- Makefile
    +- [and_some_other_cmake_stuffs]
```

目前程式限制（應改進的缺點）

只能處理單一列的圖片
圖片必須照順序輸入，不論是左至右或是右至左皆可
Feather blending處理時，不能有三張以上重疊的影像（Average就不會有影響）
僅假設圖片之間只有平移差異
不能讀png（Channel數為4個），也不能存為png（有透明）

自定義類別

層級由上至下分別為

MyCanvas	將所有跟處理Image Stitching相關包在一起的物件 包含一個MyImage的vector，圓柱投影半徑等資訊
MyImage	將所有跟圖片相關的處理包在一起的物件 包含像素值，圖片名稱，內含關鍵點，遮罩，邊界，焦距等資訊
MyFeature	將所有跟關鍵點相關的處理包在一起的物件 包含位置，描述子，大小，方向性等資訊
Match	單純只是Point2f的一對pair，方便feature matching處理

註：詳細可見stitch.h

程式流程

(1) 讀取相關檔案

MyCanvas(IMAGE_DIR, IMAGE_NUM)

IMAGE_DIR中必須包括pano.txt以給定焦距的值。（其格式見附錄）
若不存在，使用500.0作為預設值。

IMAGE_NUM可限制要讀取的圖片數，
比如資料夾裡有五張，但只想將前三張stitch一起，便將IMAGE_NUM設為3
若為-1，則讀取資料夾內所有的圖片數。

(2) 抓關鍵點

MyCanvas::detecting(DETECT_METHOD)

先對所有圖片依據其焦距不同進行**圓柱投影**
這裡我們寫了三種detect的演算法

(a) Moravec

與講義上的implementation大致相同
我們發現他會在一些較為平坦（如：白天，綠地）的區塊也抓出關鍵點
所以我們有為Moravec為Emin設上一個threshold，減少這些雜點

(b) Harris

與講義上的implementation大致相同
參數設定：
k = 0.005

(c) SIFT

這部分我們另外也參照了這篇的敘述 *Anatomy of the SIFT Method*
http://www.ipol.im/pub/art/2014/82/article_lr.pdf
參數設定：
octave_num = 4
一層octave有2+3個scale
最小scale(sigma) = 1.6

一張800*1200圖約需要5秒

(3) 關鍵點配對
MyCanvas::matching(Y_DEV)
照順序配對N張圖片，生成N-1組vector<Match>
再利用RANSAC，分別生成平移矩陣，並對圖片進行warpAffine
這邊我們會紀錄從第一張到最後一張之間y方向上的總偏差，以供之後的drifting使用
註：SIFT的matching還有些小問題

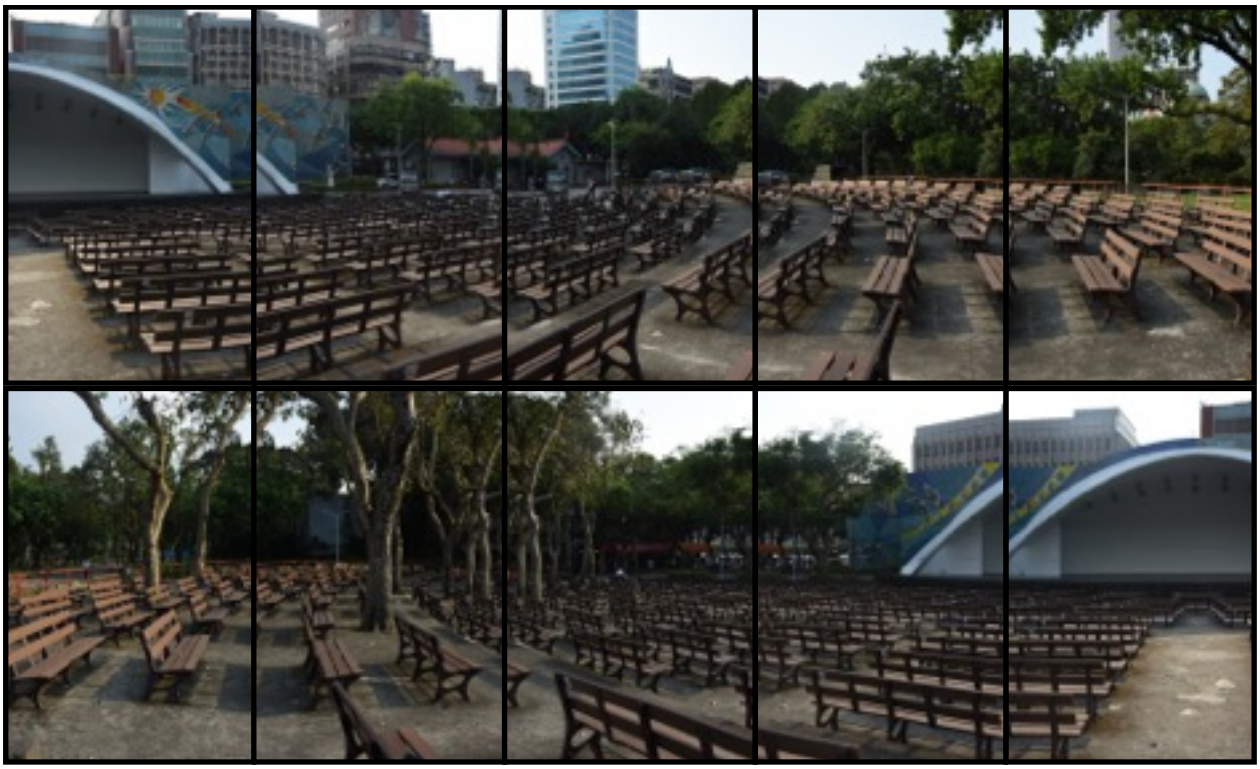
(4) Blending
MyCanvas::detecting(BLEND_METHOD)
