

# HW2 Image Stitching

資工三 b06902016 林紹維 b06902066 蔡秉辰

p.s.1. 照片為 b06902001 陳義榮所拍攝的。

p.s.2. 我們有兩組圖片，除了 report 中最後的環景圖結果外，其餘均只顯示第一組的結果

## 程式執行方式與環境

### 執行方式

```
python hw2.py [-h] [--images, -i IMAGES] [--output, -o OUTPUT]
               [--reshape_ratio, -r RESHAPE_RATIO]
               [--focal_length, -f FOCAL_LENGTH]
               [--crop_pixels, -c CROP_PIXELS]
               [--show_points, -d] [--show_pairs, -p]
```

- -h, --help: 查看幫助
- --images, -i: 輸入圖片的資料夾
- --output, -o: 輸出環景圖的路徑，預設為 `full_img.png`
- --reshape\_ratio, -r: 圖片縮放比例，預設為 0.25
- --focal\_length, -f: 圖片焦距，預設為 1100
- --crop\_pixels, -c: 投到圓柱後圖片上下被減掉的高度，預設為 70
- --show\_points, -d: 輸出每張圖片的特徵點位置
- --show\_pairs, -p: 輸出相鄰兩張圖片 match 的點的位置

若要取得我們的結果，則可以直接執行以下：

```
python hw2.py -i pic_1/
```

### 環境

- numpy == 1.18.2
- opencv == 4.3.0

## 實行細節

### 概述

本次實作的過程大致如下：

- 將組圖用 cv2 讀入後進行 cylindrical projection
- 接著透過 Harris corner detector 和 non-maximal suppression 將特徵點抓出
- 用 MSOP 的方式對每個特徵點用 64 維的向量表示
- 針對不同的向量嘗試比對出最接近的 pair
- 最後透過 linear blending 將兩兩圖片 stitch 起來成全景圖

以下將詳細描述各部分的實作內容：

## Cylindrical projection

我們首先對輸入的圖片組做座標轉換，投影到圓柱體上後，再轉回平面。我們發現，用不同的 focal length 轉換出來的圖片，會對圖片產生不同程度的形變，而我們認為當 focal length 設定在 1100 的時候對該組圖片的轉換最適合。下方由左至右分別是當 k 值在 700, 900, 1100 的結果：



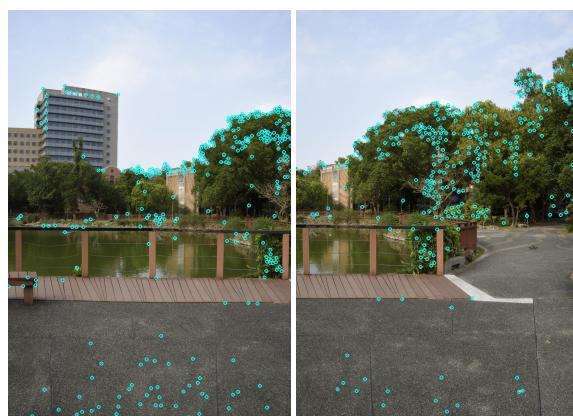
## Harris Corner Detector

因為這次圖片都是在固定焦距、縮放距離，並且相機只會繞著圓心水平移動。我們認為不需要太過於複雜的 detector，故在實作上採用最容易實作且效果也不錯的 Harris Corner Detector。

在計算每個點的 R 值步驟基本上都與上課投影片相同( $k = 0.04$ )，比較特別的是我們在選取那些點要當作特徵點時借用了 MSOP 中的 non-maximal suppression。實際選取方式如下：

1. 因為我們認為每張照片 R 值的 threshold 可能會不同，並且我們也沒有 domain knowledge 去選擇哪個 R 值較好，故我們是直接選 R 值前 3000 高的點。
2. 採用 non-maximal suppression 的方式選點，初始半徑  $r$  設為 10000，一個 while 迴圈中若被選到的點不到 250 個，則會將半徑減半並在跑一次 while 迴圈
3. 和 non-maximal suppression 有點不同的是，我們並非一選到 250 個點後就終止 while 迴圈，我會將在該半徑下符合的剩下點也都選取完。也就是說每張圖片選到的點數並不相同，但大致都會落在 250 ~ 350 這個範圍之間。

其中兩張相連的圖片的 feature point 如下：



可以看出效果還算不錯。

## MSOP feature descriptor

從上步得到的特徵點，我們首先會將各個點  $8 \times 8$  範圍內的 pixel 做出 orientation 的投票，並根據高斯分布計算不同位置的權重，得到一個表示該點方向的 gradient，接著到圖片中取下該點範圍  $80 \times 80$  的 patch，並將其作出對應方向的旋轉，使得待會取得 vector 時會是 orientation invariant。旋轉後的  $80 \times 80$  patch，我們將取中間的  $40 \times 40$  的 patch，並透過 resize 的方式讓表示該點的向量剩下 64 維度。

以下是一個特徵點的選取過程：

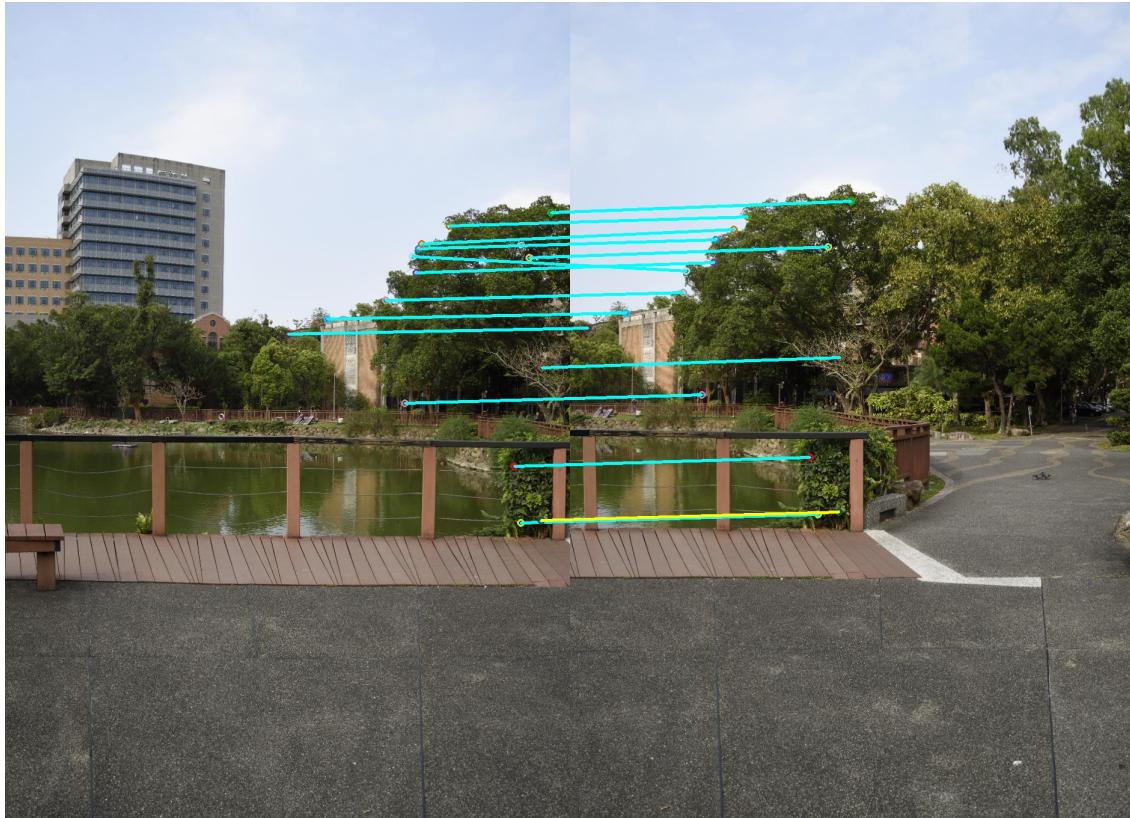


由左至右分別為  $80 \times 80$  patch, 旋轉後的  $80 \times 80$  patch, 取出的  $40 \times 40$  patch

