物件導向程式設計期末報告

基於特徵點法、光流法、直接法之影像機器人同步定位與建圖

學生：蔡承穎

系級：航太碩一控制組

學號：P46091204

Email：lung87328@gmail.com

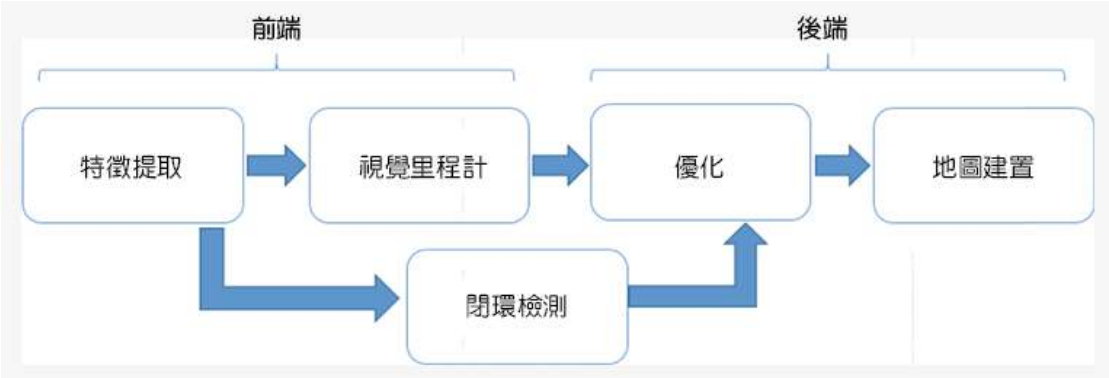
目錄

壹、報告主題摘要

一、研究動機

自主機器人在未知環境下作業，第一個需要解決的問題就是定位，也就機器人要不斷地反問自己現在大在這個環境的什麼位置，以及什麼姿態，然而建圖是定位而產生的產物，這個技術名就是為所謂的 SLAM(Simultaneous Localization and Mapping)(圖一)。

近年來，掃地機器人分為三派，分別為 Lidar based、Camera based、以及兩者皆有的，Lidar 傳回來的數值是方位與距離，然而相機所拍的照片能做的事情就較多，根據電腦視覺與深度學習就能建立出語意地圖(圖二)，故能分辨這個障礙物是否能避開，或是可穿越(圖三)。



圖一、SLAM 的基本架構

圖二、語意地圖

圖三、可穿越的障礙物

二、研究目的

因為是 Camera based，所以這也是所謂的 Visual SLAM 又名 v-SLAM，然而現今的 SLAM 方法可以大致分為三大類，特徵點法、光流法、直接法，各有優缺點，故打算寫出一個 GUI 介面，輸入可以是影片或自身相機，並且比較三種 SLAM 方法的 RMSE，故可判定其準確率為何，甚至可以先將影像預處理，例如 CLAHE+ORB-SLAM2[3]，其中三種方法分別為: ORB[1]、LK、DSO[2]。

三、部分完成的成果

目前完成的部分為(1)相機內參的校正、(2)ORB、LK 的 Feature matching、(3)ORB+SolvePnP+Bundle Adjustment、(4)Draw Trajectory and 3D point。

另外、尚未完成的部分為 LK+SolvePnp+Bundle Adjustment、DSO 的建立。

關鍵字：v-SLAM、CLAHE、ORB、LK、DSO

貳、前言

一、ORB 特徵點

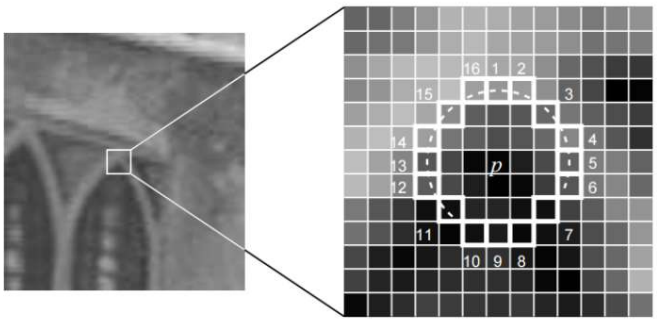
特徵點法需要兩個步驟，一為尋找關鍵點，二為描述子描述。

(一) FAST 關鍵點

主要檢測局部像素灰階變化明顯的地方，流程為(1)選取像素 p，亮度I_p。(2)設定一個設定值 T。(3)以像素為中心選取半徑為 3 的圓上的 16 個像素點。(4)圓上有連續的 N 個點亮度大於$I_p + T$或小於$I_p - T$，就可能是特徵點，N 通常取 12 (圖三)。

(二) BRIEF 描述子

將關鍵點計算描述子，BRIEF 是一種二進位的描述子，是利用 1、0 來描述，比如說關鍵點附近的兩個像素，例如 p、q，如果 p 比 q 大取一，反之取 0，如果取了 128 個 p、q，就會得到 128 維的 0、1 組成向量，故當要做特徵匹配時，計算其 Hamming distance。



圖三、FAST 關鍵點

二、光流法

Lucas-Kanade 光流法是 Sparse optical flow，對灰度作一階泰勒展開可得(1)，接著我們假設灰度不變可得(2)

$$I(x+dx,y+dy,t+dt)\approx I(x,y,t)+\frac{\partial I}{\partial x}dx+\frac{\partial I}{\partial y}dy+\frac{\partial I}{\partial t}dt-(1)$$

$$I(x+dx,y+dy,t+dt)=I(x,y,t)-(2)$$

將上式作個整理可得

$$\frac{\partial I}{\partial x}\frac{dx}{dt}+\frac{\partial I}{\partial y}\frac{dy}{dt}=-\frac{\partial I}{\partial t}-(3)$$

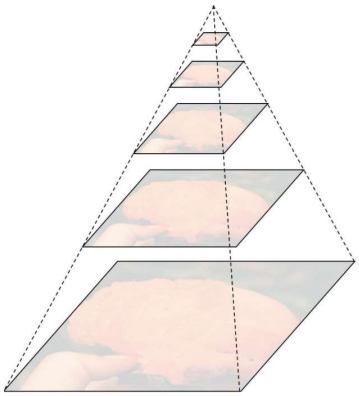
考慮一$w\times w$視窗，該視窗內像素具有同樣的運動

$$\begin{bmatrix}I_x & I_y\end{bmatrix}_k\begin{bmatrix}u\\v\end{bmatrix}=-I_{t_k},k=1,...,w^2-(4)$$

由最小平方法可得

$$\begin{bmatrix}u\\v\end{bmatrix}^*=-(A^TA)^{-1}A^Tb-(5)$$

倘若相機運動過大可利用影像金字塔概念解(圖三)



