Image Put-Back

104064518 張容瑜 104064552 蔡博丞

COMPUTER VISION FINAL PROJECT

105.1.18

104064518 張容瑜 104064552 蔡博丞

一、前言(Propose)

由於現代生活中旅行的盛行以及相機的普及，大眾經常在旅遊各景地之餘，隨手拍下看到的景色，最終上傳至個人電腦中做為收藏。而當使用者疏於分類，而導致電腦中的資料夾充滿了雜亂的圖片，此時打開資料夾只會看到凌亂的各景色混雜在一起，就算想要分類整理，也難從單一的局部景色照片回憶起屬於哪種分項。我們此次的期末專題即是想解決這樣的困擾，希望設計出一種由電腦操作的方法，達成電腦自動分類不同場景、且拼接成全景，以方便辨識及收納的目的。

二、流程概述(Work Flow)

（流程圖如上）

一開始將一個雜亂的照片資料夾傳入，經過分類處理器處理後，將一組組的資料分項傳入排序流程（即為拼接圖像時，判斷圖像目前要被拼接的位置。），再將影像拼接，及匯出。本次專題主要撰寫的程式內容為分類(Classification)以及排序(sorting)的程式碼，其它使用到 SIFT 以及 Image Stitching 的程式碼是參考網路上既有的資源加以改寫。

三、分類(Classification)

設計分類時，我們主要使用到 SIFT（Scale-invariant feature transform）的方法來找出不同圖片的特徵點，再根據兩張不同圖片對應相同特徵點的地方，計算出有多少對應符合的 matches，最終再用 RANSAC（Random sample consensus）算出

這些 matches 中有多少比例是 inliers，根據一個固定比例，判斷出兩張圖片是否是同一景色（意指判斷是否一起匯出拼接），以下會以程式碼輔助的方式，說明我們如何設計出分類的方法。

```
%% Load a list of images

imgList = dir('./data/*.jpg');

IMAGES = cell(1, length(imgList));
for i = 1 : length(imgList),
   IMAGES{i} = imread(['./data/' imgList(i).name]);
end
```

```
%% classification

S=size(imgList);
S=S(1);% 就是到底有幾張圖片
num_matches=zeros(S,S);
num_inlier(a,b)=zeros(S,S);
```

首先將圖片資料夾傳入，以及判斷出資料夾一共有多少張圖片待分類(Size=S)。這裡先創造出兩個 S * S 的矩陣來存取等等算出的 matches 以及 inlier 的數量。

```
for a=1:S

  for b=1:S
    if (a<=b)

      [numMatches, inlier] = sift_mosaic(IMAGES{a}, IMAGES{b});

      num_matches(a,b)=numMatches;

      num_inlier(a,b)=inlier;
    end

  end

end

num_matches=num_matches'+num_matches;
num_inlier=num_inlier'+num_inlier;
```

接下來用雙重迴圈，跑遍資料夾中的任兩張圖片，這裡使用 Sift_mosaic 這個 function 計算出任兩張圖片對應的 matches 及 inliner 的數量（Sift_mosaic 函式會在最後說明）。由於 num_matches 以及 num_inlier 這兩個矩陣必定是對稱矩陣，所以為了減少計算量，去除掉重複計算的情況得以加速，只需要計算出上三角矩陣及可，

再做一些簡單計算就可以得到一對圖片（i，j），它的 mathes 和 inlier 個別儲存在 num_matches（i，j)及 num_inlier（i，j）中。

```
compute=zeros(S,S); % computer用來存取哪些東西會相連

for i=1:S
  for j=1:S

    if (i<=j)

      compute(i,j)=num_inlier(i,j)/num_matches(i,j);

      if (compute(i,j)>=0.6&&num_matches(i,j)>60)

        compute(i,j)=1;

      else
        compute(i,j)=0;
      end

    end
  end
end
```

接下來設計一個Computer的S*S矩陣，其內儲存的數值就是對所有（i，j）的 num_inlier（i，j）/num_matches（i，j）數。這裡設定若兩張圖片的inlier數目占有matches的比例大於60%，則判斷這兩張圖可以做拼接，為同一景色。然而，有時候兩張圖片本身的matches數目就非常少，所以很容易達成inlier的比例大於60%的條件，所以我們在多加上一個條件，限制最少的matches數目大於60，才放入有機會是同張景色的配對。這兩個多加的數值條件是根據經驗，由嘗試了多次的資料，大概推斷出來的數據。（以下是Compute矩陣的示意圖，當符合配對條件是，給予數值1；反之，給予 0 。下圖情況即為(A1,A2,A4)&(A3,A5)個別為一組景色。）

