Digi VFX Assignment II

Team 35

R11922189 邱議禾 R11921117 王鈞

1. 程式使用方式

在code處直接執行main.py

1. 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



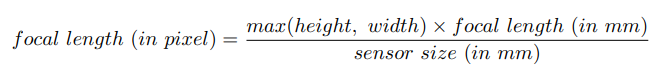
1. Warp to cylindrical coordinate

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

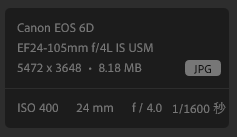
因為不想要有旁邊的黑邊影響，因此我們是採用反函數的做法

對於新圖片的每個pixel: ，利用

求出新的圖像，其中s = f

f通過公式

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
5. 求矩陣m的det和trace，用R = det(m) – k \* (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，是為了使ks = 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)

1. 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)》](https://medium.com/@russmislam/implementing-sift-in-python-a-complete-guide-part-1-306a99b50aa5)
2. 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
3. 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

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」m1和「y軸位移」m2，因此我們假設我們的Energy Function 為

優化函數為

可以推導出 close for solution

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 *θ* 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」m1和「y軸位移」m2，我們另外透過機率估計來決定*k*與*n*，估計函數如下

*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

由於我們只有水平和垂直方向位移，只有兩個變量，因此我們假設n=3，P=0.99，p=0.5。

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配合SIFT descriptor，找出了特徵點，但其實我們忽略了SIFT中的orientation assignment，原因是我們的相片都沒有rotation與scale，因此我們才採用Harris corner detection與SIFT的local image descriptor。

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，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