Final project

ORB-SLAM & COLMAP

作業內容

錄製自己的影片，使用兩種不同的視覺定位方法(ORB-slam & COLMAP)
建立相機軌跡並比較。

1. 拍攝場地，COLMAP建立場地模型
2. 錄製一段影片，匯入COLMAP得到定位的軌跡
   這段影片的移動路徑必須是環形的，最後走回起點
3. 使用同一段影片直接在orb-slam下定位，得到orb-slam的定位軌跡
4. 以1.的colmap模型當作基準，把3. orb-slam的軌跡轉到COLMAP的坐標系，畫在同一張圖上
5. 計算軌跡起點和終點的誤差並比較
   為了計算實際上的誤差(ex.每公尺誤差多少公分)，要紀錄自己實際走的距離有多長

作業內容

繳交檔案：將以下檔案包成zip檔，命名為group_0.zip

1. 一組一份10頁以內的報告，轉成pdf檔
2. code
3. test video
4. 實驗過程影片(orb-slam跑的過程)

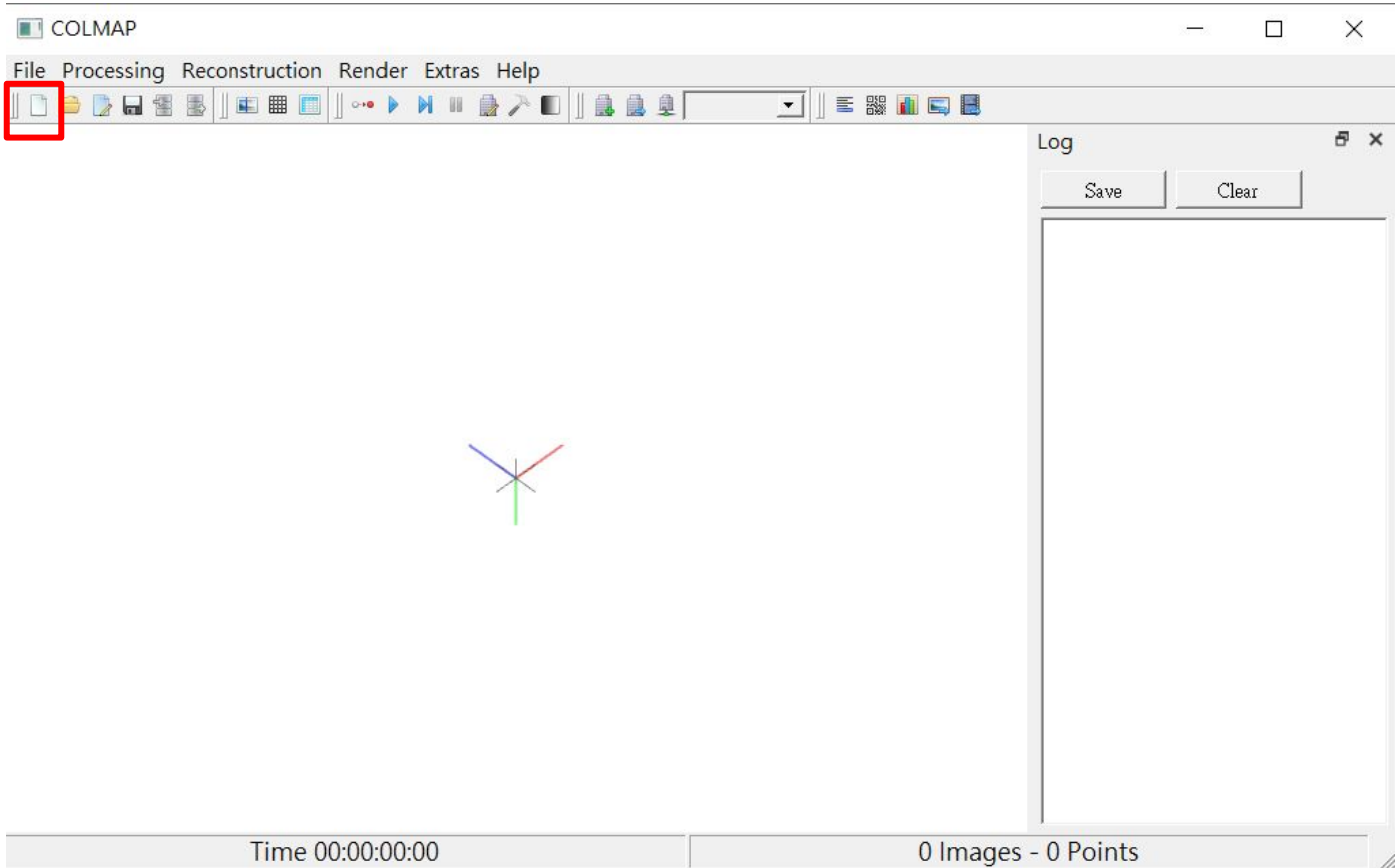
繳交期限：6/16(四) 晚上 23:59前

COLMAP

- https://github.com/colmap/colmap/releases?fbclid=IwAR38THauVythCkkbdUs4fcjv85muGyr34wHMIUqREK9v5dZrSsEKMHZbinQ
- COLMAP-3.6-windows-no-cuda.zip即為windows免安裝版
- 先建模型，再用測試影片重建相機位置

COLMAP

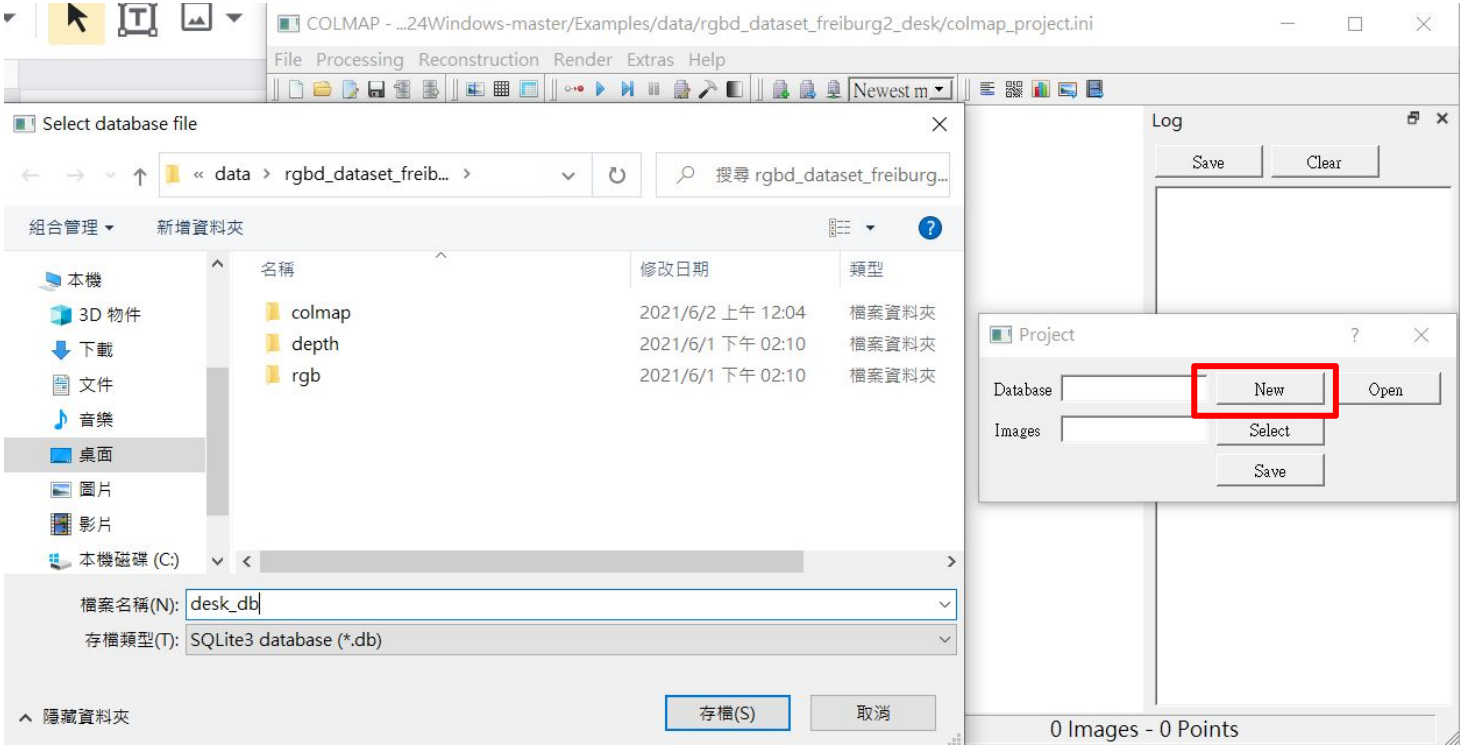
1. new project



COLMAP

2. Database

在要存的資料夾打一個名稱，會存成.db檔

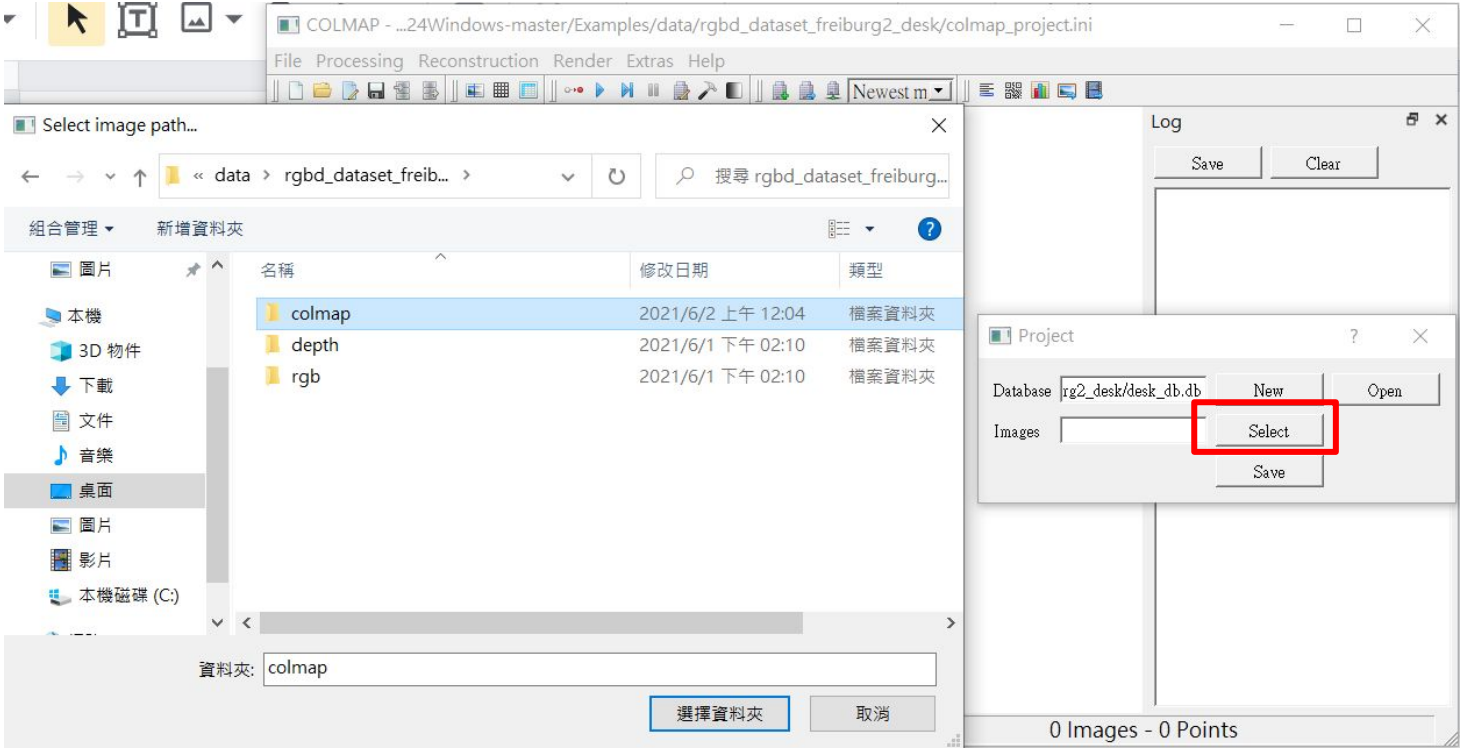


COLMAP

3. Image

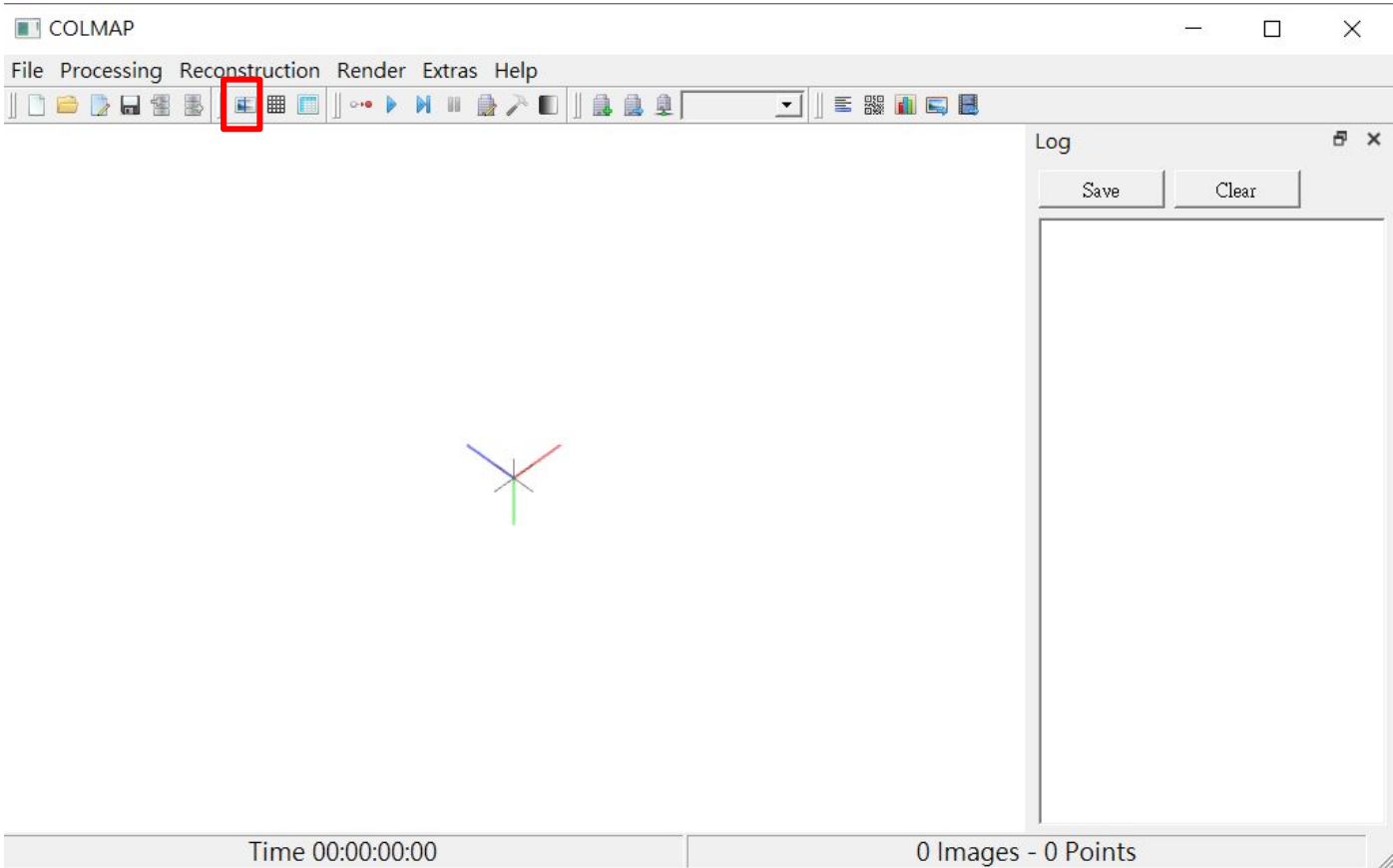
選擇輸入圖片資料夾

然後按Save



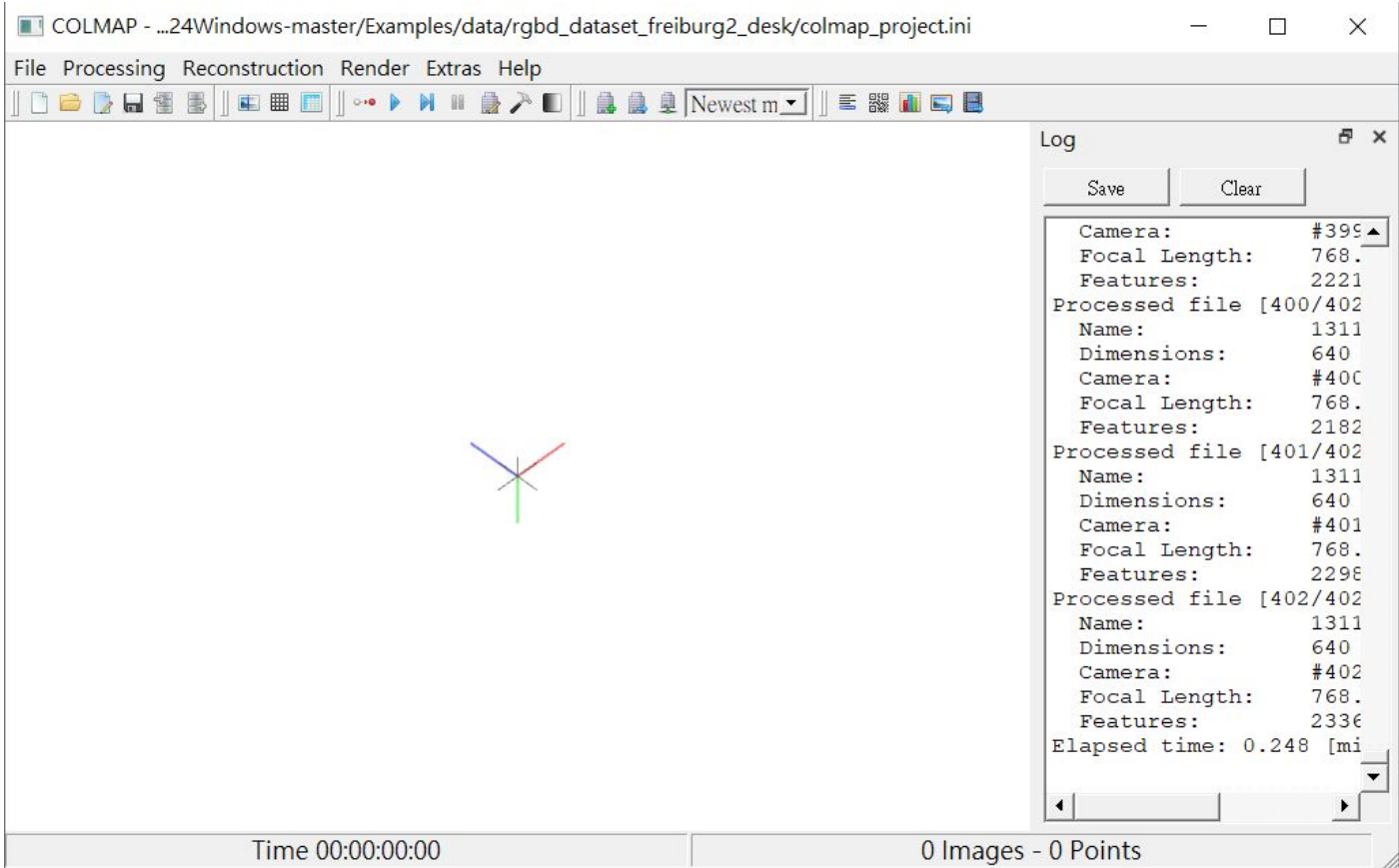
COLMAP

4. Feature Extraction



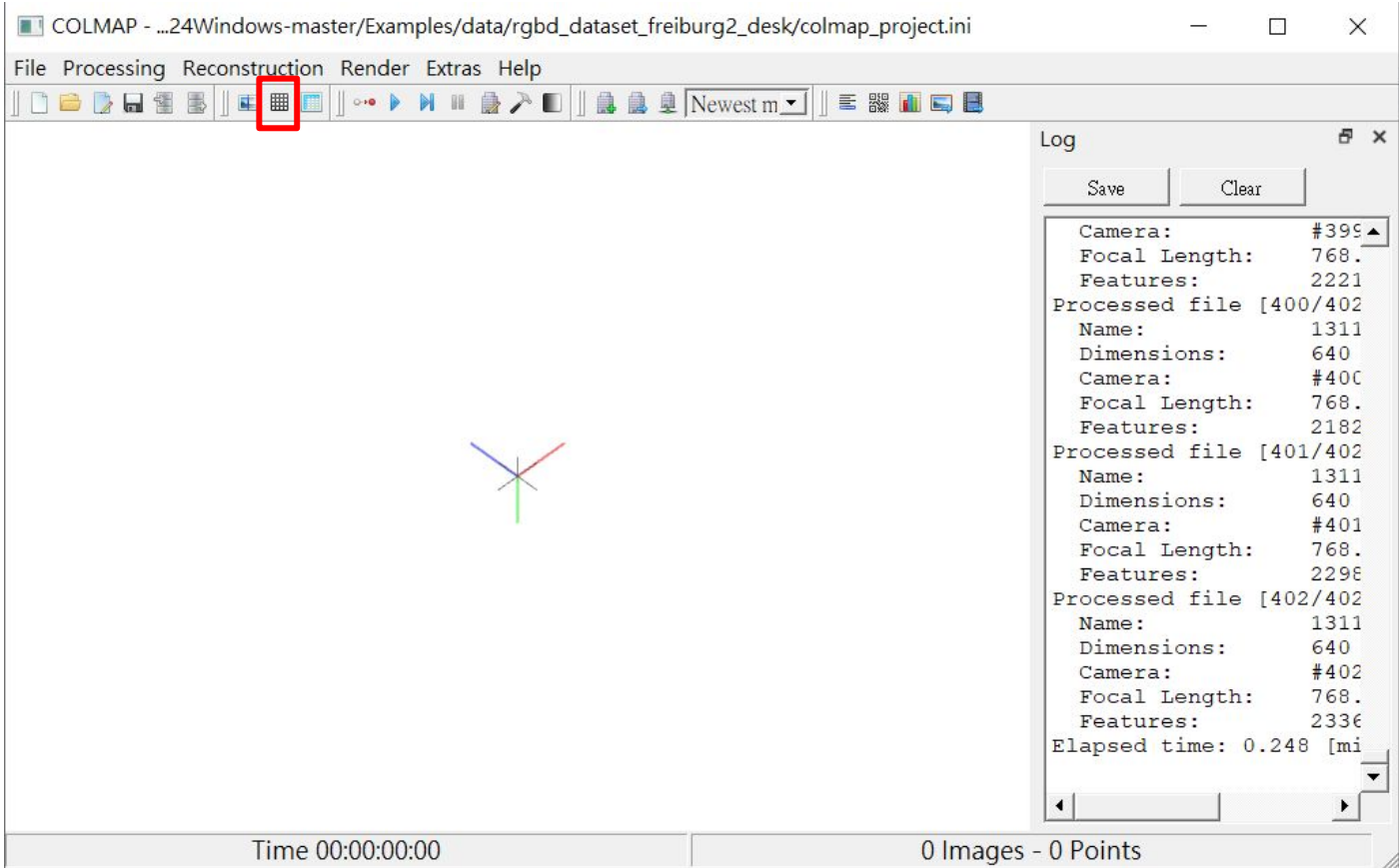
COLMAP

4. Feature Extraction



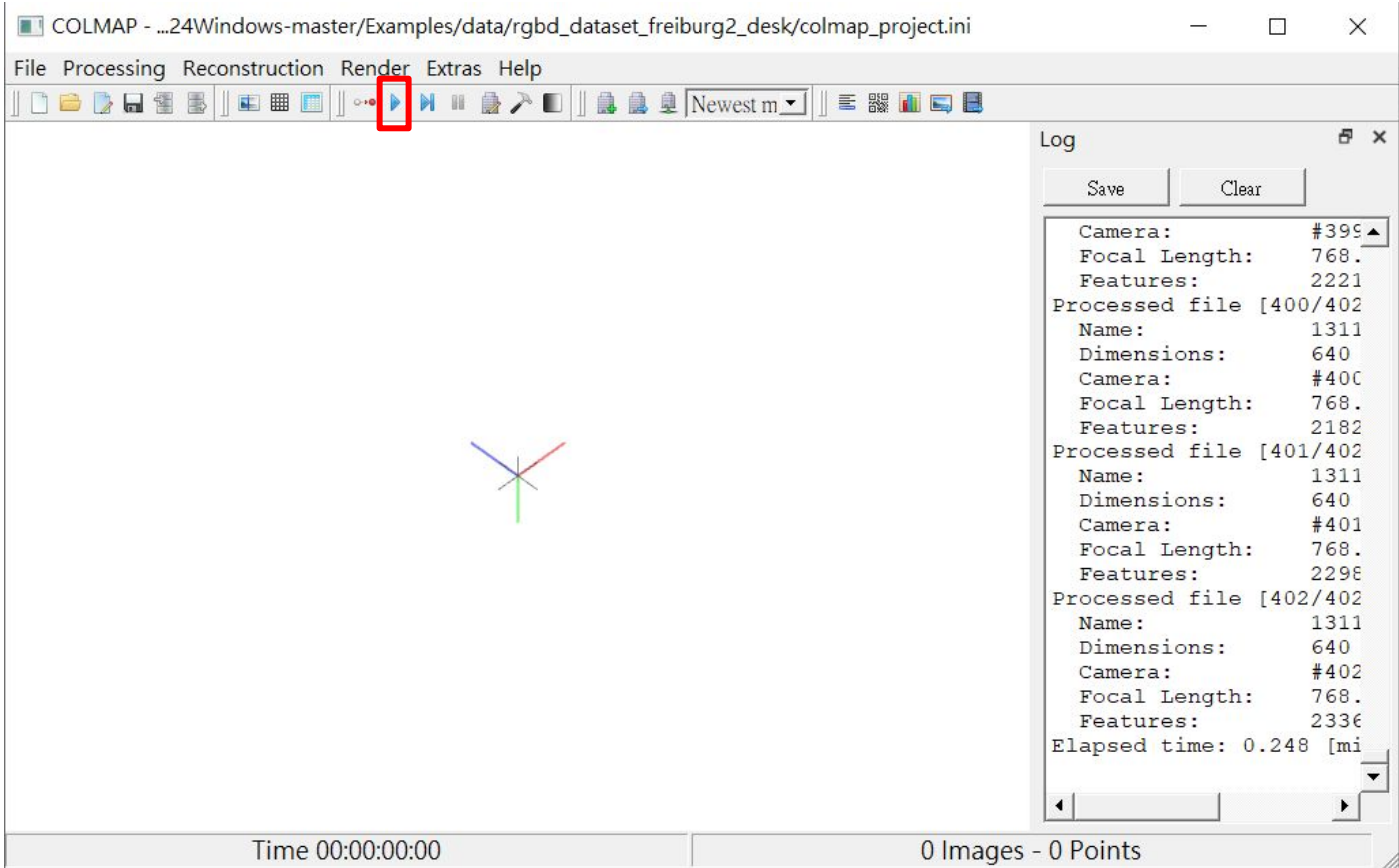
COLMAP

5. Feature Matching



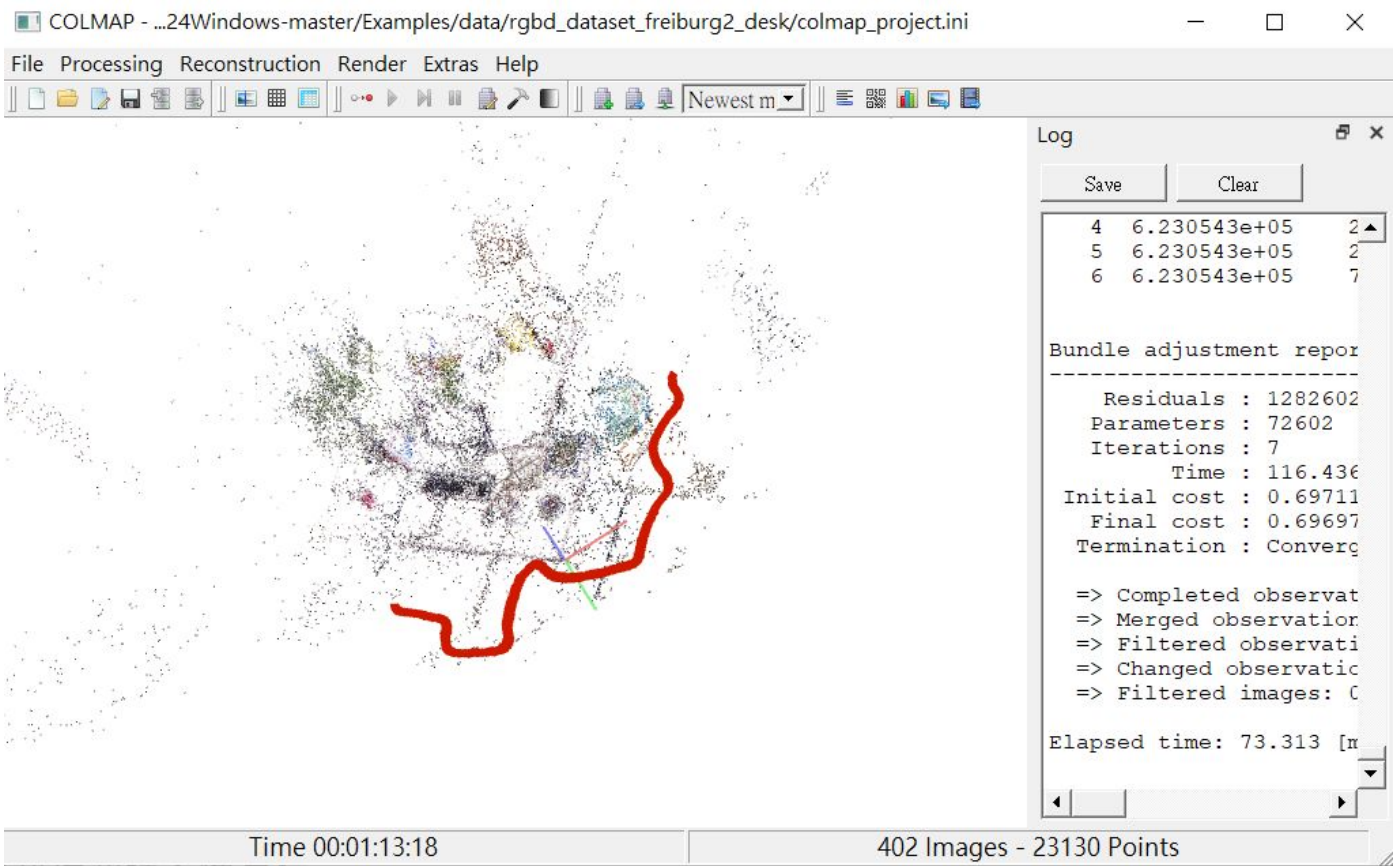
COLMAP

6. Start Reconstruction



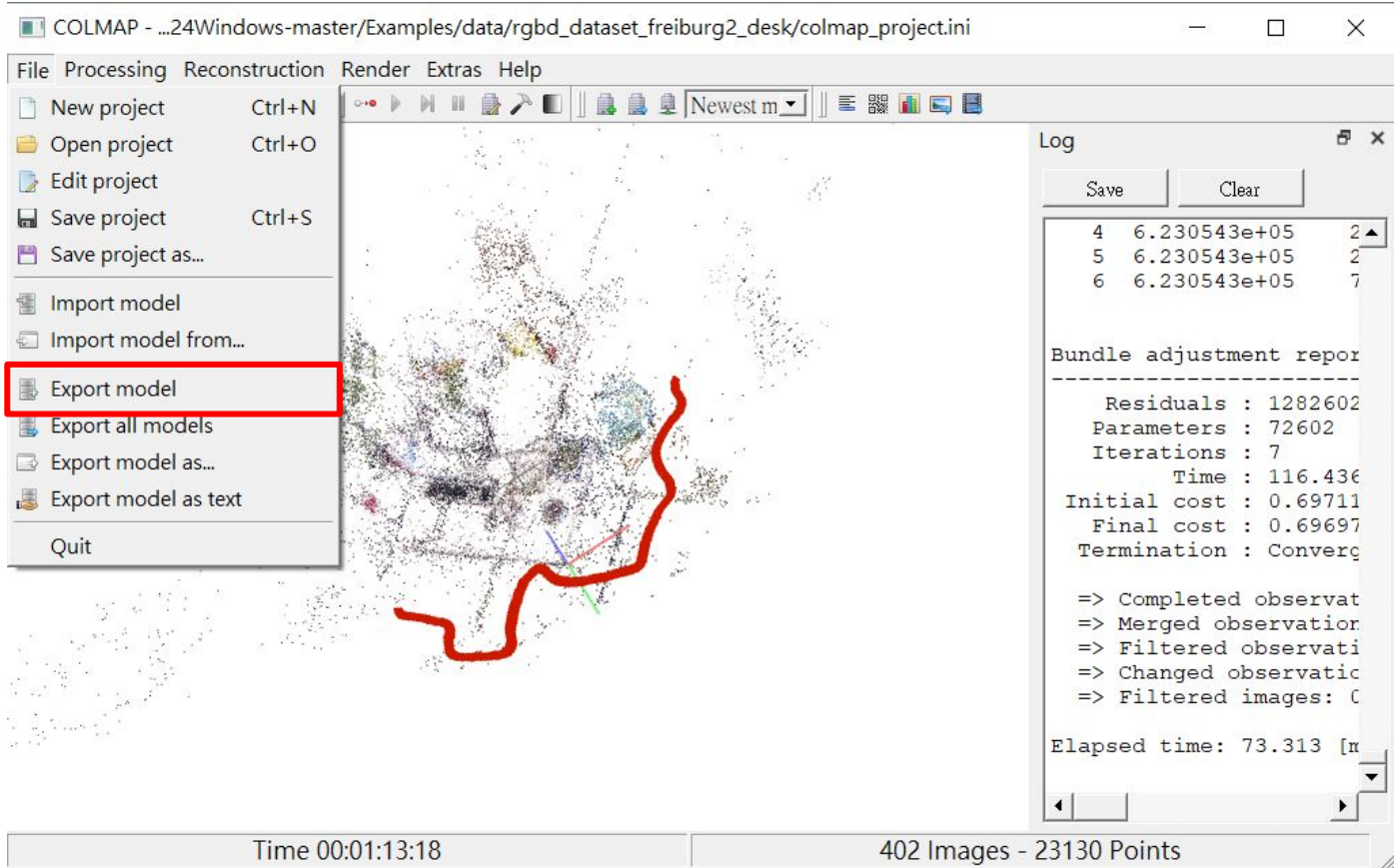
COLMAP

6. Start Reconstruction



COLMAP

7. Export model



COLMAP

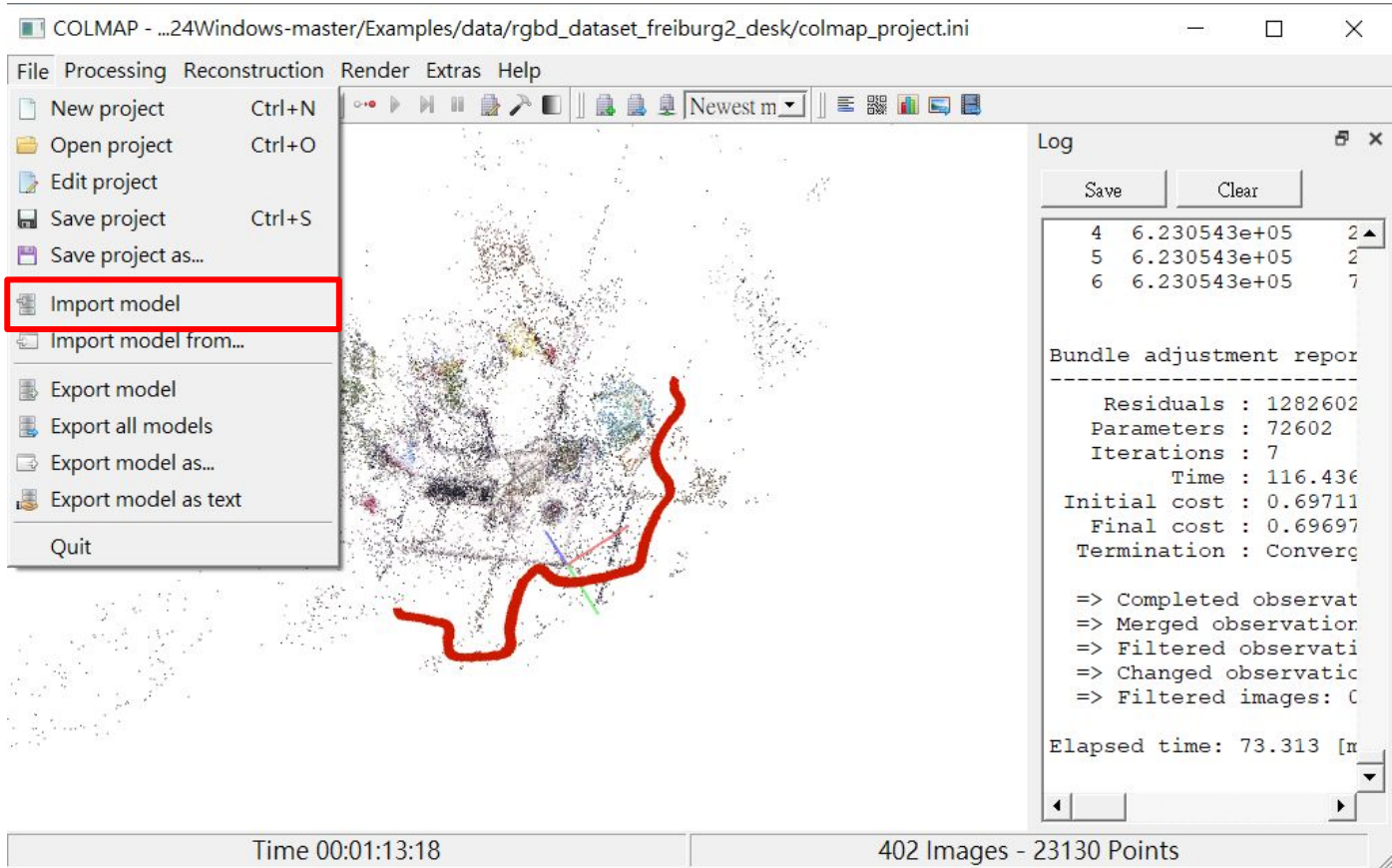
8. 把測試影像加進原始影像資料夾裡

可以分成兩個資料夾放

COLMAP

9. Import model

把剛剛Export的模型讀入



COLMAP

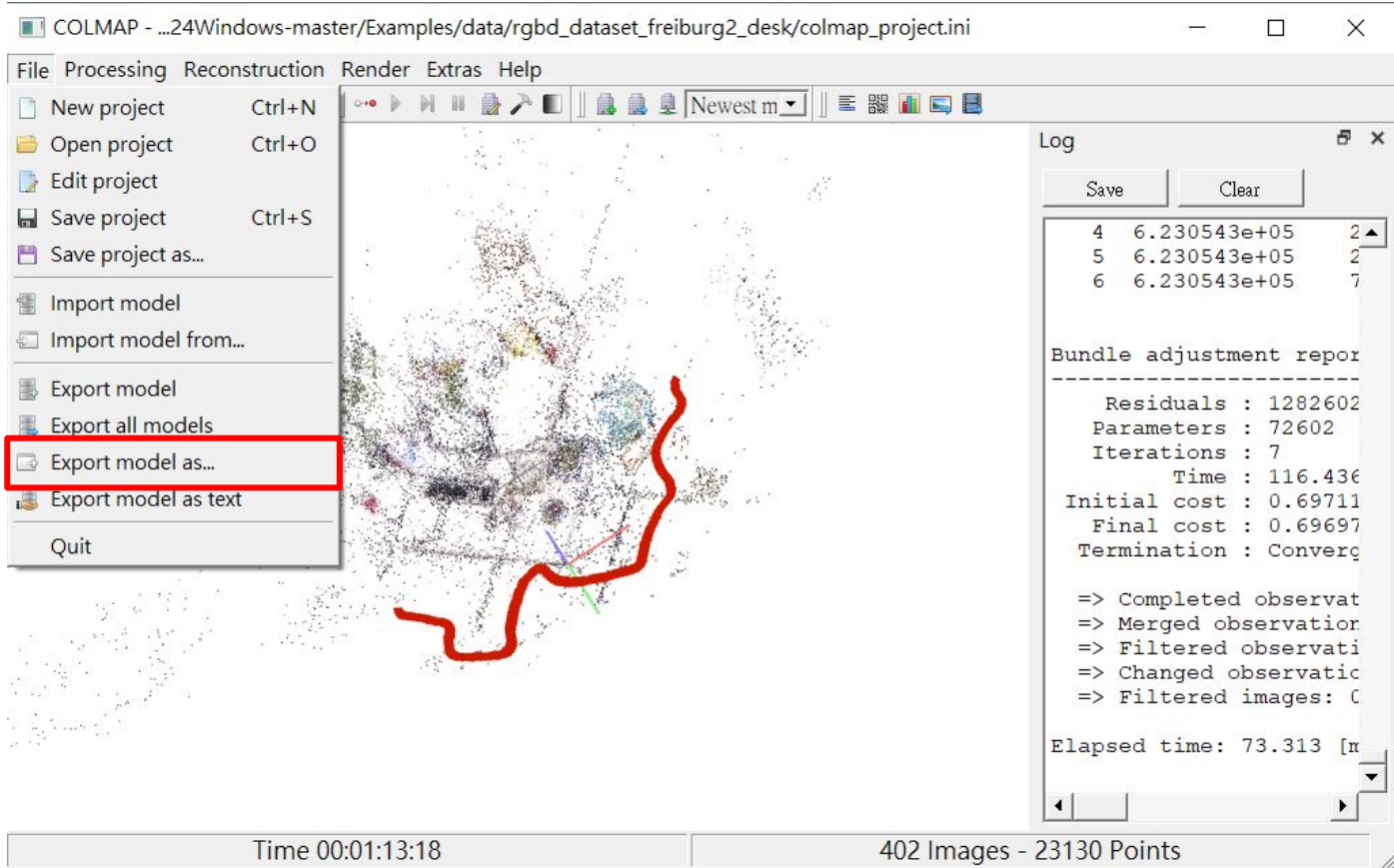
10. 重複4~6步驟

COLMAP會自動找還沒處理過的影像，也就是第8步新增的那些測試影像

COLMAP

11. Export model as

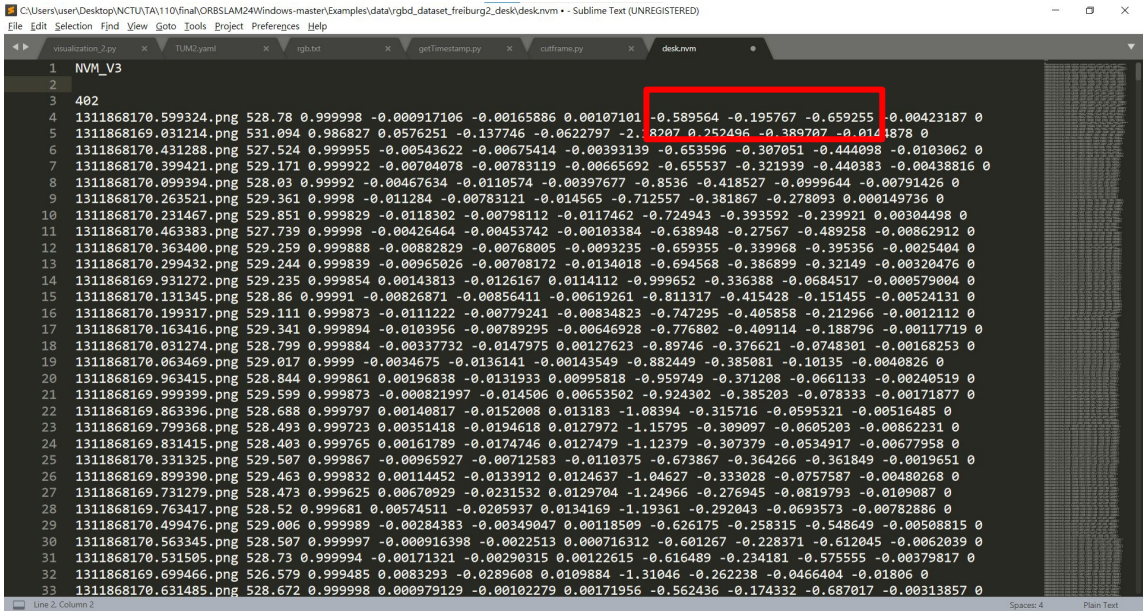
存成.nvm檔



COLMAP的output

.nvm file

<Camera> = <File name> <focal length> <quaternion WXYZ> <camera center> <radial distortion> 0

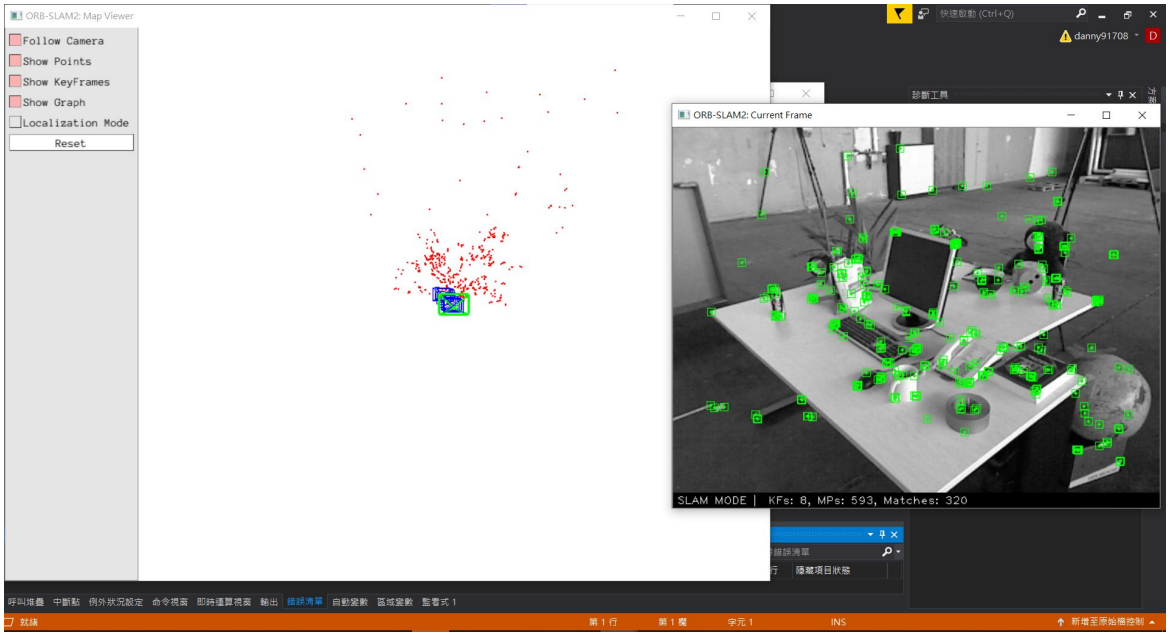


ORB-SLAM3

- Linux https://github.com/UZ-SLAMLab/ORB_SLAM3
- Windows https://github.com/melhashash/orbslam3-windows
  - Prerequisite from github
    - OpenCV: required at leat 3.0, tested with 3.4.13
    - CMake GUI: tested with 3.18.2
    - Visual Studio: tested with Visual Studio 2019
  - Prerequisite from TAs
    - Git: newest
- 助教有提供一個裝好的VMware虛擬機檔案 password: drone，如果真的裝不起來可以用

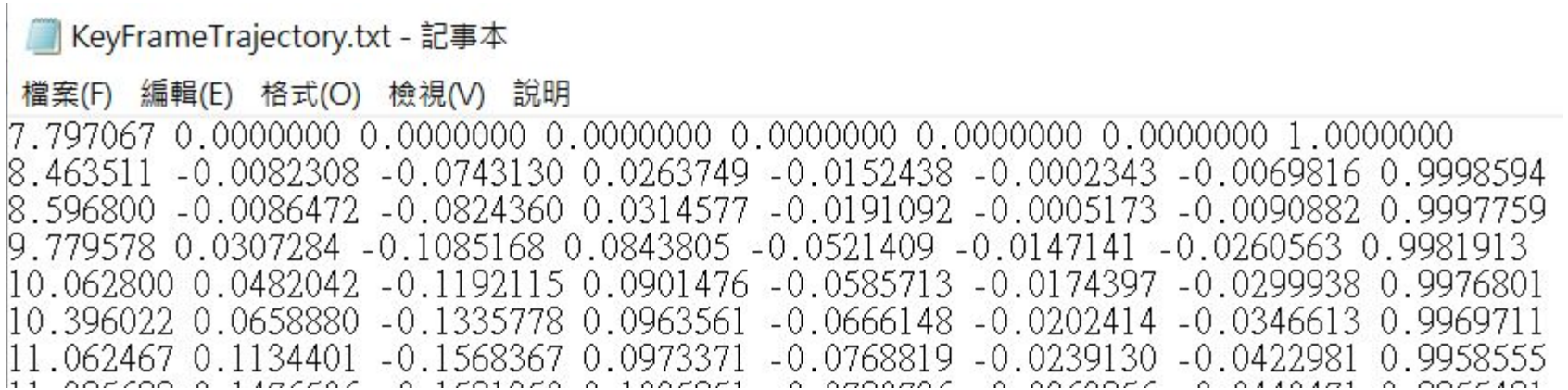
Test

- ./mono_tum [path_to_vocabulary(.txt)] [path_to_settings(.yaml)] [path_to_sequence(image folder)]



ORB-slam的Output

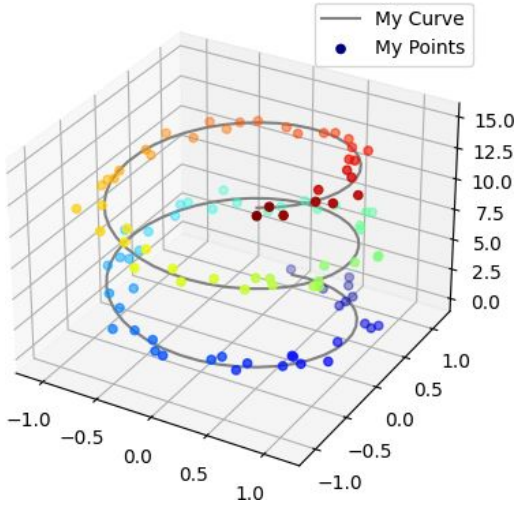
在 build 資料夾中 KeyFrameTrajectory.txt



依序為 timestamp, 相機(x, y, z), 相機旋轉(q_x, q_y, q_z, q_w)

Evaluation

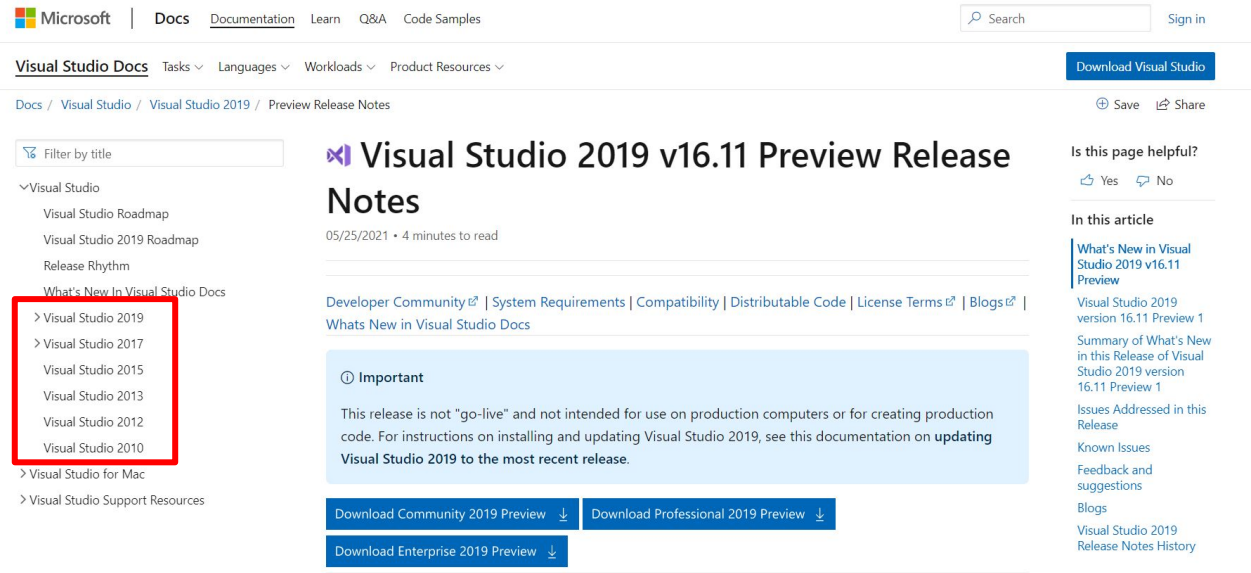
- 把COLMAP重建的相機軌道當作Ground truth
- ORB-SLAM的坐標系和COLMAP的坐標系不一樣
- 把ORB-SLAM重建的相機軌道轉換到COLMAP的坐標系，和Ground truth計算誤差
- 計算ORB-SLAM環形軌跡頭尾的誤差



附錄:安裝ORB-SLAM時可能出現的問題及逐步解決方法(版本為ORB-SLAM2)

Prerequisite

- Visual Studio
  - 可以從學校的filezilla載
  - 官網 : https://docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes-preview

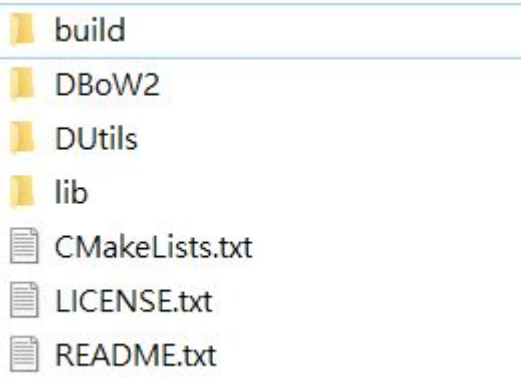


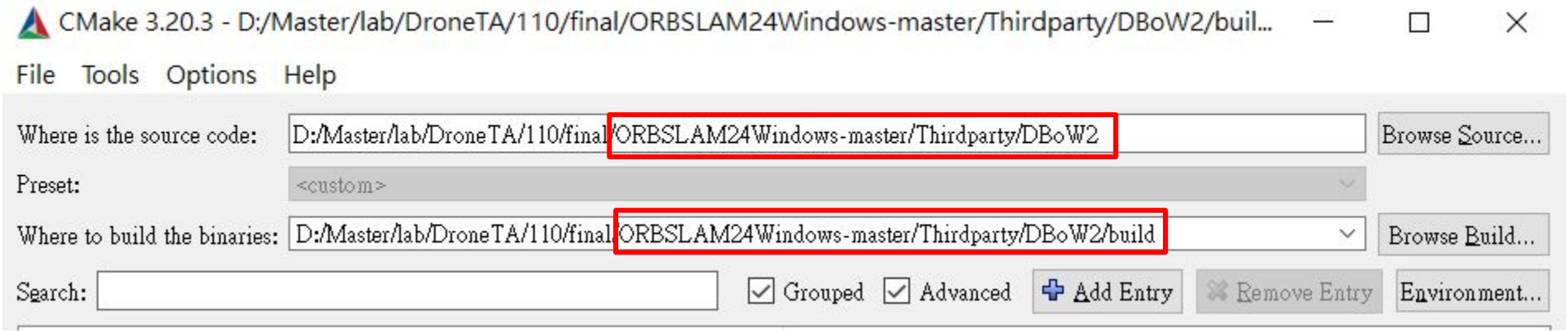
Steps

- Compile the projects in **Thirdparty** folder
  - DBoW2
  - eigen(not need to build)
  - g2o
  - Pangolin
- Build ORBSLAM

Steps - DBoW2

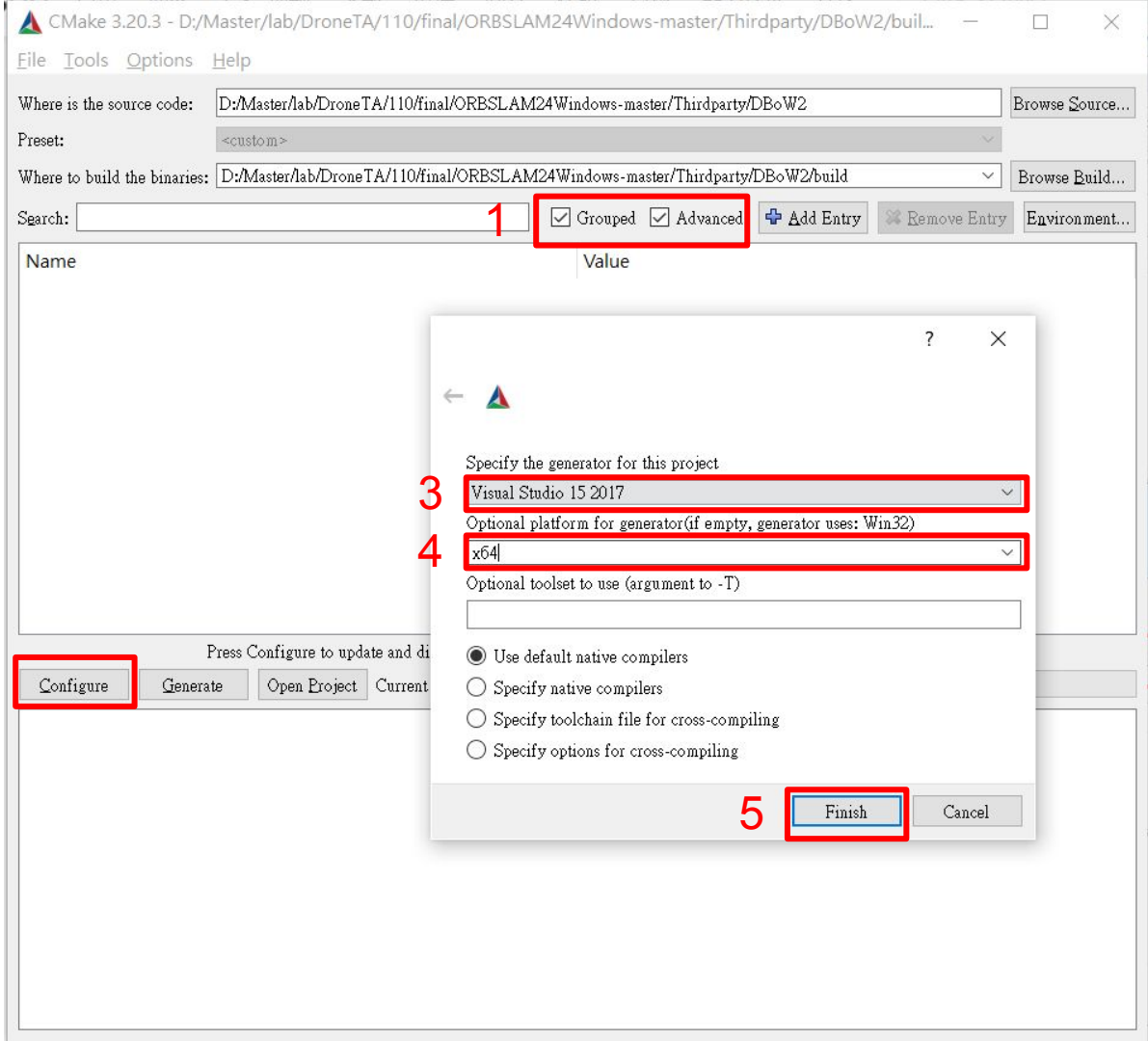
- 新增folder “build”
- Cmake
  - Browse Source.. 選*DBoW2*
  - Browse Build.. 選*DBoW2/build*





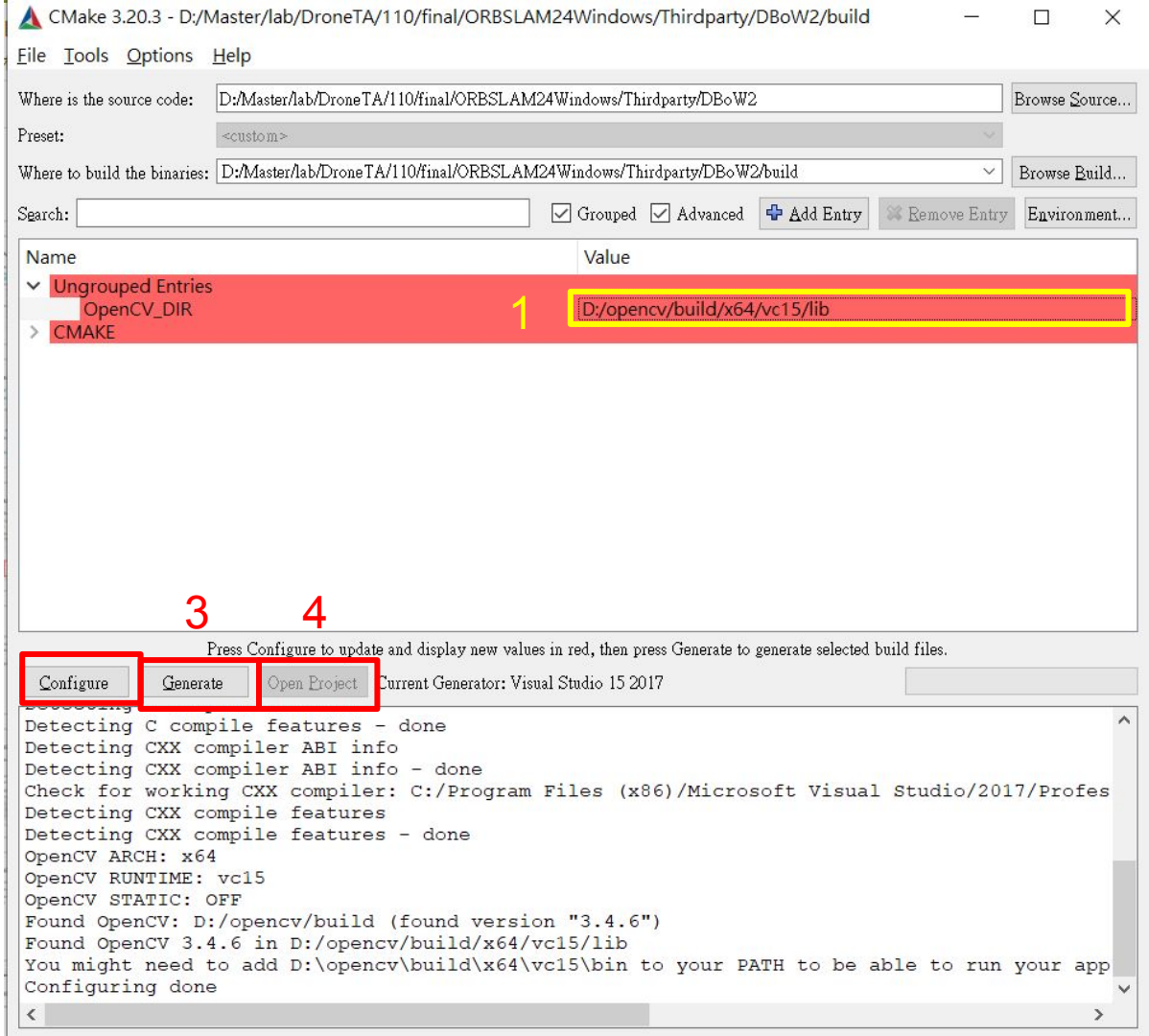
Steps - DBoW2

1. 把Grouped和Advanced勾起來
2. 點configure
3. 依自己的VS選版本
4. 選x64版本
5. 點Finish



Steps - DBoW2

1. 把opencv的lib path填上去
2. 再Configure一次
3. Generate
4. Open Project



Steps - DBoW2

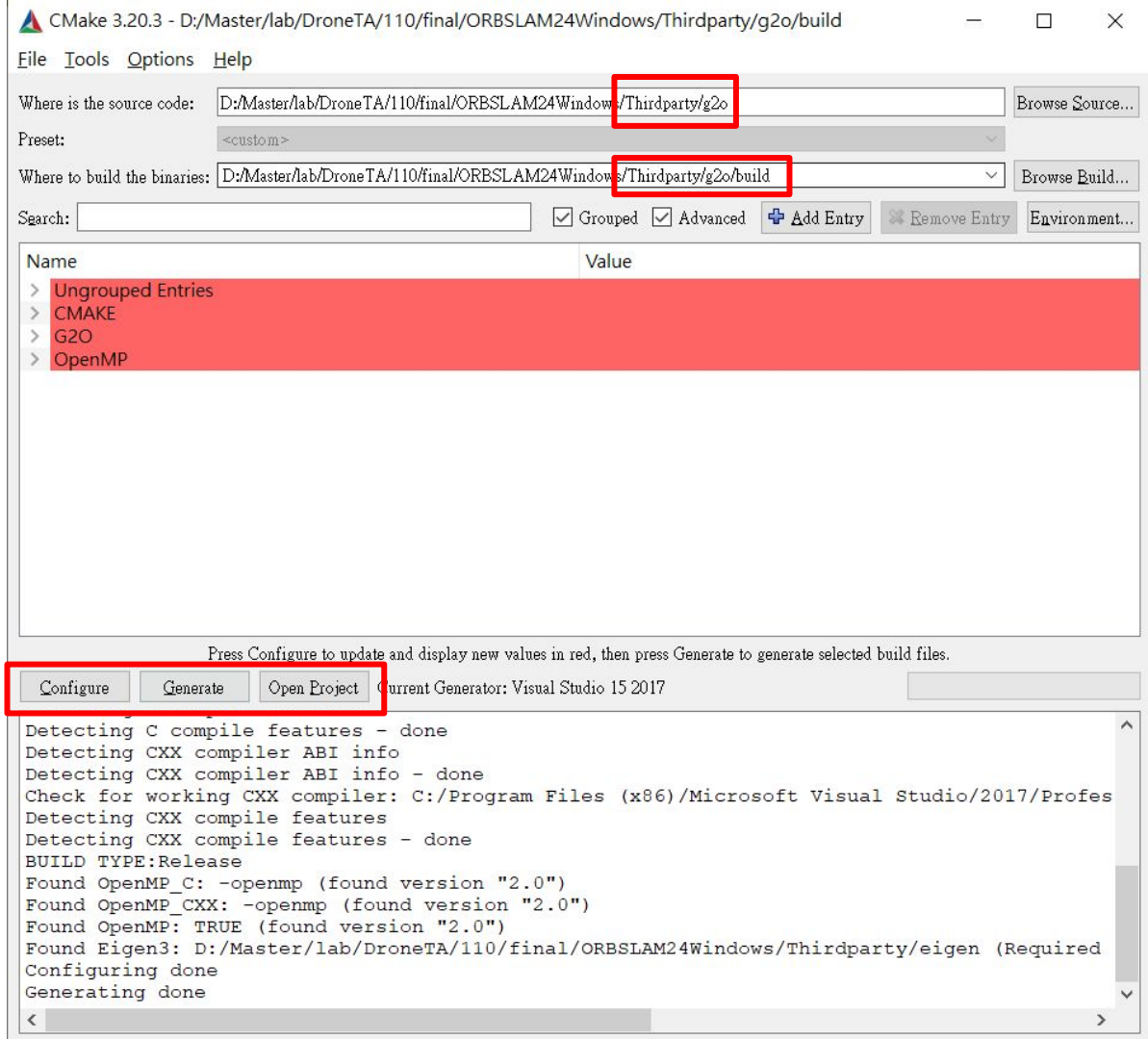
1. 選Release模式
2. 在ALL_BUILD項目點右鍵選擇"建置"
3. DBoW2 build完成！

```
========== 建置: 3 成功、0 失敗、0 最新、0 略過 ==========
```

Steps - g2o

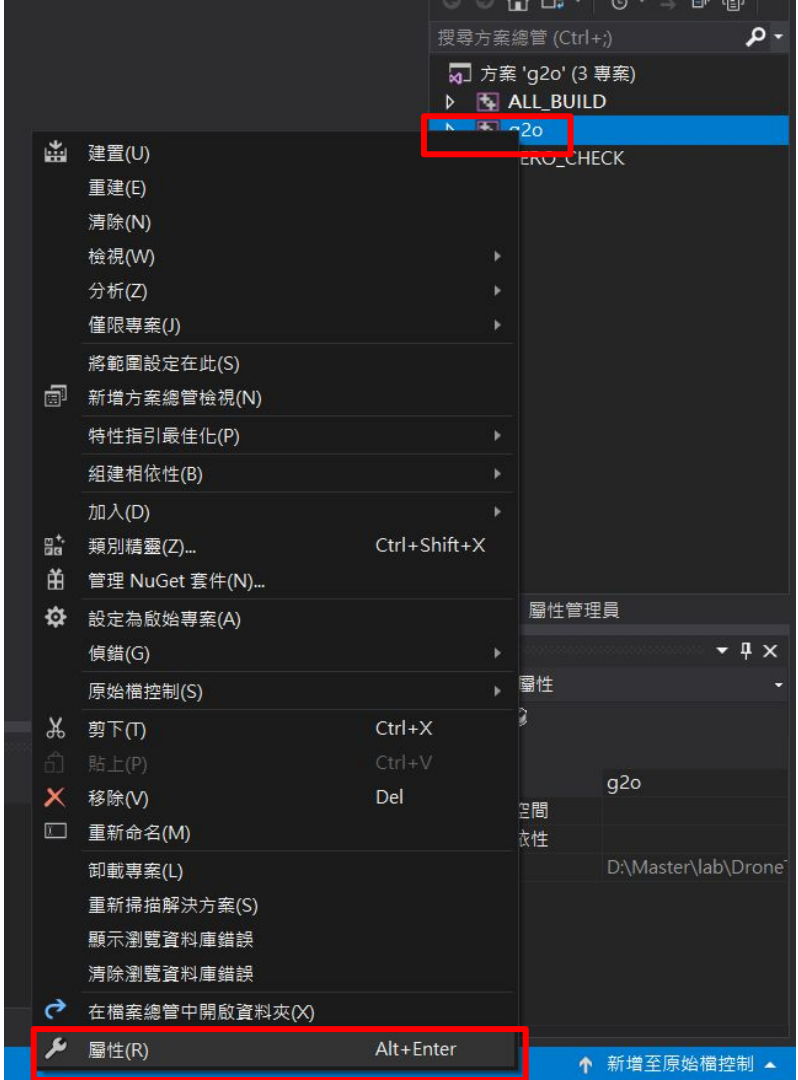
- Cmake和DBoW2一樣

(不用理中間的紅框)



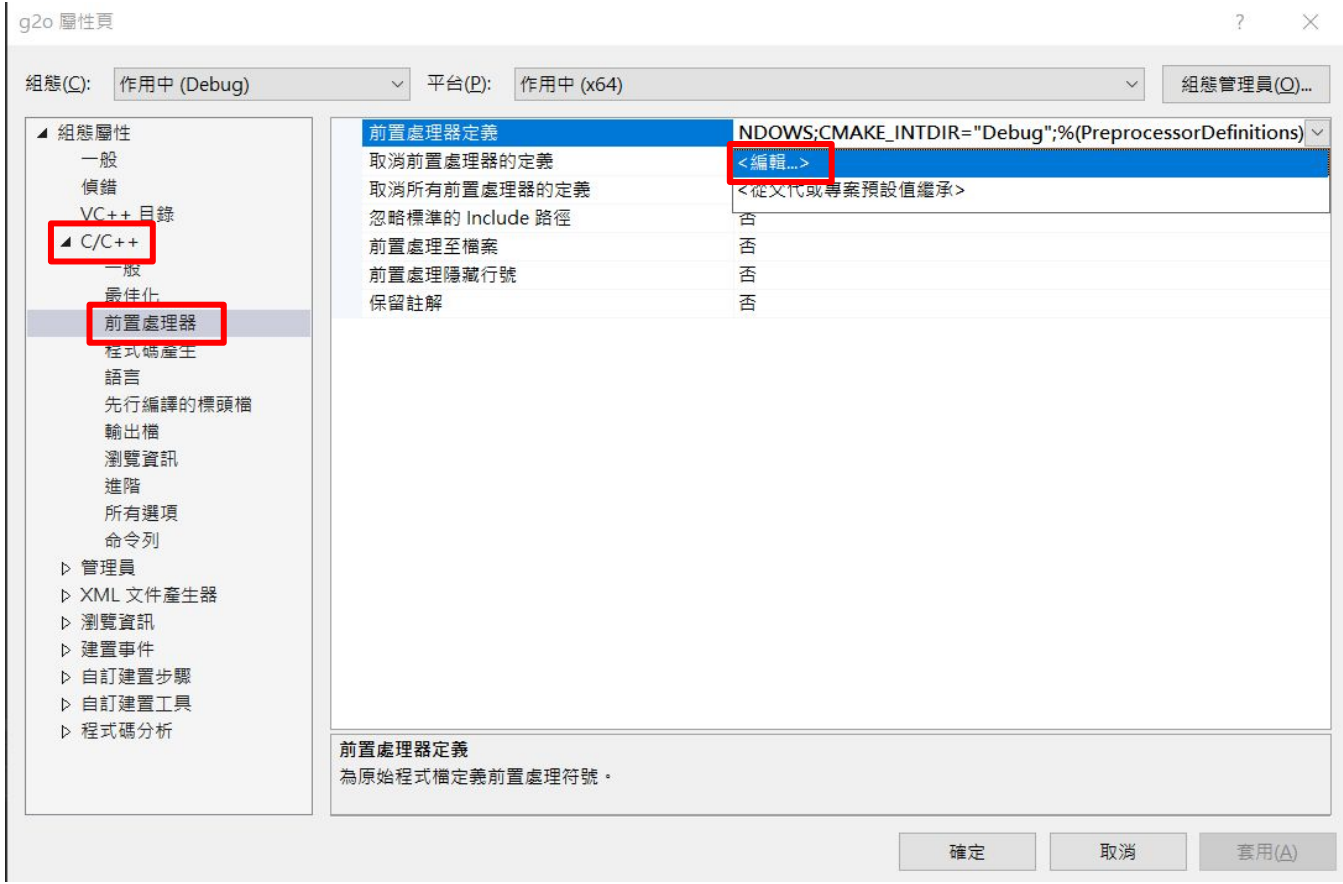
Steps - g2o

1. 點g2o項目右鍵
2. 選屬性

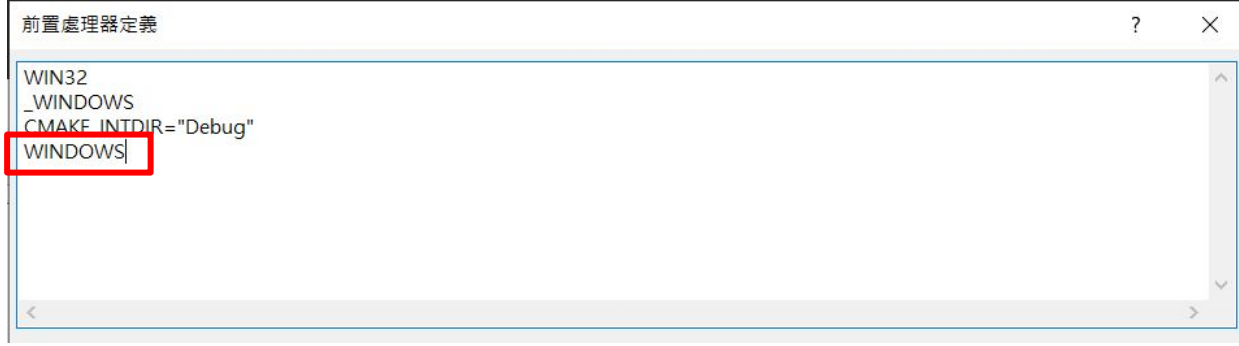


Steps - g2o

1. C/C++
2. 前置處理器
3. 編輯



Steps - g2o

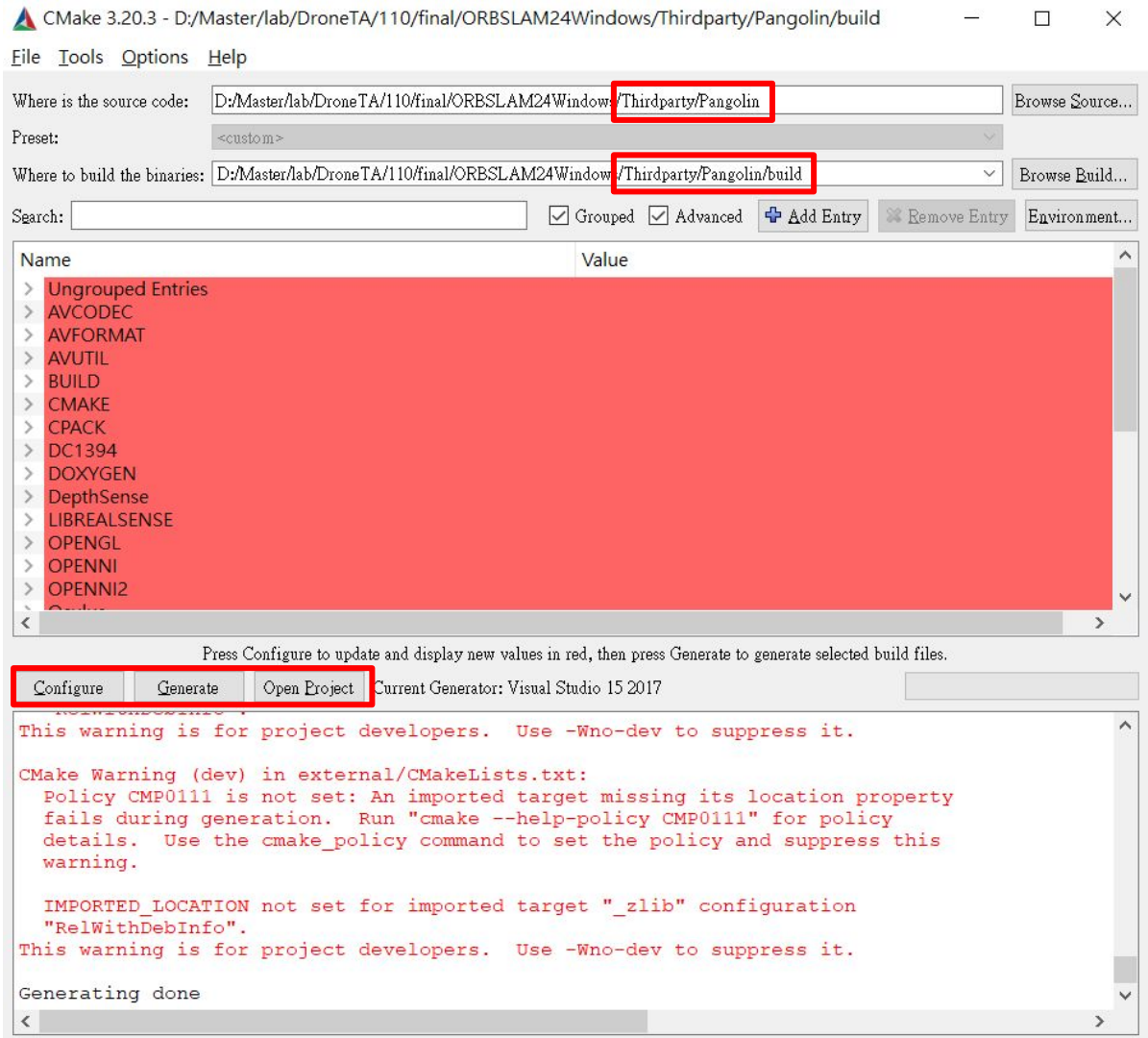


1. 加WINDOWS在最下層
2. 選Release模式
3. 在ALL_BUILD項目點右鍵選擇"建置"
4. 會有一個失敗
5. 一樣的動作再加WINDOWS，再建置一次
6. g2o build完成！

```
========== 建置: 1 成功、0 失敗、2 最新、0 略過 ==========
```

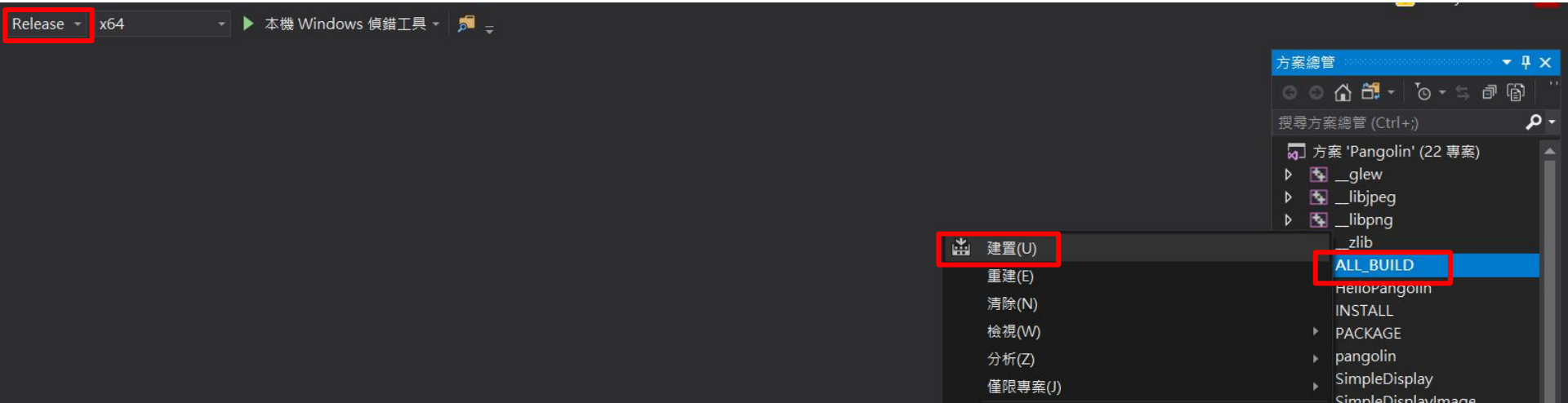
Steps - Pangolin

1. Cmake同上
2. 有很多紅框不理他→



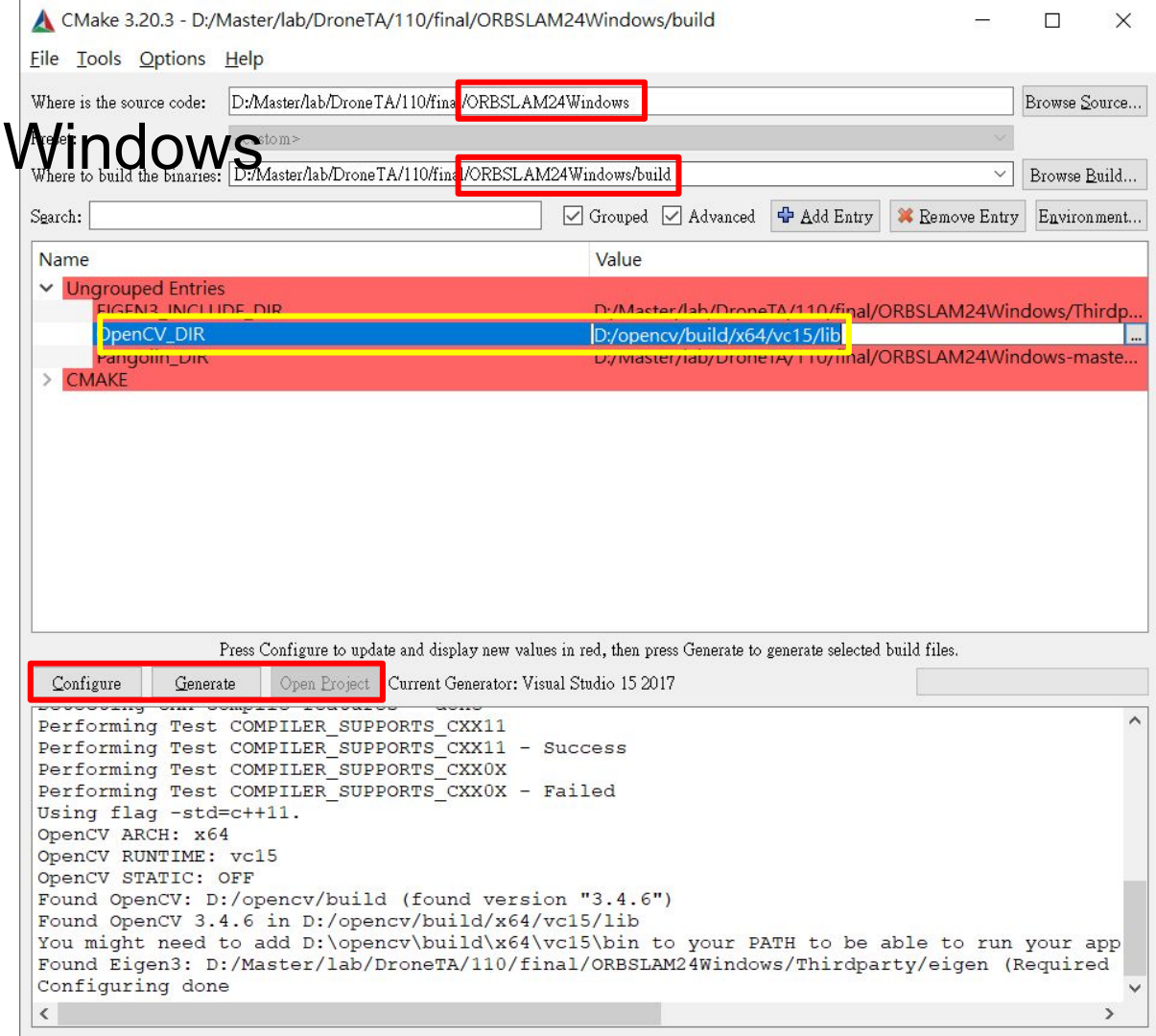
Steps - Pangolin

1. 選Release模式
2. 在ALL_BUILD項目點右鍵選擇"建置"
3. pthread.lib的失敗不用理他
4. Pangolin build完成！



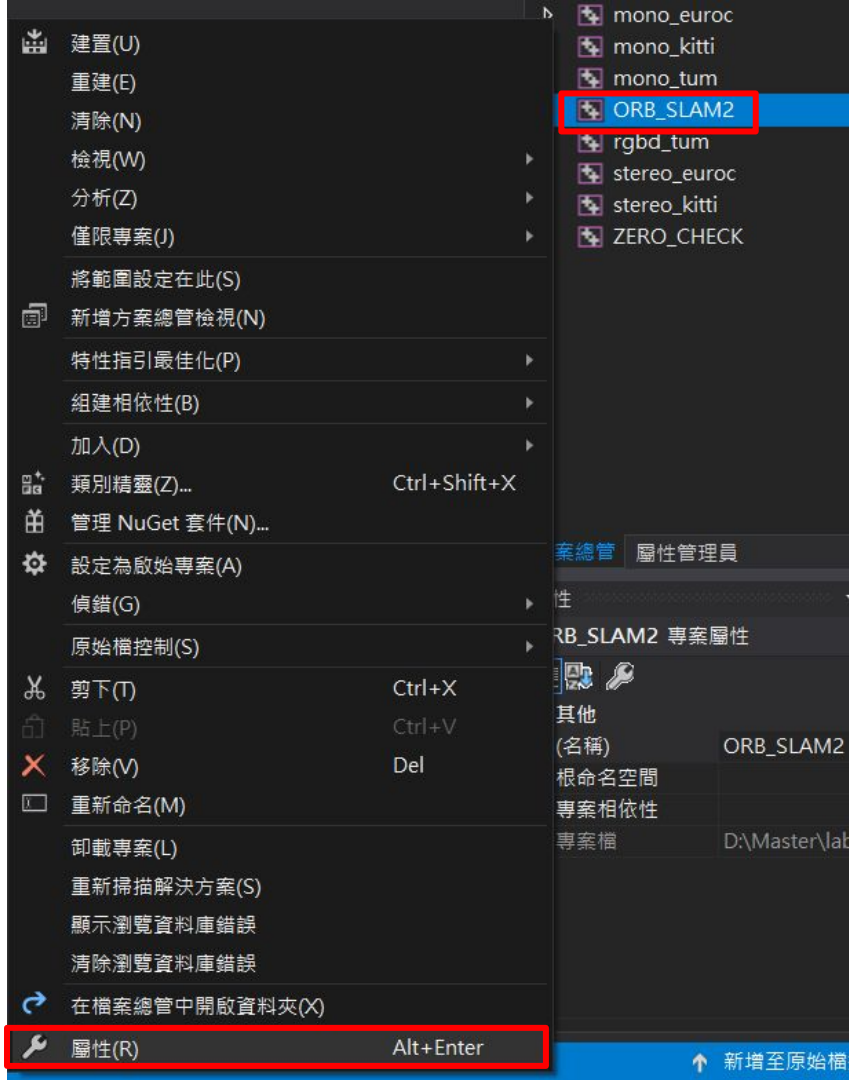
Steps - ORBSLAM24Windows

1. Cmake同上
2. 會報錯
3. 填上opencv的lib path
4. Configure
5. Generate
6. Open project



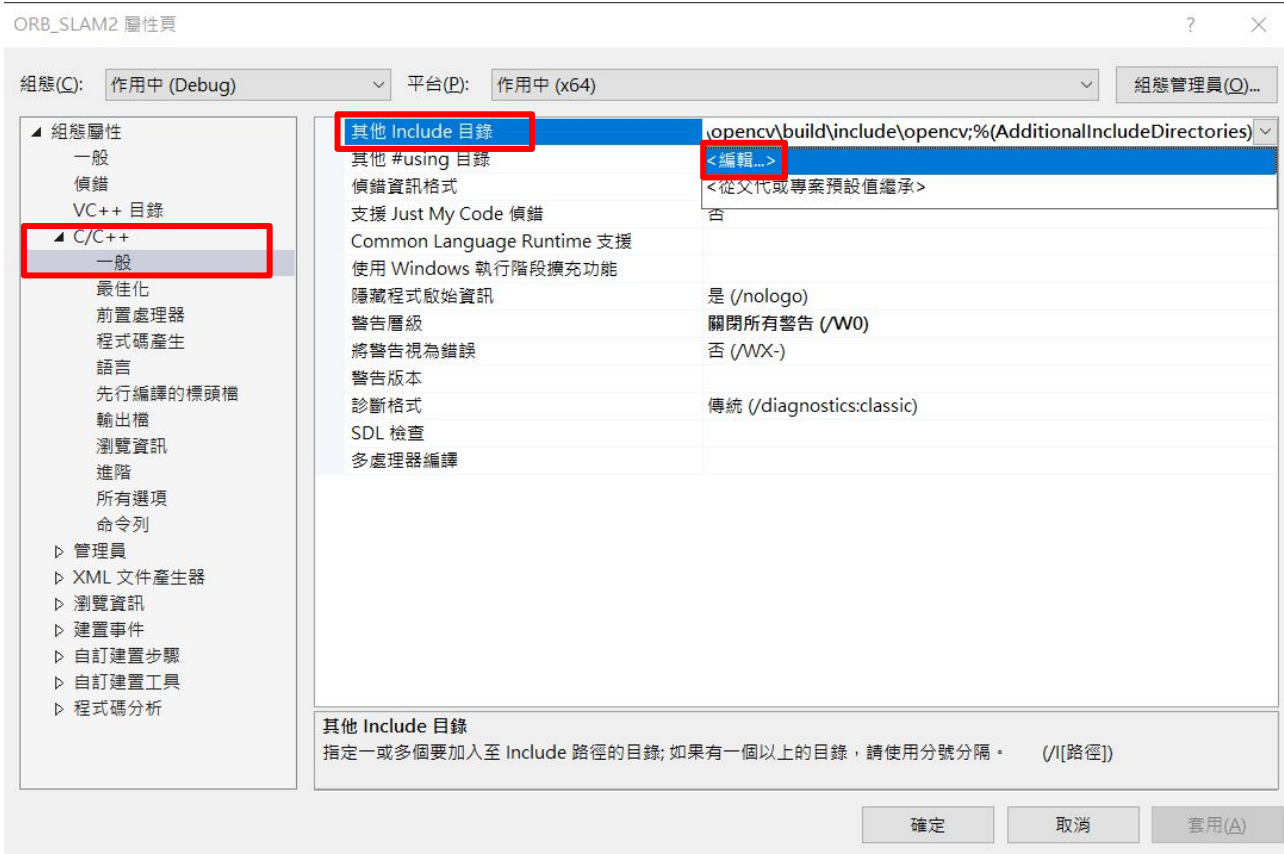
Steps - ORBSLAM24Windows

1. 選Release模式
2. 在ORB_SLAM2項目點右鍵
3. 屬性



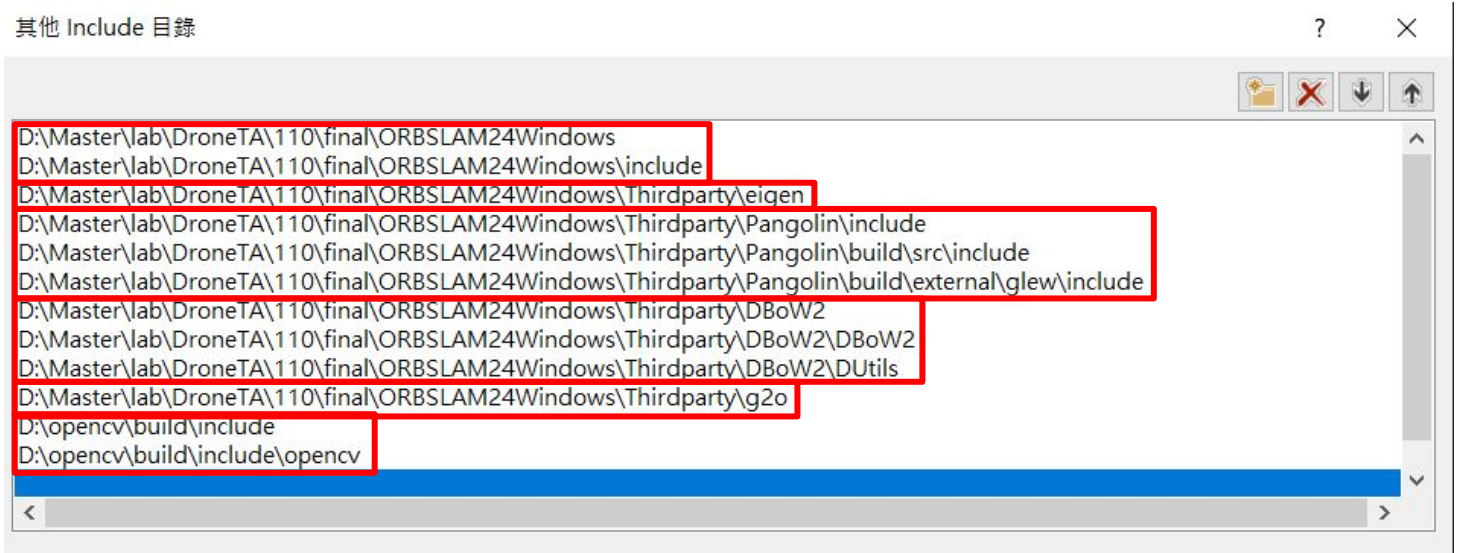
Steps - ORBSLAM24Windows

1. C/C++ → 一般
2. 其他include目錄
3. 編輯



Steps - ORBSLAM24Windows

1. 把缺的include path補上
2. 總共12個



Steps - ORBSLAM24Windows

1. 選Release模式
2. 在ORB_SLAM2項目點右鍵選擇"建置"

```
========== 建置: 2 成功、0 失敗、0 最新、0 略過 ==========
```

Steps - ORBSLAM24Windows

1. 在mono_tum項目同上再做一次

   (補include)

   (建置)