	A1	A2	A3	A4	A5
A1	1	1	0	0	0
A2	1	1	0	1	0
A3	0	0	1	0	1
A4	0	1	0	1	0
A5	0	0	1	0	1

```
compute=compute'+compute;
for i=1:S
   compute(i,i)=1;
end
%到這邊就知道到底哪兩張是合在一起

A=group(compute,1);%(第一組)
Z=check(A,S,compute) ;%%Z就是最後匯出的分類矩陣
```

相同地，Compute 矩陣也是對稱矩陣，計算及儲存的方式跟前面兩個矩陣相同。直到這一步就大致構造出了判斷兩張圖是否可以拼接的依據。接下來我們自行撰寫了兩個副程式(group&check)，以達到我們搜尋正確配對的目的。Group 副程式是用來根據 Compute，判斷哪些圖片是一組，且回傳同組的圖片代碼出去。例如圖片（1，3，5）是同一張景色，可做拼接。則我輸入 group(Compute,i)（這裡 i 無論放 1,3,5 皆可），就會回傳出一個向量[1,3,5]。（以下為 group 的程式碼）

```
function [v] = group( M,x )

compute=M;
k=find(compute(x,:));
k1=k;
L=size(k);

for i=1:L(2)
temp=find(compute(k(i),:));
k1=union(k1,temp);
end;
```

```
while ( size(k1,2)~=size(k,2))

k=k1;
L=size(k);

  for i=1:L(2)
  temp=find(compute(k(i),:));
  k1=union(k1,temp);
  end

end

v=k1;
```

但考慮到單單使用 group 副程式，必須自行確認多次還有哪些圖片沒有被分組到，所以又自行多寫了一個 check 副程式，目的是讓電腦自己計算好各組，一次自動跑出來所有結果。Check 主要是延用第一次使用 group 的結果，檢查是否還有剩下沒被分類到的圖片。這裡的原理使用集合的方法。首先用 AAA 向量儲存目前已被分類好

的照片個數（例如上述的[1,3,5]，AAA 就儲存了 3 。），然後用 BBB 向量儲存照片一共的數目。當 AAA 不等於 BBB 時，表示還有照片沒有被分完，則繼續使用 group 函式。進入 while 迴圈之後，先看剩下來的照片到底是哪些，儲存在 remain 這個向量裡。不失一般性，再取 remain 的第一個元素去做分類，分完之後再將這些被分好的分組和之前原有的分組做聯集，存入 A1，而下一次的 AAA 即是目前 A1 的元素個數，再繼續連代操作直到所有圖片都被分類完為止。我們將分類好的所有圖片儲存在矩陣 Z 裡面。

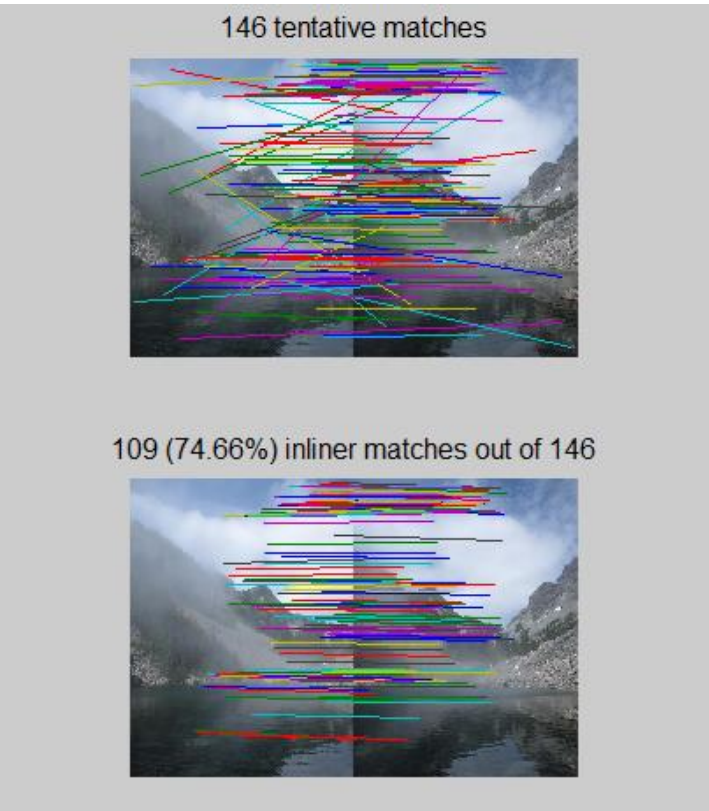
```
function [Z] = check( A,S,compute )

B=[1:S];
A1=A;
B1=B;
AAA=size(A);
BBB=size(B1);
Z=zeros(S,S);
Z(1,1:AAA(2))=A;
i=1;
```

```
while (AAA(2)~=BBB(2))

remain=setdiff(B1,A1);
v2=group(compute,remain(1));
vv=size(v2);


Z(i+1,1:vv(2))=v2;


A1=union(A1,v2);

i=i+1;
AAA=size(A1);
BBB=size(B1);
end
```

