VFX hw2 Author: Shannon Lee 李尚倫 studentID: r07921001

Images Stitching(Panoramas)

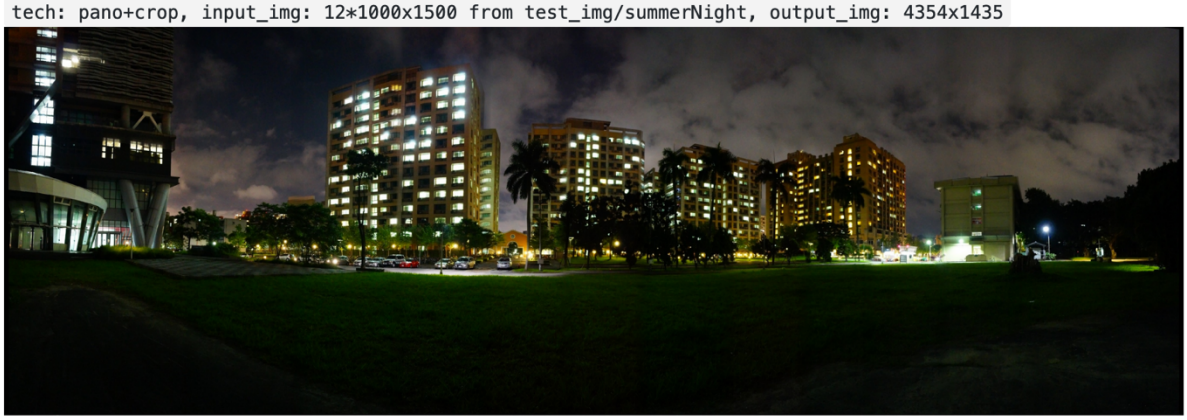
A python implementation of **Images Stitching(Panoramas)[1]**.

- feature detection[2] `SIFT detector, SIFT discriptor`
- feature matching `Brute force(2-norm distance), flann(kd-tree, knn-search)`
- image matching `RANSAC finding shift`
- blending `Linear filter on edge, Linear filter on overlapRegion, Naive overlap stitching`
- end to end alignment `Scattering y displacement`
- cropping

Artifacts

summerNight:

`tech: pano+crop, input_img: 12*1000x1500 from test_img/summerNight, output_img: 4354x1435`



DEMO_parrington:

`tech: pano+align+crop, input_img: 18*384x512 from test_img/parrington, output_img: 4565x502`

DEMO_grail:

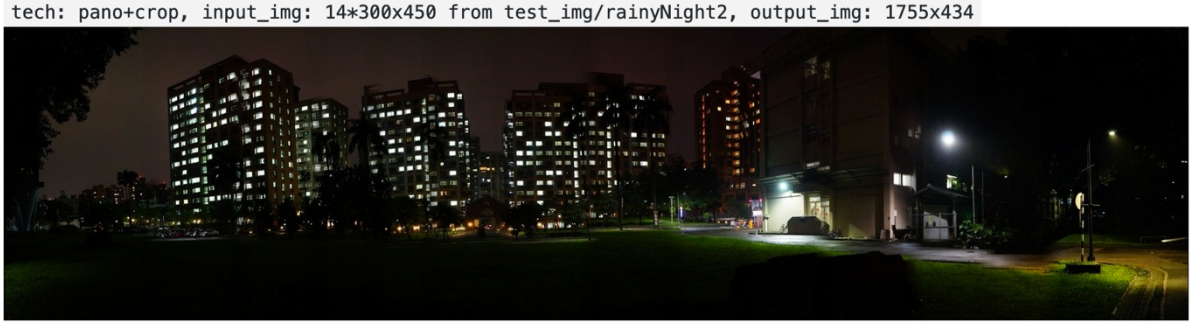
`tech: pano+align+crop, input_img: 18*384x512 from test_img/grail, output_img: 4102x499`

chasedByDog:

`tech: pano+crop, input_img: 10*1000x1500 from test_img/chasedByDog, output_img: 4421x1268`

rainyNight:

`tech: pano+crop, input_img: 14*300x450 from test_img/rainyNight2, output_img: 1755x434`



midnightStreet:

`tech: pano+crop, input_img: 12*533x800 from test_img/midnightStreet, output_img: 3823x728`

dormitoryNight:

`tech: pano+crop, input_img: 6*533x800 from test_img/dormitoryNight, output_img: 1632x734`

Check the original size in result_img/
Or https://github.com/shannon112/VFXolo/blob/test/project2_panorama/README.md

Instruction to run:

1. Create a folder of images
   `For the images in folder, tne filenames should increase according to the view of landscape from the left side to the right side` e.g.

```
DSC09994.jpg is the left side of landscape
DSC09995.jpg is the middle of landscape
DSC09996.jpg is the right side of landscape
```

2. Create info.txt in the image folder
   `The focal lengths in info.txt are gotten from autostitch(win7 old ver.), and should list from top to bottom with filenames increasing.` e.g.

```
DSC09994.jpg 628.57
DSC09995.jpg 631.311
...
```

3. Run the image stitching program at VFXolo/project2_panoramas e.g.

```
python2 src/main.py test_img/dormitoryNight
```

4. If you are not satisfied with the result, you can tune the parameters in `constant.py`

Files Structure

Files structure

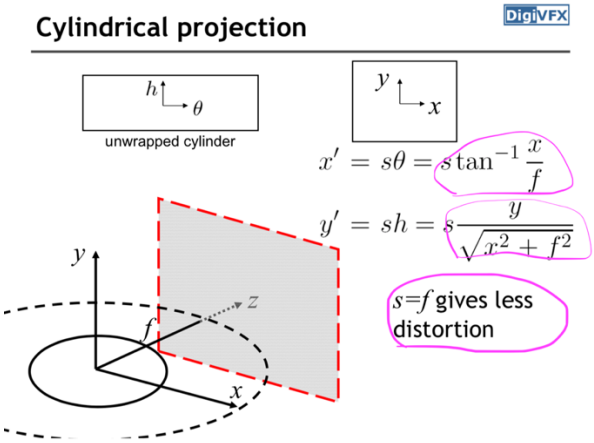
```
├── project2_panoramas
│   ├── src       #source/executable code
│   ├── testing_src   #some testing code while developing
│   ├── raw_img     # 6000x4000 raw images, 5 collection
│   ├── result_img     # pano, aligned_pano, cropped_pano of each collection
│   ├── test_img     # small size image of raw_img
```

Source code is available on:
https://github.com/shannon112/VFXolo/blob/test/project2_panorama/README.md

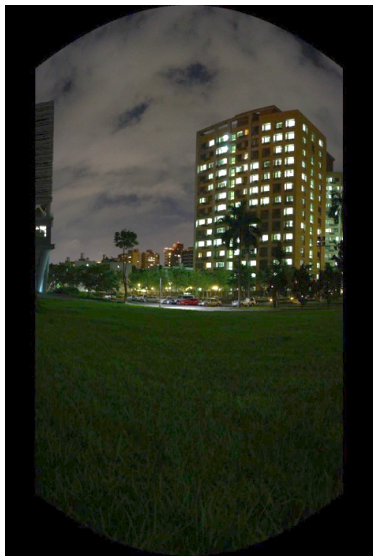
Details of My Implementation

0. Image warping

我這裡實作的warping基本上就是套用講義上的式子[5]，把每個pixel從x,y轉換到x',y'



而focal length的部分則來自於autostich(old win7 version)的pano.txt，基本上如同講義所言，focal length越後的話兩側的distortion越大，而選擇恰到好處的focal length 才能使圖片拼起來的效果最好，關於這點後面的conclusion還會再討論。下面左圖是Focal length =1500.2，右圖是592.85。