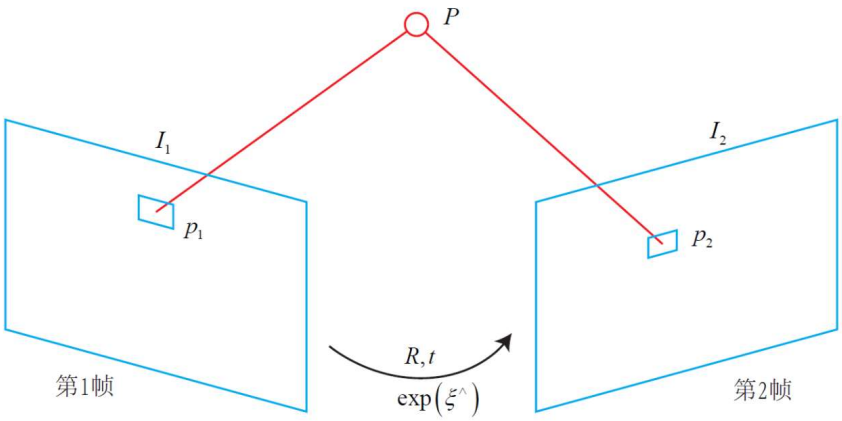
圖三、FAST 關鍵點

三、直接法

考慮 P 點在兩個相鄰 Frame 的投影，可得式(5)

$$p_1 = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_1 = \frac{1}{Z_1} KP$$

$$p_2 = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_2 = \frac{1}{Z_2} K(RP + t) = \frac{1}{Z_2} K(TP)_{1:3} \qquad \text{-(5)}$$



圖四、相鄰 Frame 的投影

由於不是特徵比對，代表不是要最小化重投影誤差，而是光度誤差，也就是兩個像素的亮度誤差(6)(7)

$$e = I_1(p_1) - I_2(p_2) \text{-(6)}$$

$$\min_{\xi} J(\xi) = \| e \|^2 \text{-(7)}$$

能做這種優化的前提，是基於灰度不變

在直接法中，假設一個空間點在各視角下，成像灰度是不變，比如有 N 個空間點 P_i

整個相機位姿估計問題變成(8)

$$\min_{\xi} J(\xi) = \sum_{i=1}^{N} e_i^T e_i \text{-(8)}$$

$$e(\xi \oplus \delta\xi) \approx I_1(\frac{1}{Z_1} KP) - I_2(\frac{1}{Z_2} K \exp(\xi^\wedge) P + \frac{1}{Z_2} K \delta\xi^\wedge \exp(\xi^\wedge) P)$$

$$q = \delta\xi^\wedge \exp(\xi^\wedge) P = TP$$

$$u = \frac{1}{Z_2} Kq \qquad \text{-(9)}$$

根據(9)式，q 為 P 在第二個相機坐標系下的座標，u 則為他的像素座標

$$e(\xi \oplus \delta\xi) \approx I_1(\frac{1}{Z_1}KP) - I_2(\frac{1}{Z_2}K\exp(\xi^\wedge)P + u)$$

$$\approx I_1(\frac{1}{Z_1}KP) - I_2(\frac{1}{Z_2}K\exp(\xi^\wedge)P) - \frac{\partial I_2}{\partial u}\frac{\partial u}{\partial q}\frac{\partial q}{\partial \delta\xi}\partial\xi \text{ -(10)}$$

$$= e(\xi) - \frac{\partial I_2}{\partial u}\frac{\partial u}{\partial q}\frac{\partial q}{\partial \delta\xi}\partial\xi$$

(1) $\frac{\partial I_2}{\partial u}$ 為 u 處的梯度

(2) $\frac{\partial u}{\partial q}$ 為投影方程式關於相機坐標系下的 3D 點導數

(3) $\frac{\partial q}{\partial \delta\xi}$ 為轉換後的 3D 點對轉換的導數

$$J = -\frac{\partial I_2}{\partial u}\frac{\partial u}{\partial \delta\xi}$$

$$\frac{\partial u}{\partial \delta\xi} = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z^2} & -\frac{f_x XY}{Z^2} & f_x + \frac{f_x X^2}{Z^2} & -\frac{f_x Y}{Z} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} & -f_y - \frac{f_y Y^2}{Z^2} & \frac{f_y XY}{Z^2} & \frac{f_y X}{Z} \end{bmatrix} \text{ -(11)}$$

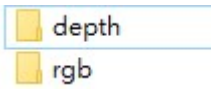
有了 Jacobian Matrix (11)，就能直接代 Gauss-Newton 法或 Levenberg-Marquadt 法，這兩個方法比較 Naïve，不贅述。

四、三種方法的比較

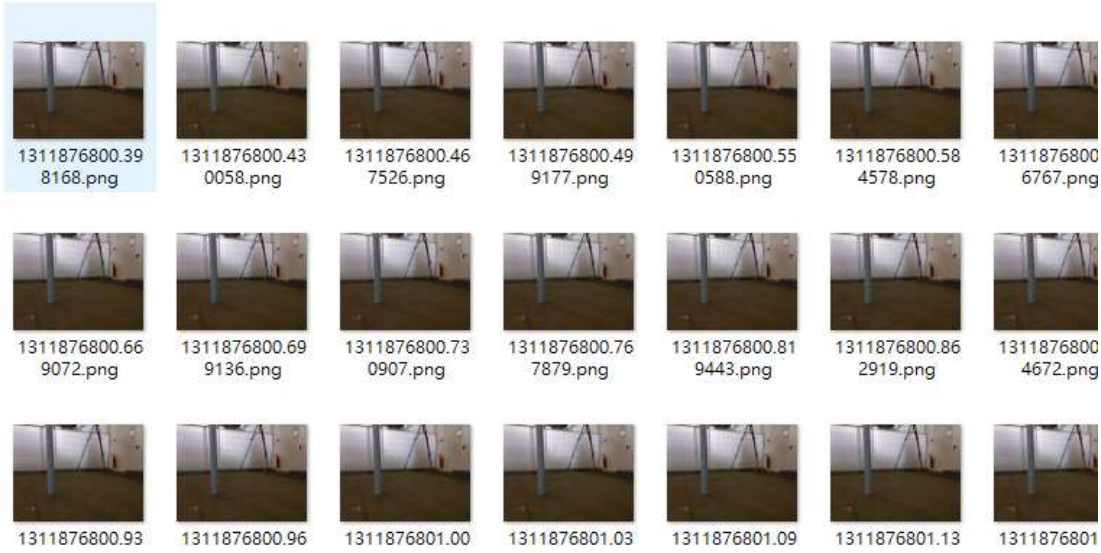
特徵點法	直接法	光流法
對於運動太大有魯棒性	只要關鍵點有梯度即可	不需要描述子、匹配特徵點
	取點建議500個點以上	關鍵點的多少都可以
	可以建立稀疏至稠密地圖	可以建立稀疏至稠密地圖
	非常快速，適合資源受限	
不宜特徵點過多過少	圖像無梯度，對優化無貢獻	灰度不變假設問題
花很多時間在計算描述子與匹配	灰度不變假設問題	
只能建構稀疏地圖		