這裡我們寫了兩種blending的演算法
(a) average
將一個像素上所有重疊的影響取平均，最簡單的blending方式之一
由於我們的feather演算法有瑕疵（不能重疊三張以上），
因此有時候average會是我們使用的演算法。
但通常來說，由於其在圖片交界上會有明顯的界線，會比feather的效果還差
(b) feather
計算圖片彼此之間的重合區，邊界線等值，計算alpha, beta值
並將兩張圖片作線性重合

(5) Cropping —— 分為兩種
MyCanvas::weakCropping( )
將四個邊界上，「大量多餘」的黑邊去掉
只留下「必要」的黑邊
註：這有些難以解釋，請見後面的圖片說明

MyCanvas::strictCropping( )
將四個邊界上，「所有」的黑邊都去掉
只留下中間的彩色圖片
註：因為我們的演算法假定圖片為彩色的，因此這裡會有限制，不適用於黑白圖片

(6) Drifting
為了改善多張圖片stitch完之後可能的誤差累積
我們對於疊合後的影像做一個global affine warp
y’ = y - ax, 其中 a = Y_DEV / 影像的寬度

接下來我們使用所選用的artifact以及課程提供的parrington做圖片說明（artifact的source images存於input/best/）：

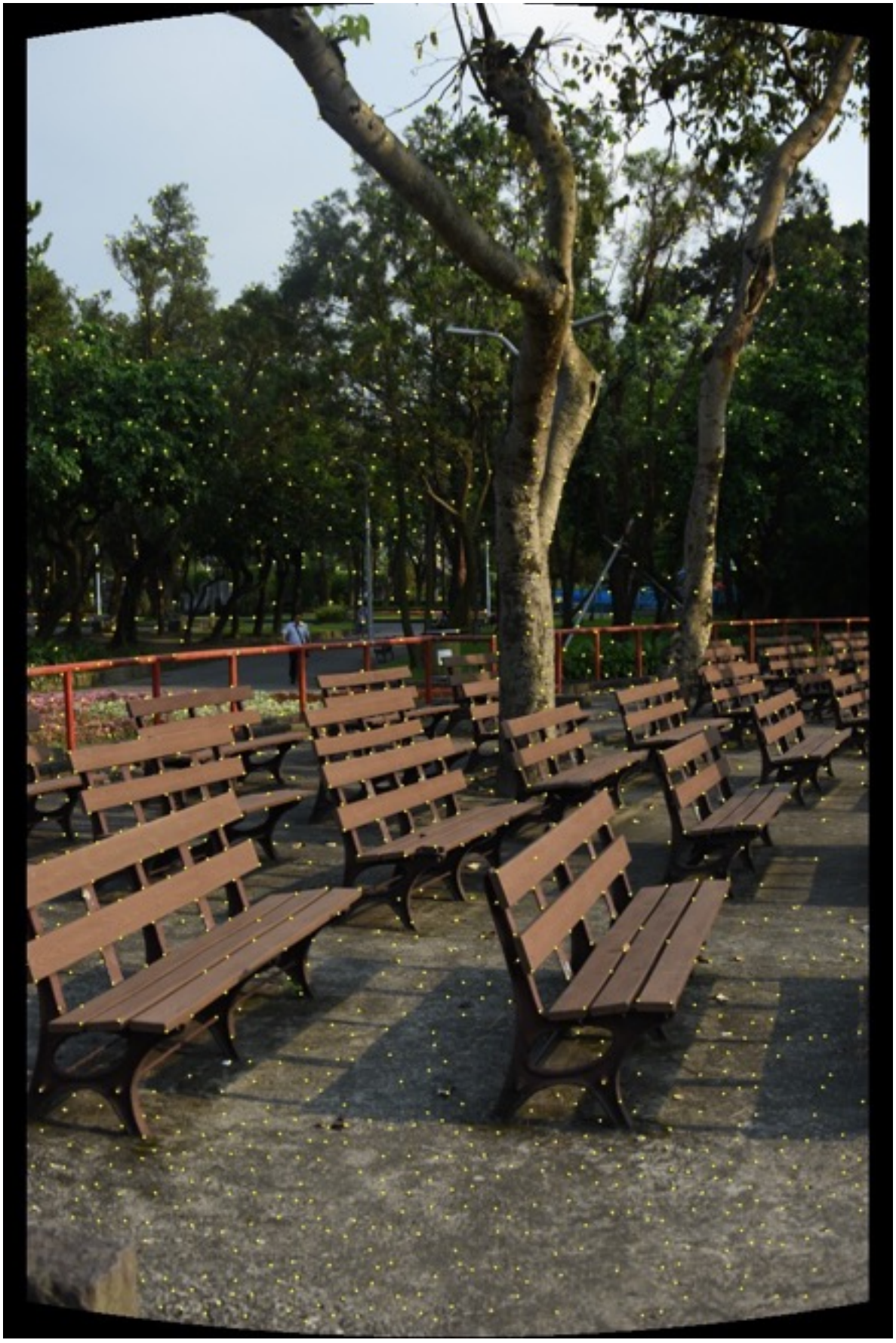


相機/鏡頭規格

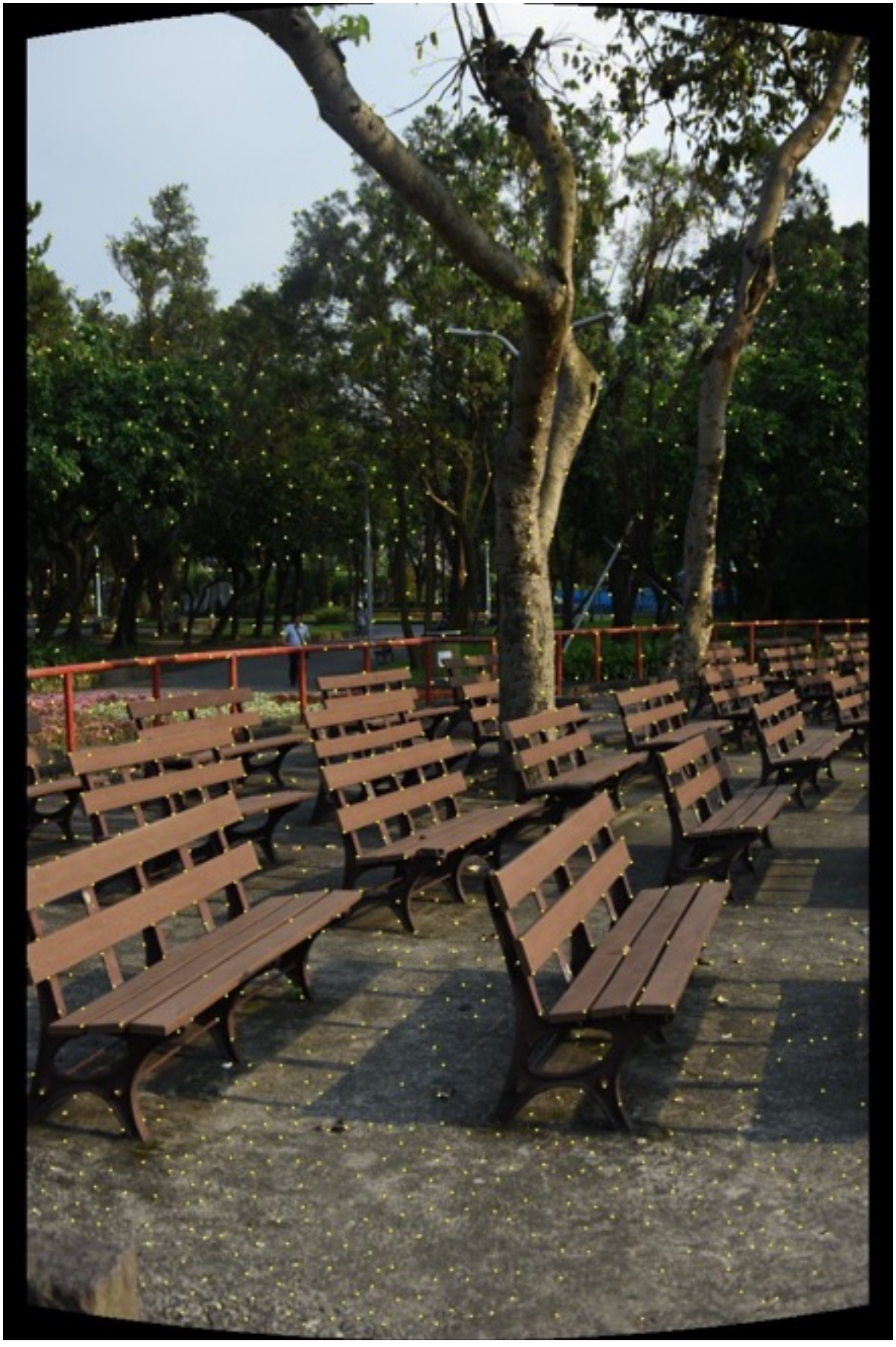
Nikon d7200
Nikon AF-S DX NIKKOR 18-105mm F3.5-5.6G ED VR
f 5.6
快門 1/250
ISO100

圖片解析度經過壓縮為800*1200

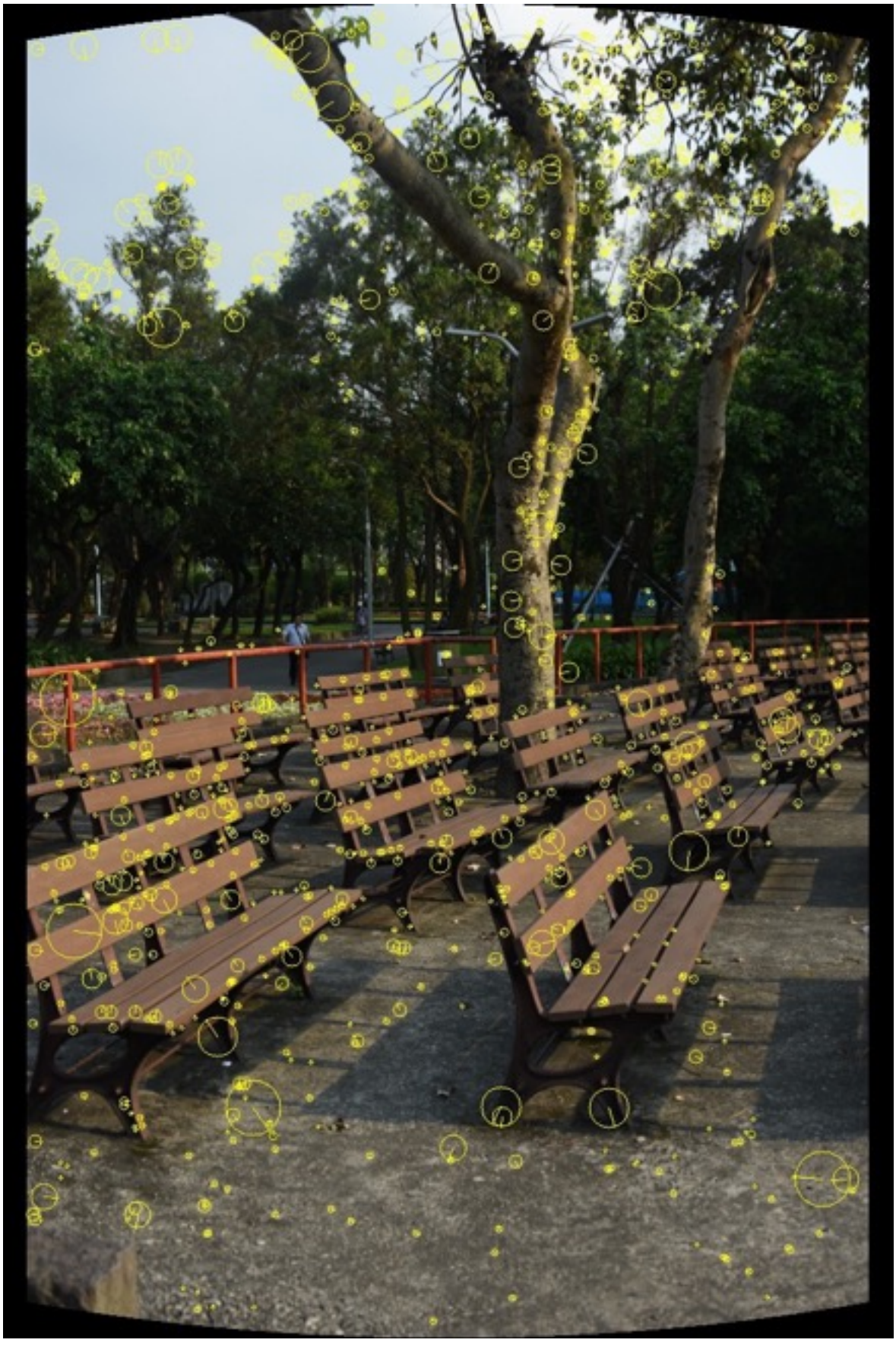
Moravec



Harris



SIFT



Average

Feather

before cropping

