[VFX HW#2 : Image Stiching]

R05922124 黃雅博
R05922101 楊騏瑄

Part 1:cylindral projection

對每張照片原本的 xy 座標採用

x’ = x – x_center

y’ = -y + y_center

把各點換成以中心為原點且 y 軸上下顛倒的坐標系表示，接著進行圓柱投影

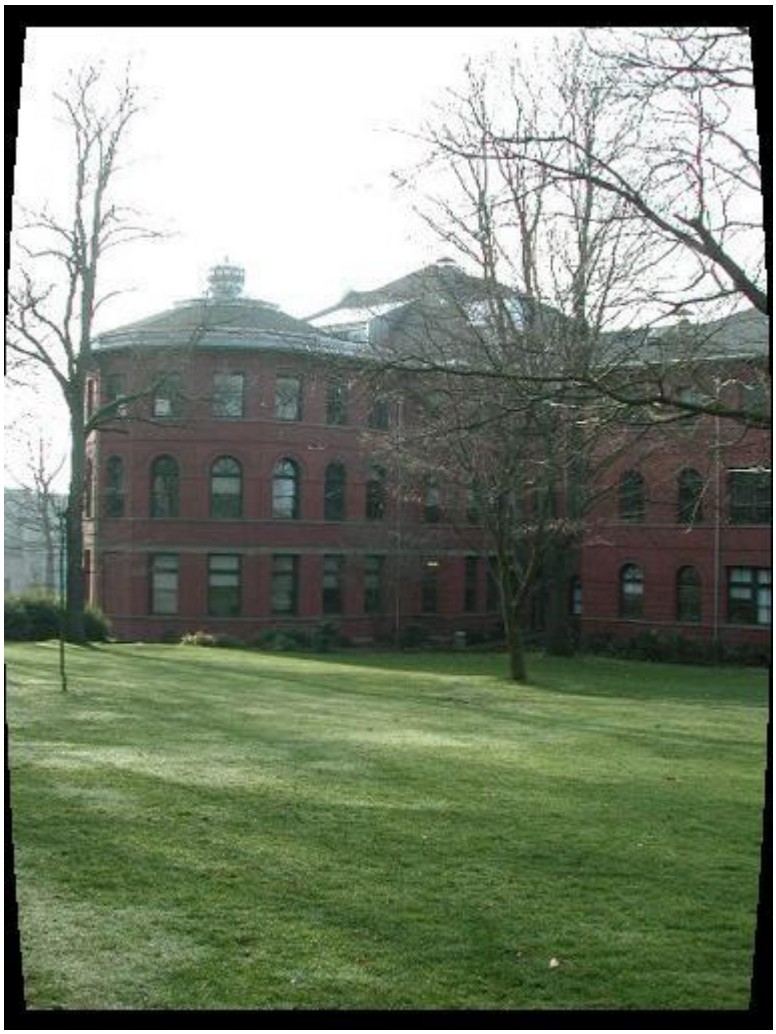
x’’ = f * arctan(x’ / f)

y’’ = f * y’ / sqrt(x’ * x’ + y’ * y’)