表一、三種方法的比較

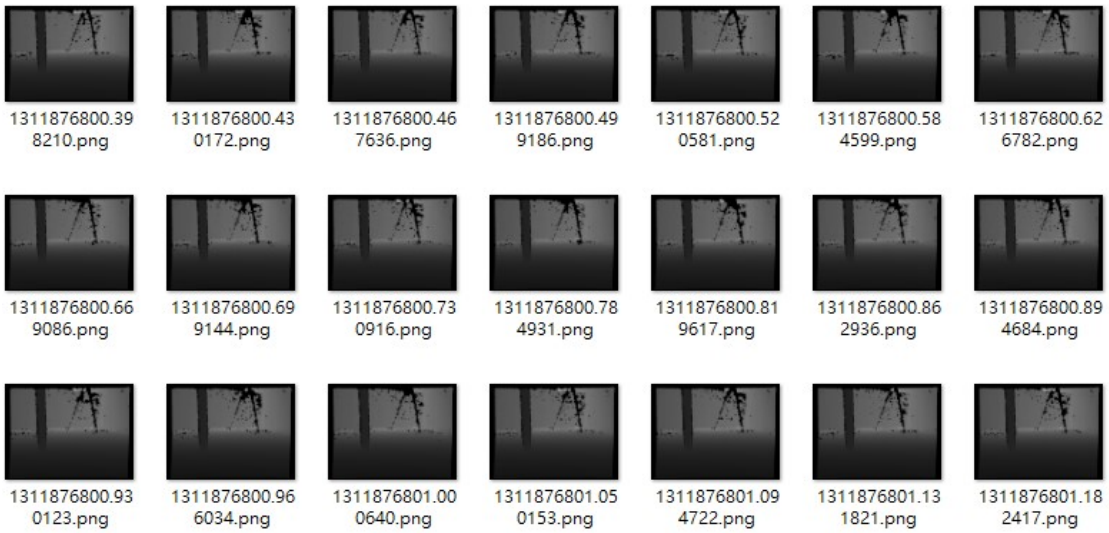
根據上表可知，三種方法在不同環境會有不同的效果，故希望使用者可以任意輸入所需的場景(圖五、六、七)。



圖五、RGBD 資料夾



圖六、RGB 圖



圖七、D 圖(深度圖)

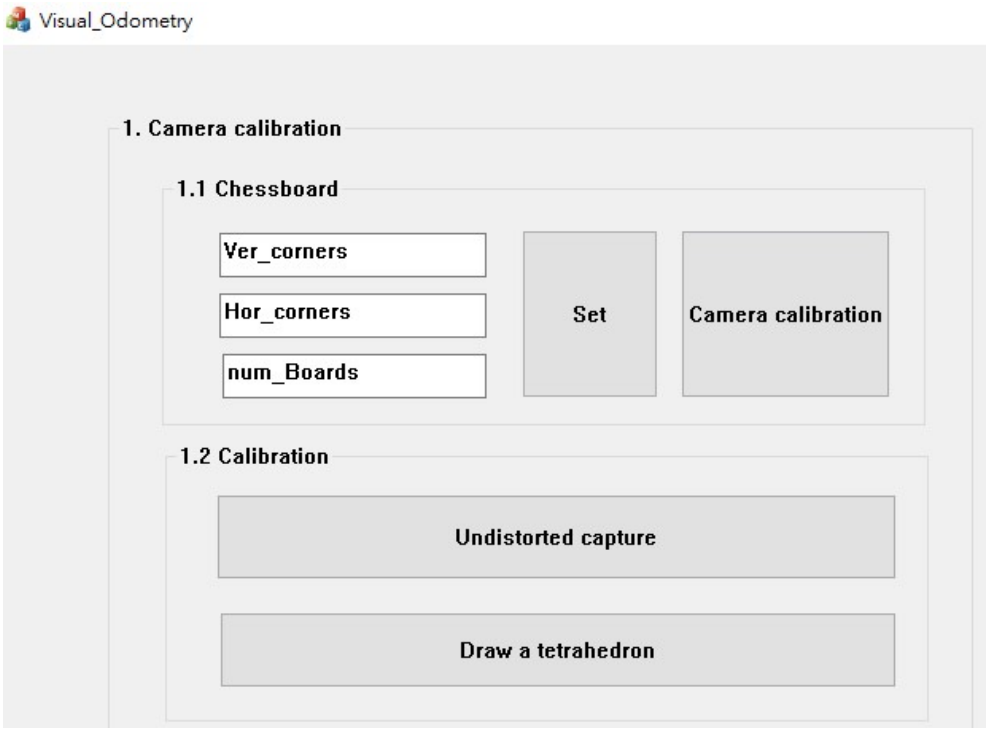
利用 SLAM 方法得到其位姿資料，而劃出軌跡圖，其中位姿表示位置加上姿態，這裡會用$t_x,t_y,t_z,q_x,q_y,q_z,q_w$來儲存計算結果，前三者為平移表示位置用，後四者為四元數表示姿態用。