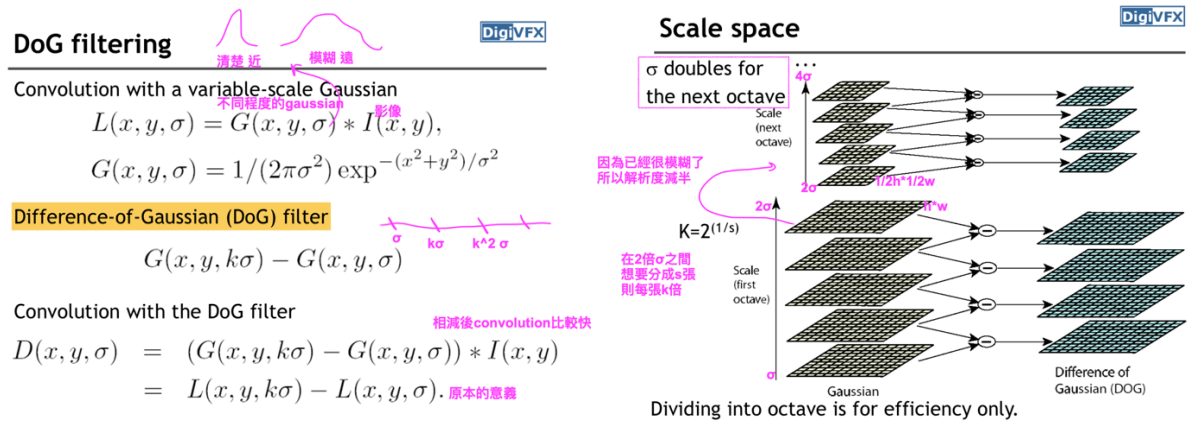
1. Feature detection

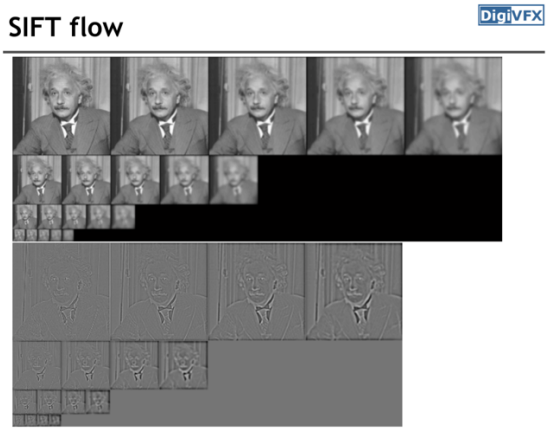
這裡我實作了SIFT演算法的detector和discriptor，大致上是照著paper[2]和講義上所講的，並參考[3]，修改而來。

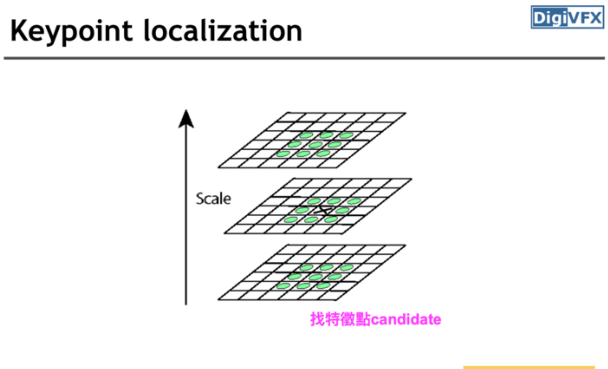
a. SIFT detector — Scale-space extrema detection 和 Keypoint localization

共有4個octave，分別為原圖的2x, 1x, 0.5x, 0.25x scale
每個scale分別做6層gaussian，得到5層DoG

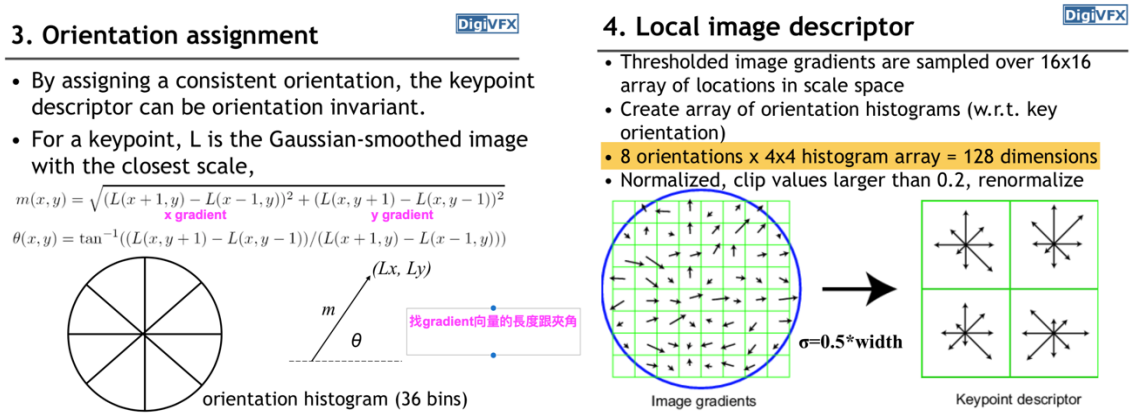
然後每個點再和鄰近的26個點相比，較大或較小者被labeled as extrama
然後再針對這些extramas找出Accurate keypoint localization







