Digi VFX Assignment II

Team 35

R11922189 邱議禾 R11921117 王鈞

一、程式使用方式

在 code 處直接執行 main.py

二、拍照

我們選擇在醉月湖畔，使用相機並且架設腳架，透過旋轉腳架拍攝不同角度的場景照片，由於當天遊客眾多，為了在影像中去除走動的遊客，我們同一個角度的場景會拍攝 5 張以上影像，再透過算圖片每個 pixel 的中位數，得到沒有行人走動的影像，最後再拼接這些影像。

原始影像：圖中有行人

經過處理後的影像

三、Warp to cylindrical coordinate

與講義中公式(下圖)相同，我們建立了一張計算出大小的結果圖

$$x' = s\tan^{-1}\frac{x}{f}$$

$$y' = s\frac{y}{\sqrt{x^2+f^2}}$$

因為不想要有旁邊的黑邊影響，因此我們是採用反函數的做法

對於新圖片的每個 pixel: (x', y')，利用

$$\mathrm{x} = \tan(\mathrm{x}'/\mathrm{s}) * \mathrm{f}$$

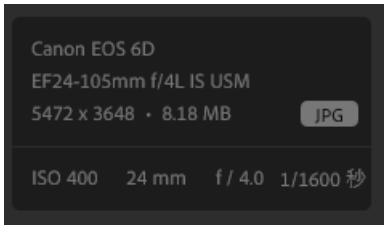
$$\mathrm{y} = \mathrm{y}'/\mathrm{s} * \sqrt{\mathrm{x}^2 + \mathrm{f}^2}$$

求出新的圖像，其中 s = f

f 通過公式

$$focal\ length\ (in\ pixel) = \frac{max(height,\ width) \times focal\ length\ (in\ mm)}{sensor\ size\ (in\ mm)}$$

我們所用的相機是 Canon EOS 6D，感光元件是 36 x 24 mm，焦距是 24mm，因此推算出 focal length in pixel 是 3648



四、Feature detection

（一）Harris Corner Detector

我們順著七個步驟完成 Harris Corner Detector 去實現 feature detector

1. 計算水平垂直向量 ix, iy，這部分我們用 Sobel 去實現
2. 計算 ixx, ixy, iyy
3. 對 ixx, ixy, iyy 做 Gaussian Blur，得到模糊後的 sxx, sxy, syy
4. 建矩陣 $m = \begin{bmatrix} sxx & sxy \\ sxy & syy \end{bmatrix}$
5. 求矩陣 m 的 det 和 trace，用 $R = \det(m) - k * (\mathrm{trace}(m))^2$ 得到 corner 響應矩陣 r

6. 對 r 取 threshold，其中為了提升成效我們用了 OTSU(大津)二值化
7. 對得到的結果取 non-maximum suppression

完成 feature detect 以後，會接續 SIFT 後面的 orientation assignment 完成 feature descriptor

（二）SIFT

我們通過以下步驟完成 SIFT 演算法：

1. Scale-space extrema detection: 根據圖片大小決定 octave 的層數，由於我們有做 down-sampling，根據 SIFT 的原始論文提到，k 的值之所以設定為$2^{1/s}$，是為了使 $k^s = 2$，代表通過 s 次模糊化以後，結果會與 resize 成一半解析度一樣的模糊程度，其中 s 參考原始論文設為 3，原始 sigma 設為 1.6
   模糊化以後通過同一個 octave 層級的圖片相減得到高斯差分圖(DoG)
2. keypoint localization: 根據 DoG 圖上的 extrema point，反推在原始解析度下正確的 x, y 座標，這部分根據講義我們利用 gradient 與 Hessian 去計算，其算法參考於這篇文章《Implementing SIFT in Python: A Complete Guide (Part 1)》
3. Orientation assignment: 找出 keypoint 以後，我們用 Sobel 求出水平與垂直梯度，並計算出極座標(m , Θ)並用矩陣儲存。接著用 Gaussian 對 keypoint 周邊做 Gaussian 並加權於 m，我們把Θ的單位轉換為角度，並直接除 20 即可放進 20 個 bins，之所以求 20 是我們發現得出來的效果較好。求出來的最高票與第二高票我們用 0.8 的比值作為 threshold，若高於 0.8 的話則拆分為兩個特徵點
4. Local image descriptor: 與 Orientation assignment 相同，我們同樣求出水平與垂直梯度，以及極座標矩陣，接著我們對每個 keypoint 周邊的 16*16 格進行特徵點的描述。對於離邊界不到 16 的 keypoint，我們則是直接將 window 的中心拉到 16 的位置。
   接著同樣利用投票的方式，將 16*16 切成 16 個 4*4，然後把每一個小塊的投票結果存進 bin 中，再存進 4*4 的投票結果統計 area 中，最後我們用 flatten 把 area 攤平成 1*128 的向量作為特徵向量。

五、Feature matching

（一）Exhaustive search

我們針對儲存的 description 對 image 兩兩比對，我們用的比對方式是用 cosine_similarity，其中公式如下

$$cosine\_similarity = {kp1 \cdot kp2}/{norm(kp1)norm(kp2)}$$

並找出對於每個 kp1 而言，相似度最高的 kp2，其中 kp1 與 kp2 表示來自不同 image 的兩個 keypoint，另外如果 kp2 第二相似的結果跟最大的結果的比值大於 0.9 的話，直接放棄不計，視為未成功辨識。