以下是 Z 矩陣的範例圖，這個例子下每一列是一個組別，圖片資料夾由代碼為 1-22 的照片所組成，根據此結果，可回傳一列列的 Z 矩陣的向量，交給下一個 Sorting&Stitching 的流程。

	1	2	3	4	5	6
1	1	2	0	0	0	0
2	3	0	0	0	0	0
3	4	5	6	7	8	9
4	10	11	12	13	14	0
5	15	16	0	0	0	0
6	17	18	0	0	0	0
7	19	20	21	22	0	0

（下圖是判斷 num_inlier/num_matches 的示意圖，相同景色的兩張圖片，其數值必須 > 0.6。）



在這個分類(classification)步驟中，所用到的 source code 為 Sift_mosaic，我們運用這個 code 回傳給我們任意兩張圖片間的 sift 所 match 的數目、RANSEC 後 inlier 的數目，得到這兩個資訊後，我們就能做分類。

四、影像拼接(Image Stitching)

由分類(Classification)完，我們可以得到每種場景確切擁有哪些的圖片，但由於在做 stitching 時，需要圖片次序的問題，也就是需要是完完全全的知道圖片在某個場景中得確切位置，但是，我們現在所得到的資訊並不是排序好的，它們是亂的，只知道他們是屬於同個場景而已，因此我們在做 stitching 前，會先將這些圖片做 sorting(方法如下)。

Step1:

首先，我們 stitching 的方法是會以排好次序中間的那張圖片為基準做影像拼接(此為 source code，feature based 的方法)，因此從原本相鄰兩張間的 homography matrix 會經由連續相乘，變成是對應於中間圖片的 homography matrix。例如，現在有 5 張排好順序的圖片(I1 -> I2 -> Iref -> I4 -> I5)，以 Iref 右邊為例，I1 與 I2 間有 homography matrix H1 ，I2 與 Iref 間有 homography matrix H2，經由 H1 和 H2 兩個矩陣的相乘，就能得到 I1 對應於 Iref 的 homography matrix H，其他圖片也都是如此。

Step2: (寫於 function FindTheRef.m 裡)

由我們的 data 可以知道，每種場景中，一定有一張圖片是與所屬場景中其他的圖片都有重疊到的， 因此我們一樣先用 RANSAC 找出每張圖片之間相連的關係，在這邊的 threshold 會設的比分類時來的高(我們設為 0.7)，且 inlier 數目要大於 0.6 乘以 sift match 的數目)，若大於這個數目就代表正在判斷的某張圖片與其他張圖片間的關係是相連的，會給予它一票，這樣的目的在與確切地找出圖片彼此之間的關係，以便做排序。(表格一、每張圖片之間的相連關係，若相連為 1，反之為 0。表格二、每張圖片的得票率，票數越高代表與這張圖片相連得越多張)

	1	2	3	4	5	6
1	0	0	1	1	0	1
2	1	0	1	1	0	0
3	1	1	0	1	0	1
4	1	1	1	0	1	1
5	0	0	0	1	0	1
6	1	0	0	1	1	0

(表格一)

1	2	3	4	5	6
3	3	4	5	2	3

(表格二)

Step3: (寫於 function FindTheRef.m 裡)

由 step2 的結果，我們就能做排序，以投票數最高的那張(表格二藍色圈起來的，第 4 張圖片)當做基準 Iref，放在最中間，然後先從右邊開始排，找投票數最低的(第 5 張圖片)當做 Iref 右邊的第一張，然後再找與第 5 張相連的圖片， 由表格一可以知道是第 6 張，於是把這放在 Iref 右邊的第二張，於是右邊排完了，換排左邊(前面排過的圖片就不會再排了)，一樣先找投票數最低的(若票數相等會從相等的圖片中隨便挑一張，所以挑到第 2 張圖片)，然後在找其對應關係且其票數也是剩下未排到圖片中票數最低的(第 1 張圖片)，最後一張就是第 3 張圖片。

