CSIE5429 3D Computer Vision with Deep Learning

Applications HW 1

Cheng-En Tsai

Email: r10525028@ntu.edu.tw

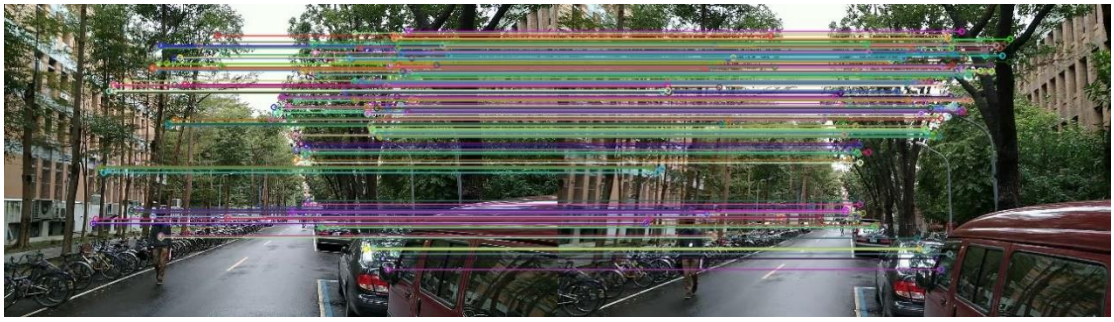
- Briefly explain your method in each step
  - Camera calibration

使用助教提供的 camera_calibration.py 與 calib_video.avi 計算相機的內部參數矩陣，內部參數矩陣計算結果如下：

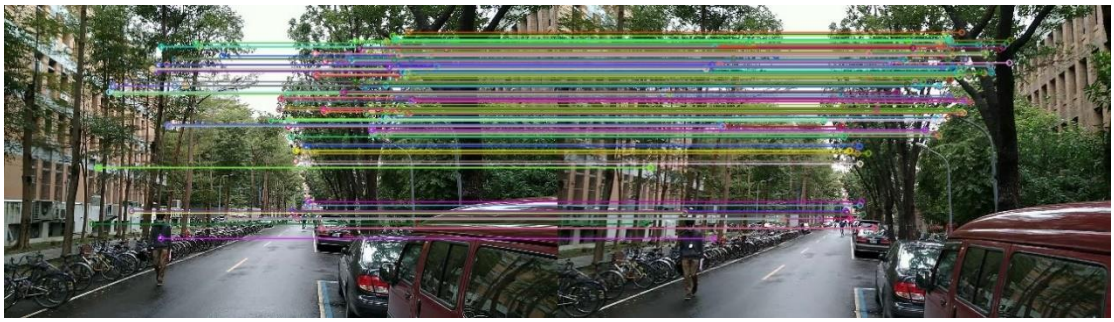
```
Camera Intrinsic
[[524.75727349   0.         315.19705   ]
 [  0.         525.8798624  183.92187059]
 [  0.           0.           1.        ]]
Distortion Coefficients
[[ 1.23562270e-01 -9.32110713e-01 -2.84784337e-03 -1.44326608e-04
   1.76649399e+00]]
```