最後再做一次把原點移到左上角 y 軸顛倒的變換得到投影後的照片

x’’’ = x’’ + x_center

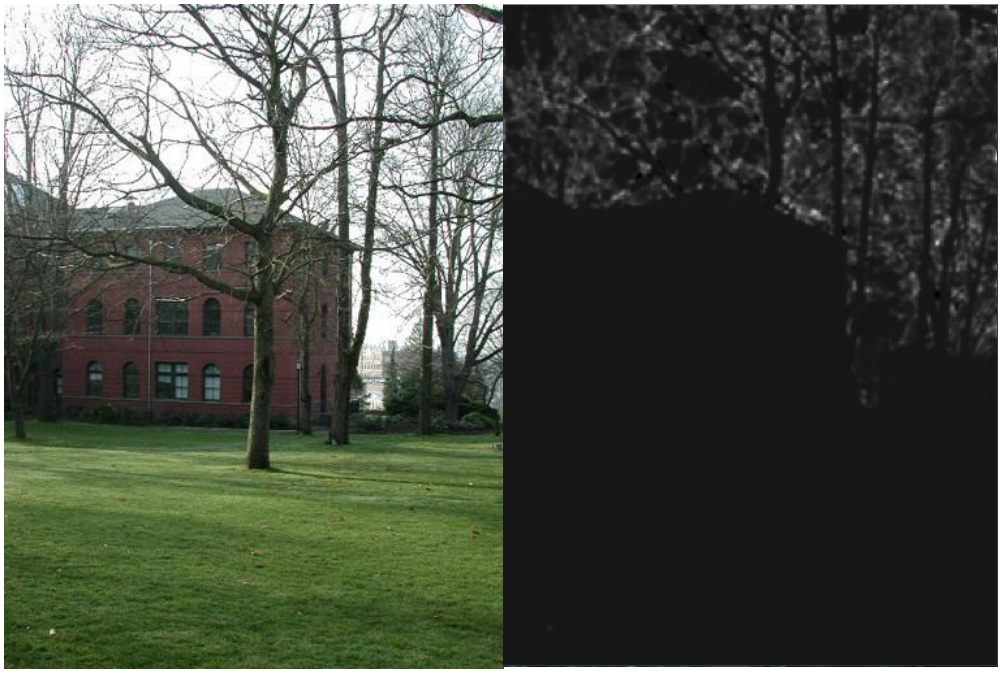
y’’’ = -y’’’ + y_center



(經過投影完的照片)

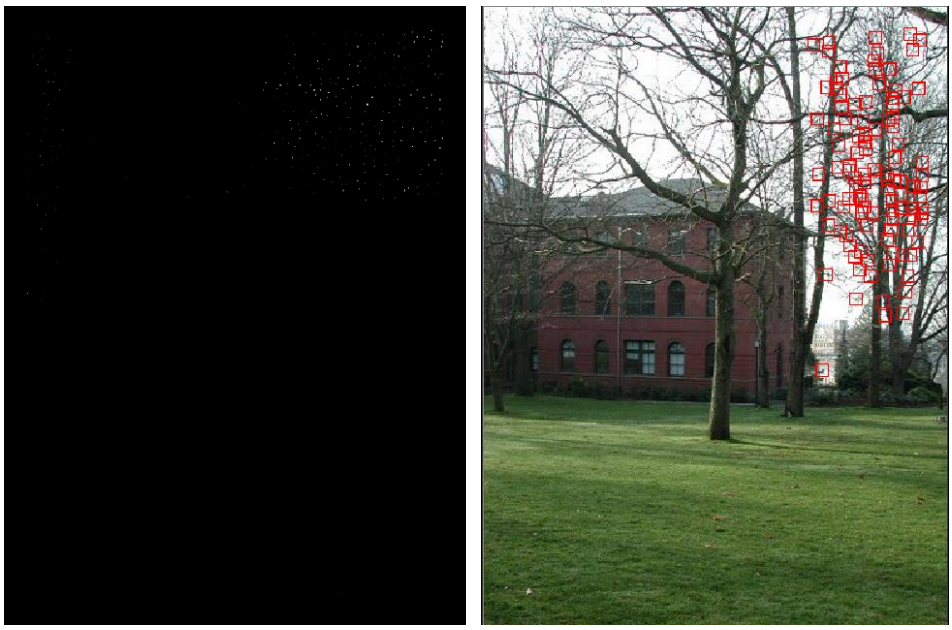
Part 2 : features matching

Feature detection 使用上課提過的 **Harris Corner Detector**. 利用 numpy 與 scipy 做 gradient 與 gaussian filter, 求得 Response. (**getResponse**)



(左邊原圖, 右邊 response map)

再將 response 邊邊 20pixel 之 response 設為 0, 避免太邊緣的 feature.



(nms 後的白色點點, 之後會轉為 feature point(紅色區域))

non-maximum suppression 利用 scipy 的 maximun filter 搭配 threshold 來實作.