after weak cropping (注意周圍的黑邊)

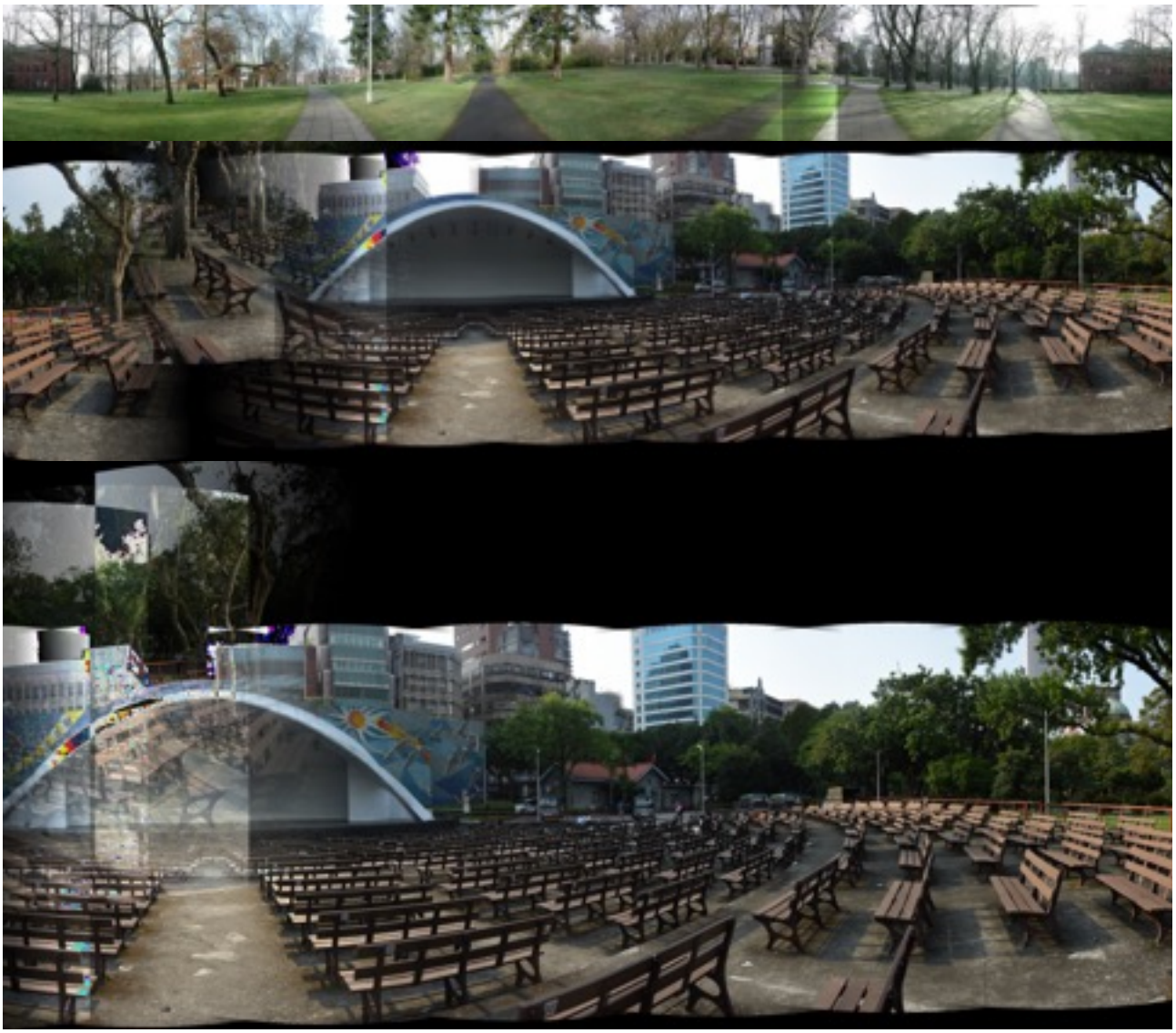
after drifting

after strict cropping => RESULT artifact

我們最後選用的圖：harris, feather

我們的SIFT features matching成果：

並不是很好，整體上，還是有很多瑕疵。推測部分也是因為feather的漏洞

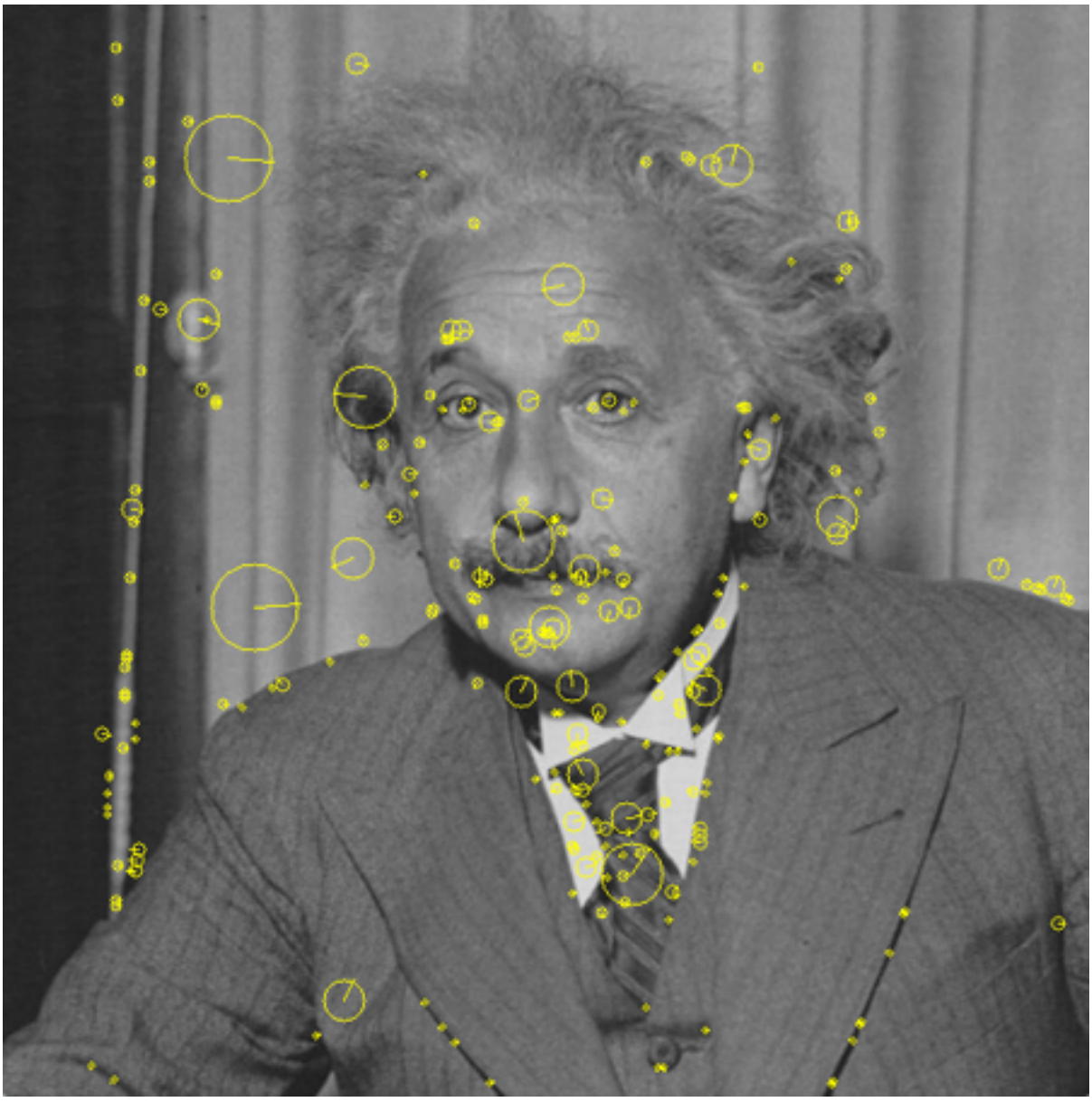


結論與心得

Moravec和Harris其實在我們的實驗成果是差不多不錯的，但仍然會發現一些鬼影。
很可惜SIFT的配對正確率不夠高，沒辦法做出完全合格的成品
在解決圖片輸入必須照順序的問題時，有想到將所有圖片兩兩找matches，並以matches數量作為邊的權重，做成graph後，利用Dijkstra Algorithm找longest path，實驗過後發現這樣的方法似乎不大行。
