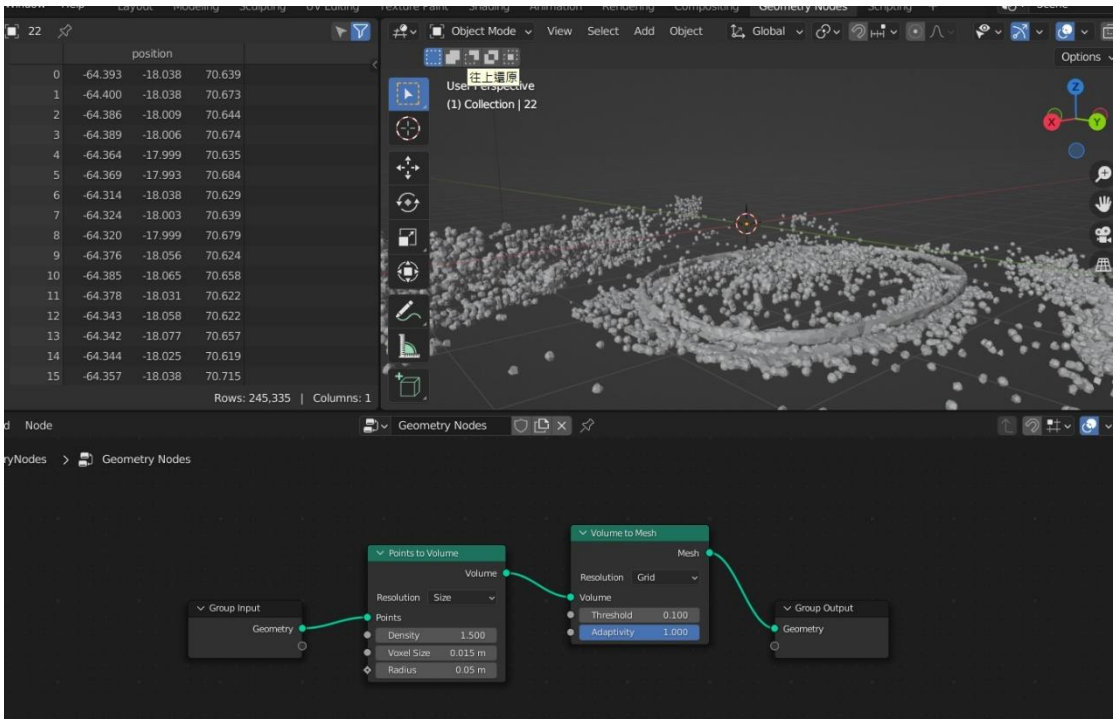
3DCV Homework 2 Report

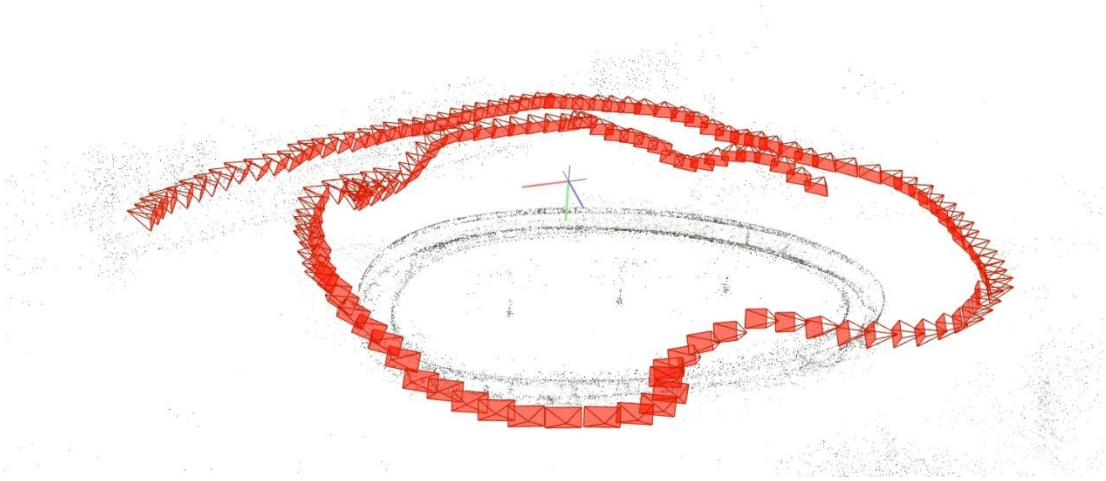
Problem 1:

Problem 1 最主要的部分其實我是在想辦法過濾 3D Mesh 重建的 Outliers，但效果實在不是很好，我用了 MeshLab 去做 Remove Isolated Pieces，但不知道為什麼會全部的 voxel 都被去除，所以後來我是在 Blender 先手動刪除明顯在遠處跟範圍外的 voxel，然後再用 MeshLab 去 merge 在 radius2 以內的 voxel 來降低噪點。(不過有些噪點的 cluster 還是存在，最多就是 merge 的小一點)。Noise 的量推估跟拍攝的角度還有過程有關，可能晃動的影響造成 data 的品質不是很好。降低完噪點後的點雲最後再送回 Blender，利用 Blender 內建的 Geometry Node 將點雲轉換成 Volume 後再轉換成 Mesh。

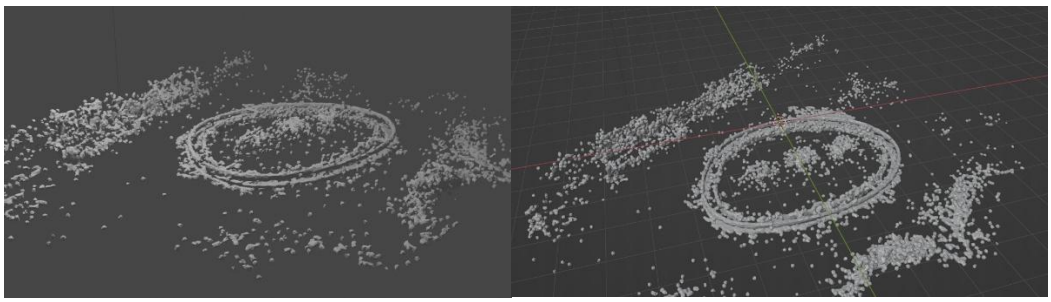


而使用 COLMAP 去提取點雲(Q1-1)的過程大致還算順利，最一開始我是直接把影片丟給 COLAMP 去使用 video frames 來做 reconstruction，然後他就爆炸了，我嘗試了幾次測試後(主要是拿助教提供的校門的 dataset)，發現應該是電腦性能的緣故跟一次吃太大的 Input frames 所以導致 COLMAP crash。所以我用 OpenCV 寫了一個簡單的提取 frmae 的 code(read_frame.py)，每 16 幀取一次的方式來降低 Input frame 的數量。讀取 dataset 的方式也改用 Individual image 後就有成功完成 Sparse reconstruction 了。

Demo: https://youtu.be/AQ1_OvjuWZE



COLAMP 重建結果



Blender Mesh 最終重建結果

Problem 2:

這次 Problem 2 其實最花時間的部分應該還是 RANSAC + P3P 的實作，尤其是 P3P 內部有需多數學公式的推導跟簡化，下面會著重在這部分進行說明。

P3P 演算法實作原理的細節，主要參考了網上這篇文章的說明: P3P 相機姿態估計數學推導，而 RANSAC 的部分則是以課堂上的講義為主。LLM 的部分使用了 GPT5 和 Gemini 2.5 (因為 debug 時 GPT 一直鬼打牆，我就又跑去問 Gemini)，主要是用來做 debug，也用來查詢 o3d、scipy、cv2、numpy 等一部分函數的功能跟範例。另外其實做到最後篩選出 inliers 後我不是很清楚再來要做什麼後，最後 LSE 的優化方式也是直接問 LLM 的 scipy.optimize.least_squares 的函式也是 GPT 推薦的。

那麼這之後會分成 3 個部分說明 RANSAC + P3P 的實作、error 計算和 visualization、以及 2-2 的 AR 渲染。

RANSAC + P3P 求解 R、T 矩陣的大致步驟如下 sudo-code 所示

Descriptors Matching and ratio test (這裡是直接搬助教作業 1 的 code)

#RANSAC 實作

For iter = 100:

    Sample 3 points

    R, T = Solve p3p()

    Calculate all project error by R, T #Project 3D point back to 2D point, and compare with ground truth 2D point

    Determine inliers and num of inlier

    Compare with max_inliers, if more, replace max_inliers

#LSE 最終優化

least_squares(R_max, T_max) #Use the R, T matrix with maximun inliers as initial

Solve p3p():

c12, c13, c23, d1, d2, d3 = get cosine()

#Get cosine between 3 points (WCS) and their ray line (OP1, OP2, OP3)

Dis 12, Dis13, Dis23 = sqrt(np.sum((P – P)**2)) #Get distance between 3 point

Solve quartic function by np.root() and remove complex and negative solutions

Solve a, b, c by np.root() as defined in lectures

By a, b, c, d1, d2, d3, we get 3D points in CCS, then we can solve and get RT by the transformation from CCS to WCS

Move the origin of both into the centroid of 3 points; thus, we remove the effect of translation, and calculate R by linear algebra

Use R to calculate T and return R, T

以上是大致流程，code 內還有一點簡單的註解，大致上最複雜跟花時間的應該還是 P3P 的流程還有如何解中間的一大串方程式，另外我也查了不少資料確認

camera matrix 和 distortion 影響的時間點(運算是要先削去哪個還有 project 回去 2D 點時計算的順序)。另外我解參數 y 時 (算 P3P 內的 a, b, c 時)，我改用了 sympy 的 package，這是 LLM 我用的，跟 np.root()一樣是解方程用的 function，主要是我 debug 時不知道為什麼那段一直出錯(可能我有什麼 key 錯了)，反正 Gemini 推薦我用後就跑過去了，所以我也就不改回 np.root()了，另外就是他那個參數有點醜，比較難算。

實作時，是分別先跟 cv2 的 solveP3P 和 solvePnPRansac 兩個 function 對答案，先是確認 P3P 的演算法正確後，再確認 RANSAC + LSE 後的答案是否正確。另外因為有使用 LSE 做優化，這部分會跑得比較久一點，希望能加速的話可以試著拉高 ration test(0.25↓)和 RASANC error(0.35↓)的容許標標準，降低最終 inliers 的點對數減少 LSE 優化的計算時間

再來是 error calculation 和 visualization，error 比較麻煩的是旋轉誤差，位移誤差，直接算歐是距離就好，旋轉的話，因為四元數表示的(x, y, z, w)中 w 正好表示 cosine，所以算出四元數後 normalize 再丟 arccos()會比較方便，這邊我覺得角度值會直觀點，但因為 $\cos(0) = 0$，直接拿 cos 當誤差好像也行? 總之我的作法是先將兩個 rvec 轉成 Rotation matrix，然後$R_{est} \cdot {R_{gt}}^{-1} = R_{diff}$ ，轉成 quaternion 後如上所述計算誤差角度。

$$Median\ rotation\ error: 0.0005138637276310517\ (Angle)$$

$$Median\ translation\ error: 0.0012402617321008527\ (Distance).$$

旋轉及位移誤差中位數

Visualization 的部分是先以[0, 0, 0, 1], [0.15, 0.1, 0.4, 1], [-0.15, 0.1, 0.4, 1], [0.15, -0.1, 0.4, 1], [-0.15, -0.1, 0.4, 1]這五點構成的一個朝向+z 軸的四角椎為目標(CCS)，反投影至 WCS，而反投影矩陣就是$-R^{-1} \cdot T$，來計算這五點的 WCS 座標，最後用 o3d 畫出來。o3d function 的使用主要是從助教 transform cube 的 sample code 搬，還有從作業 slide 提供 o3d 官網找方法。(也有問了下 GPT 點跟線的分別呈現方式)

相機位置可視化結果

最後是 2-2 的 AR 渲染，這部分和 visualization 也有點像，先準備好一個在 WCS 座標下的 cube voxel，位置預設在 (0, 0, 0)到 (1, 1, 1)的立方區域，實際擺放位置、大小和旋轉，則是使用 transform cube 跑出來的(rotation,translation,scale)調整。放置完 cube 後計算到相機(-T)的距離，然後再進行排序。最後是每個 voxel 投影到相片上，實際的操作跟再 RANSAC 找 inliers 算誤差的動作是一樣的，需要額外注意的是在 RT 矩陣運算完後要先檢查 CCS 中 Z 座標是否大於 0(位於相機前)，若沒有的話則不需在畫面中著色。再來是計算回 2D 座標，這邊則是檢查座標是否若相片的範圍內[(0, width), (0, height)]，若超出範圍的話也無需著色。

然後這邊提一個比較好笑的 bug，一開始我是一個 pixel 對一個 voxel 的，結果因為點太小根本看不到。我找了老半天 bug 想說都怎麼 cube 沒畫出來，後來把 image 上畫的點調大(4*4 pixel)就看得到了。

最後附上 demo 的 youtube 連結 https://youtu.be/xEeA6rmdoU8