參、程式說明

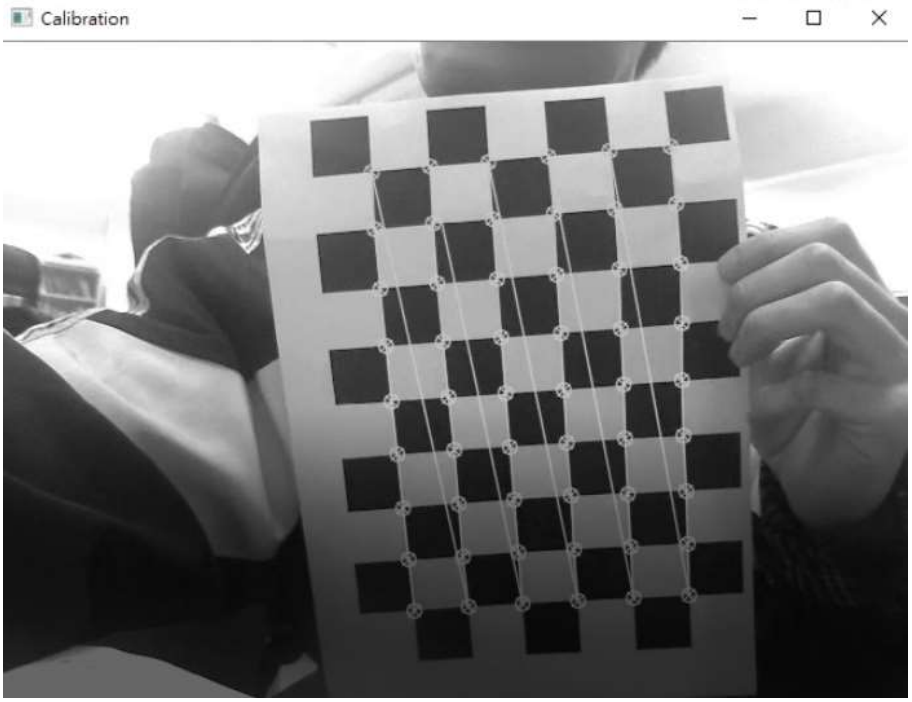
一、使用者介面與說明

(一) Camera calibration (圖八)

因為希望可以輸入自己的圖片，故可先將自己的相機做內參校正，這裡不贅述，直接使用 OpenCV::Find Chessboard 技巧即可(圖九)，也就是設定自己的 Chessboard 的 Corner 數，與打算用幾張照片。

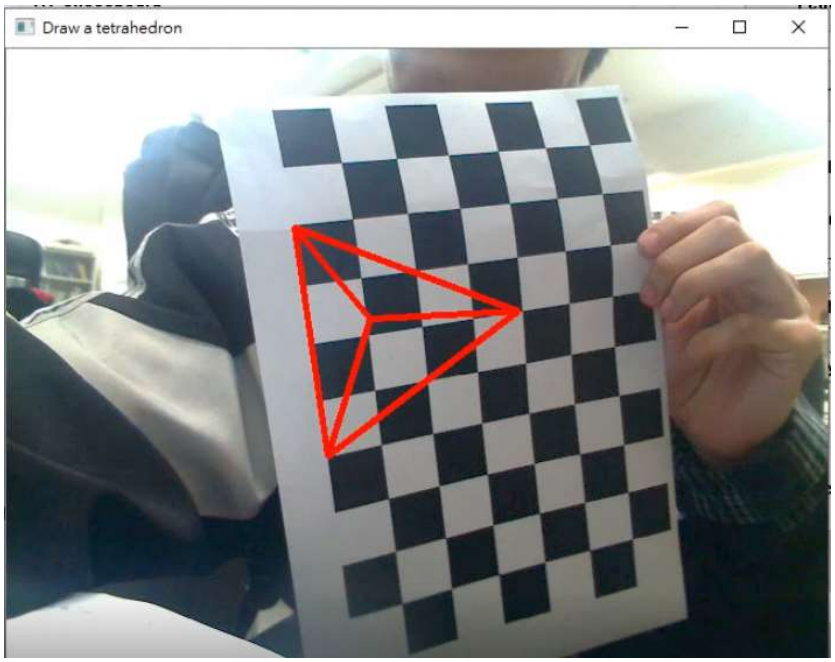


圖八、Camera calibration

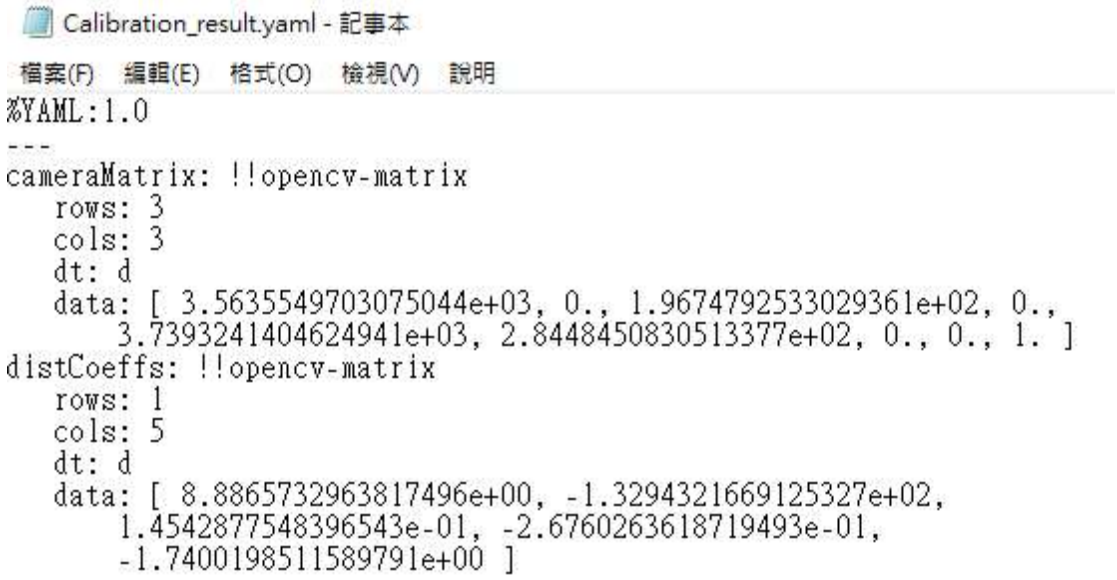


圖九、OpenCV::FindChessboard

倘若想知道校正的精確度多高，即可按下畫四邊形做個驗證(圖十)，結果若是好的，會自動儲存(圖十一)。



圖十、畫四邊形


Calibration_result.yaml - 記事本

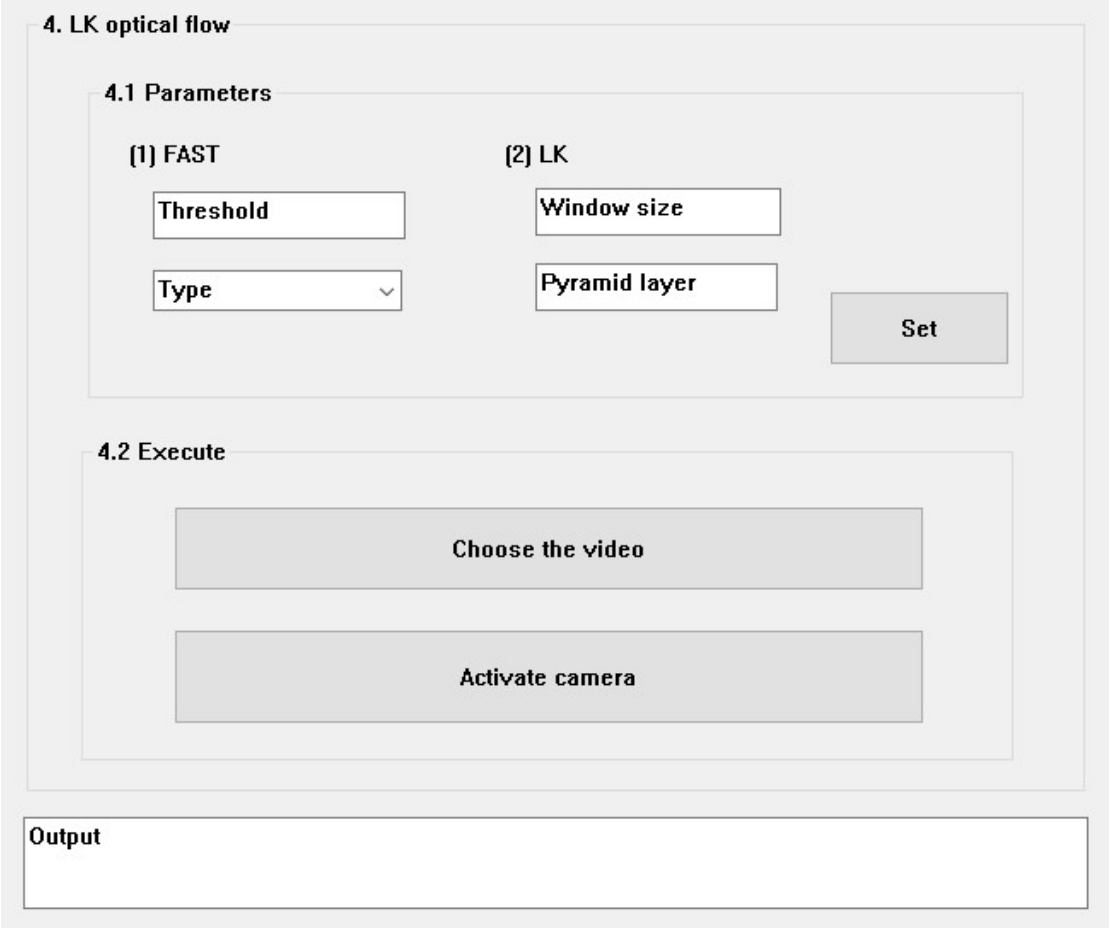
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
%YAML:1.0
---
cameraMatrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 3.5635549703075044e+03, 0., 1.9674792533029361e+02, 0.,
       3.7393241404624941e+03, 2.8448450830513377e+02, 0., 0., 1. ]
distCoeffs: !!opencv-matrix
   rows: 1
   cols: 5
   dt: d
   data: [ 8.8865732963817496e+00, -1.3294321669125327e+02,
       1.4542877548396543e-01, -2.6760263618719493e-01,
       -1.7400198511589791e+00 ]
```

圖十一、相機內參校正結果

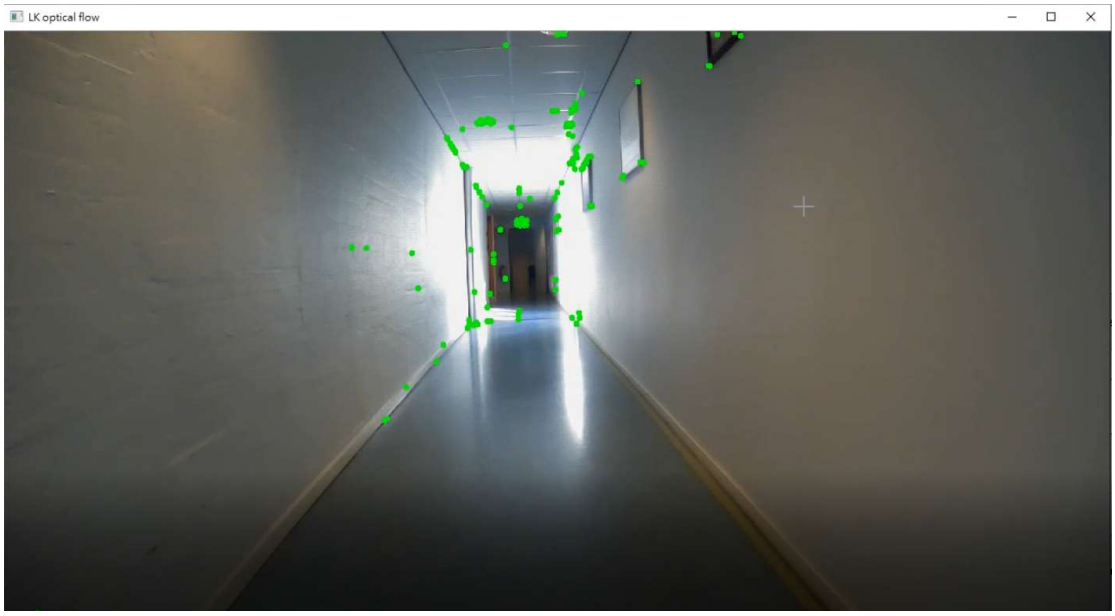
（二）光流法的參數設定(圖十二)

要使用光流法時，要設定的東西為兩大項，分別為 FAST 角點的 T 值與 N 點個數、Window size 大小與金字塔層數，在前面已經敘述過，這裡不贅述，

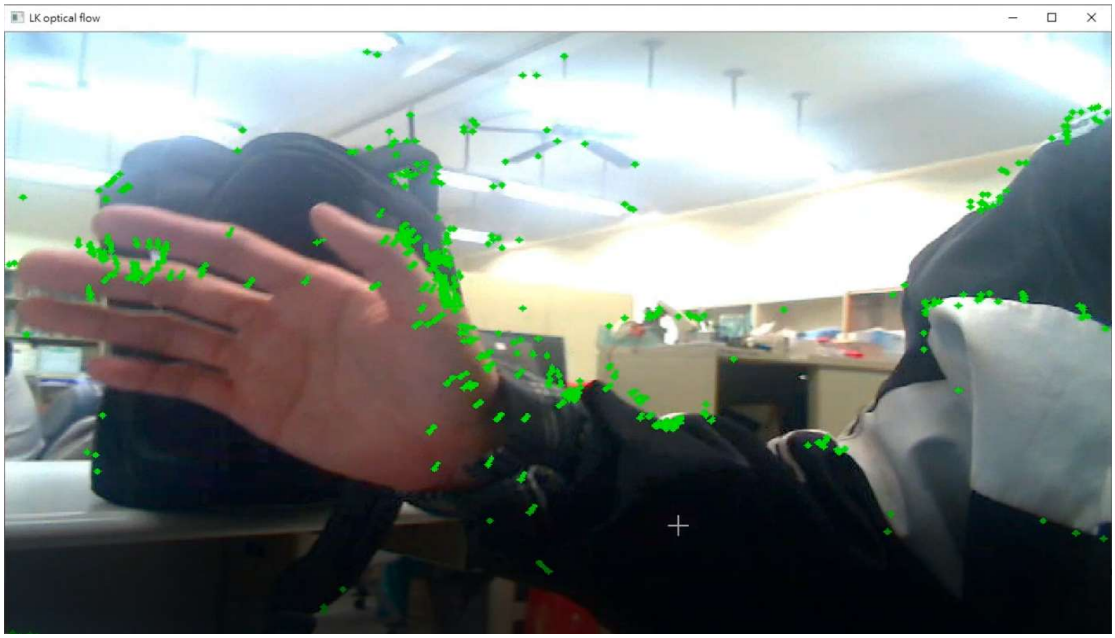


圖十二、光流法參數設定

設定完參數後，可以選擇影片觀察光流是否可使用在你的場景(圖十三)，或是選擇自己的相機(圖十四)。



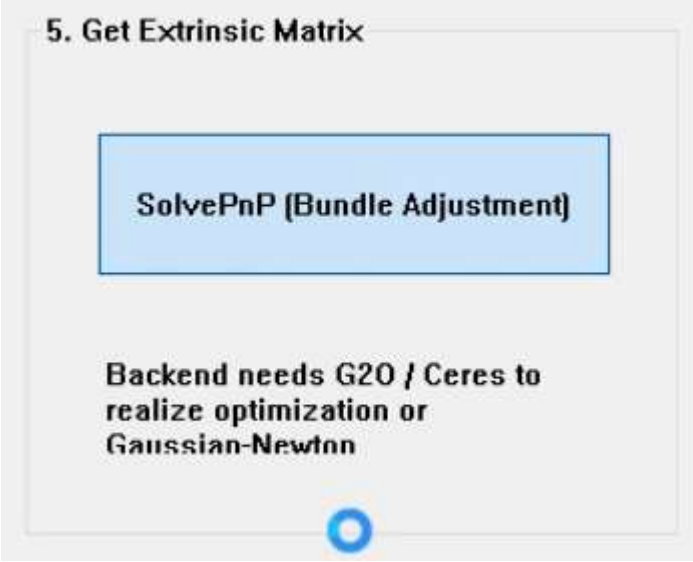
圖十三、光流法追蹤影片



圖十四、光流法追蹤相機

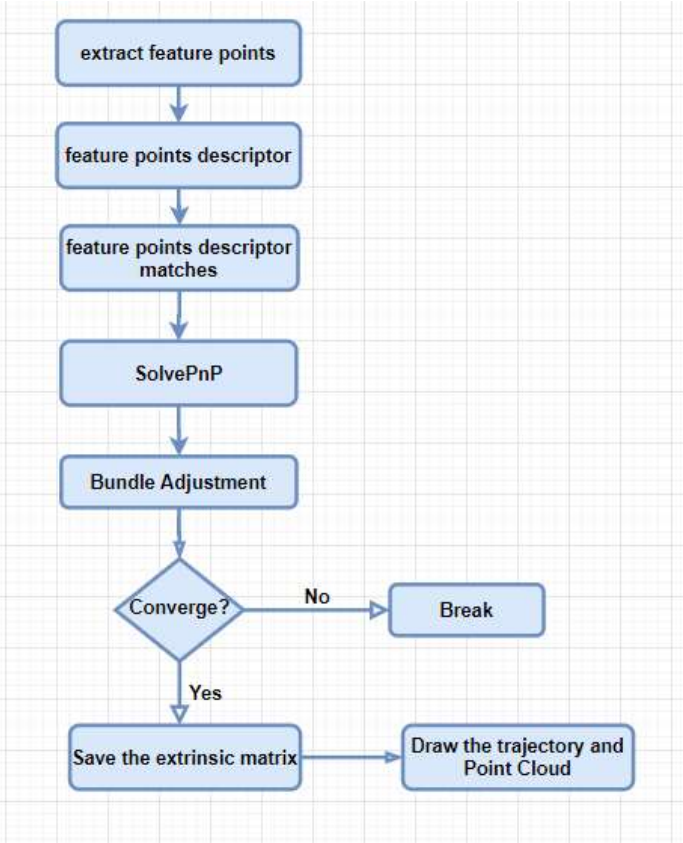
(三) SolvePnP + Bundle Adjustment

這裡我利用 OpenCV 的函式 SolvePnp 計算位姿(圖十五)，利用這個函式需要的是 3D 點搭配 2D 點(圖十六)，因為計算結果不一定是最佳解，可以搭配 Bundle Adjustment 解，因為這裡會牽扯到太多關於李代數與擾動模型，故不贅述，程式流程圖(圖十七)。



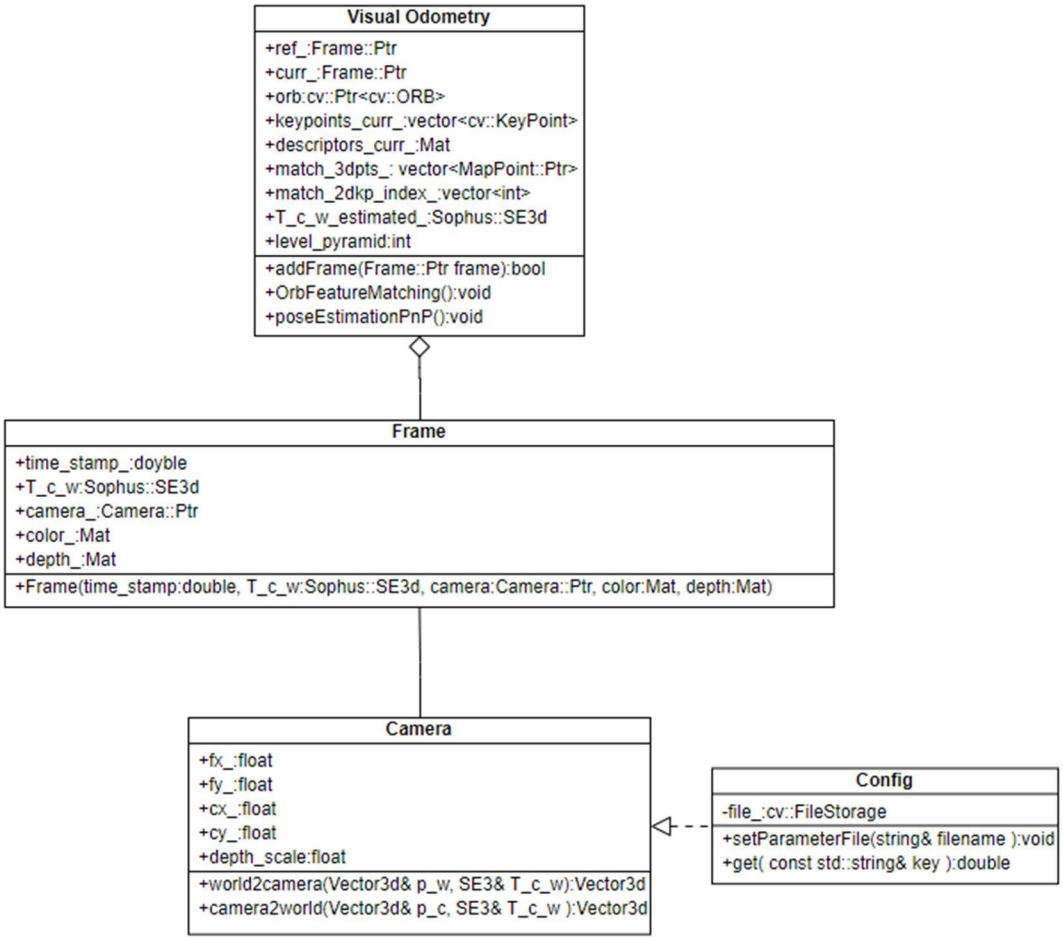
圖十五、計算位姿功能

圖十六、一張平面圖+深度圖與一張平面圖



圖十七、SLAM 流程

另外(圖十八)為 UML 圖，我們可以先寫個 Config 類讀取相機內參檔案，然後丟給 Camera 類，接著輸入很多個 Frame，相鄰的 Frame 丟給 Visual Odometry 類，計算其位姿後，做儲存，位姿以李代數做儲存，Sophus 李代數與 Eigen 矩陣是可以互轉的，這裡不贅述。

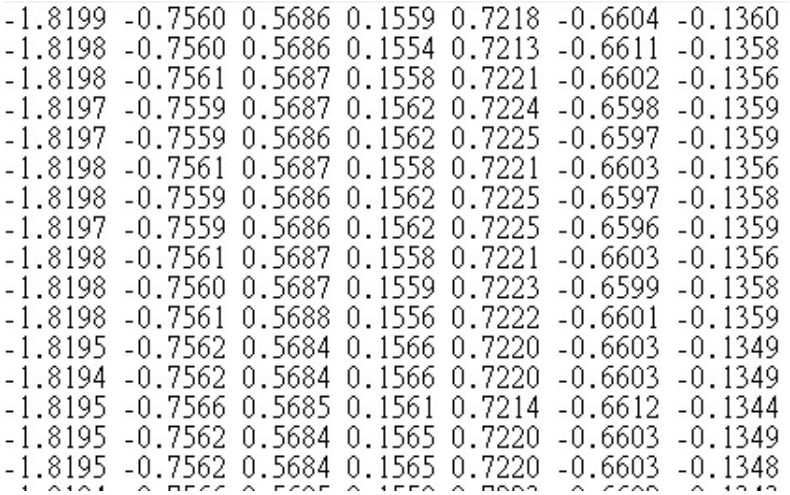


圖十八、SLAM 的 UML 圖

(四) 畫出軌跡圖

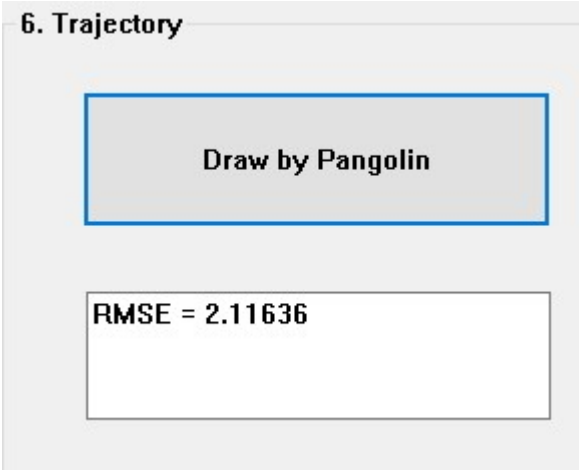
根據前面敘述，我們將計算結果儲存至 estimated.txt 檔(圖十九)，形式為$t_x, t_y, t_z, q_x, q_y, q_z, q_w$，相同的 Groundtruth.txt 檔(圖二十)。

圖十九、位姿計算結果

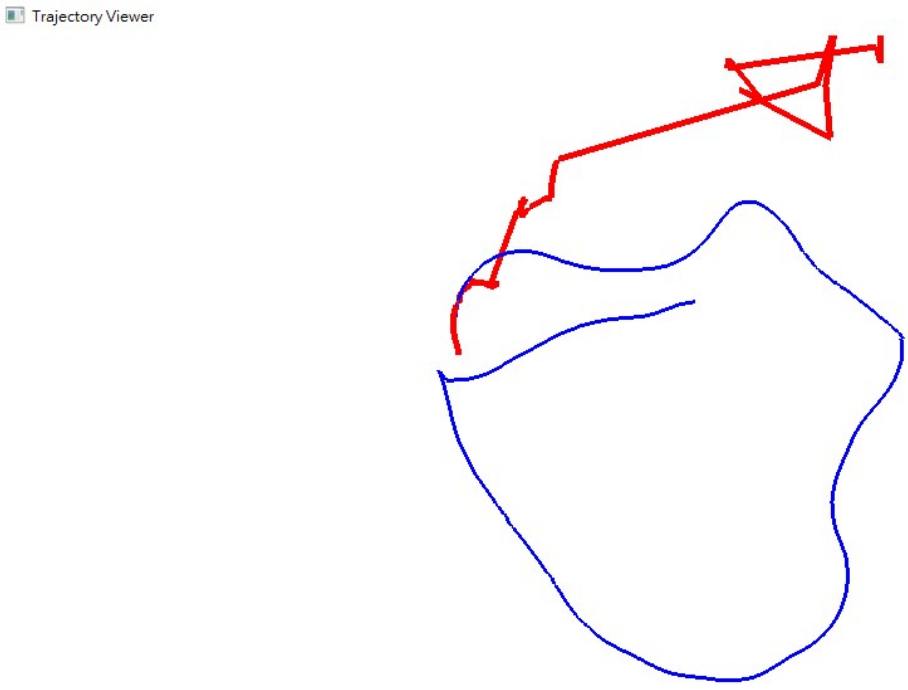


圖二十、Groundtruth.txt

接著可以按下 Draw by Pangolin 按鈕(圖二十一)，可得真實路徑與預測的圖形(圖二十二)，與 root mean square error，這裡因為特徵缺失緣故，以及優化位姿部分尚須改進，故誤差較大。



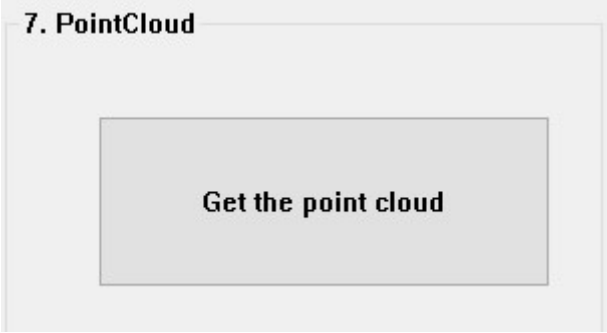
圖二十一、按鈕與誤差值



圖二十二、軌跡圖

(五) 畫出點雲

因為有照片與位姿，故可直接畫出點雲圖(圖二十三、二十四)。



圖二十三、按鈕

圖二十四、點雲圖

二、使用工具總覽

所需工具為 (圖二十五)：

(1) Sophus：計算李代數使用。

(2) Pangolin : 輕量級 OpenGL，用來畫軌跡圖與點雲圖。

(3) OpenCV：影像處理必要。

(4) glew：Pangolin 的依賴項。

(5) Eigen：計算矩陣使用。

圖二十五、所需工具

三、遭遇困難

SLAM 優化庫通常會使用 G2O，但它較多依賴項，加上功力不足，只能在 Linux 下建起來，無法在 Windows 環境下成功建立，故只能手寫 Gauss-Newton 做優化。

SolvePnP 僅限定 3D-2D，加上我寫的演算法只適用於 RGB-D 相機，故未來可以多加入 2D-2D、3D-2D、3D-3D，對極幾何解、SolvePnP 解、ICP 解，並且有三個按鈕分別是，單眼相機、雙眼相機、RGB-D 相機，這樣才可以適用所有使用者。

四、例外處理

