

3DCV 2025

HW2 COLMAP & Camera Relocalization

姓名:林敬哲
學號:R14521607

=====

執行步驟:

#python=3.10

```
git clone https://github.com/NTU-CSIE-3dcv-TA/homework2-linjing0927.git
cd homework2-linjing0927
```

```
pip install -r requirements.txt
sudo apt install ffmpeg
sudo apt install colmap
```

download data file into homework2-linjing0927

q1)
python q1_run_sparse.py
python q1_mesh_poisson.py
<https://youtu.be/dQjmc5jXg8>

q2)
python 2d3dmatching.py
python transform_cube.py
python q2_ar.py
https://youtu.be/Rau51fnJZ_Y

=====

LLM is used -本次作業有使用chatgpt5 輔助完善程式碼架構，在程式碼有衝突時提供解決辦法，並且協助統整報告重點整理，在我不熟悉操作時也會詢問解決辦法

Problem 1

method:

Q1-1

輸入為一段.mov格式影片，首先以固定 **2 fps** 抽取影像並輸出至q1/images/
之後特徵提取使用 COLMAP 的 SIFT 特徵描述子進行特徵點偵測與描述
最大影像尺寸設為 **2000 像素**

由於輸入影像為影片幀序列，使用 sequential_matcher 僅對臨近幀進行匹配
使用稀疏重建

將 COLMAP 的模型轉換成 .ply
並由 points3d_txt_to_ply() 讀取 points3D.txt, 將座標與顏色欄位轉換成 points3D.ply
以供 Open3D 顯示與 Poisson 重建使用

執行完成後, 會自動啟動 Open3D GUI 顯示稀疏點雲結果:

- 每個三維點以 RGB 顏色呈現
- 可透過旋轉與縮放觀察整體結構
- 若顯示為稀疏且分散, 代表相機軌跡或特徵不足

Q1-2

使用Open3d mesh

讀取 Q1-1 輸出的 q1/output/points3D.ply 並自動估計點雲法向量半徑

利用 Poisson 方法生成封閉曲面, 重建深度設定為 10

Poisson Surface Reconstruction: 以輸入點雲的法向量場為依據, 求解泊松方程以近似隱函數 $f(x, y, z)$, 並將其零等位面作為最終曲面

為減少雜訊與三角面過多問題, 加入輕度平滑與面數下修

```
POISSON_DEPTH = 10
SMOOTH_ITERS = 3
TARGET_FACES = 150_000
```

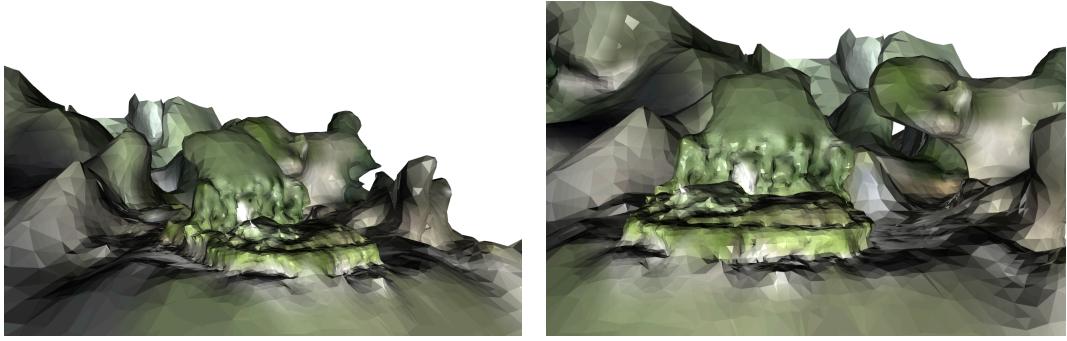
輸出結果



原圖



點雲



mesh model

Problem 2

method

Q2-1 2D–3D Matching & Camera Pose (自己實做 P3P + RANSAC)

讀 `data/images.pkl` / `train.pkl` / `points3D.pkl` / `point_desc.pkl`

以 `train.pkl` 建「3D 模型的描述子索引」

以 `point_desc.pkl` 提供每張驗證影像的 2D keypoints 與描述子

兩者做 2D→3D 對應後，進入 P3P + RANSAC 解相機外參

針對每張validation影像，程式從 `point_desc.pkl` 讀入該影像的 keypoints (`kp_query`) 與描述子 (`desc_query`)，並以 **FLANN** 最近鄰搜尋 進行比對。每個 query 描述子會找出兩個最近鄰，透過 **Lowe's ratio test**(比例閾值 0.8)保留唯一可信的匹配，產生一組 2D–3D 對應點。

隨後進入 `pnpSolver()` 進行姿態估計。實作**P3P + RANSAC**:在 RANSAC 外迴圈中，每次隨機抽取 3 對匹配點以求得候選姿態。P3P 根據三個 2D–3D 對應與相機內參 KKK(1868.27, 1869.18 為焦距；主點在 (540,960))計算多組可能的 $[R|t]$ 解。之後進行 **cheirality** 檢查，確保大部分 3D 點的深度為正，再以所有對應計算重投影誤差(`cv2.projectPoints()`)以像素距離作為評分指標。

RANSAC 迴圈重複多次(約 2000 次)，取內點數最多且平均誤差最小者為最終解。為避免無效解，設置最小內點數(20)與誤差閾值(3 像素)。最終的 `rvec` 與 `tvec` 轉換為「camera-to-world」形式後輸出至 `data/poses_est.csv`。接著以 ground truth 姿態計算旋轉角誤差與平移誤差，分別取中位數作為整體表現指標。

