

VFX Project #2 Report

a10409007 劉健彬

編譯環境

Matlab R2016b + Image Processing Toolbox

實作內容

1. Taking Photos on a tripod

使用設備：Canon 6D / 腳架

使用腳架固定相機在一個位置，然後利用腳架上的角度參數旋轉相機，

每旋轉間隔大約 20 度，拍一張，總共拍了 17 張（1824 x 2736）。

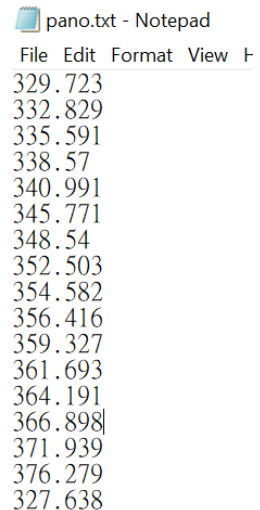
2. Warp to cylindrical coordinate

1) 讀取照片

首先讀取一組 17 張的照片。

2) 獲取每張圖片的 focal 值

通過 autostitch (32bit windows version) 軟體，讀取拍攝的 17 張照片，得到 pano.txt，裡面記錄了估算的每張照片的焦距值。將多餘的訊息刪除，僅留下每張照片的焦距值，並用換行符相隔（如右圖）。



```
pano.txt - Notepad
File Edit Format View H
329.723
332.829
335.591
338.57
340.991
345.771
348.54
352.503
354.582
356.416
359.327
361.693
364.191
366.898
371.939
376.279
327.638
```

3) 將原來座標各點資訊對應到圓柱座標上

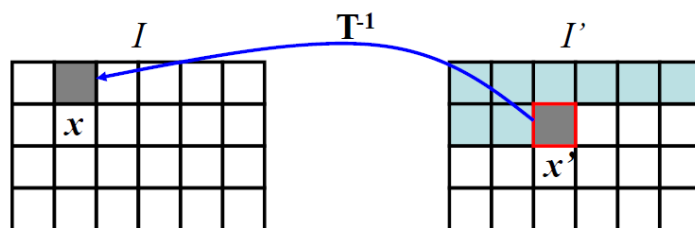
直接用拍攝的照片進行 image stitching 的結果會是 Rectilinear Projection，不是很符合人的習慣，所以要將原本的每一張照片進行 Cylindrical projection。通過投影片裡提到的公式以及演算法，將原來照片每個像素的 XY 座標轉換到圓柱座標上。

$$x' = s\theta = s \tan^{-1} \frac{x}{f}$$
$$y' = sh = s \frac{y}{\sqrt{x^2 + f^2}}$$

Inverse warping

DigiVFX

```
iwarp(I, I', T)
{
    for (y=0; y<I'.height; y++)
        for (x=0; x<I'.width; x++) {
            (x,y)=T-1(x',y');
            I'(x',y')=I(x,y);
        }
}
```



3. Feature Detection

因為拍攝的照片不涉及到會發生 scale 的情況，所以採用相對其他兩種比較沒那麼複雜的方法，Harris corner detector.

1) 將照片轉成灰階圖

將每張照片都轉為灰階圖。

2) 減少 noise，計算 x 和 y 方向的 derivatives

為了減少 noise，對每張圖做 gaussian filter，並且根據方程式計算 x 和 y 方向的 derivatives。

$$I_x = G_{\sigma}^x * I \quad I_y = G_{\sigma}^y * I$$

3) 計算每個像素的 xy 方向的乘積

根據方程式，計算每個像素的 x 和 y 方向各自的乘積以及 xy 的乘積。

$$I_{x^2} = I_x \cdot I_x \quad I_{y^2} = I_y \cdot I_y \quad I_{xy} = I_x \cdot I_y$$

4) 根據方程式，對每個值進行 gaussian filter

$$S_{x^2} = G_{\sigma'} * I_{x^2} \quad S_{y^2} = G_{\sigma'} * I_{y^2} \quad S_{xy} = G_{\sigma'} * I_{xy}$$

5) 求出 R

為了決定怎樣的特徵值 λ_1 和 λ_2 才是夠大的，定義 Response R。
k 的取值範圍是 0.04~0.06，經過測試，在自己拍攝的照片中取
0.04 效果相對比較好。並且根據方程式及上一步驟中算得的值算
出 R 矩陣。

$$R = \det \mathbf{M} - k(\text{trace} \mathbf{M})^2$$

$$\begin{aligned} \det \mathbf{M} &= \lambda_1 \lambda_2 \\ \text{trace} \mathbf{M} &= \lambda_1 + \lambda_2 \end{aligned}$$

6) 設定 threshold

對 R 設定一個 threshold,超過這個值的 pixel 就抓取下來，因為這個
pixel 就有可能對應到 corner。

4. Feature Description

通過上一步驟 Feature Detection 之後，找出了 feature 在哪裡。為了下一步能
夠比較兩兩圖片之間的 feature，所以要對 feature 進行描述。

取了上下左右各 2 個 neighbor，形成 8 維的 vector，作為 feature points 的
Descriptor.