  - Feature matching（orb feature matching 結果）

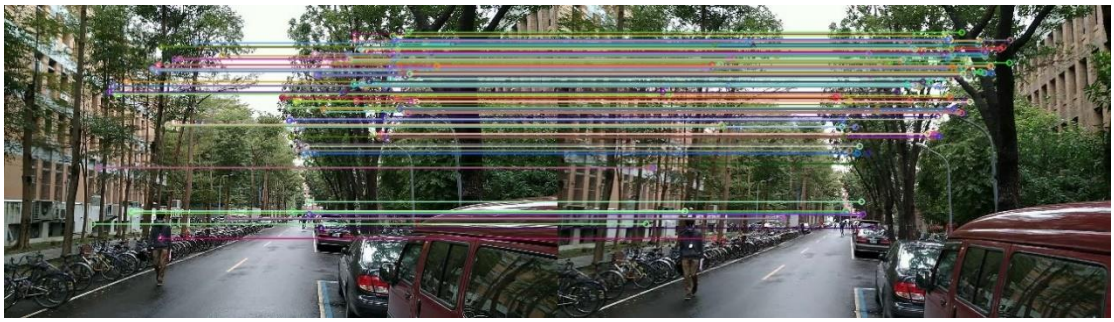
使用 orb 匹配結果



經過評估 essential matrix 後 inlier 的點



經過 recover pose 後 inlier 的點



- Pose from epipolar geometry

相機 pose 的計算流程如下，為了達到 scale consistent 我使用了上課講義內的公式，我隨機選擇同時出現在 2 張影像中的 3 維點來計算出 t，由於 t 會不斷地改變，所以我選擇 t 陣列中的中位數。

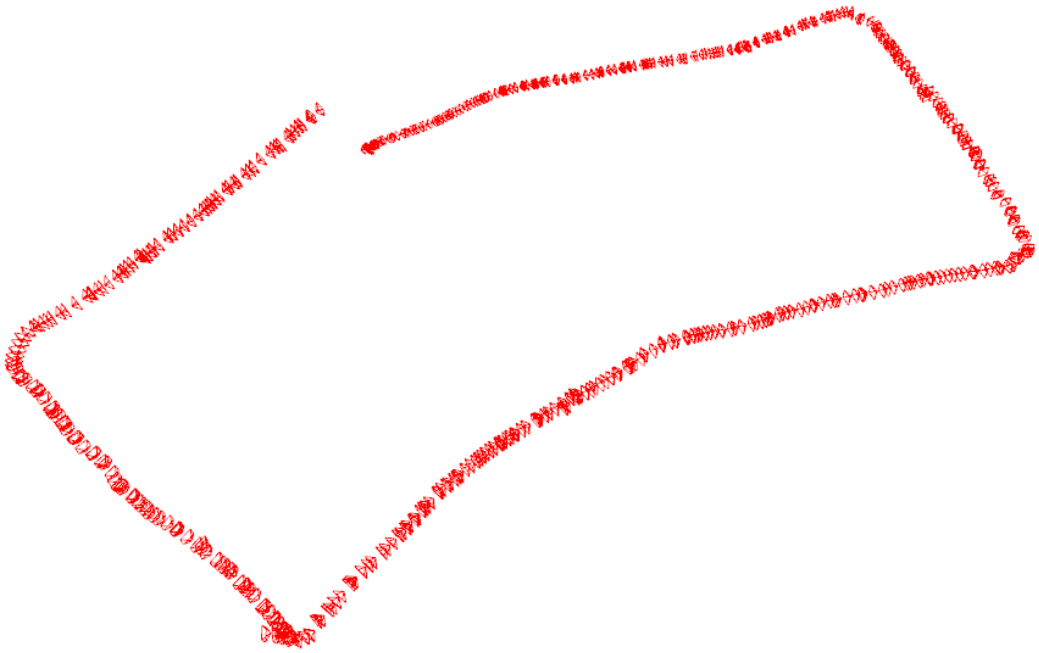
$$\frac{\left\| {}^{k-1}\boldsymbol{t}_{k} \right\|}{\left\| {}^{k}\boldsymbol{t}_{k+1} \right\|} = \frac{\left\| {}^{k}\boldsymbol{X}_{k-1,k} - {}^{k}\boldsymbol{X}'_{k-1,k} \right\|}{\left\| {}^{k}\boldsymbol{X}_{k,k+1} - {}^{k}\boldsymbol{X}'_{k,k+1} \right\|}$$

```
#TODO: compute camera pose here
# ------------------Step2 Extract and match feature between Img_k+1 and Img_k -----------------
kp0, kp1, points0, points1 = self.get_correspond_points("orb", img0,img1)
# draw_match(img0, img1, points0, points1)

# ------------------Step3 Estimate the essentail matrix -----------------
E, inlier = cv.findEssentialMat(points0,points1,self.K,cv.RANSAC,0.999,1.0)
points0, points1 = self.sort_inliers(points0,points1,inlier)
# draw_match(img0, img1, points0, points1)

# ------------------Step4 Decompose the E_k,k+1 into  R_k+1 and t_k+1 -----------------
rerval, R, t, inlier, triangularpoints = cv.recoverPose(E, points0, points1, self.K, distanceThresh=1000)
points0, points1, points3d = self.sort_inlier3d(points0, points1, triangularpoints, inlier)
# draw_match(img0, img1, points0, points1)
```