b. SIFT descriptor — Orientation assignment和Keypoint descriptor



計算每個keypoint的orientation，然後再取Local features，針對每個keypoint取16×16大小的像素，再平均切為4×4的cells，每個cells取梯度及角度值統計為8個bins的Histogram，合併成128維度的資料，最後進行L2-Normalizing，即可得到代表該keypoint的descriptor

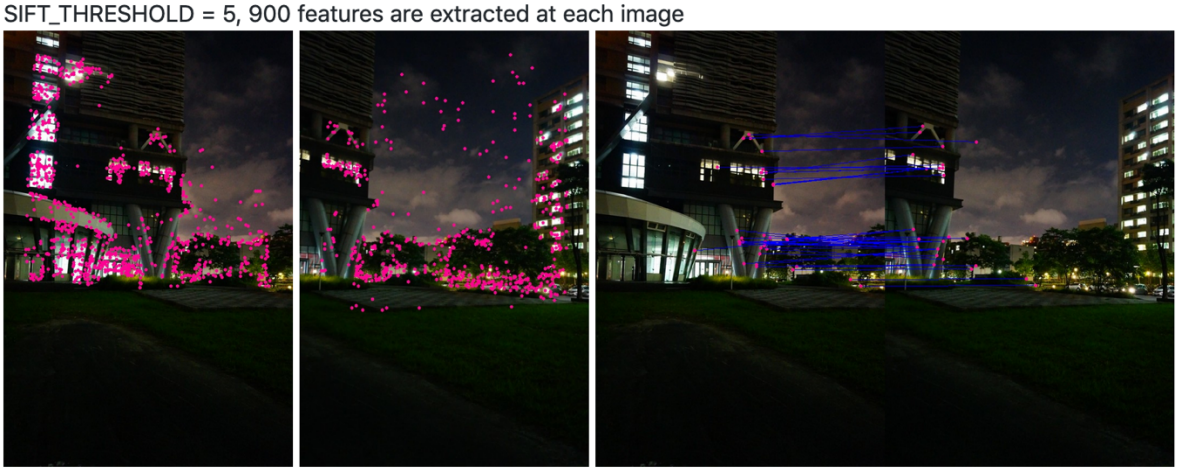
c. Result

最終對每張進來的圖給出keypoint和descriptor，如有n個features的話，keypoint會是nx4的array，前2個是position，後兩個是orientation，descriptor則會是nx128的array。下圖為不同threshold下的成果。

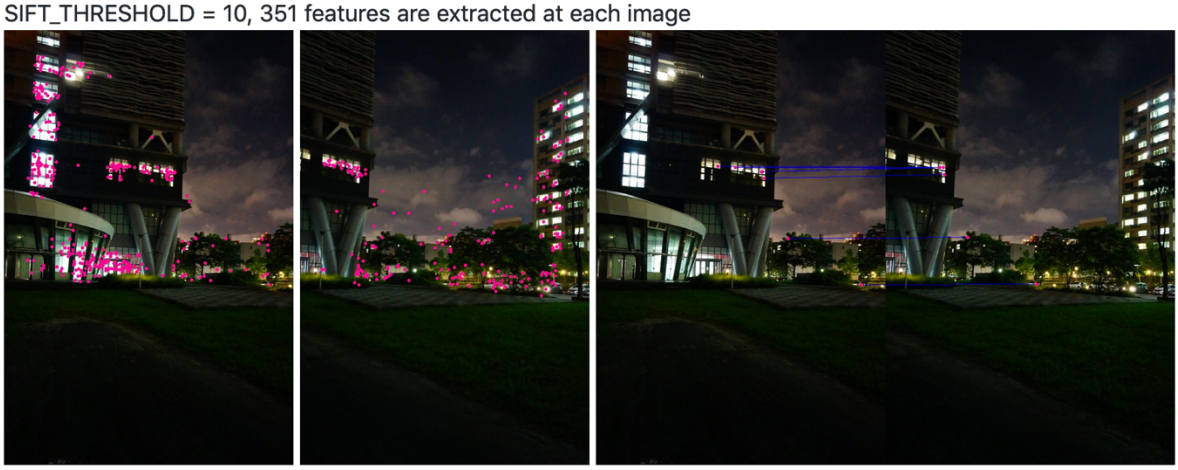
All results use brute force feature matching

