

無人機自動飛航與電腦視覺概論

Final Project: ORB-SLAM & COLMAP

第一組 組員:黃聖偉 蔣沅均 鄒年城

視覺定位方法

● 方法介紹

基本上視覺定位就是feature matching的應用。

本次實驗使用SLAM (simultaneous localization and mapping)方式, 利用連續的影像來同時建地圖與定位自身。

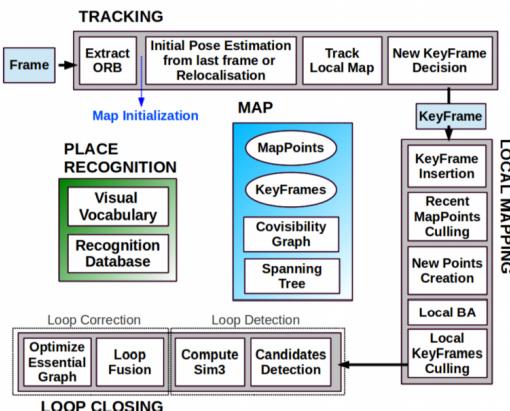
這種方法因為不需要事先蒐集資料, 因此對陌生場景適應性比較強, 但同時精準度也會比較差(因為估測誤差會持續累積), 而且因為需要同時建模與定位所以難度較高, 為了達成此一目的, 必須利用蒐集到的影像的「連續性」。

○ ORB-SLAM

- 這是一個只需要單鏡頭相機的視覺定位演算法, 能夠在沒有慣性、深度資訊等額外輔助下依然有良好表現。
- 如圖所示, 主要由三個部分組成:

Tracking, Mapping, Loop Closing

- **ORB-SLAM** (*IEEE Transactions on Robotics, 2015.*)
 - only using camera, based on ORB feature



(擷取自上課投影片p12)

1. Tracking

擷取特徵，並與地圖進行比對，以此建構出自身的位置。

其中最主要的部份為Feature detection – **ORB**

- 從本課程lab11的實驗結果可知ORB的計算速度非常快，使得此演算法能有real-time的應用價值。
- 經過對FAST演算法的修改後(oriented FAST)，ORB獲得較好的scale/rotation invariance特性。

2. Local mapping

利用Tracking找出的KeyFrame進行地圖更新。

小缺點是建出來的地圖是sparse representation，所以人較難以此看出地圖上的物件結構。

3. Loop closing

視覺定位這種相對性的定位方法會有誤差累積的問題，因此需要有一個降低誤差的手段。

此法概念為，比較現在的frame和過去見過的景象是否相似，如果相似就代表現在位置很可能是過去見過的那個位置，以此就可以消除這段行徑期間所產生的誤差。

○ COLMAP

此演算法基本上由兩大部分組成：

Structure-from-Motion (SfM) + Multi-View Stereo (MVS)

- Structure-from-Motion

1. Feature detection - **SIFT**
2. Feature matching

因為本次為自己錄影，所以選擇sequential matching即可

3. 建立場景圖與位置

經過以上步驟即可獲得sparse representation與相機位置

(本次lab只做到此步驟)

- Multi-View Stereo

利用SfM的輸出與圖片融合，做出dense 3D場景圖。

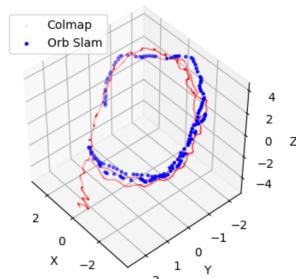
reference: <https://colmap.github.io/tutorial.html>

● 實作差異比較

	優點	缺點
ORB-SLAM	運行速度快，接近realtime。	需要在影片撥放時才抓取特徵，導致影片後期才開始定位。 會漏掉一些特徵比較不明顯的照片。
COLMAP	可以先訓練特徵，所以可以得到整個影片的相機位置。 幾乎不會漏掉照片。	運行時間非常久，60秒影片要跑一整天。

實作流程

- 前置步驟
 - i. 拍攝SLAM要使用的影片
 - ii. 使用python將影片切成多張照片
 - iii. 取出一些照片給colmap訓練用
 - iv. 建立紀錄timestamp與照片檔案關係的rgb.txt
- ORB-SLAM
 - i. 照著有點複雜步驟編譯好ORB-SLAM
 - ii. 執行ORB-SLAM，ORB-SLAM會在執行時抓取特徵，抓取成功後再計算相機軌跡
 - iii. ORB-SLAM輸出相機軌跡
- COLMAP
 - i. 下載COLMAP
 - ii. 開啟新的COLMAP專案
 - iii. 利用前置步驟取出的照片訓練COLMAP
 - iv. 利用COLMAP建立出剩下照片的相機軌跡
 - v. 將COLMAP建立的NVM相機軌跡輸出
- 後製處理
 - i. 讀取並parse ORB-SLAM相機軌跡
 - ii. 讀取並parse COLMAP nvm相機軌跡
 - iii. 將ORB-SLAM相機軌跡做線性轉換來與COLMAP對齊
 - iv. 將兩者相機軌跡利用matplotlib繪製成3D圖形
 - v. 計算對齊相機軌跡的相關係數平方(R square)



繪製出的3D圖形

程式內容

● ORB-SLAM transform

- 使用伸縮矩陣、旋轉矩陣、平移矩陣來將ORB-SLAM座標與COLMAP座標mapping在一起。透過伸縮矩陣與三軸旋轉矩陣相乘，再加上平移矩陣，可以大致將兩者的座標對齊。
- 下圖擷取程式中 `orbslamTransform(traj, params)` 函數中各種矩陣的程式碼，最後得到一個transfer矩陣與原位置相乘再平移來完成座標轉換。

```
posMatrix = np.array([[x],  
                     [y],  
                     [z]])  
  
scaleMatrix = np.array([[xScale, 0, 0],  
                      [0, yScale, 0],  
                      [0, 0, zScale]])  
  
XrotateMatrix = np.array([[1, 0, 0],  
                           [0, np.cos(xRad), -np.sin(xRad)],  
                           [0, np.sin(xRad), np.cos(xRad)]])  
  
YrotateMatrix = np.array([[np.cos(yRad), 0, np.sin(yRad)],  
                           [0, 1, 0],  
                           [-np.sin(yRad), 0, np.cos(yRad)]])  
  
ZrotateMatrix = np.array([[np.cos(zRad), -np.sin(zRad), 0],  
                           [np.sin(zRad), np.cos(zRad), 0],  
                           [0, 0, 1]])  
  
shiftMatrix = np.array([[xOffset],  
                         [yOffset],  
                         [zOffset]])  
  
transferMatrix = \  
    np.matmul(ZrotateMatrix,  
              np.matmul(YrotateMatrix,  
                        np.matmul(XrotateMatrix, scaleMatrix)))  
  
newPosMatrix = np.matmul(transferMatrix, posMatrix) + shiftMatrix
```

● Auto tuning

- 由於相機軌跡的圖形不是對齊XYZ軸，所以要將兩個圖形旋轉、伸縮、平移對齊非常困難與不直覺，因此我使用程式以窮舉角度、平移、伸縮的方式自動找到R square最高的位置，一開始需要將窮舉step設定的比較大才能在合理時間找到一個最佳位置，之後再將範圍與step縮小來fine tune，就能使R square越來越高，最後R square約為0.98，colmap與Orb slam幾乎完全對齊。
- 下圖擷取程式中 autoTuneParam() 穷舉的程式，其中 range 是根據上一次 autotune 的結果來調整，不過 step 是我手動調整的。

```
xRotateRange = range(
    self.param["degree"][0]-5, self.param["degree"][0]+5)
yRotateRange = range(
    self.param["degree"][1]-5, self.param["degree"][1]+5)
zRotateRange = range(
    self.param["degree"][2]-5, self.param["degree"][2]+5)

xOffsetRange = np.arange(
    self.param["offset"][0]-0.2, self.param["offset"][0]+0.2, 0.1)
yOffsetRange = np.arange(
    self.param["offset"][1]-0.2, self.param["offset"][1]+0.2, 0.1)
zOffsetRange = np.arange(
    self.param["offset"][2]-0.2, self.param["offset"][2]+0.2, 0.1)

scaleRange = np.arange(
    self.param["scale"][0]-0.2, self.param["scale"][0]+0.2, 0.1)

bestParam = self.param
bestR2 = self.calculateR2()

for scale in scaleRange:
    for xRotate in xRotateRange:
        for yRotate in yRotateRange:
            for zRotate in zRotateRange:
                for xOffset in xOffsetRange:
                    for yOffset in yOffsetRange:
                        for zOffset in zOffsetRange:
                            self.param = {
                                "offset": (xOffset, yOffset, zOffset),
                                "degree": (xRotate, yRotate, zRotate),
                                "scale": (scale, scale, scale)
                            }

                            transferOrbSlamTrajectory = orbslamTransform(
                                self.orbSlamTrajectory, self.param)

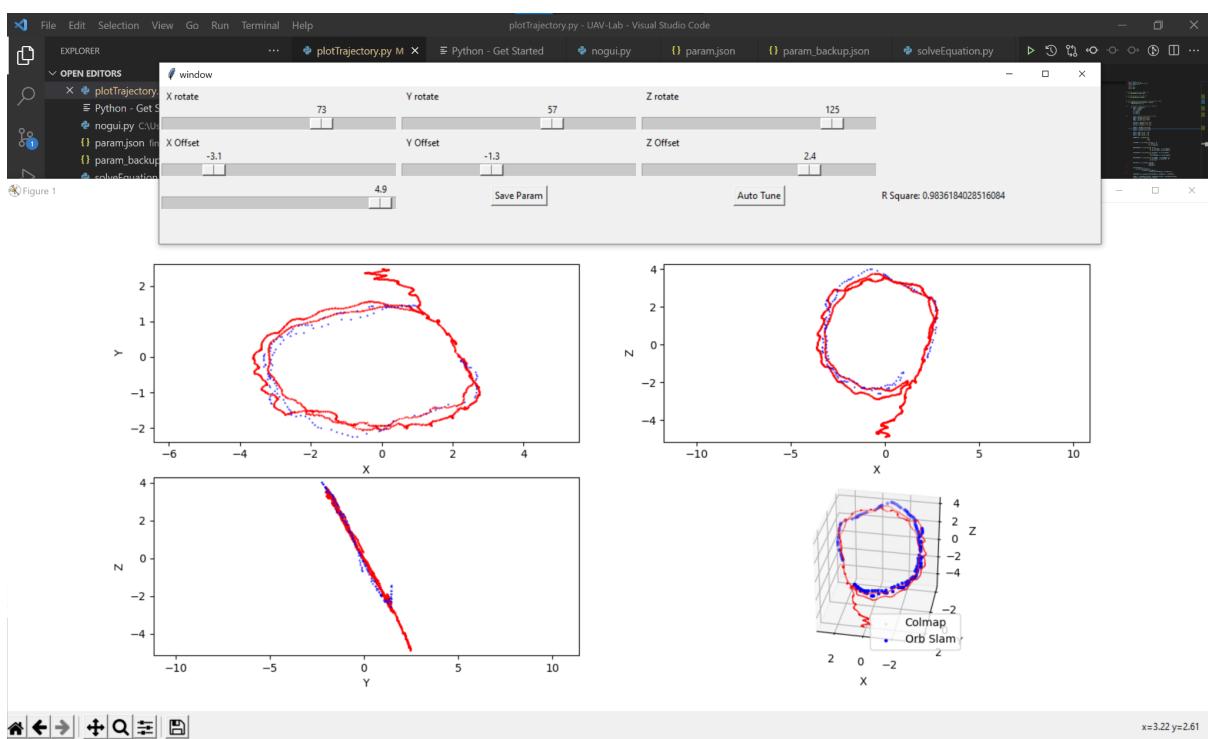
                            print(self.param)
                            r2 = self.calculateR2()

                            if(r2 < bestR2):
                                bestParam = self.param
                                bestR2 = r2

print("Best Param: ",end="")
print(bestParam)
print("Best R Square: ",end="")
print(bestR2)
```

