# 3DCV&DL HW2

R11922196 林佑鑫

## Problem 1:

### 1-1:

一開始試著做 P3P + RANSAC，到最後一步判斷 real R, T 時就算多放一個點進去判斷正確的 R,T 也無法去到剩下一個，不知道問題出在哪一步，後來索性直接改用比較熟悉的 DLT + RANSAC 做。

Pseudo Code:

Step1. 對每張圖，RANSAC 做 20 次。每次選六個對應點，做 step2, step3 的 DLT 後得到 R, T; 再用 point3D 跟 R, T 內積得到預測的 point2D，與眞正的 point2D 算 error，error 小於 threhold 15 的點當作 inliers，總共做 20 次選 inliers 最多的 best_R, best_T 回傳。best_R, best_T 即爲這張圖的 camera pose。

Step2. DLT 用選到的六個對應點根據 lambda * **u** = K × A × **X** 轉換而來的矩陣，每個點可得 2 X 12 的 vector of a11, a12, ⋯,a34 線性組合，六個點能得到 12 X 12 的矩陣 M，對 M 做 SVD 取 V_H 最後一列，有 12 個變數組成 R', T'，但因爲有經過常係數 c 縮放且 R'不保證正交，不是最終的 R。

Step3. 承上 Step2 DLT，對 R'做 SVD 得 U', D', V_H' ,常係數 c 爲±trace(D')/3，R 是 sign(c) ×U×VH，T 是 c * T'。代入 M 檢查決定 c 的正負號。回傳 R, T。

### 1-2:

一開始沒做 undistort 的誤差:

```
Median of relative rotation angle differences: 0.005803901346432731
Median of translation differences: 0.023241857559172026
```

加上 undistort 的誤差:

```
Median of relative rotation angle differences: 0.003958943117388924
Median of translation differences: 0.003981762113556889
```

Discussion:

雖然一開始的誤差看起來已經很小，但當時我沒發現公告中說可以用 cv 解 distortion 的 function，我也只找到從 undistortion 到 distortion 的算法，沒有找到怎麼回推正確的位置的算法，所以當時我的實作中並沒有考慮 distortion。

之後運用了 cv. undistortImagePoints 之後可以看到我的誤差有明顯提升，rotation 提升三分之一的精準度，translation 甚至變成了將近六倍精準度。由此可知內部參數，外部參數，形變參數對 camera calibration 來說都十分重要。

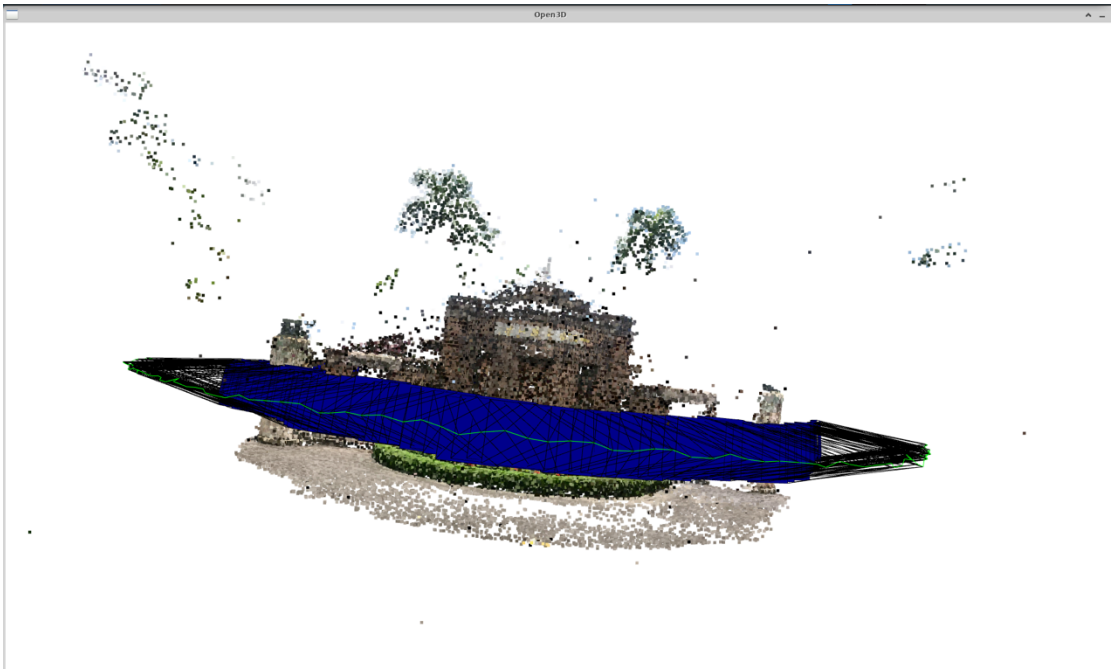
## 1-3(只用 train_image):

首先要先對 image 做 preprocessing，因爲 image 包含 train 跟 validation image，若一起畫出來則會有兩條線。而且 dataframe 中 image_id 是亂的，如果沒做 preprocessing 的話綠線會因爲 index 亂掉而到處亂飛。

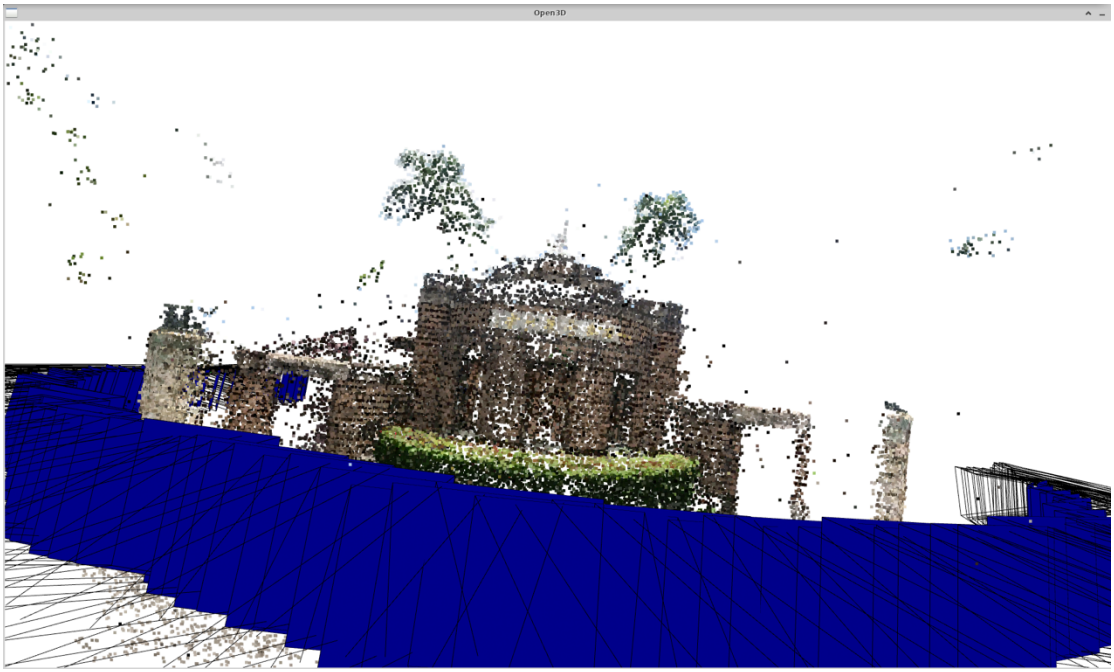
對每張 image 的 camera pose R, T 我用 R, T 結合的反矩陣跟圖片中的固定五個點(中心＋正方形四角)做內積，可以得到在三維空間中的點。中心內積的結果是成像中心，另外四個點是相機的 orientation，就是題目要求的五個點。線條分綠線與黑線，綠線是每五個點就連在一起(上一個成像中心到下一個成像中心)，黑線則是成像中心到其他四點以及四點的邊，如此就能得到金字塔型的圖。最後就是底部要塗滿藍色，我用 triangularmesh，把底面的正方形當作兩個三角形畫出。

這邊有一點要注意：triangularmesh 有 normal 的差異，所以三角形的 index 順時針或逆時針對上色的面有影響，若設定錯誤則顏色會顯示在三角形的另一面(背對相機那面)則大部分的正方形都會是白的。

遠：



近：

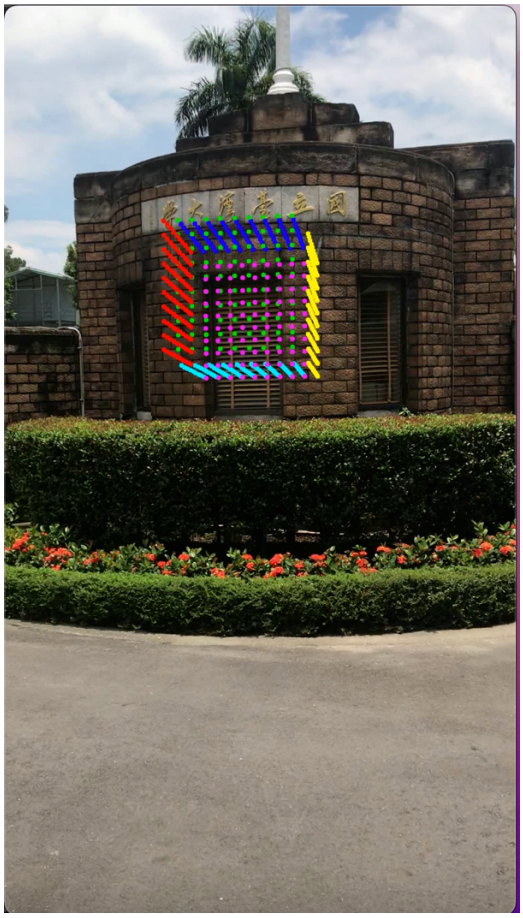


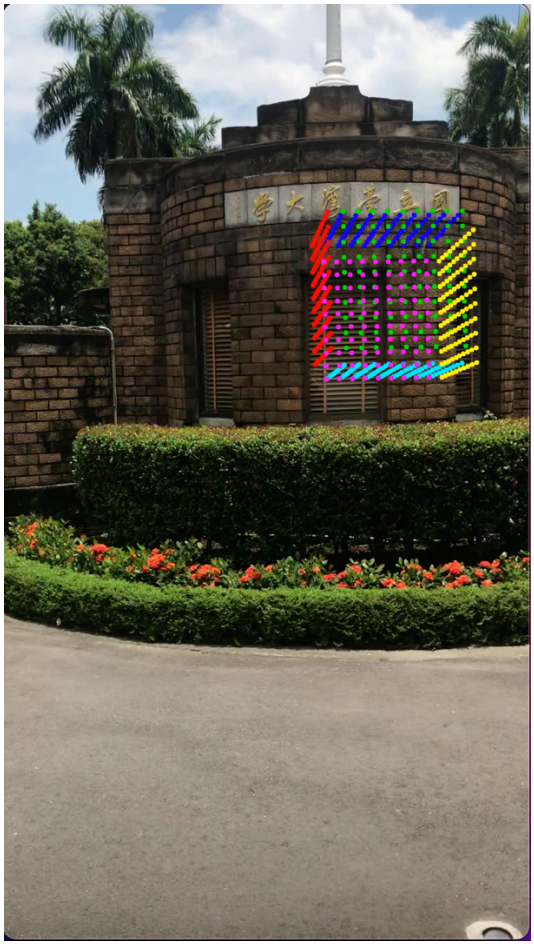

## Problem 2:

Problem 1-3 只用 train_image，這題只用 valid_image。

首先用 sample code 得到的正方體的頂點(8 個)做 interpolation 得到 world coordinate 中所有點(6 X 10 X 10)的位置，並根據點所在的 plane 對應不同顏色。

之後，將第一題找 camera pose R, T 的 function 搬過來，對每張 image 找出 camera pose。已知 intrinsic, extrinsic parameters 和點在 world coordinate 的位置，就能算出每個點在 image 上的 pixel 位置，之後用 cv circle 畫在圖上，將圖串成影片就得到 AR_video.mp4。





## Hardware:

1. CPU: i9-10980XE
2. GPU: Nvidia RTX 3090

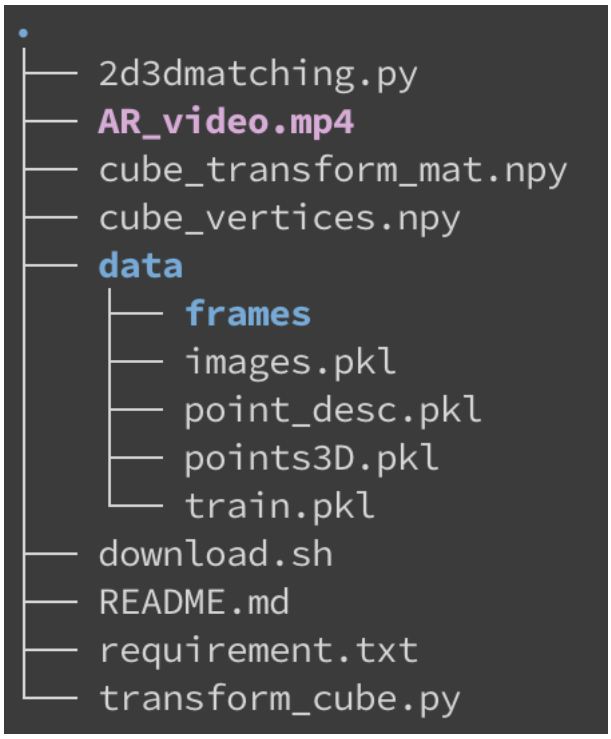
## Environment:

Ubuntu 20.04 / Python 3.8.10

## Modules:

1. numpy==1.23.4
2. scipy==1.9.3
3. pandas==1.5.1
4. opencv-python==4.6.0.66
5. open3d==0.15.2
6. tqdm==4.46.1

## File structures:




AR_video.mp4, *.npy 在執行過 transform_cube.py 才會出現

## Commands:

### Environment:

pip install -r requirements.txt

### Download:

bash download.sh

### Reproduce Question 1:

python3 2d3dmatching.py

### Reproduce Question 2:

python3 transform_cube.py

請先用(shift) + A/S/D/Z/X/C/V 對 cube 做平移、旋轉、縮放，關閉視窗後產生 AR_video.mp4