SIFT_THRESHOLD = 3, 1757 features are extracted at each image

SIFT_THRESHOLD = 5, 900 features are extracted at each image



SIFT_THRESHOLD = 10, 351 features are extracted at each image



原圖可見：
https://github.com/shannon112/PythonSIFT/blob/master/README.md

2. Feature matching

這裡我實作了Brute force 和 kd-tree+knn-search (參考[3]並使用flann lib)

Feature matching DigiVFX

- Exhaustive search
  - for each feature in one image, look at *all* the other features in the other image(s) n*m 慢
- Hashing
  - compute a short descriptor from each feature vector, or hash longer descriptors (randomly)
- Nearest neighbor techniques
  - *k*-trees and their variants (Best Bin First)

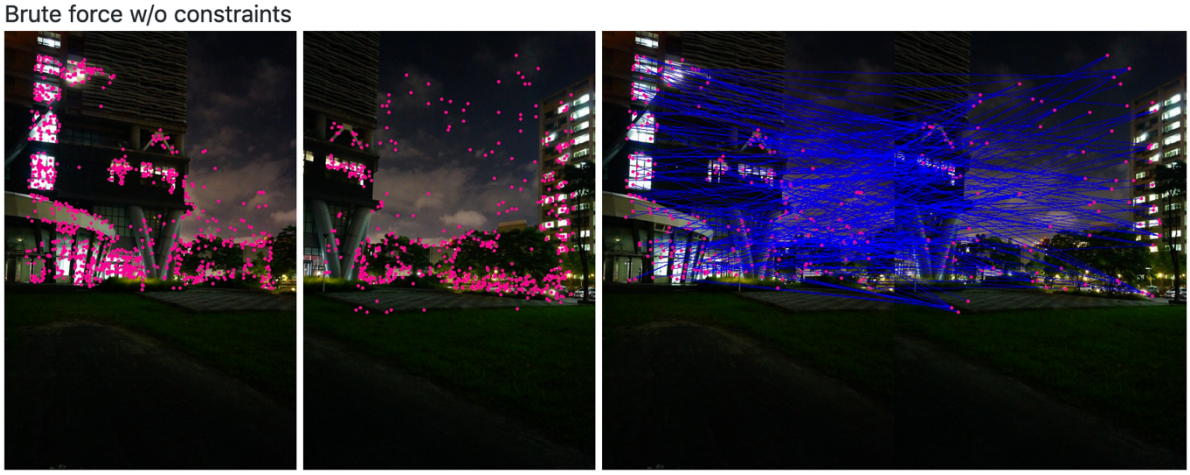
Brute force(2-norm distance) 法就是簡單的雙迴圈計算2-norm，而flann(kd-tree, knn-search)則是call library這裡就不多敘述，僅為做為比較。但以結果來說運算速度並沒有差太多(相較前面提取feature的時間，此差異並沒很有感)，matching的品質亦差異不大。

此外，因為假設照片品質良好的情況下，我將可能match的兩點的x的差和y的差做了filter，如果該match的兩點y值跳躍太大則不取，x值平移太小或太大也不取，而以下則是兩種方法和有無加入constraint得結果比較。