5. Features matching

有了上一步驟取得的 Descriptor 之後，接下來就要比較 feature 之間的相似程度。

1) 計算兩張照片 feature 之間的距離

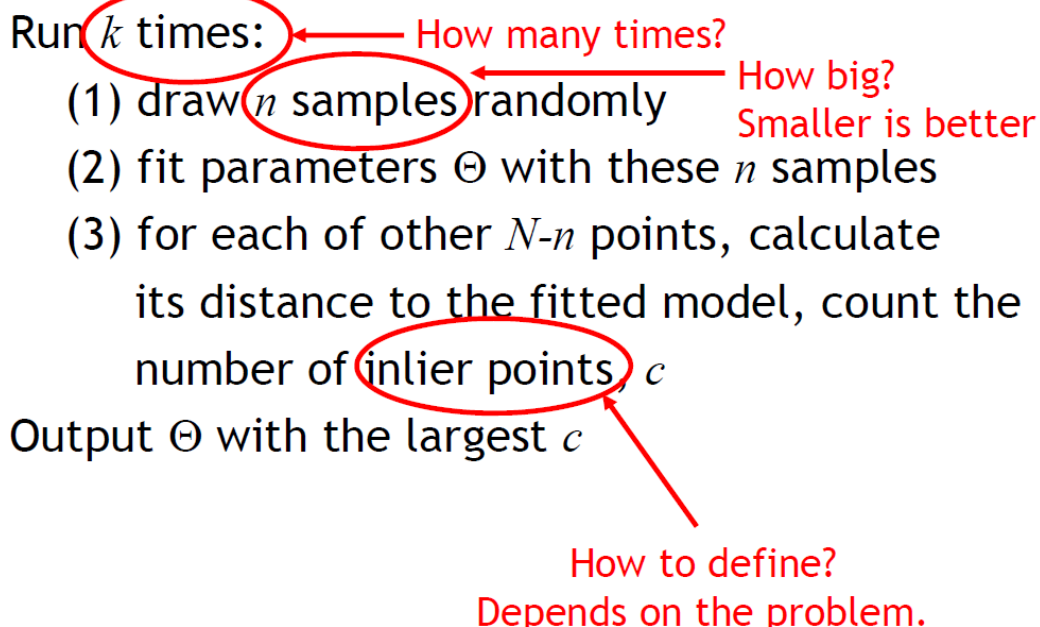
用距離來表示兩張照片之間的相似程度，距離越短則代表相似性越高。

2) RANSAC 演算法

找出來相對應的 features，不一定是 inliers，如果直接計算的話，如果裡面有一個差異很大的 outlier，則算出來的 model 會被影響。所以在這裡要利用 RANSAC 演算法去避免 Outlier.

根據投影片中的演算法，重複 run k 次，每次都隨機從裡面取出 n 個點，利用這 n 個點估計 Θ ，接著把剩餘的點 ($N-n$ 個) 再帶入新的 model 裡算出有多少點是 inlier。重複 k 次之後能讓 inlier 個數最多的 Θ 值就是算出的解。

RANSAC algorithm



其中涉及到 k 和 n 兩個參數值的取值問題。根據關係式， p 是真正 inlier 的機率，而 P 是演算法跑了 k 次之後成功的機率。在固定 n 值的時候， p 越大， k 則越小。相反固定 p 時， n 越大，所需要的 p 就越大。而 p 通常未知，所以 n 取小一點比較好。所以在程式中將 P 值設為 0.999， n 值設為 3。

6. Images matching

利用 inlier match 中的 feature 對應的座標做 image matching. 透過 Model

$Ax = b$ 計算位移量，即圖片 1 要位移多少才能和圖片 2 接合。

7. Images blending

將兩張照片拼接的時候往往會因為拍攝的角度或是光源問題導致顏色上有無法，所以要做 blending. 程式中使用了線性的方法進行 Blending，使兩張圖的接合處不會太過突兀。

處理過程及效果討論



結果圖

解決的問題

1. 直接拿拍攝的照片跑程式會非常慢，因為圖片太大，所以後期為了方便測試將所有圖片的尺寸從 1824 x 2736 轉到 300x450。
2. 有些圖的 feature 非常多，常常集中在某個角落，常會發生在一些例如草地之類的場景，導致 feature 太多程式跑很慢。所以試著做了一下 Non-Maximal Suppression，不過最後效果還不算特別好。

仍然存在的問題

1. Drift 問題沒有解決，最後拼接的照片會逐漸地往上移。
2. Blending 沒有做的很好，圖片交合處仍然很明顯。整體色調和亮度也一致。

References

- 課堂投影片
- M. Brown and D. G. Lowe, Recognising Panorama, ICCV 2003.
- Chris Harris, Mike Stephens, A Combined Corner and Edge Detector, 4th Alvey Vision Conference, 1998