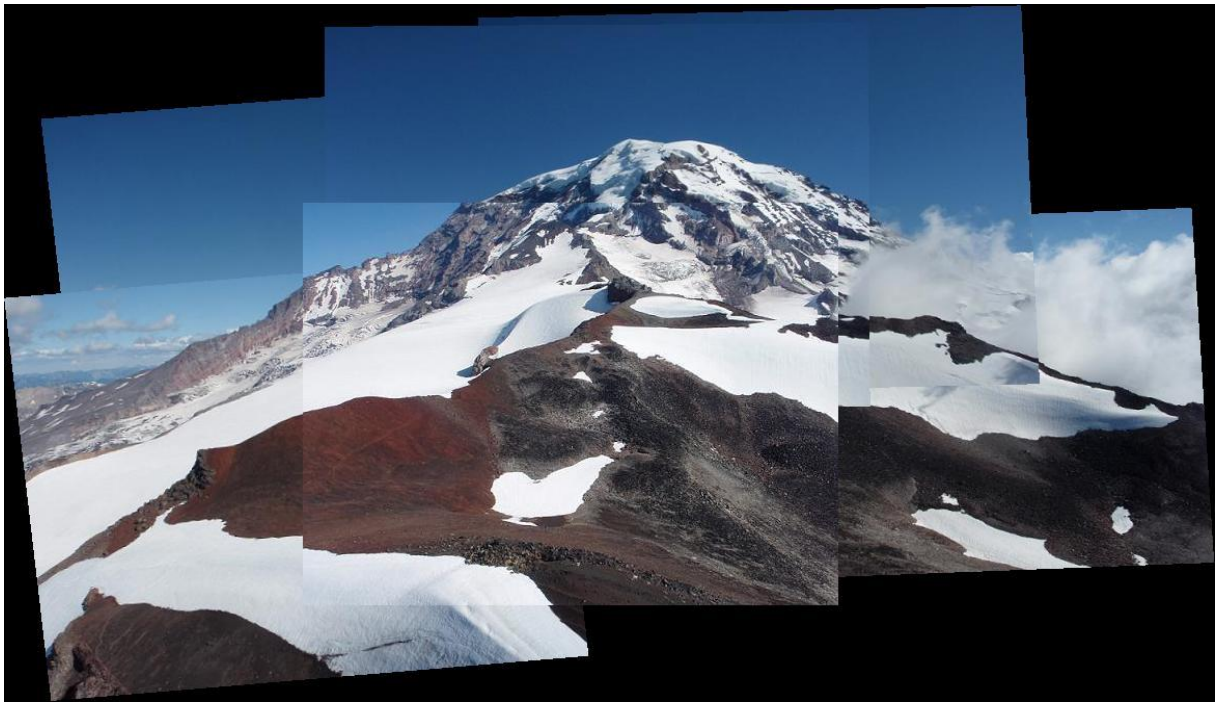
Step4: (寫於 function FindTheRef.m 裡)

由 step3 我們就能得到某個場景的順序關係，I3 -> I1 -> I2 ->I4-> I5 -> I6，然後將得到的次序送到 stitching 的 source code，就能得到拼接好的場景(成果放在五、結果)。

五、結果(Results)