倘若本身使用者沒有相機，按下 activate camera 會跳出 messagebox (圖二十六)

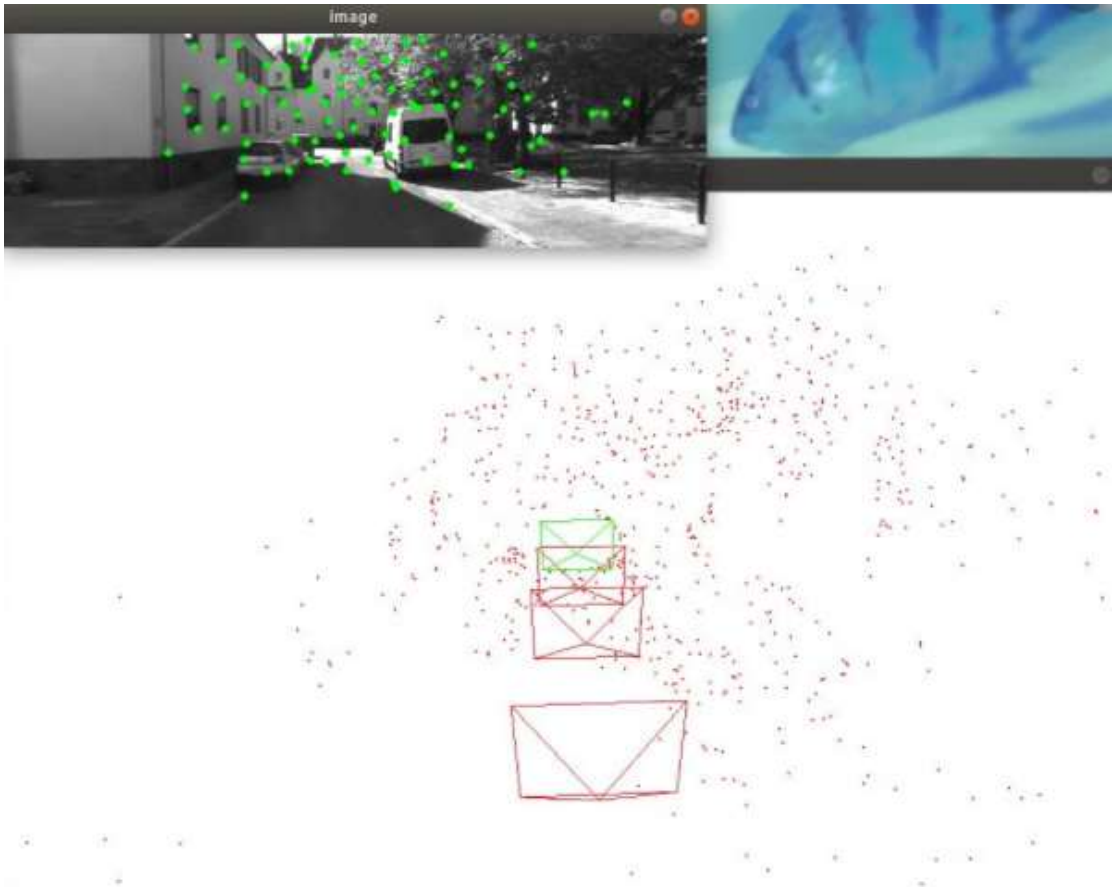
圖二十六、Error message

肆、結論及未來展望

(1) 目前程式只有 ORB+SolvePnP 計算位姿功能，並沒有實現 LK+SolvePnP 功能，未來可以加入之，並且與直接法做一個比較，

(2) 增加一個按鈕是可以選擇自己的相機的形式，單眼、雙眼、RGB-D，並且拍照片儲存後，直接畫出軌跡。

(3) 增加一個按鈕選擇資料夾，選擇照片與深度圖，畫出軌跡，甚至是影片呈現(圖二十七)，這裡我只有在 Linux 下實現。



圖二十七、光流法

伍、參考文獻

1. Mur-Artal, R., J.M.M. Montiel, and J.D. Tardós, ***ORB-SLAM: A Versatile and Accurate Monocular SLAM System***. IEEE Transactions on Robotics, 2015. **31**(5): p. 1147-1163

2. Engel, J., V. Koltun, and D. Cremers, ***Direct Sparse Odometry***. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018. **40**(3): p. 611-625.

3. Yang, W. and X. Zhai. **Contrast Limited Adaptive Histogram Equalization for an Advanced Stereo Visual SLAM System**. in 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). 2019.