- Result visualization: 以下呈現出使用 orb feature 的結果，此外

我另外比較了使用 sift feature 的結果，結果發現到因為 sift 偵測到更多的特徵點，所以計算結果會比 orb 好更多，但也會花更多時間。

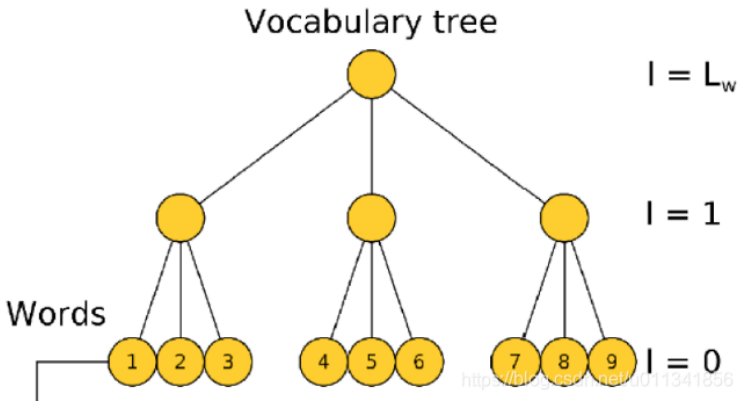
Orb feature (total time 125.62sec):



Sift feature (total time 156.90sec)

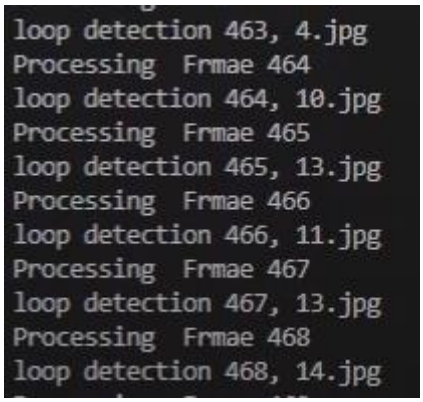
- Youtube link
  - ORB feature: https://youtu.be/jTRPueav8nI
  - SIFT feature: https://youtu.be/Mfknrjl1T1I

- Loop detection: 此部分有與吳泓毅(r11922029)同學一起討論。
  - 程式參考來源: https://github.com/itswcg/DBow-Python
  - 方法: 我參考的程式碼是實作 DBoW 演算法，此演算法輸入是 descriptor，針對這些 descriptor 不斷地 clustering，樹的深度與 clustering 是可以自己手動輸入。



  - 應用: 我將字典設定為(k= 5, L = 4)，由於相鄰的照片分數會相當高，所以程式碼有寫假如照片編號相減大於 400 才可以印出" loop detection "。
  - Loop 偵測結果: 本程式碼 loop detection 偵測結果有: (#456, #2), (#463,#4), (#464, #10), (#465, #13), (#466, #11), (#467, #13), (#468, #14), (#475, #24), (#489, #36), (#490, #36), (#492, #36), (#494, #39), (#499, #40), (#506, #44)。

- Loop detection 結果：



- How to execute my codes: python vo.py ./frames
  - 如果要更換不同的 feature 需修改 vo.py 第 206 行 get_correspond_points()的第一個變數

- Package used and the environment:
  - Python: 3.9.7
  - open3d: 0.15
  - numpy: 1.20
  - opencv: 4.6.0