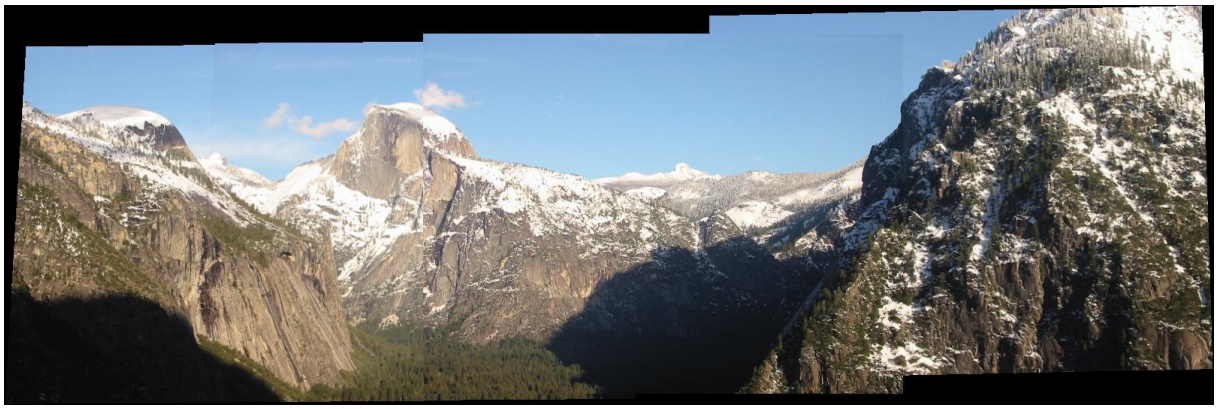
以下我們總共試了 4 個場景。

A、6 張圖片的場景

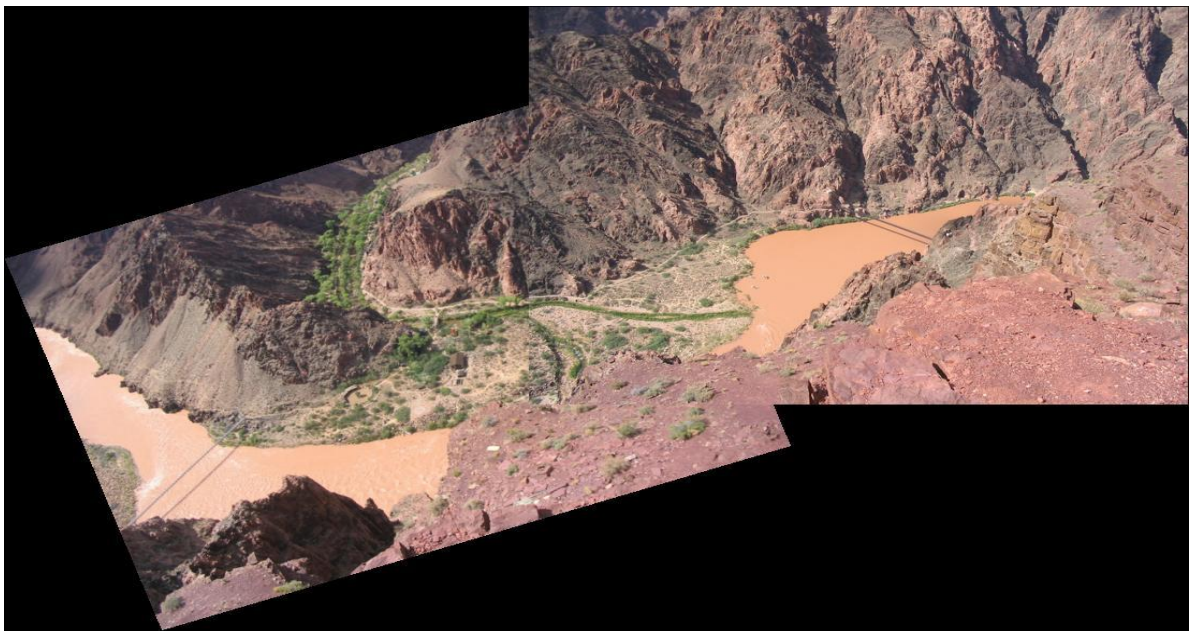


B、5 張圖片的場景

C、4 張圖片的場景



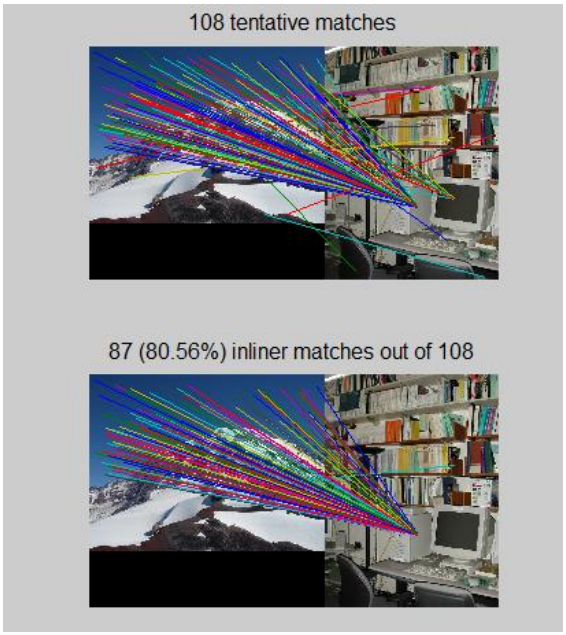
D、2 張圖片的場景



六、問題與討論

下圖，為我們在再分類(classification)遇到的問題，可能在不同場景的圖片間會有重大錯誤的 match，我們覺得會如此的原因，可能是因為 match 到的地方都是很單調的，色調也都很平滑的且相似的。

目前想到解決的方法是，在我們原先分類的方法後，在加上找出圖片中 saliency 的地方，判斷圖片之間 saliency 的地方是否為相同，希望能透過這樣的方法解決這樣的問題，然後確切的分類出來。



七、參考資料(Reference)

1、Brown, M. and D. Lowe. Automatic Panoramic Image Stitching using Invariant Features. International Journal of Computer Vision. 2007.

2、Szeliski, R. Image Alignment and Stitching: A Tutorial. Microsoft Research. 2004