## Feature mapping

對於相連的兩張圖，我們採用暴搜的方式計算對於左圖每個特徵點，右圖的哪個特徵點的 feature vector 和其最相近。我們採用  $l_2$  距離平方當作評斷基準(以  $d^2$  表示)。為減少誤差，在右圖中的特徵點我們只考慮在和左圖該特徵點垂直方向距離相差小於 50 個 pixel，且水平位置在左圖特徵點之左邊的點。

接著在所有 pair 中，我們只取  $d^2$  小於 20 的 pair。我們希望至少會選到 5 組 pair 來做底下的步驟，但因為可能  $d^2$  小於 20 的 pair 會少於 5 組。此時我們會逐步加大 threshold，直到有選到 5 組以上的 pair 為止。以下為前述 detector 的兩張圖片做 mapping 的結果：



每個線條均為 mapping 的 pair，而黃色的線條為最終選定的主要位移方向(下面會有詳細說明)。而至於採用不同的  $d^2$  threshold 造成結果的差異，會在本章結尾的補充實驗做完整的展示。

## Blending and Stitching

因為我們的 mapping pair 數量並不會太多(<30)，所以比起隨機去選某個 pair 當作主要位移方向，我們不如遍歷每個 pair，使其當作主要位移方向，並計算其它 matching pair 的  $l_2$  距離誤差，取最小的當作我們主要位移方向。

而在取完每兩張之間所需的位移方向後，我們即可將所有圖片黏成一張環景圖。黏合方面我們採用的是 linear blending。並在所有圖片都黏和完後，我們發現其會有往下的趨勢。故我們也有做投影片上提到的 drifting，並最後再將底下黑色色塊裁減掉。兩組圖片的結果如下：



可以看出雖然有些地方有一點糊，沒有到黏合得很完美，但整體而言環景圖的效果還算是不錯。

## 補充實驗

### 不同 $d^2$ threshold 起始值之差別

調整初始  $d^2$  threshold，並看最終輸出圖片之結果的差異：

- $d^2$  threshold = 25



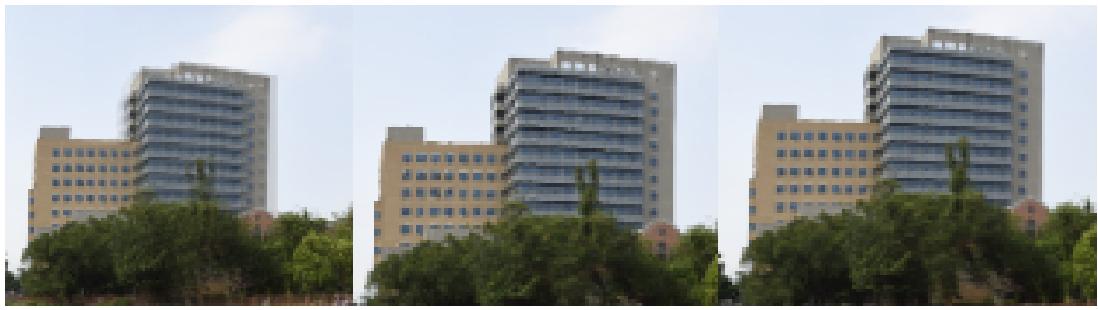
- $d^2$  threshold = 20



- $d^2$  threshold = 10



- threshold 依序為 25, 20, 10 的細部圖：



可以看出在 threshold 為 25 和 20 時有較明顯之差異(e.g. 天數館在 threshold 為 25 時較為模糊) · 但在 threshold 為 20 和 10 之差異就沒有這麼大。我們認為是因為 threshold 設為 25 過大 · 導致會有較多錯誤或不好的 matching pair · 進而導致結果變差。

## 參考資料

---

- Cylindrical projection <https://gist.github.com/royshil/0b21e8e7c6c1f46a16db66c384742b2b>