3DCV HW2 Report

電子所碩一 R10943117 陳昱仁

一. Problem 1-1

1. Camera re-localization

這次實作的 camera re-localization 整體分為以下幾個步驟。

Pseudo code: camera re-localization

Step 1 找出每個 Point 的 model

Step 2 使用 KNN 以及 ratio test 找出適合的 2D-3D correspondences

Step 3 使用 PnPRanac 來回傳 camera poss

整體 work 如下:

Step 1: 實作了 Outliers removal 找出 model

Step 3: 實作了 P3P 以及 DLT 兩種方法來實現 PnP

2. PnPRansac

Pseudo code: PnPRansac

Step1 取樣隨機點，若使用 P3P 取 4 點，DLT 則取 6 點

Step2 用選定的 method (P3P, DLT)得到 camera pose

Step3 求此 pose 的 inliers 數量，並記錄擁有最多 inliers 的 pose

Step4 回到 Step1 直到疊代次數夠多

Step5 最後欲求的 pose 為擁有最多 inliers 的 pose

在本次資料中，存在不同 point，WCS 座標相同情形，若在 Step1 遇到此情形，形同點數減少不滿足點數，則重新取點。

3. P3P

使用四組對應點，運用上課講義的方式求解。

Pseudo code: P3P

Step1 使用 cv2.undistortPoints 來轉成 ccs 中在 image plane 上的點

Step2 由 WCS 及 CCS 座標求出係數並求方程式的根(多解或無實數解)

Step3 由根找出長度 a, b, c 以及 CCS 中 optical center (多解)

Step4 由 optical center 找出可能的 camera pose

Step5 在找出的 pose 中，用第四個點從 WCS 投射到 CCS 並找出最佳解

4. DLT

使用六組對應點

Pseudo code: DLT

Step1 使用 cv2.undistortPoints 來轉成 ccs 中在 image plane 上的點

Step2 將六個點的矩陣排列好

Step3 解 SVD 得到 P 矩陣

Step4 使用 singular value 得到 scale

Step5 用 sacle 得到最後的 camera pose

5. Outliers Removal

觀察資料可以得知，同個 Point ID 會出在多個 image 上，原先範例 code 是將 training image 上同個 Point ID 所有的 SIFT 取平均值，當作此 Point ID 的 model，再用 knn 找 match。但由下圖可知，紅框處明顯在此維度和其他 image 上差異甚大，將此 outliers 算入平均會使誤差加大。

	POINT_ID	XYZ	RGB	DESCRIPTORS
0	1	[1.6093346, -1.1848674, 1.610395]	[87, 87, 77]	[46, 43, 12, 11, 10, 5, 19, 37, 24, 16, 8, 9, ...
1	1	[1.6093346, -1.1848674, 1.610395]	[87, 87, 77]	[39, 42, 34, 14, 15, 12, 13, 31, 29, 11, 8, 7,...
2	1	[1.6093346, -1.1848674, 1.610395]	[87, 87, 77]	[47, 57, 39, 12, 12, 11, 9, 20, 43, 26, 13, 7,...
3	1	[1.6093346, -1.1848674, 1.610395]	[87, 87, 77]	[38, 58, 39, 12, 11, 11, 13, 16, 35, 20, 12, 8...
4	1	[1.6093346, -1.1848674, 1.610395]	[87, 87, 77]	[32, 38, 31, 19, 15, 6, 11, 32, 28, 14, 6, 10,...
..	...	...	...	...
56	1	[1.6093346, -1.1848674, 1.610395]	[87, 87, 77]	[25, 41, 34, 24, 29, 14, 17, 23, 18, 15, 19, 2...
57	1	[1.6093346, -1.1848674, 1.610395]	[87, 87, 77]	[19, 41, 31, 26, 31, 14, 19, 23, 16, 16, 20, 3...
58	1	[1.6093346, -1.1848674, 1.610395]	[87, 87, 77]	[16, 33, 34, 28, 33, 14, 19, 22, 11, 22, 24, 2...
59	1	[1.6093346, -1.1848674, 1.610395]	[87, 87, 77]	[43, 37, 19, 11, 2, 2, 11, 43, 23, 14, 11, 11,...
60	1	[1.6093346, -1.1848674, 1.610395]	[87, 87, 77]	[43, 43, 29, 14, 8, 5, 15, 42, 22, 10, 8, 11, ...

因此實作 Outliers Removal，實作方法如下

Pseudo code: Outliers Removal

Step 1 求出 abs($x - x$.mean) 及 x.std

Step 2 求出每個 vector $\leq$ m $*$ x.std 個數，最多 128，最少 0

Step 3 若此 vector 128 維超過 80%滿足 step 2，判定為 inliers，否則 outliers

Step 4 將所有 inliers 取平均得到新的 model

二. Problem 1-2

1. Experiments and Results

誤差除了題意要求的 median error，也另外求出 mean error 和 max error。Median 較能反映該法大致上表現，若某幾個 frame 誤差特別大，median 也不會相差太遠，仍能當作判定基準，但若某幾個 frame 誤差大，會反映在 mean error 上，因此用 mean 可以看出是否有幾次做的誤差較大，max error 則是找出所有結果誤差最大的 error，在 part1-3 若有某個 frame 偏離嚴重，在畫出的軌跡圖會明顯看出偏離，在此可直接用 max error 直接看出。由左至右分別為 median/mean/max error。

比較不同方法之間的 error，比較的 model 如下
P3P: P3P + RANSAC，有考慮 camera distortion
DLT: DLT + RANSAC，有考慮 camera distortion
OpenCV: cv2.solvePnPRansac，有考慮 camera distortion
Distortion: P3P + RANSAC，不考慮 camera distortion

觀察下表實驗結果，和預期相同，OpenCV 誤差及時間都是最好的結果。P3P 和 DLT 相比有考慮到 geometry constraint，在所有參數都未改的情形，P3P 誤差皆比 DLT 還來得低，但時間較久，可以認為 P3P 是犧牲時間來換取精準度。另外同樣為 P3P，未考慮 camera distortion 誤差也比 P3P 來得大不少。

	Trans error	Rotate error	Time
P3P	0.0079 / 0.0127 / 0.1167	0.0028 / 0.0040 / 0.0264	332s
DLT	0.0765 / 0.2716 / 4.0375	0.0200 / 0.0617 / 0.9468	231s
OpenCV	0.0001 / 0.0002 / 0.0032	0.0000 / 0.0001 / 0.0003	185s
Distortion	0.0224 / 0.0328 / 0.2454	0.0050 / 0.0264 / 1.2252	x

2. Reject outliers results

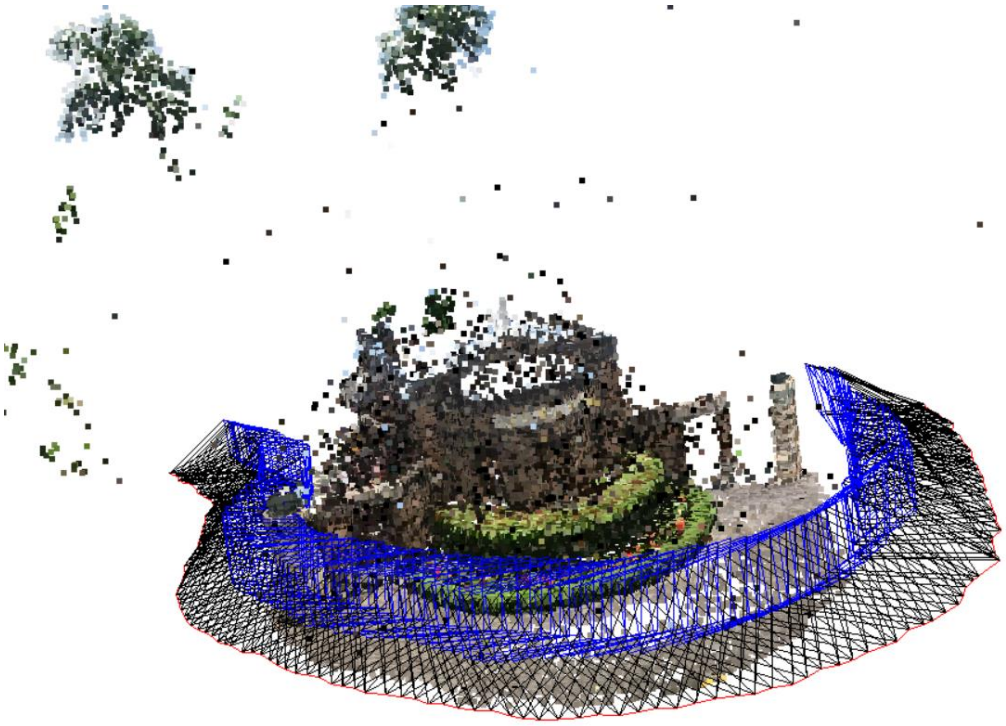
下表為實驗結果，由結果可知，此演算法在 m = 3 的情形，有使用 Outliers removal 下所有結果皆比沒使用此法好，結果和預期相同，最後繳交的版本是有 Reject outliers。

P3P	Trans error	Rotate error
Reject outliers	0.0079 / 0.0127 / 0.1167	0.0028 / 0.0040 / 0.0264
Include outliers	0.0096 / 0.0144 / 0.1548	0.0030 / 0.0044 / 0.0264

三. Problem 1-3

繪製軌跡圖需由 part1-1 算出的 pose 求出，且需要考慮 distortion 的情形，求出每個 frame CCS 中五個點在 WCS 的座標，分別是 image plane 四個 corners，以及 CCS 座標原點(optical center)。

因知道 image size 是(1920, 1080)，CCS corners 座標可求，轉換得到 WCS 座標後。再將點之間相連。為了方便辨識，image plane 用藍色表示，CCS 座標到四個 corners 連線用黑色線，trajectory line 使用紅色線表示，結果如下圖。



四. Problem 2

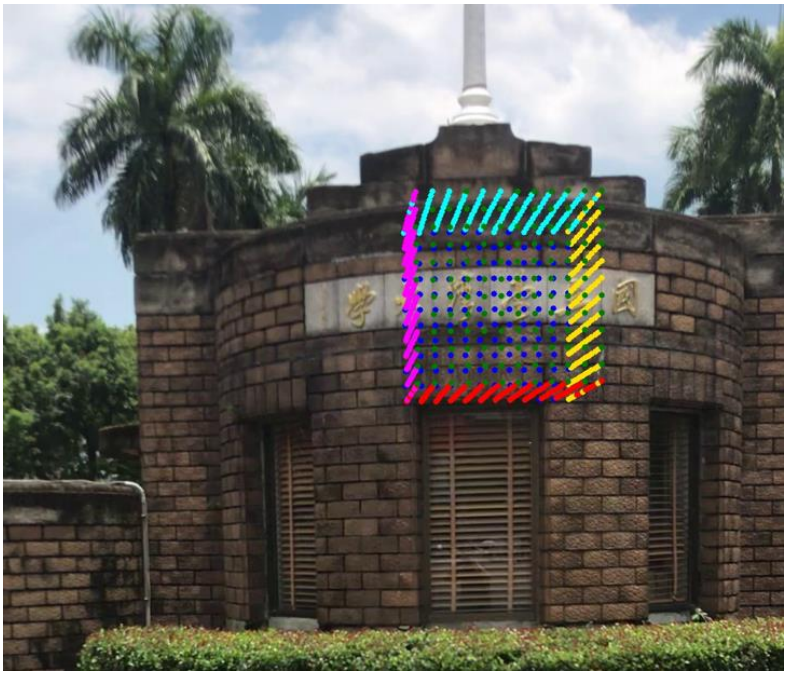
預先得出 cube points 在 WCS 座標，每張 frame 畫出 cube 的演算法如下

Pseudo code: Painter's Algorithm

Step 1 將 cube points 在 WCS 中由 Z 軸遠至近排列

Step 2 由遠至近，將點轉成 pixel 座標，若沒超出圖片範圍則畫在圖上

此演算法較遠的點會先畫，較近的點會最後畫出，因此在圖片上，若有不同深度但 pixel 座標相同的情況，只有較近的點會在圖上呈現。使用了 cv2.circle 畫圓，整體得到如下圖的效果。最後再將所有 frame 轉成 gif。



五.結果與討論

這次的實作除了基本的 P3P+Ransac，還實作了 DLT 以及 Outliers Removal 來實驗比較效能。需注意的事項如下

在 part1-3 以及 part2 需要作圖的部分，需要注意的是，valid image 並沒有按照順序排好，需要額外處理排序的問題。

實作的 Outliers removal 若 m 的值太小反而會去掉太多 inliers 導致 error 更大，m 太大等同所有點都為 inliers，最後實驗出 m = 3 表現最佳。

P3P 講義作法是先平移在旋轉，要得到最後 R 矩陣，使用 R = - R.dot(T)