All results use SIFT_THRESHOLD=5

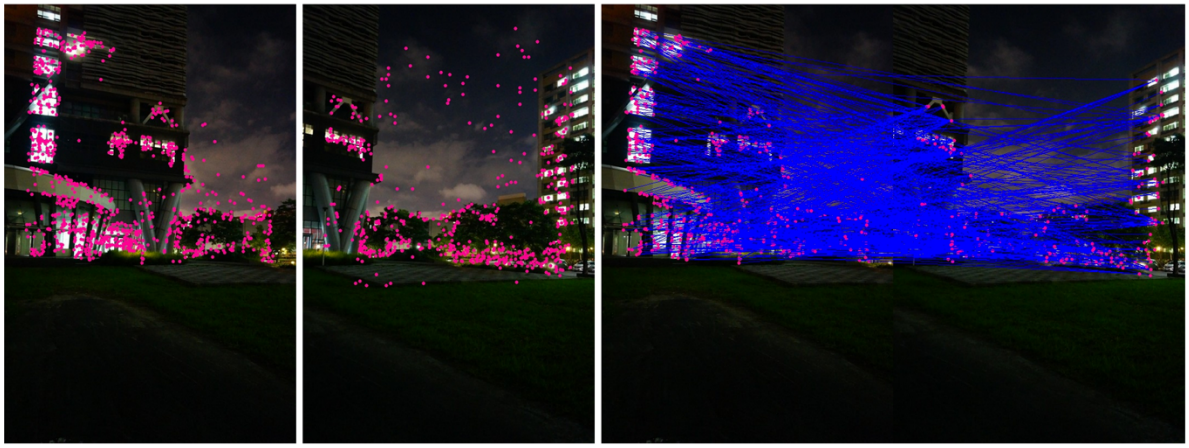
Brute force w/ constraints

Brute force w/o constraints



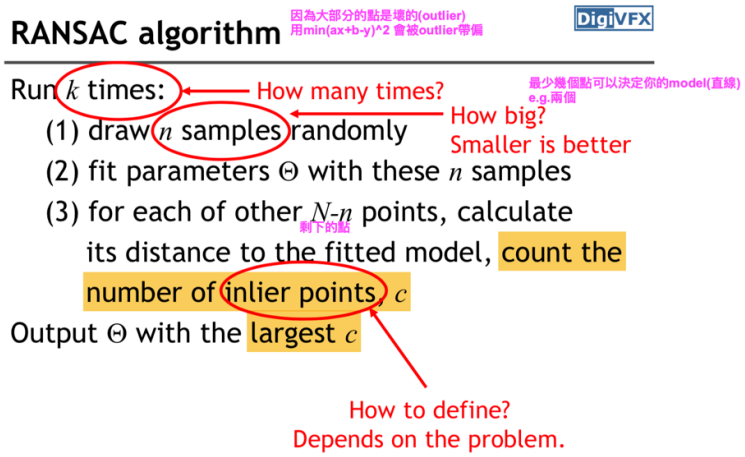
flann w/ constraints

flann force w/o constraints



3. Image matching

這裡我實作了RANSAC來找到最佳shift，參考[4][6]

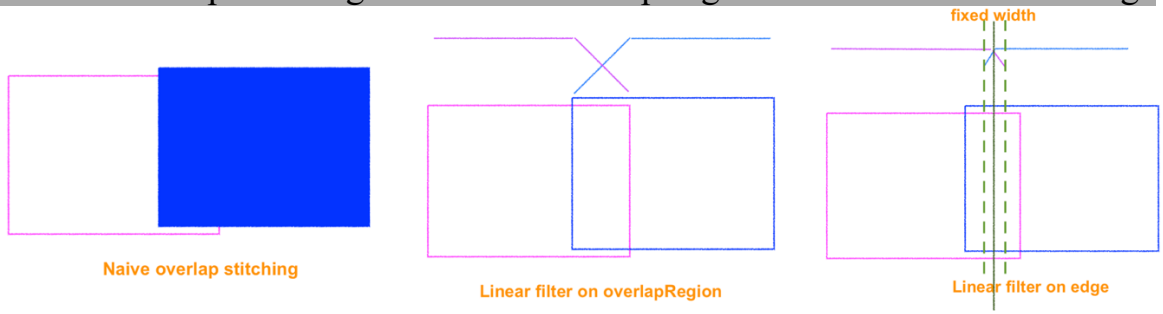


基本上就是隨機取一個matched pair(含前一張照片的點座標和後一張照片的點座標)，算出shift後，套用回去所有matched pairs的後一張照片的點座標，算出預計的前一張照片點座標，然後和該matched pair真正的前一張照片點座標做2-norm distance，小於threshold的即為inliner，最後取inliner最多的shift作為最後的shift。

