

VFX HW2

**Image Stitching 實作 : feature detection /matching,
image matching, bundle adjustment and blending**

R05922033 廖苡彤、R05944045 陳卓晗

I. Brief Introduction:

我們在生活中拍攝照片時，往往會受到鏡頭的限制，而無法將各個視角的畫面都一收眼底。然而，我們可以藉由一些影像處理的方式來疊合多張小視角的照片，重現身歷其境的視覺效果。

下圖為相機拍到的相片(FOV: $50 \times 35^\circ$)，視角受到鏡頭的限制：



下圖為人再不轉動頭部的前提下，大致能見到的景像(FOV: $200 \times 135^\circ$)：



下圖則是180度的panorama(FOV: $360 \times 180^\circ$)：



這次作業就是要實作image stitching：將不同視角拍得的照片，透過影像縫合的方式接成一張全景(panorama)的影像。

II. Steps for implementing image stitching:

Image stitching主要可以分成兩個部分：alignment 和 blending，前者主要focus在geometric registration，後者則是focus在photometric registration。

以下為image stitching的主要步驟：

(geometric registration)

step1. 計算拍攝好的多張不同視角照片的focal length，將其投影到圓柱上。

step2. 找出每張照片的feature points。

step3. 將不同視角的照片兩兩進行feature points的matching。

step4. 利用matched feature points來進行影像的疊合。

(photometric registration)

step5. 主要是使用poisson blending來使影像的接合更smooth一些。

III. Algorithm:

1. Cylindrical Projection:

2. Feature Detection:

Features又被稱作interest points或salient points，指的是在一張image中具有代表性的那些點，而我們可以很輕易地藉由local information，判斷出一張照片裡的一個feature point和另外一張照片裡的一個feature point是同一個點。

這部分我們是實作Harris corner detector，因為老師課堂中提到做panorama並不需要使用到SIFT那麼複雜的演算法，而我們在固定相機拍攝的前提下，並不會使特徵點在scale下有變化，因此也不需要使用到multi-scale的harris corner detector。

Feature detection的演算法如下：

- 算出x和y方向上的derivatives得到 I_x 與 I_y :
- 計算 $I_{x2} = I_x * I_x$, $I_{y2} = I_y * I_y$, $I_{xy} = I_x * I_y$
- 對 I_{x2} , I_{y2} , I_{xy} 做 Gaussian Blur得到各個pixel的和 S_{x2} , S_{y2} , S_{xy}
- 定義 $M = [[S_{x2}, S_{xy}], [S_{xy}, S_{y2}]]$
- $R = \det(M) - k(\text{trace}(M))^2$ ，為其corner detection function。
 - $\det(M) = S_{x2} * S_{y2} - S_{xy} * S_{xy}$
 - $\text{trace}(M) = S_{x2} + S_{y2}$
- R 越大的地方越可能是Feature Point
- 定義Threshold，取 $R > \text{Threshold}$ 的點為Feature Point

值得注意的是，我們比較了兩個決定如何從那麼多>Threshold的feature point裡選取我們想要的數量的feature point的方式：

(1) **Strongest Feature Point** (HarrisTop.m，最後一個參數設為0)

- 根據R值大小將所有的Feature Point進行descend的排序，從R值大的feature point開始選取，直到取完想要的數量的feature point
- 可以藉由定義localRadius，決定多大半徑的圓裡只會有一個Feature Point，來避免搜尋到的Feature Point太靠近

(2) **Non Maximum Suppression** (HarrisTop.m，最後一個參數設為1)

這個演算法是為了避免feature point都集中在特定區域，主要的做法是先建一個空的cornerList，把R值最大的點放入cornerList，再把距離和cornerList中的點大於一個指定半徑(一開始是令其等於 $\min(\text{Img_長}, \text{Img_寬})$)的點放入cornerList中，再縮小半徑。重複以上步驟，直到cornerList中的點和我們要求的feature point數量一樣的時候停止。

Result:

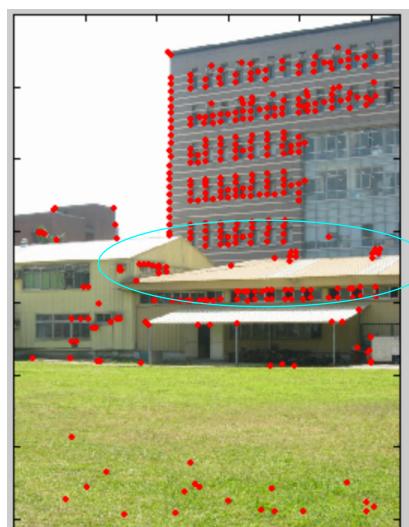
原圖



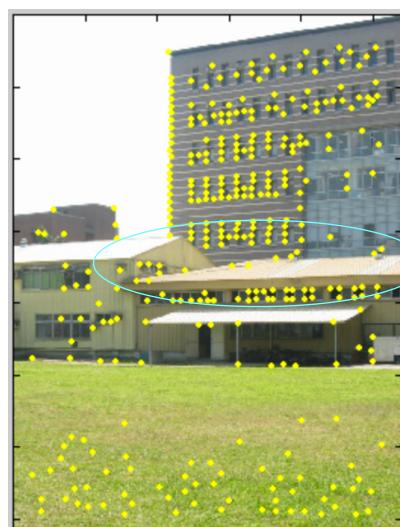
每個pixel 的R值作圖



Strongest Feature Point



經過NMS後



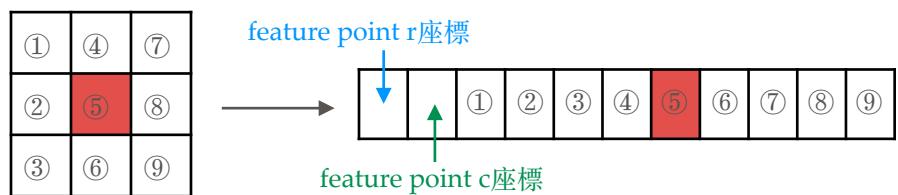
(比較圈圈內的feature point，可以發現NMS後的feature point比較不會過於集中)

3. Feature Descriptor:

在找完feature point的位置後，我們需要為每一個feature point derive “feature description”，這些feature description可以說是feature point的unique特色，讓電腦得以根據這些feature description，決定不同張圖裡的兩個feature point到底是不是同一個點。

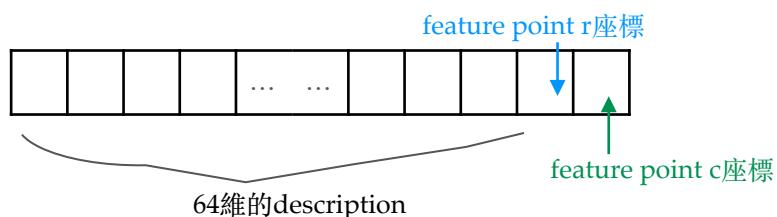
我們這邊實做了兩種feature descriptor:

- (1) 第一種是最簡單的feature descriptor，將該feature point周圍(包含自己)的intensity值作為這個feature point的descriptor (*simpleDescriptor.m*)，如下圖(紅色方框為feature point所在pixel)：



- (2) 第二種是**MSOP feature descriptor**，也就是將Bias / gain normalized sampling of local patch 做為我們的feature descriptor (*FeatureDescription.m*)，具體作法如下：

- 根據feature detection 所找到的feature points的位置，以每個feature point為中心，框出一個40x40的patch
- 對每個40x40的patch做Gaussian blur，即右式： $P_l(x, y) * g_{2 \times \sigma_p}(x, y)$
- 取得8x8 oriented patch sampled at 5 x scale
- 對pixel 的亮度做Bias / gain normalization : $I' = (I - \mu) / \sigma$
 - 使其mean 為0，variance為1。
 - subtract mean可以使其具有 shift invariance的特性；而divided by sigma則可以使其具有 scale invariance的特性。
- 經由上面的作法可以取得64維的feature description
- 另外，為了方便取得每個feature description對應到的feature point，我將feature point的位置存在feature descriptor的第65和66維



4. Feature Matching:

計算好每個feature point的descriptor後，我們便可以根據descriptor將不同image的feature point進行matching。

這個部分，我們實作了Exhaustive Knn Search及Kdtree的Knn Search：

(1) Exhaustive Knn Search:

主要是計算兩張input image 的所有feature points的descriptor間的L2 distance，將這些distance進行ascending sort，取得距離最小的前幾名作為我們的matched feature points。

不過這樣的做法非常慢，複雜度為 $O(n^2)$ ，另外也會有match錯的情形。

(2) Kdtree的Knn Search:

主要是利用matlab 內建的kd-tree function，分別計算“第一張image”的所有feature point在“第二張image”裡的2 nearest neighbors，以及“第二張image”的所有feature point在“第一張image”裡的2 nearest neighbors，考慮到第二近的feature point和第一近的feature point的距離要大於“第一近的feature point的距離乘上一個threshold”，再將match到的feature point在兩張image裡的位置存起來，格式如下：

row in Img1	col in Img1	row in Img2	col in Img2
...
...
...

match 到的feature point 在第一張 image 的座標位置

match 到的feature point 在第二張 image 的座標位置

此方法的計算速度較快，而且也較精準。

Result:



紅色方框內為第一張 image，沒框起來的部分為第二張image

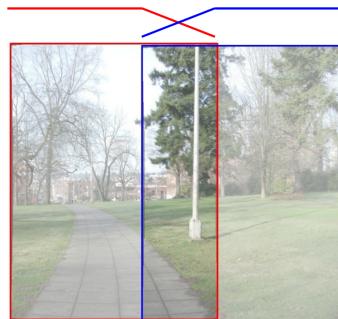
5. RANSAC/Image Stitching:

6. Blending:

兩張image在得到對齊的關係後，因為有些區域會互相重疊，所以還需要透過blending的方式才能夠生成完整的影像，我們在這邊實作了兩種方式：

(1) linear Blending(weightBlend.m) :

透過影像邊界遞減的方式來互相疊加，也就是交界區域越靠近誰，誰的影像像素乘以的權重就越大，示意圖如右：



我們的實作方式是為重疊區域建立兩個linear mask，在組合圖的時候，兩張照片重疊的區域分別乘上這個mask，便可以得到blending的結果。

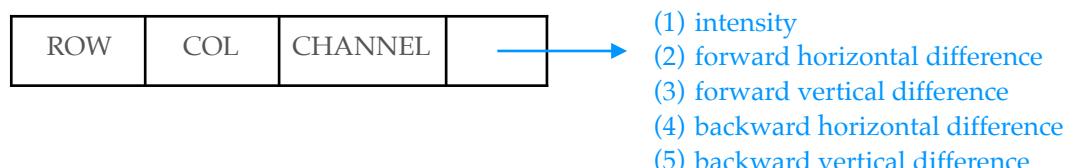
但是，linear blending有一個很大缺點，那就是其重疊區的中間會因為兩張影像的權重差不多，所以，如果兩張image對齊的結果不是那麼完美的話，可能會有鬼影產生。

(2) Poisson Blending (Gradient Domain Blending)(poissonBlend.m): **BONUS PART**

這個方法是源自於比較常見的應用：poisson editing，其目的是要將一小塊目標圖貼到一個底圖上，並藉由調整目標圖使得其梯度和目標圖原來接近但邊界部分則保留底圖的亮度。

我們這邊實作的方式如下：

- 先取得每張input image 的水平與垂直的forward gradient 與 backward gradient，和原圖的intensity一起存成一個4維的tensor : gradFeat



- 接著，根據下一張image對前一張image的位移(offset)，我們將所有gradFeat的四維資訊貼上，重疊時皆採取右邊的info覆蓋左邊的info的方式，如右圖：



- 接下來，我們利用兩種方式解poisson equation：

- ① 利用網路上提供的matlab toolkit，使用fast fourier transform的方式解poisson equation，得到blending的結果
- ② 利用Gauss-Seidel Method的方式遞迴地逐步更新重疊部分的pixel值，即遞迴解poisson equation，使得貼上去的圖的gradient和原圖的gradient愈接近愈好，來達到blending的效果

實作下來，我們發現直接使用toolkit在速度和效果上，都比遞迴的去寫poisson equation好。

result:



linear blending



poisson blending

比較上面這兩張blending offset（也就是對於前一張圖的位移）相同的image，可以發現若是使用linearblending且沒有匹配好，便會有鬼影的現象產生；而使用poissonblending則不會有這個問題(可以比較兩張圖的紅色方框內的影像)。

另外，值得注意的是，poisson blending的過程中是使用double 的型態在進行運算，所以poisson blending的input image必須要是double型態，否則會造成運算上的錯誤，產生的圖會如下：



7. Drift:

BONUS PART

8. Recognizing panoramas:

這個部分是實作2003年ICCV的paper <Recognising Panoramas>，主要實作目標為：不按照順序丟image進去我們的程式，程式會自動比對，找出image相接的順序，並以正確的順序進行stitching。

IV. Result:

V. What we have learned: