

# Final Project ORB-SLAM & COLMAP

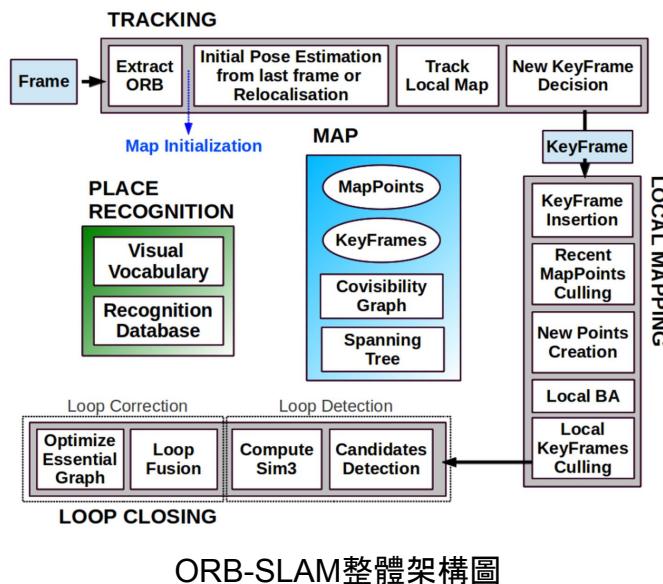
無人機自動飛航與電腦視覺概論

第4組／游騰德、鄭程哲、唐宇謙

## 比較兩種方法之間的差異

### ORB-SLAM

ORB-SLAM是一個非常知名的視覺SLAM系統，其核心是使用ORB (Oriented FAST and rotated BRIEF) feature，基於ORB的feature matching和relocalization，讓他擁有較佳的視角不變性。此外，新增三維點的feature matching效率更高，因此能更及時地擴充場景。



整個ORB-SLAM是三大塊、三個流程同時執行，分別是追蹤、建圖、閉環檢測。

### 追蹤 (Tracking)

主要是從影像中提取ORB feature，根據上一frame進行pose estimation，或者透過全域性重定位初始化位姿，然後追蹤已經重建的local map，最佳化位姿，再根據一些規則確定新的key frame。

## 建圖 (LocalMapping)

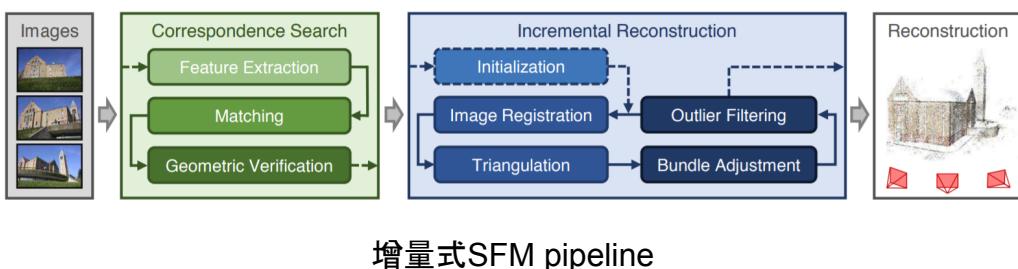
主要完成local map構建。包括對key frame的插入，驗證最近生成的mapping點並進行篩選，然後生成新的mapping點，使用區域捆集調整(Local BA)，最後再對插入的key frame進行篩選，去除多餘的key frame。

## 閉環檢測 (LoopClosing)

主要分為兩個過程，分別是閉環探測和閉環校正。閉環檢測先使用 WOB 進行探測，然後透過 Sim3 演算法計算相似變換。閉環校正，主要是閉環融合和 Essential Graph 的圖最佳化。

## COLMAP

COLMAP是一款結合SFM (Structure-from-Motion, 從運動中恢復結構) 和MVS (Multi-View Stereo, 多視圖立體) 的三維重建Pipeline, 可用於稀疏重建和稠密重建。其使用的SFM演算法會先進行feature extraction/matching, 以及後續的幾何校驗來取得 scene graph, 作為後續增量式重建的基礎。



## 檢索匹配

輸入一系列的圖片，可以得到幾何校驗後的圖像匹配關係。COLMAP中為了使結果盡可能準確，用SIFT進行了feature matching，輸出的scene graph中圖像是node，幾何校驗後的配對是edge。

## 增量式重建

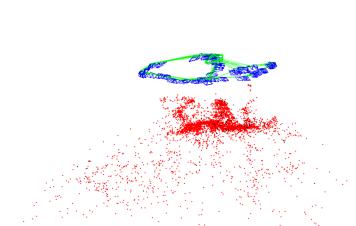
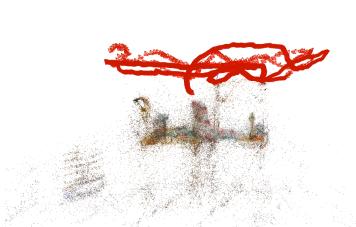
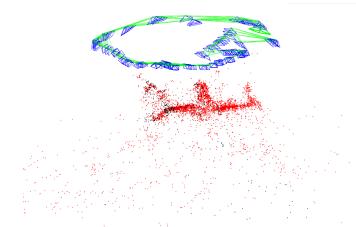
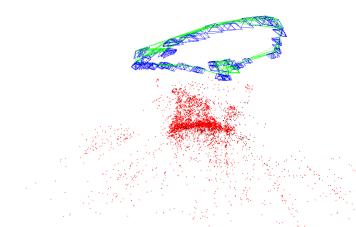
輸入scene graph，可以得到相機pose和3D標點。首先會進行初始化，選擇scene graph中相機間可視區域多的兩視角進行。接著根據已有的SFM模型進行pose estimation。特別的是，這裡用了新提出的三角化場景的方式，提高了重建場景結構的完整性與 robustness。接著用捆集調整(BA)做優化，消除累積誤差。由於增量式重建僅會影響鄰

近frame，所以沒有必要每次都做全域捆集調整(Global BA)，只有當模型成長到一定程度以後才會用，可以分擔時間消耗。

## 模型比較

在模型比較的部分，我們的實驗是先將ground truth image丟進COLMAP訓練模型，再分別將相同的testing data丟進COLMAP以及ORB-SLAM進行測試，而在COLMAP訓練過程中，我們不使用預設的Exhaustive feature matching，而是使用效率比較好的Sequential feature matching，因為我們的image數量足夠多，而且這種影片類型的圖片較為連續與重疊，不需要使用Exhaustive feature matching將每張圖片與其他全部圖片對照處理，也能得到不錯的模型結果。

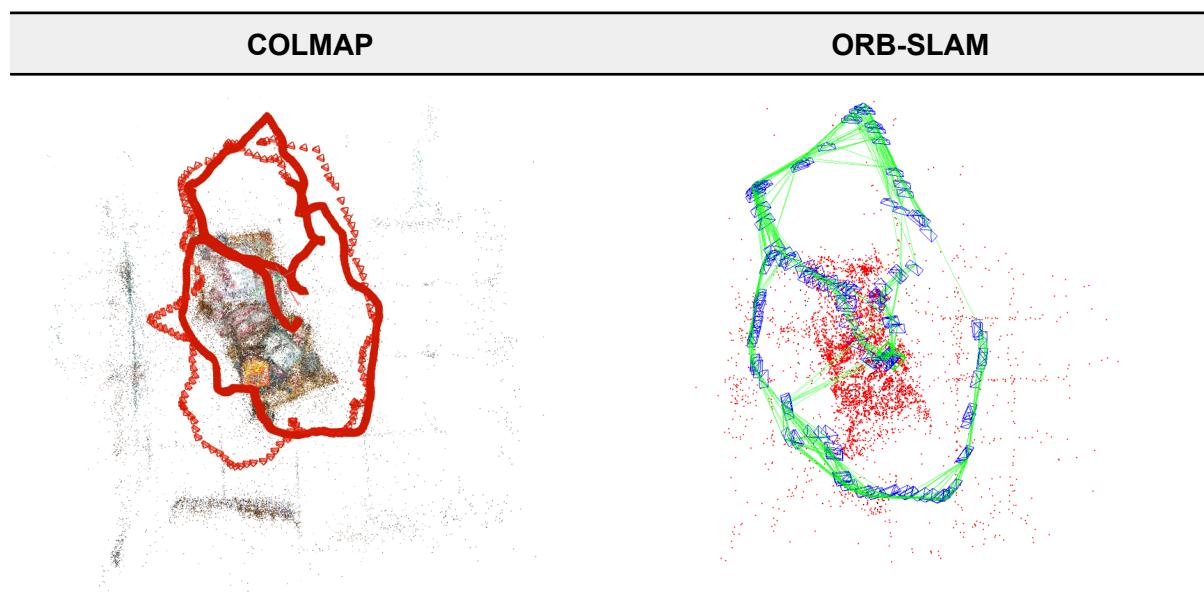
下面的圖表我們比較原始圖片以及COLMAP、ORB-SLAM各自建出來的模型效果，模型截圖有稍微調整視角以利模型識別，與原始圖片視角有些許差距。

Ground Truth	COLMAP	ORB-SLAM
		
		
		

由實驗結果可知，滿明顯可以看出COLMAP的模型比較完整，更能還原初始圖片的輪廓，得到的Key frame座標點也較多；此外，COLMAP還有針對顏色作判別，從上表中可以看到COLMAP建立出的模型是有還原顏色的，這點也幫助我們更能清楚看見物件的輪廓。反之，ORB-SLAM雖然也能夠建立出滿完整的模型，但座標點少以及少了顏色判別的缺點，讓ORB-SLAM的模型只能看出一個模糊的輪廓。

## 相機定位

在相機定位的部分，我們統一將模型以俯瞰的視角截圖來比較兩個方法在相機定位下的差異，最後在將模型中得到的座標點進行轉換與繪圖，將COLMAP模型當成ground truth與ORB-SLAM模型計算error。



COLMAP圖中有兩條軌跡，較稀疏的一條是使用training data訓練的軌跡，而較稠密的一條是用testing data測試的軌跡，而ORB-SLAM就是單純用testing data訓練的相機定位軌跡。所以我們將COLMAP稠密的testing軌跡與ORB-SLAM比較，可以看出兩者在相機定位的表現並沒有相差很多，ORB-SLAM雖然key frame的數目較少，但也能夠完整重建相機的位置。

## ORB-SLAM實驗過程影片

[ORB-SLAM實驗過程影片連結](#)

## 座標轉換

由於ORB-SLAM和COLMAP的座標系統不同，為了要比較兩者的差異，我們需要對ORB-SLAM的座標進行轉換，以符合COLMAP的座標系。

轉換的過程包含以下的動作：平移、縮放、旋轉。其中我們選擇ORB-SLAM的第一個點當作基準點，稱為ORB0，與其對應的COLMAP點稱為COL0，也就是說轉換完成之後，ORB0會與COL0完全重合。除此之外，我們尚須另一點來作為校正，我們選擇ORB-SLAM的第二個點來作為校正點，稱為ORB1，與其對應的COLMAP點則稱為COL1。

以下過程為座標轉換示意，綠色部分是ORB-SLAM，藍色部分為COLMAP，橘色為旋轉軸和旋轉角，為了有更佳的視覺效果，點的標示的位置與實際位置並不相同。

### 平移

為了能夠縮放和旋轉，我們首先將2個模型的中心點重合併放置在原點座標上，作法是將2個模型的所有點都看作是向量，並且將所有向量減去各自模型的平均向量。

### 縮放

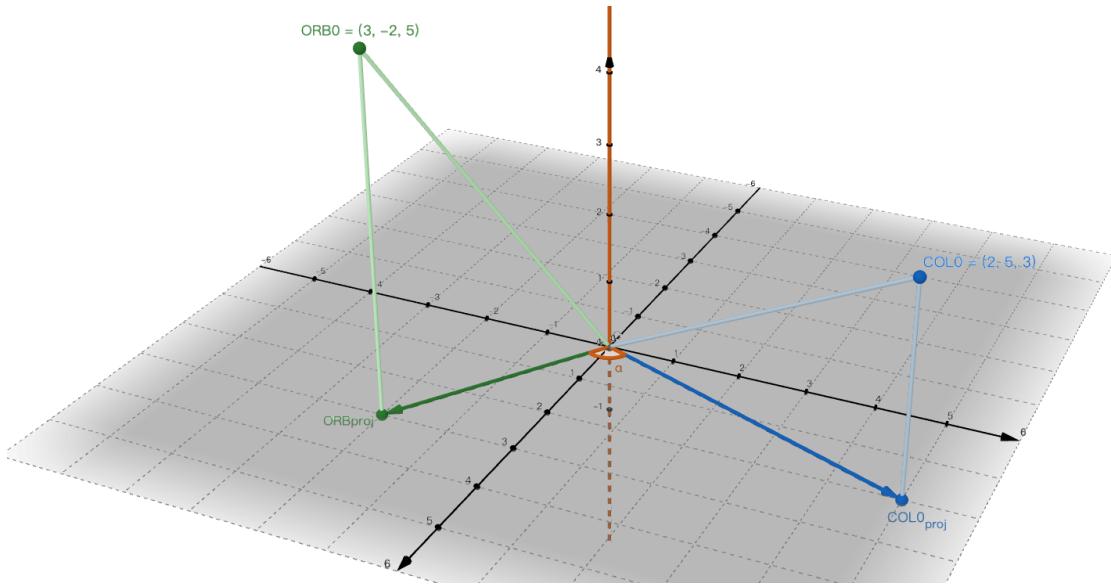
對齊中心點之後，我們將ORB-SLAM的所有向量縮放 $scale$ 倍。其中  
 $scale = |COL0|/|ORB0|$ ，如此一來，我們便能確保ORB0與COL0的長度一致，也就是兩個模型的比例大小一致。

### 旋轉

我們採用羅德里格旋轉公式的原理，在給定轉軸向量 $v$ 和旋轉角度 $\theta$ 後產生旋轉矩陣 $R$ ，利用此一方法我們將模型進行3次旋轉。

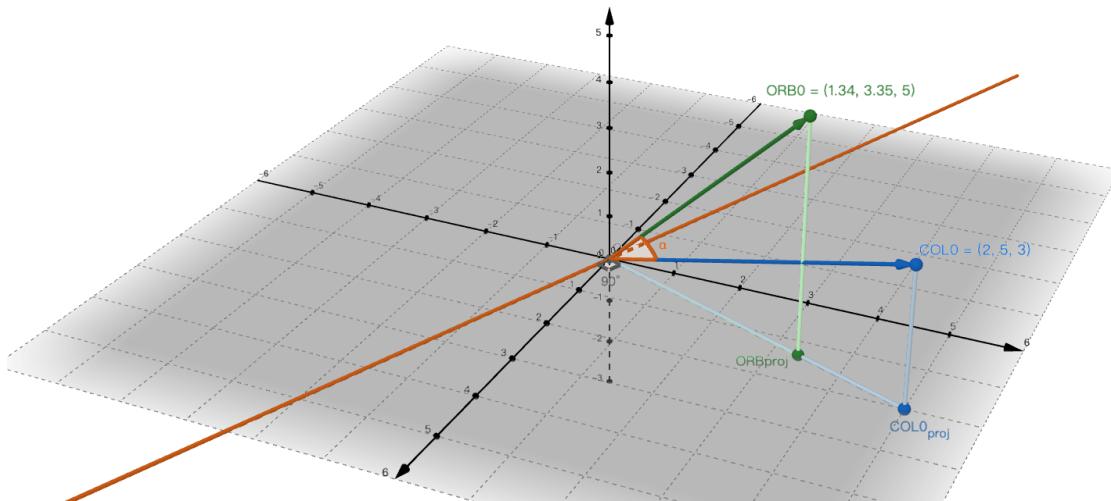
### 旋轉1

首先，我們轉動ORB-SLAM模型，使得ORB0的XY平面投影向量（深綠向量）與COL0的XY平面投影向量（深藍向量）平行，此一旋轉之轉軸為 $z$ 軸（橘線），旋轉角度 $\theta$ 為兩平面投影向量之夾角（橘角）。如下圖所示：



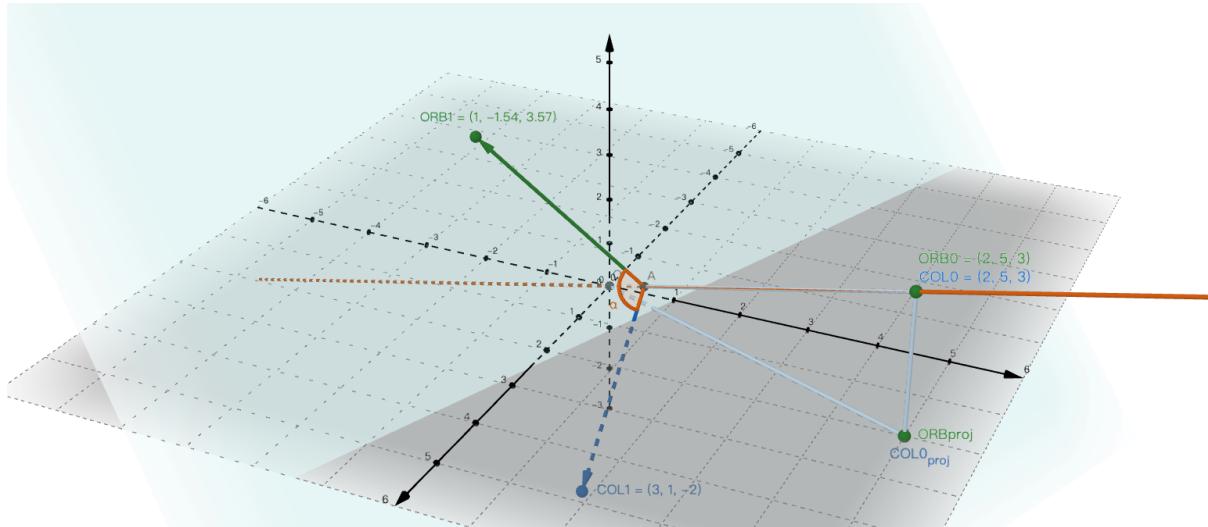
## 旋轉2

接下來，我們轉動ORB-SLAM模型，使得ORB0與COL0重合。此一旋轉之轉軸為在XY平面上與ORB0的XY平面投影向量垂直之直線(橘線)，旋轉角度 $\theta$ 為ORB0向量(深綠向量)與COL0向量(深藍向量)之夾角(橘角)。如下圖所示：



## 旋轉3

完成上方兩個旋轉之後，ORB0與COL0已經重合，接者，我們以ORB0向量方向的直線(橘線)為轉軸，使得ORB1與COL1盡可能靠近。此一旋轉對於ORB1而言，旋轉中心點為ORB1投影至轉軸的點(A點)，因此旋轉角度 $\theta$ 為向量( $A, ORB1$ )(深綠向量)與向量( $A, COL1$ )(深藍向量)的夾角(橘角)。如下圖所示：



## 平移

完成上述步驟之後，我們將COLMAP模型的中心點平移回到原本位置，方法是將本COLMAP模型加上原本的COLMAP平均向量。而ORB-SLAM模型也加上原本的COLMAP平均向量，保持ORB0與COL0重合。

至此，座標轉換在經過上述步驟後便全部完成。

## 座標轉換實例

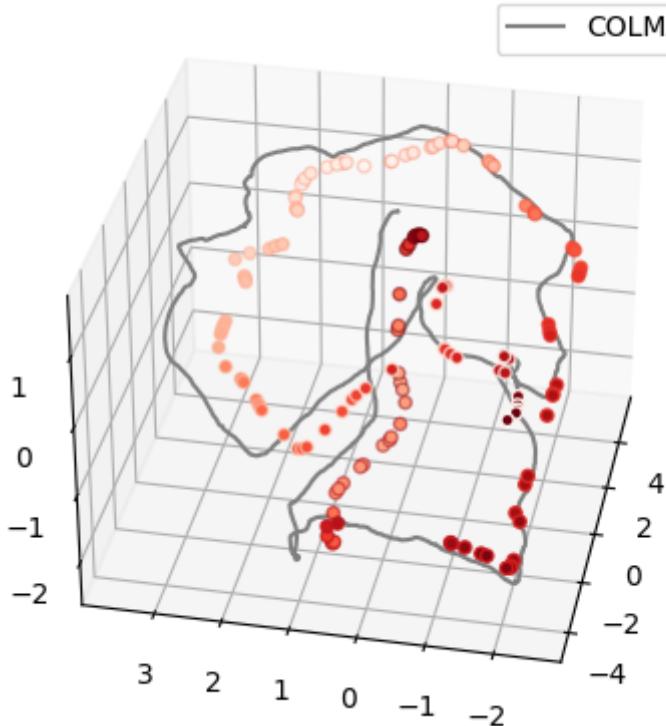
以下為實際的點在座標轉換過程中log出來的結果，數值四捨五入至小數點下第二位。

	ORB0	COL0
轉換前	(0, 0, 0)	(-2.75, -1.89, -0.06)
平移、縮放後	(0.22, 2.66, -2.10)	(-2.78, -1.94, -0.04)
旋轉1後	(-2.19, -1.52, -2.10)	(-2.78, -1.94, -0.04)
旋轉2後	(-2.78, -1.94, -0.04)	(-2.78, -1.94, -0.04)
旋轉3後	(-2.78, -1.94, -0.04)	(-2.78, -1.94, -0.04)
平移後	(-2.75, -1.89, -0.06)	(-2.75, -1.89, -0.06)

	ORB1	COL1
旋轉3前	(-2.64, -2.06, -0.04)	(-2.63, -2.06, 0.04)
旋轉3後	(-2.64, -2.05, 0.03)	(-2.63, -2.06, 0.04)

## 兩個相機軌道的圖

下圖紅色點為ORB-SLAM的座標轉換結果，灰色線為COLMAP ground truth。



## 誤差與統計

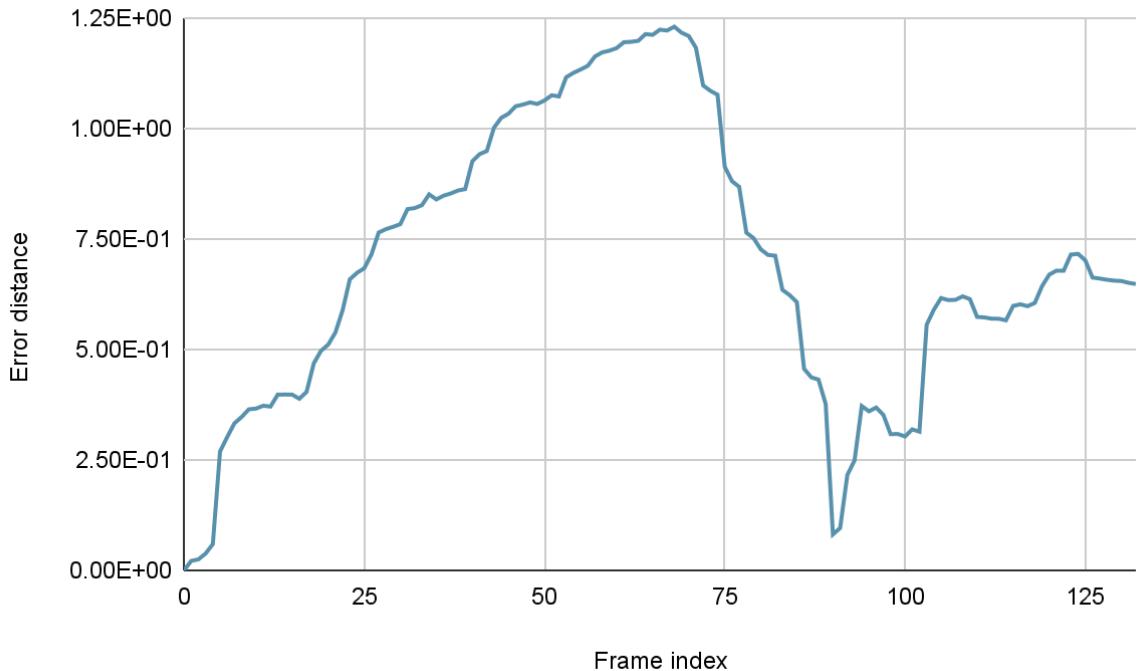
### 平均誤差

轉換後，ORB-SLAM與COLMAP的平均誤差為0.6946736225437562。

### 誤差分析

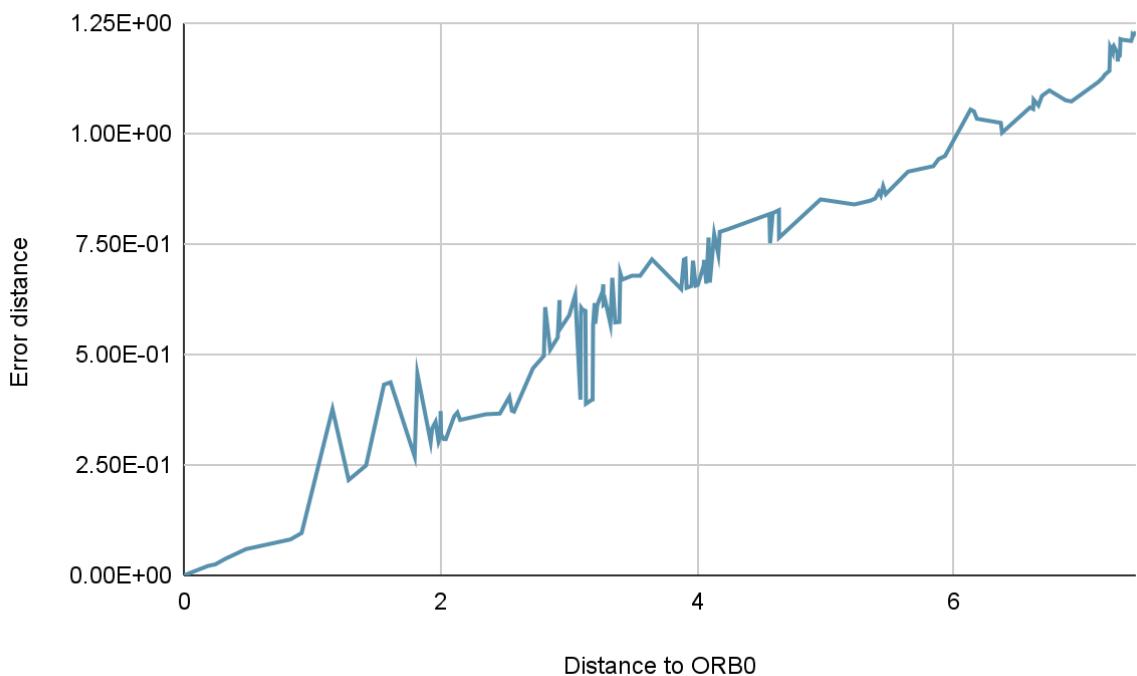
#### 誤差與拍攝時間順序比較

下表中橫軸為ORB-SLAM中各點的時間序，縱軸為該時間點的誤差，從圖中不足以看出（或不明顯）因為時間越往後而使得誤差越大的對應關係。



### 誤差與距離第一點的距離比較

下表中橫軸為ORB-SLAM中各點與ORB0的距離，縱軸為該距離下的誤差，從圖中得以看出，距離越遠、誤差越大，因此可以推論造成誤差的主要因素應和「與第一點的距離」有關。



## 結論與未來探討

這次的Final Project中，我們學會了使用ORM-SLAM和COLMAP兩種視覺定位方式，了解到其各自的運作原理，並且比較了兩種方式建模所產生的差異。此外，也研究了座標點在三維空間中的轉換技巧，包含模型的平移、縮放，以及使用羅德里格旋轉公式的原理來旋轉空間中的點，最後統計了兩種模型的誤差並進行分析。

除此之外，在安裝過程中，我們也遇到許多困難，例如spec中僅提到ORB-SLAM2在Windows和Linux的安裝方式，而我們則是設法使macOS上也能運行。除了需要下指令安裝OpenCV與Cmake之外，由於版本因素，還需要更改ORB-SLAM、Pangolin和DBoW2中的CMakeLists，並需在部分cc和hpp檔案中的增加include標頭，才得以成功編譯。另外，COLMAP 3.6版本在macOS上或因動態連結問題而無法執行，因此我們使用COLMAP 3.5 Release版本來取代。

將平面的圖像轉換為立體空間中的點本就不易，而在這次實驗當中，可以看出使用了不同方式的ORM-SLAM和COLMAP，大致都能產生不錯的結果。我們或許可以再設法克服特徵點較少之處的缺陷，例如牆或地版，並將點雲進一步轉變成為有意義的線及面，來進行更多應用。在未來，我們認為當這類視覺定位的技術日益成熟之後，將可以應用於室內導航、自動駕駛等更為廣泛的領域。