HW2

Q1-1:

這題我是用 DLT 去實作的。原本是想按照期望用 p3p 去實作，但 debug 難度過高，寫完後完全找不出 bug，後來就決定用 DLT 去實作。用 DLT+ransac 去實作。我寫的 P3P 的 code 也在檔案中

Pseudo code:

For every image:

讀資料，R,T=DLT(7 points)，跑 50 次。

根據 R 跟 T 計算自己的 point2D

自己的 point2D 跟 ground truth 算 error

看有多少個點的 error 比 threshold=10 小，當作是 inliners。挑出 inliners 最多的那些 inliners，用這些 inliners 重新算一次 R、T 當作最終的 R、T

def DLT:

挑出內在參數跟每個點的對應點

根據公式去設定方程組的參數，做 SVD，VH 的最後一列就是解。

不過因為 svd 求出的 R 沒有正交，所以還要再做一次 SVD 才能得出解。我基本上是根據 https://zhuanlan.zhihu.com/p/58648937 去

實作的。

我把跑出來的 R 跟 T 存起來，方便 Q2 使用。我也放到雲端了，連結如下:

https://drive.google.com/file/d/1XglaGXHcoHgkuz7pcUb3rGREwjcGRmk4/view?usp=sharing

Q1-2:

我的 angles 跟 transaction 的 mean 是 0.004551910869514348 跟 0.028752429130272456。

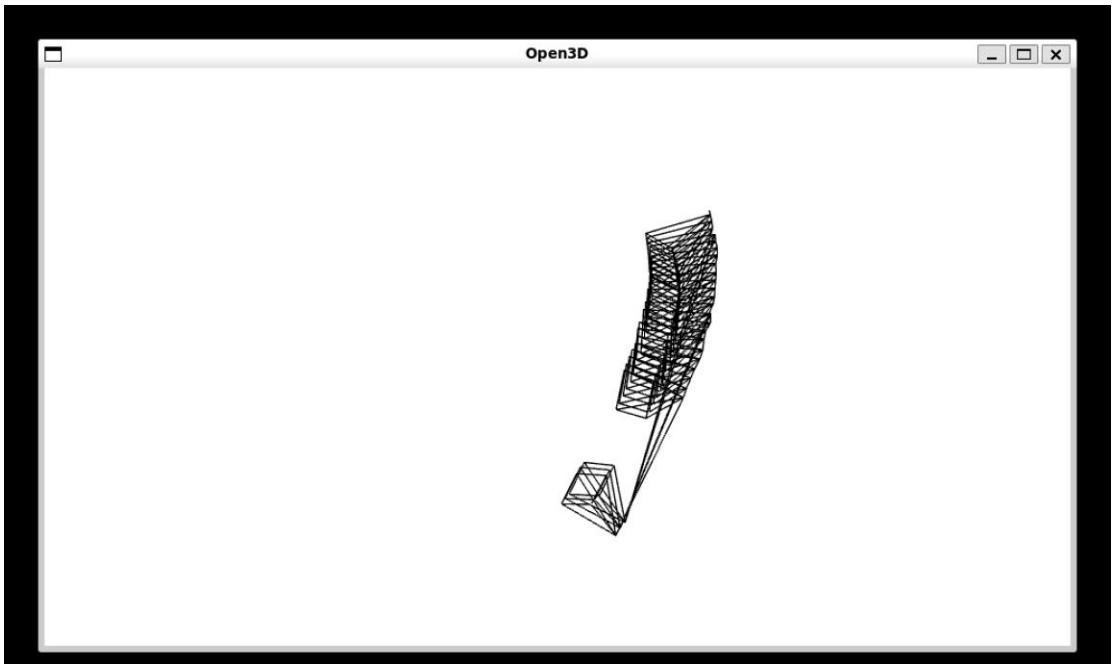
可以發現其實 DLT+ransac 的誤差還滿小的。沒 ransac 的話誤差可以會很大，我沒實作出來 p3p，但理論上應該會更小，畢竟是專門解 pnp 的算法。我測試了沒有處理 distortion 的狀況。結果是 0.002310478461115535 0.01446038776236965。比原本的還要低。理論上應該是要高一些，應該是隨機性和計算誤差的緣故。有時候會發現 angle 部分會一直出現錯誤，後來發現是因為誤差導致 arccos 的計算超過範圍。

Q1-3:

我的電腦畫圖有點問題。單純跑沒修改過的 transform_cube.py 的結果如下:

沒辦法準確顯示大門的樣子。可能是 wsl 的問題。

所以我就只能顯示畫軌跡跟相機角椎的圖片



我的作法是根據前一題得到的 r 跟 t 再加上一維去算反矩陣，這樣就可以從相機座標反推世界座標。設定好相機座標下角錐的座標，就可以一起做矩陣乘法來反推角椎的世界座標。再用 open3d 把對應的點連線起來即可。

Q2:

這題跟 Q1-1 一樣都是用 valid images，所以我選擇在第一題就把每個 image 對應的 R 跟 T 存成檔案，然後在這題拿來用。就把內在參數跟外在參數乘起來，乘上 cube，得到 cube 在 image 上的位置，然後把每個 R 跟 T 對應的影像讀進來，把 cube 放上去。Painter's algorithm 的部分只要調一下點的順序就可以了。難點在於格式的複雜和 function 的不熟悉。

最後成果存成 ntu.mp4 檔。我把我跑出來的的 ntu.mp4 放到雲端端了。連結如下:

https://drive.google.com/file/d/1_K_FPeoMXohIrnOou7_SIsl6PHRD63E8/view?usp=sharing

Environment:

Ubuntu 20.04

Command:

For Q1-1,1-2:

```
python3 p1.py
```

For Q1-3

```
python3 p1-3.py
```

For Q2:

python3 p2.py

還請先跑 Q1 再跑後面的。因為我 Q1-3 跟 Q2 都有用到 Q1-1 跑出來的資料。

Youtube 連結:

https://youtu.be/STO2VsTNXN0