學號:r13522859 姓名:蔡翔羽

Problem1

1.我用以下指令先把影片切成 img 到我指定的 images 資料夾

ffmpeg -i /home/ivmlab3/Downloads/IMG_4041.MOV -vf fps=4 frame_%04d.jpg

2. 用這個指令做特徵提取

colmap feature_extractor --database_path database.db --image_path ../images/

3. 按照影像順序匹配特徵

colmap sequential_matcher --database_path database.db

4. 建立 sparse 資料夾後，進行 SFM

ivmlab3@ivmlab3-Nuvo-6108GC:~/3dcv_hw/homework2-asd30627/images$ colmap mapper --database_path database.db --image_path ../images/ --output_path sparse/

5. 影像校正，產生去畸變的影像和相機模型到 dense/

```
colmap image_undistorter \
    --image_path ../images/ \
    --input_path sparse/0 \
    --output_path dense \
    --output_type COLMAP
```

6. 開 colmap GUI

colmap gui

import model 進來

7. 網格重建

(3dcv) ivmlab3@ivmlab3-Nuvo-6108GC:~/3dcv_hw/homework2-asd30627/images$ python /home/ivmlab3/3dcv_hw/homework2-asd30627/mesh_from_colmap.py --colmap_model_dir sparse/0 --method poisson --poisson_depth 10 --density_trim 0.03 --target_tris 200000 --out mesh_poisson_clean.ply

我寫了一個 mesh_from_colmap.py 是使用 open3d 套件做:

1.讀點雲（points3D.bin）

2.(可選)下採樣、法向估計

3.用 Poisson 或 Ball Pivoting (BPA) 做網格重建

4.清理／平滑／（可選）簡化

5.輸出 mesh_out.ply，並可彈視窗預覽與存截圖。

連結在這裡
https://youtu.be/b3LNH_xdlKQ
此外，因為我覺得室外可能雜訊太多，效果不好，所以我做了另一個室內烏薩奇娃娃的版本，做法是改全用 GUI 做，影片連結在這
https://youtu.be/pLKekowNKMw

用到 LLM(Chatgpt)部分:

我自建的檔案 mesh_from_colmap.py 中除了 Poisson 與 Ball Pivoting (BPA)大多為我寫的以外(有問他輸入有哪些)，其他大多功能依靠 LLM(Chatgpt)的幫忙

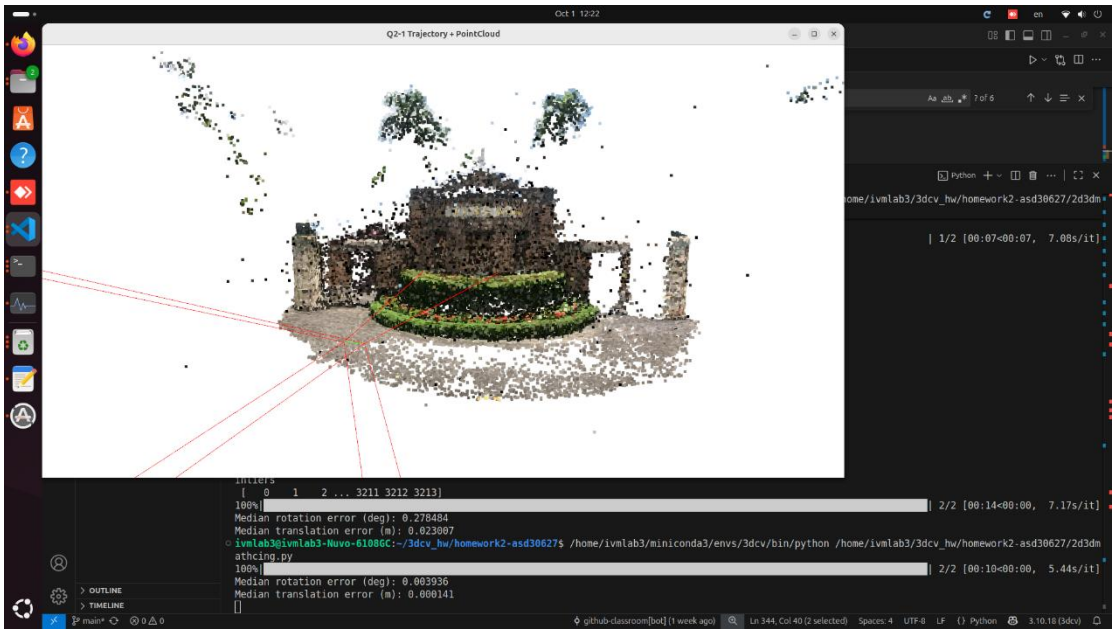
Problem 2
以下為我的第二題 YouTube link，我影片後面有放完整 output.mp4
https://youtu.be/mwuXAdTAsRA

2-1

一般版本的 slovepnp

1. 我拿 2d 與 3d 的資訊進來 pnpslover 後用 cv2.BFMatcher 做 KNN matching（k=2），再跑 Lowe ratio test 過濾誤配。
2. 拿經過以上方法後，合格的點用 solvePnPRansac 粗估位姿，只取 inliers 後再用 solvePnP 精化位姿
3. 把估計結果轉成相同型態，計算誤差，旋轉：rvec → 四元數 → 旋轉誤差（角度），平移：tvec → 與 tvec_gt 的 L2 距離
4. 把 (rvec, tvec) 轉成「相機到世界」的 4×4 變換矩陣，用來畫軌跡
5. Visualize 中，建點雲後畫每一幀相機的金字塔線框=>畫相機軌跡=>設定視角並顯示

   以下為結果:



手刻 ransec_pnp:

1. 比對建立 2D–3D 對應：BFMatcher.knnMatch(k=2) + Lowe ratio 0.75；若 good < 6 → 失敗。
2. 整理 uv_all（2D 像素）、Pw_all（3D 世界）。
3. RANSAC：

   先固定 seed、初始化 best_inliers/best_model/max_trials。

   迴圈直到 trials < max_trials：

   (1)隨機抽 **3** 對應。

   (2)用 p3p_numeric(Pw3, uv3, K) 取多組候選 (Rc, tc)。

   (3)對每個候選：用 project(Rc, tc, Pw_all) 投影全點（無畸變）。

   reproj < thresh_px 且 z > 0 判內點。

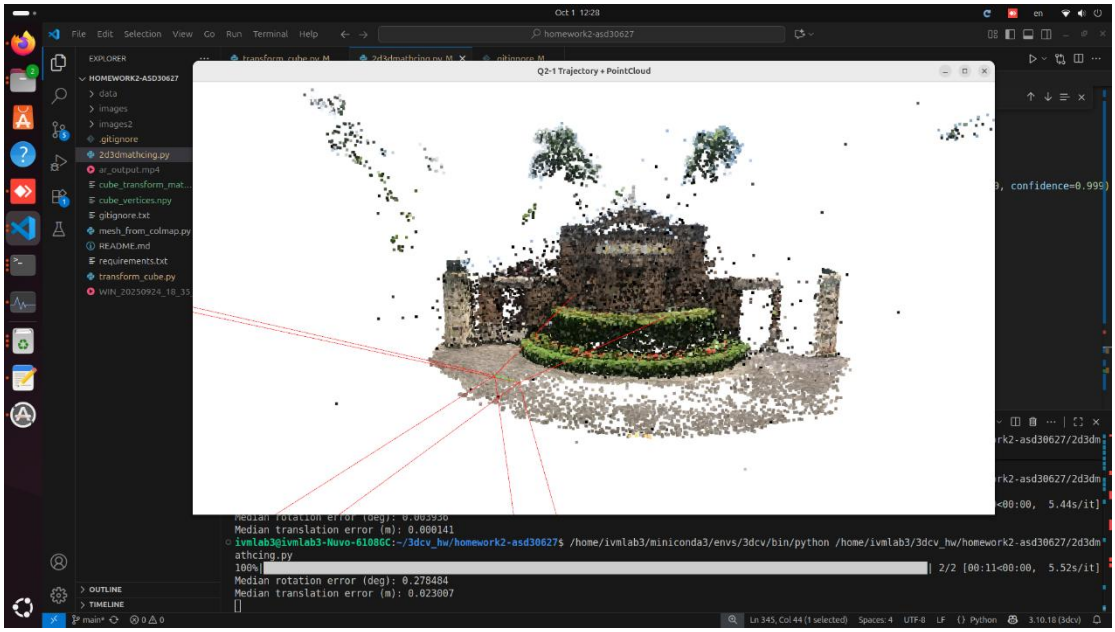
   若更好 → 更新最佳；用 w=|inliers|/N、樣本數 s=3 自適應更新

max_trials。

trials += 1。

4. 收斂與輸出：若 best_model 且 |best_inliers| ≥ 3 → 回傳 (rvec, tvec, inliers)；否則失敗。

以下為結果:



Pseudo code

```
FUNCTION ransec_p3p(query, model, K, thresh_px, max_iter, confidence, ratio=0.75, seed=1428):

  # 1) 建立 2D–3D 對應
  matches ← BFMatcher.knnMatch(desc_query, desc_model, k=2)
  good ← { m | m.distance < ratio * n.distance }
  IF |good| < 6: RETURN (False, None, None, [])

  uv_all ← [kp_query[m.queryIdx]]
  Pw_all ← [kp_model[m.trainIdx]]
  N ← |Pw_all|

  # 2) 初始化
  SET RNG(seed)
  best_inliers ← ∅
  best_model ← None
  trials ← 0
  max_trials ← max_iter

  # 3) RANSAC 主循環
  WHILE trials < max_trials:
    I ← 隨機抽樣 3 個索引（不重複）
    uv3 ← uv_all[I]; Pw3 ← Pw_all[I]

    candidates ← P3P_NUMERIC(Pw3, uv3, K)   # 多候選 (Rc, tc)

    IF candidates 為空:
      trials ← trials + 1
      CONTINUE

    FOR EACH (Rc, tc) IN candidates:
      (uv_hat, z) ← PROJECT_WITH_K(Pw_all, Rc, tc, K)   # 無畸變
      reproj ← L2(uv_hat - uv_all)   # 每點誤差
      inliers ← { i | reproj[i] < thresh_px AND z[i] > 0 }

      IF |inliers| > |best_inliers|:
```

```
            best_inliers ← inliers
            best_model ← (Rc, tc)

            w ← max(1e-9, |inliers| / N)
            s ← 3
            eps ← clamp(1 - w^s, 1e-12, 1 - 1e-12)
            max_trials ← min(max_iter, ceil( log(1 - confidence) / log(eps) ))

        trials ← trials + 1

    # 4) 輸出
    IF best_model 存在 AND |best_inliers| ≥ 3:
        (Rc, tc) ← best_model
        rvec ← Rodrigues(Rc)
        tvec ← tc
        RETURN (True, rvec, tvec, best_inliers)
    ELSE:
        RETURN (False, None, None, [])

FUNCTION p3p_numeric(Pw, uv, K,
                             max_iters=30, tol=1e-8,
                             init_scales=[1.0, 0.5, 1.5, 2.0],
                             cheirality_z_min=0, dedup_round=1e-6):

    f ← NORMALIZED_RAYS_FROM_UV_AND_K(uv, K)   # 3 條單位像射線
    base ← 平均世界邊長的量級估計
    solutions ← []
    seen ← ∅

    FOR s IN init_scales:
        λ ← [base*s, base*s, base*s]
        μ ← 1e-2

        # LM 迭代（殘差/雅可比對 λ，降低邊長差異）
        REPEAT up to max_iters:
            r ← RESIDUALS( λ , f, Pw)        # 不寫公式，只表示對三邊長的差
            J ← JACOBIAN( λ , f)              # 對 λ 的偏導
            Δ ← SOLVE( (JᵀJ + μI), -(Jᵀr) )
```

```
        λ_new ← λ + Δ

        IF NORM(r_new) < NORM(r):     # 接受
            λ ← λ_new; μ ← μ * 0.5
            IF NORM(Δ) < tol: BREAK
        ELSE:                         # 拒絕
            μ ← μ * 2.0

    IF 任一 λ ≤ 0: CONTINUE    # 物理解（正深度）

    Pc ← 組出三個相機座標點(λ, f)
    (Rc, tc) ← KABSCH_ALIGN(Pw → Pc)    # world→camera
    zc ← Z_COMPONENTS(Rc * Pw + tc)
    IF ALL zc > cheirality_z_min:
        key ← ROUND(Rc, tc, dedup_round)
        IF key NOT IN seen:
            seen.ADD(key)
            solutions.APPEND((Rc, tc))

RETURN solutions
```

Problem 2-1

實驗結果比較：

從實驗結果來看，手刻方法的表現較差，尤其在處理包含較多噪聲或數據不一致的情況下。儘管手刻方法使用了 RANSAC，這仍然無法保證在所有情況下都能夠達到 cv2 方法的穩定性。這是因為手刻方法過於依賴精確的初始化條件，以及迭代的數值優化過程，當數據中有較多不一致或誤差時，可能會導致收斂到錯誤的解。

討論與觀察:

手刻方法的優勢：

手刻的 P3P 方法允許更靈活的控制與自定義，特別是在數學模型和優化過程中。對於非常準確的數據，手刻方法可以達到較高的精度。

手刻方法的劣勢：

手刻方法的收斂性較差，尤其是當初始條件不夠準確或數據中包含較多噪聲時，會導致錯誤的解。需要手動調整參數，且計算時間較長。

cv2 方法的優勢：

cv2 的 P3P 方法基於現成的函數，具有較強的 robustness，並能處理實際應用中的噪聲。cv2 內建的 RANSAC 演算法有效過濾掉錯誤匹配，並且計算速

度較快。

cv2 方法的劣勢：

cv2 的方法對於極為精確的數據來說，可能無法提供手刻方法那麼精細的解答，並且當數據非常乾淨時，RANSAC 可能會丟失一些潛在的 inliers。

使用 LLM(Chatgpt)部分:

1. Visualize 整理數據的部分以及 o3d 套件如何實現 camera to world transformation
2. ransec_p3p 中的 while 迴圈如何處理 p3p_numeric 出來的點
3. p3p_numeric 中如何實現高斯-牛頓法的迭代、Kabsch **演算法**、寫與用雅可比矩陣等

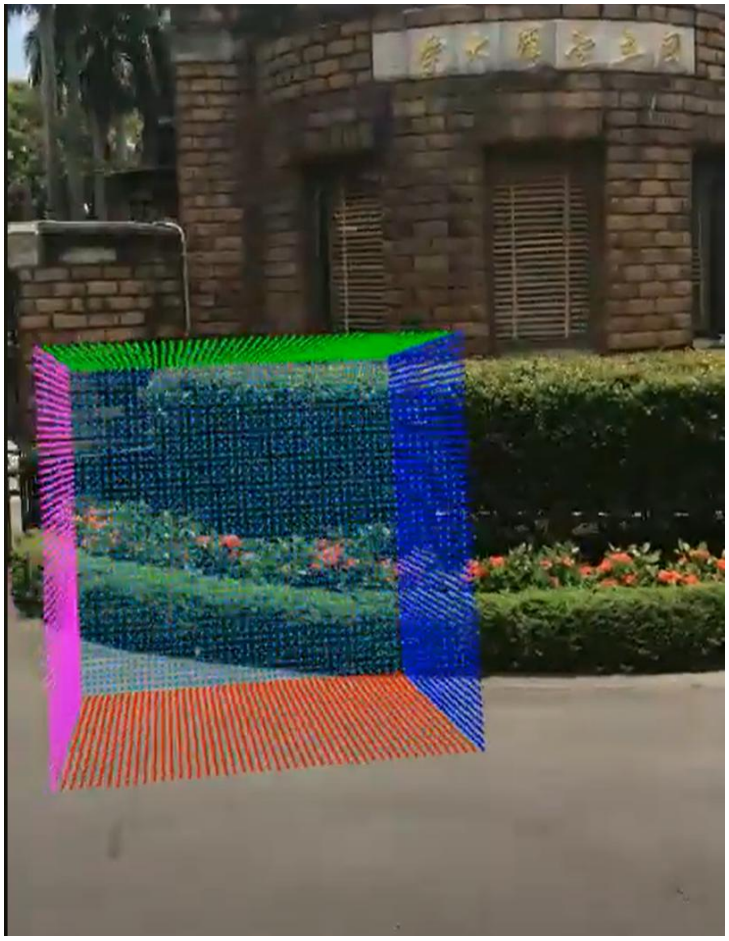
Problem 2-2

1. 存立方體變換矩陣 cube_transform_mat.npy（由 R_euler, t, scale）。
2. 存立方體局部點 cube_points.npy 與 顏色 cube_colors.npy。
3. 定義內參 K、畸變 DIST 設定影像資料
4. Load 回立方體資料、顏色從 RGB 轉 BGR。
5. 將立方體點雲變換到世界座標。
6. 由姿態欄位建立世界→相機的外參
7. 丟到相機座標檢查前後（正深度）
8. 點用 OpenCV 投影到像平面
9. 把投影點畫到影像上
10. 輸出影片

觀察與討論:

我覺得這個作業很有趣，因為我有買一個 AR 的遊戲機，可以玩射箭、音樂遊戲等等的，原來那些物體是這樣被建立出來了。

我有觀察到雖然我把方塊、有顏色的點雲重建了，可是在其他畫面還看到一團壓縮、閃爍、變動的殘影，我有做一些嘗試想要解決這個問題，但最後只是改善，沒辦法完全清除，所以我應該不知道真正導致這樣的原因



Youtube link https://youtu.be/mwuXAdTAsRA，我影片後面有放完整 output.mp4

使用 LLM(chatgpt)的部分:

1. 我寫的步驟有 bug，例如讀檔、資料處理等，有遇到困難會請他幫忙