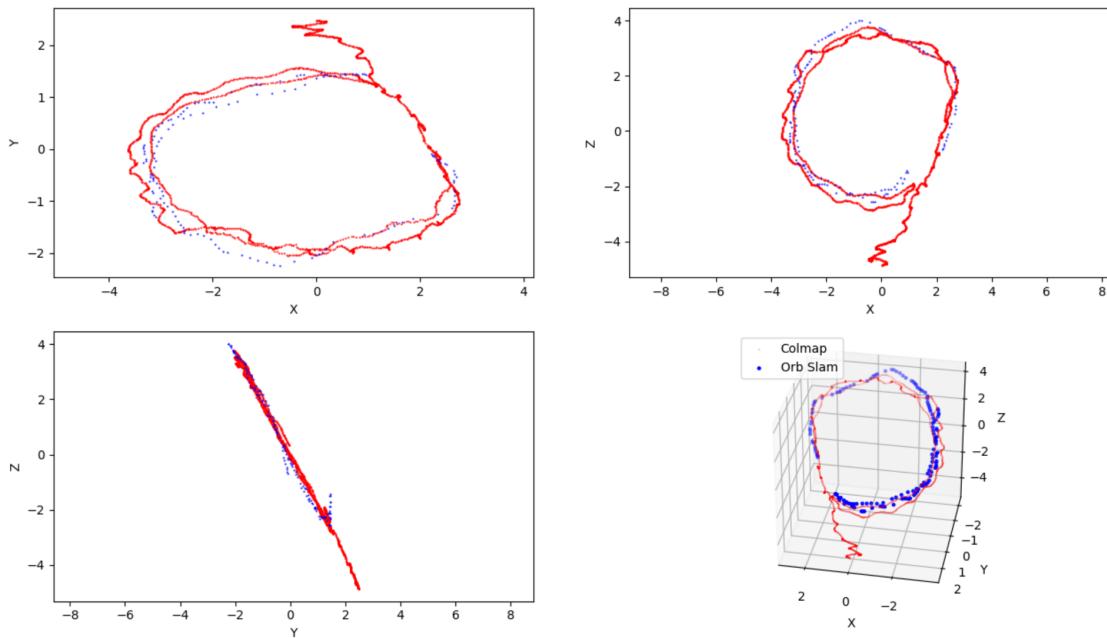
程式介面

- 使用tkinter library來建構gui
- 利用scale物件來挑整orb slam座標的選轉、位移、伸縮等參數。
- 上方三個scale滑桿 Xrotate、Yrotate、Zrotate分別用來控制ORB-SLAM座標以X、Y、Z軸為選轉軸的轉動角度。
- 中間三個scale滑桿 Xoffset、Yoffset、Zoffset分別用來控制ORB-SLAM座標X、Y、Z座標的平移。
- 下方scale滑桿用來挑整ORB-SLAM座標縮放比例
- Save Param按鈕用來儲存現在滑桿的每個參數至param.json。
- Auto Tune按鈕用來開始auto tune, 不過我沒有用thread寫, 所以按下auto tune後gui會卡住, 但python程式有繼續執行auto tune的運算。
- R square用來顯示現在參數下ORB-SLAM與COLMAP的相關係數平方R Square值。
- 程式開啟的第二個視窗則是matplotlib所繪製出的結果。



建模與定位結果比較

- 紅色為COLMAP，藍色為ORB-SLAM。
- 前三張圖為XY平面YZ平面及XZ平面的投影，可以藉由這三個投影來調整參數。
- 第四張圖就是ORB-SLAM與COLMAP在空間中的軌跡了。
- R square約為0.98。