4. Blending and stitching

這裡我實作了三種stitching方式：Linear filter on edge, Linear filter on overlapRegion, Naive overlap stitching

a. Naive overlap stitching　b. LF on overlapRegion　c. Linear filter on edge



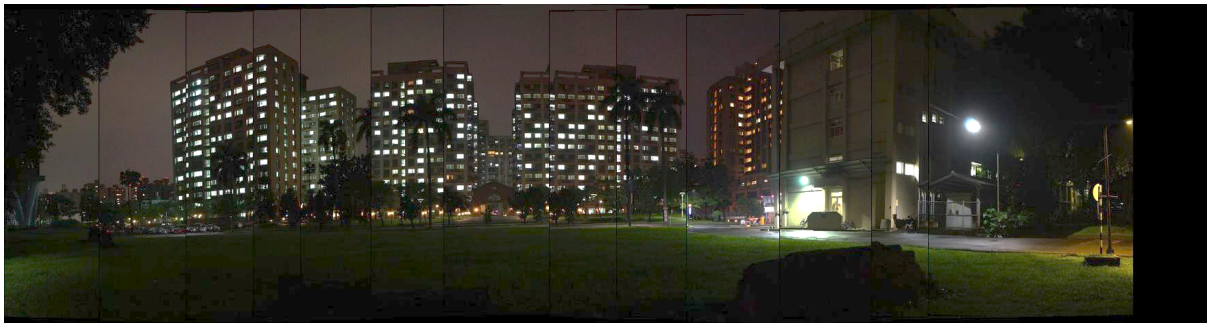
a方法實作起來做簡單，但也很容易受到照片品質亮度不一所影響，會留下明顯的不連續顏色接縫，如果shift不夠好的話也會直接看到建築物斷層。而b方法則是用線性的方式做blending，可以完全消彌邊界，但壞處是如果shift沒有抓得很好，因為重疊區間太大，會有明顯鬼影。而最終改良成c方法就只針對兩張照片接縫處左右各固定幾個pixels內做blending，縮小blending範圍，不易產生鬼影，但也不致於可以看得出接縫。

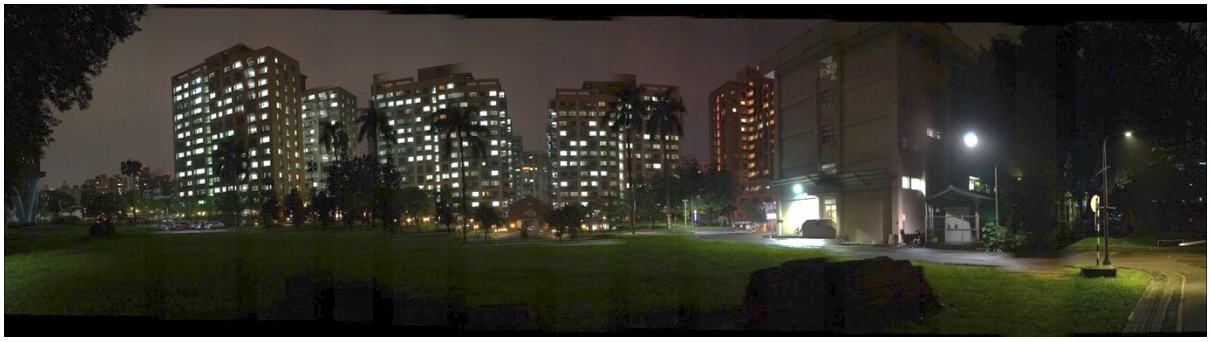
d. Comparison (原圖可見：https://github.com/shannon112/PythonSIFT/blob/master/README.md)

e.g.1 針對domitoryNight 的前兩張做比較

Naive overlap stitching　LF on overlapRegion　LF on edge

e.g.2 rainyNight 使用Naive overlap stitching(上)和Linear filter on edge(下)





5. End to end alignment

這裡我實作了針對gradually向下或向上斜的pano的end to end alignment
用的方法很簡單，基本上是參考講義上的[7]，把位移分散到各個column。