（二）knn

knn 我們使用 knnMatch 進行取樣

兩個辨識方式都將 matching 的 keypoint 記錄起來，以使用 RANSAC，image matching 的結果也記錄至 similarity matrix，讓我們知道照片的順序應該要怎麼排

六、Image matching

（一）RNASAC

由於我們已經將所有影像投影在圓柱座標上，因此我們假設場景與場景之間拼接的變量只有「x 軸位移」m_1 和「y 軸位移」m_2，因此我們假設我們的 Energy Function 為

$$E = \sum_{i=1}^{n} [(m_1 + x_i - x_i')^2 + (m_2 + y_i - y_i')^2]$$

優化函數為

$$0 = \frac{\partial E}{\partial m_1}$$

可以推導出 close for solution

$$m_1 = \frac{1}{n} \sum_{i=1}^{n} (x_i' - x_i)$$

$$m_2 = \frac{1}{n}\sum_{i=1}^{n}(y_i' - y_i)$$

不過我們無法保證 feature matching 給的所有結果都是 100%正確，因此我們在優化尋找 solution 時引入 RANSAC 演算法，算法流程如下

Run k times:

(1) draw n samples randomly
(2) fit parameters 5θ with these n samples
(3) for each of other N-n points, calculate its distance to the fitted model, count the number of inlier points c

Output θ with the largest c

演算法中 k 代表優化次數，n 代表取多少點配對作為運算 θ 的基準，θ 則是兩張影像拼接的關係「x 軸位移」m_1和「y 軸位移」m_2，我們另外透過機率估計來決定 k 與 n，估計函數如下

$$P = 1 - (1 - p^n)^k$$

P 是我們希望達到的信心水準，p 是 random sample 是有用的 sample 的機率，我們可以進一步推導 k 的 close form solution

$$k = \frac{\log(1 - P)}{\log(1 - p^n)}$$

我們假設 n=6，P=0.99，p=0.2。

七、Blending

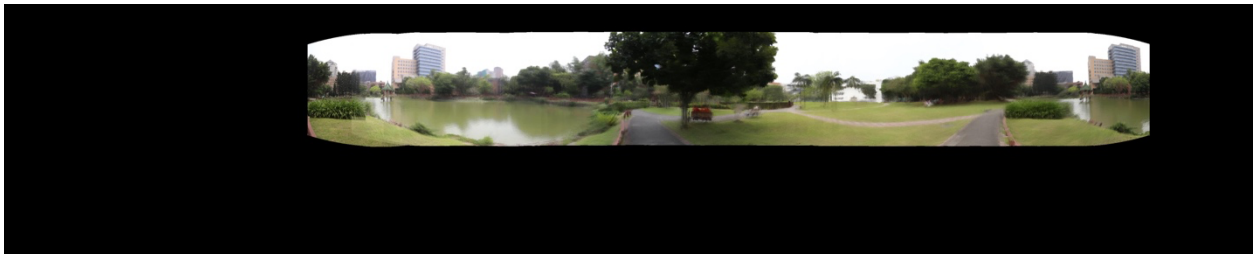
（一）Median Blending

當我們得到多張影像之間的位移變化，我們可以將多張影像疊在大的畫布上，重疊區域的 pixel 我們計算同個座標上所有數值的中位數，效果如下圖

（一）Gradient Domain Blending

八、End-To-End Alignment

我們發現最後拼接結果最左側和最右側之間會有垂直方向的位移差，因此我們去重新計算所有 pixel 的座標，將圖片映射到沒有位移差的新座標上，結果如下圖



最後將不規則區域裁切，得到最終結果

九、結論

我們使用 Harris corner detection 找出特徵點，再透過 SIFT descriptor 的 orientation assignment 和 local image descriptor 描述特徵點，近一步使用 cosine similarity 比對特徵點，得到 matching feature pair。

找出特徵點後，我們用 RNASAC 對收到的 matching feature pair 進行比對並接上圖片，使用 median blending 同時處理 end-to-end alignment ，最終裁切掉少數的垂直位移以後完成 stiching。

十、心得與收穫

本次的作業我們用了 SIFT，其中 SIFT 有許多參數上的選擇都是經過實驗，而非依據理論，這樣我們在發生錯誤的時候，會不自覺地去懷疑參數並進行調整，而且其中有一部份是。

我們在圓柱投影的時候，由於實作方式不同，因此採用了不會有留黑的作法，用反函數的概念去 fetch 原始圖片的 pixel，這也讓我們不知道如果發生殘影究竟是什麼原因。

另外在寫 SIFT descriptor 的時候，當我們完成了程式碼，但結果似乎不太好，而我們很難對 128 維的 descriptor 進行 debug，最終幾乎是重寫一份才完成了這份作業。

原先我們也有做 SIFT 的 feature detection，但是我們在 keypoint 的 localization 上出了問題，我們認為是在 quadratic fitting 的時候可能有 bug 才導致這樣的情形發生，但同樣的因為我們直接使用矩陣算法，導致我們也不知道問題的原因。

整體來說這次作業比上次還要複雜，SIFT 不是一個很好實作的演算法，裡面關於極座標的計算、角度的投票都很容易出現 bug，反而前面的 DoG 還比較好算。

十一、器材

相機 EOS 6D

focus 4.3，iOS 隨相片角度使 ISO 經自動校正後略有不同，大致都在 400 上下

使用相機腳架，並用手機操控透過 wifi 遠端連線啟動快門以避免晃動