圖片經過warp或resize之後，周圍生成的黑邊很容易干擾corner detector的運作，程式容易在圖片邊界與四角抓大量的關鍵點，造成不必要的雜點。這個問題困擾我們很久，最後我們建立一個遮罩，在每次圖片變形時，遮罩也隨之變形。而後在corner detector運算時，將落於這個遮罩的點去掉。這個方法改善了我們程式很多。
很多threshold值, 以及抓關鍵點的參數都必須image by image去調整優化，這也是將來必須考慮的改善問題。

附錄：

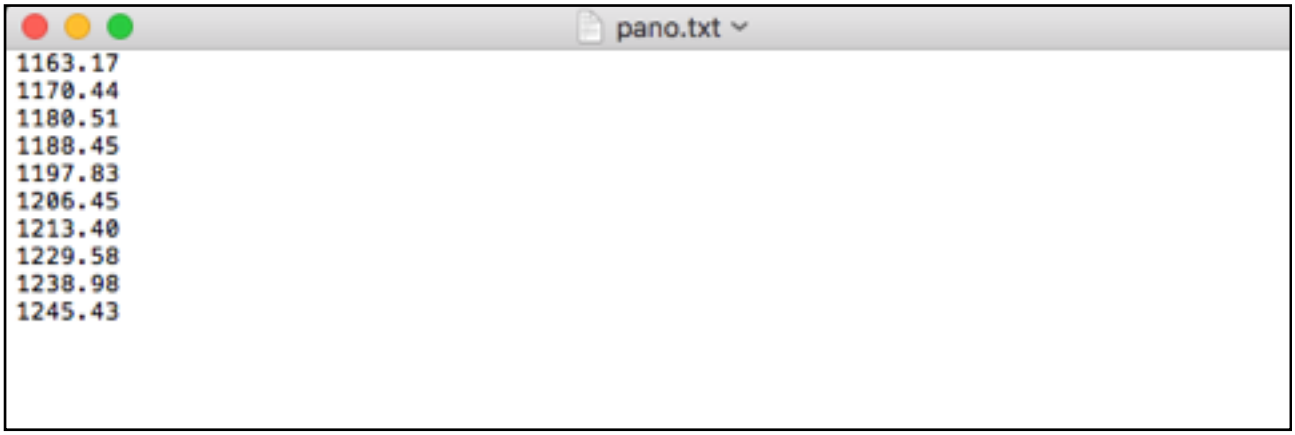
其他的相關成果



parrington, harris, feather

parrington, sift, average

pano.txt格式


pano.txt
1163.17
1170.44
1180.51
1188.45
1197.83
1206.45
1213.40
1229.58
1238.98
1245.43