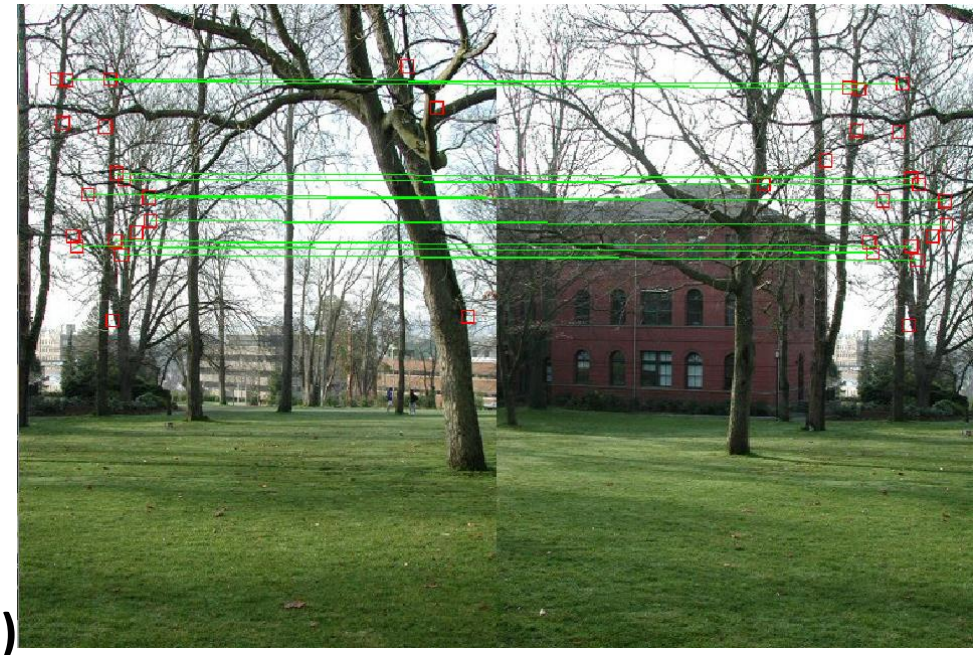
最後將 feature 包好(position & response), 再進入下一階段.

(nms 階段能控制最大 feature 之數量)

Feature description 參考上課提過的 **SIFT Descriptor** 的其中一部份 (**harris_descriptor**), 首先先對 intensity map 取 gaussian, 再取 feature point 周圍 **35*35**(自己設的不是原本的 40*40) pixel 之 intensity, 做 normailize 並縮小至 0.2 倍後, 可以得到 7*7=**49** 個 element 的 descriptor. (由於之前 clip 過 response 所以不會遇到取出圖片範圍的情況).

原先有試過直接取周圍 7*7 之 pixel 做 descriptor, 但效果很差, 如下圖 1. Matched pair 數量很少. 而用上述大 window 方法, Matched pair 會多很多(下圖 2)

(1)



(2)

Feature match 利用 scipy 之 cdistfm 窮舉出兩圖(d1, d2) 每個 feature 兩兩之間之 distance, 將 distance 之矩陣照每列排列距離大小並存入該數值之 index 至 idx 矩陣. 取第一行所有 index 為最佳配對與次佳之 index, 利用 index 回去找最佳與次佳 match 之 distance. 利用下式求 score 來過濾較差之 matches (best 與 second 距離差異不大者).

```
# get score by best / second distance
score = best_dist / second_dist.mean()
# find the indices of the best Matches by threshold
d1_match = np.argwhere(score < 0.5)
d2_match = best_dist_idx[d1_match]
```

其中, 將次佳距離做平均以後能使兩兩圖片的 match 數差異變小, 數量也會較多. (match pair 數量相對比較平均).

有 mean 之 Best match num 之 list

[70, 61, 116, 88, 103, 75, 101, 94, 100, 81, 116, 87, 71, 108, 56, 50, 122]

沒有 mean 之 Best match num 之 list

[17, 45, 57, 107, 84, 86, 56, 91, 85, 92, 75, 110, 73, 72, 90, 46, 48, 108]