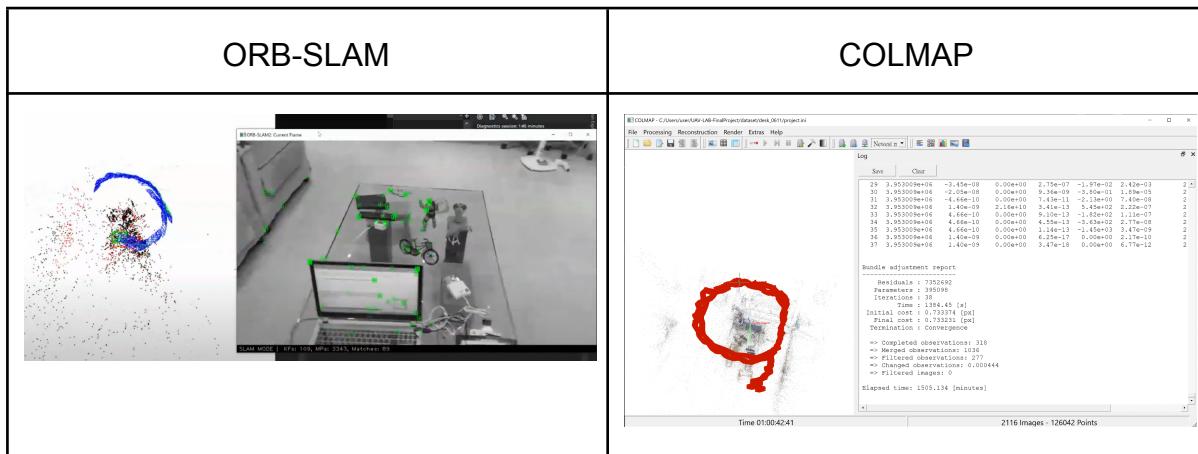
● 以實例比較做出來後的優劣處

- 可以看到COLMAP所得到點非常密集，幾乎可以連成一條線(程式中沒有設定連成線)，因為總共2116張照片都有定位出相機位置。
- ORB-SLAM的點就少很多，程式中統計出來才200個左右，因為ORB-SLAM從影片後期才開始定位，加上ORB-SLAM比較容易有抓不到特徵的情況，所以成功定位的點較少。
- 雖然ORB-SLAM結果較COLMAP差，但是在運算速度上大大勝過COLMAP，ORB-SLAM運算是realtime 30 fps的，而COLMAP運算要花一整天24小時的時間。

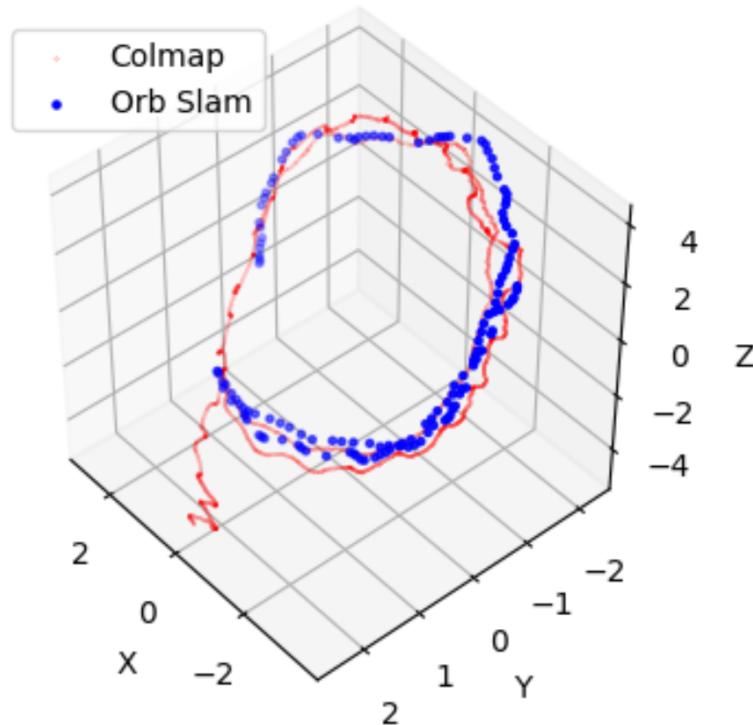
● 兩者間的誤差

- 我們原本使用方均根誤差，但由於SLAM出來的座標單位與實際長度不同，因此方均根誤差的意義好像不大，因此我們將誤差改成用相關係數平方R square來比較。
- R square結果為0.98，是高度相關，可以看出我們座標轉換是成功的，有成功將兩個座標mapping在一起。

相機軌道結果



兩者重疊後的3D圖



詳細實驗過程與結果影片如附檔