視覺化

程式利用 **Open3D** 將 `points3D.pkl` 點雲顯示出來，並為每張影像繪製一個相機錐體表示其姿態。錐體顏色以紅、藍交替顯示，並以綠線作相機軌跡。

結果：

Rotation (deg) median = **1.312**

Translation (m) median = **0.079**

討論

整體而言，系統能穩定地從驗證影像估計出相機姿態，顯示出以 SIFT 特徵為基礎的 2D–3D 對應與自行實作的 P3P + RANSAC 管線具備良好的穩健性。

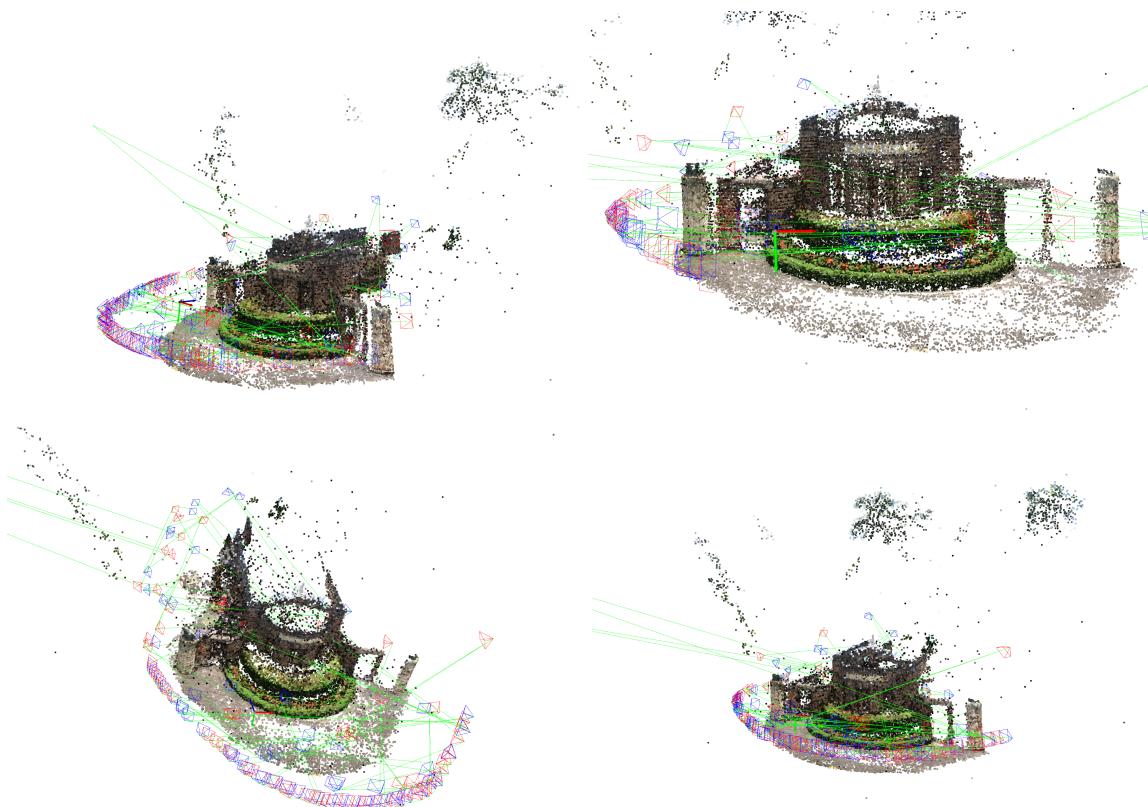
實驗結果中，旋轉誤差中位數為 1.312° 、平移誤差 0.079 m ，說明整體定位誤差相對低，對應的相機軌跡在三維視覺化中亦呈現連續分佈，與場景結構合理對應。

然而在少數幀上觀察到相機錐體位置偏離主體結構可能因為：

1. 特徵分佈不均或遮蔽：某些影像區域缺乏清晰紋理，導致 SIFT 無法提供足夠匹配點。
2. RANSAC 內點不足：若抽樣點落在局部平面或誤匹配點，可能出現局部最佳解。

可能改進方向：

- 動態調整匹配閾值：依影像品質或特徵數量自動調整 Lowe ratio 或重投影閾值。
- 融合多張影像平滑軌跡：可於姿態序列後處理中加入卡爾曼濾波或滑動平均以提升連貫性。



camera pose around the point cloud

Q2-2: Augmented Reality Cube with Painter's Algorithm

載入前一題輸出的 `data/poses_est.csv`, 其中包含每張驗證影像的旋轉向量 `rvec` 與平移向量 `tvec`, 這些參數描述了相機在世界座標系中的姿態。再讀入 `data/images.pkl` 對應出每個影像 ID 的檔名, 從 `data/frames` 中載入實際影像
執行 `transform_cube.py`, 則可讀取 `cube_transform_mat.npy`(形狀為 3×4 的 $[sR|t]$ 矩陣), 以此設定立方體在世界座標中的位置、朝向與縮放

執行 `q2_ar.py`

每張影像中, 程式以 Q2-1 求得的 (`rvec`, `tvec`) 將立方體點雲從世界座標投影至影像平面, 套用內參矩陣K及畸變參數, 同時以 (`R @ X + t`) 的 z 值作為深度。

Painter's Algorithm: 避免遠處的點覆蓋近處的點

- 先依深度由遠至近排序 (`np.argsort(depths)[:-1]`)
- 再逐點在影像上以 `cv2.circle()` 畫上彩色圓點

執行完成後.mp4檔會存在目錄下



立方體圖