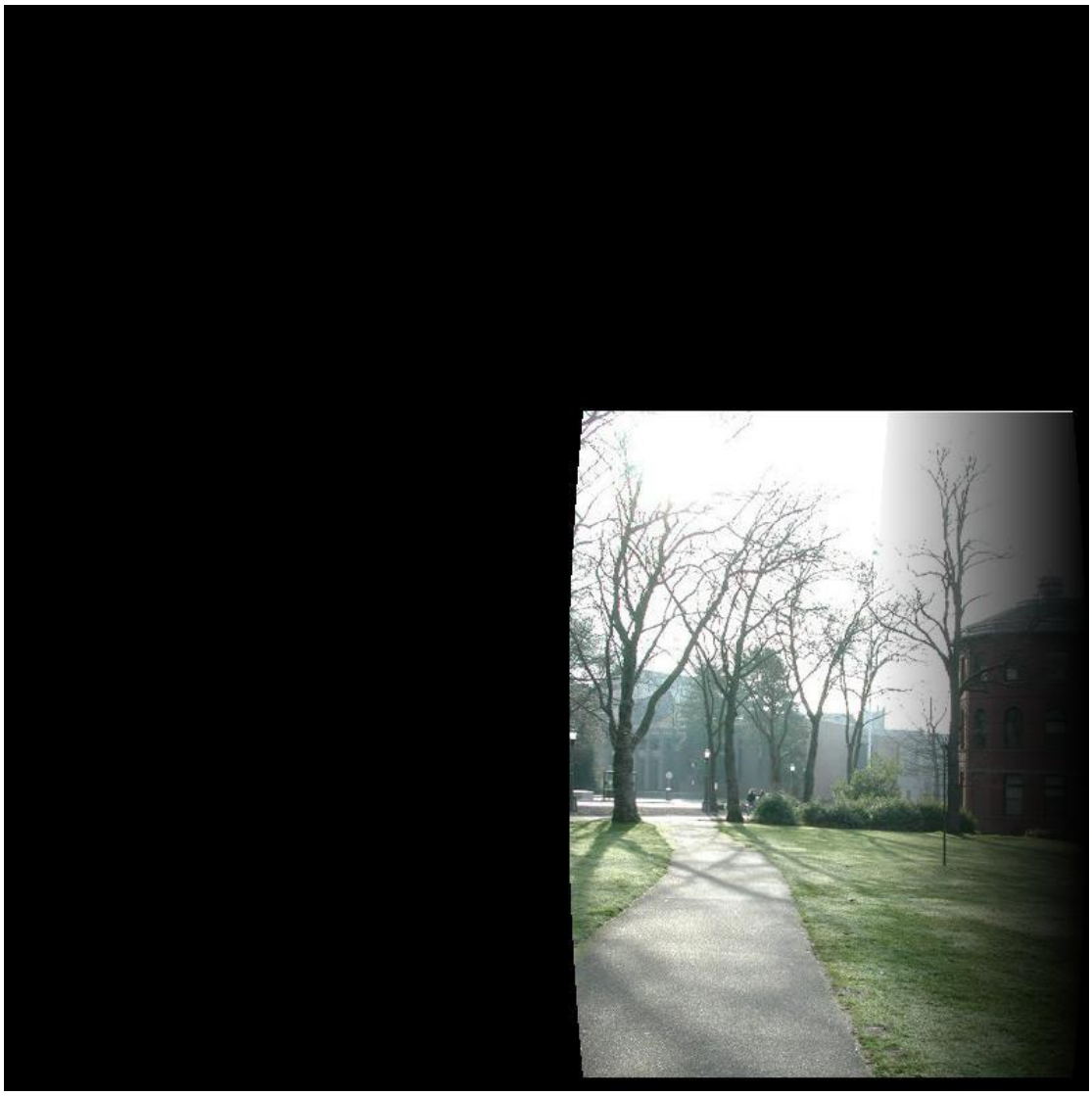
Part 3: RANSAC

一次 sample10 個'點'並假設真正 inlier 比例是 0.5，最後成功的機率 0.99

每次隨機抽到的'點'其實是 matched feature pair，可以算出 shift_x shift_y(假設在圓柱之後只有 translation)，接著平均那些所有點的 shift_x shift_y，對那些沒被抽到的人進行距離的測試，並記錄回傳最好的 shift_x shift_y，在有使用腳架的情況中 shift_y 只有幾個 pixel

Part 4:stitching and blending

利用 RANSAC 算出的資訊先開一張夠大的 panorama，接著依照輸入照片的序列以及對應的 translation 填入這張 panorama，填入的方法如下:
(1)如果自己該點是黑的就跳過
(2)如果 panorama 是黑的就填入
(3)如果兩個都不是黑的就依照 translation 做 linear interpolation 的 alpha blending



(blending 效果，右上角因為沒有與別張圖重疊所以符合(2)的情況)

結果:

(相機直立拍攝可以看到剛好在某張照片邊緣的方尖碑有扭曲的現象)

(相機橫放拍攝扭曲就較小)