- add displacement of $(y_1 - y_n)/(n-1)$ to each image after the first

不過此方法僅對較完美的dataset有用(向下方兩個作業提供的dataset)拼起來是gradually上升或下降的，而我自己拍攝的大都會上上下下，套用此方法起來效果就沒這麼好，直接crop即可。

e.g.1: parrington原pano (向右上斜)

e.g.1: parrington align後pano

e.g.2: grail原pano (向右上斜)

e.g.2: grail align後pano

6. Cropping

這裡實作的就是簡單的cropping，基本上就是先轉成灰階，然後把該image row確定是沒資訊的pixel（黑色，value=0）超過一個比例時，整個row砍掉，如此對上下裁剪完後得出最終的crop後的pano。

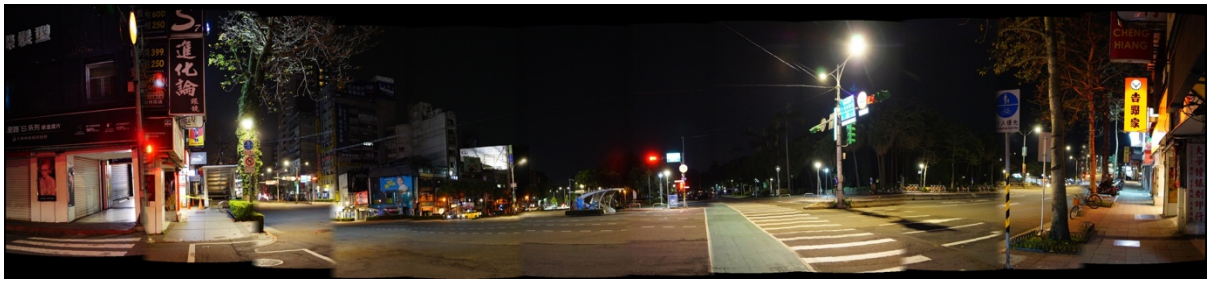
e.g.1: parrington align後pano

e.g.1: parrington crop後pano

e.g.2: grail align後pano

e.g.2: grail crop後pano

e.g.3 midnight street 原pano



e.g.3 midnight street crop後

7. Conclusion

基本上這次的作業實作下來的感想是，照片的品質(有沒有晃到旋轉到貨不連續移動)和autostich吐出來的focal length值很重要，這兩者如果沒弄

好，再怎麼做feature detection, feature matching, image matching, stitching and blending的調整結果都不會太好。因為前者如果有兩個照片之間有旋轉到或仰角改變或其他非我們原本理想假設(相機固定繞著同一軸旋轉)，都可能導致後來演算法拼不上去(因為我設計的shift只考慮x和y)。所以原本midnightStreet, rainyNight, dormitoryNight都是手扶著相機在腳架上旋轉，照片品質沒有很好，後來的chasedByDog和summerNight才改成把相機用膠帶捆在腳架上旋轉得到比較穩定的照片品質(因為相機側邊沒有固定孔QQ)，得到和demo照片(parrington和grail)差不多的結果水準。


而後者的大小也會決定後來拼貼起來時的直線會不會連續(大小不同投影到圓柱的扭曲度也不同)，但如果曲度不合適可能會有殘影。
e.g. summerNight最左邊的樓，大樓的中間處剛好是兩張圖的邊界。左圖是比原本autostitch focal length較大的參數，右圖是autostitch原本的參數。我們可以觀察到原參數在黏接觸有較好的表現，沒有殘影，但原參數的曲度較高，在下方草皮處會變得比較曲，導致接起來有波浪感。


References:

[1] M. Brown, D. G. Lowe, Recognising Panoramas, ICCV 2003
[2] Matthew Brown, Richard Szeliski, Simon Winder, Multi-Image Matching using Multi-Scale Oriented Patches, CVPR 2005
[3] PythonSIFT : https://github.com/rmislam/PythonSIFT
[4] Panoramas: https://github.com/SSARCandy/panoramas-image-stitching
[5] Image warping: VFX lecture07 slide p.36
[6] RANSAC: VFX lecture07 slide p.69
[7] End to end alignment: VFX lecture07 slide p.49