

TDCV HW3 Report

r10922061 資工所 王竑睿

Visual Odometry

1. visual odometry 是在給定相鄰兩張 frames 後，透過 match 兩張 frame 的 local feature 來找出對應特徵點，並以對應特徵點估測出 essential matrix 並分解 essential matrix 成旋轉矩陣以及平移向量來表示 camera 在此兩張相鄰 frame 的 pose 變動情形。本次作業開放使用 opencv API，因此在找到對應的特徵點後能直接使用 `cv2.findEssentialMat` 來找出 essential matrix，以及利用 `cv2.recoverPose` 來分解 essential matrix 以獲得旋轉矩陣 R 及平移向量 t 。然而 opencv 的 `cv2.recoverPose(E, points1, points2, cameraMatrix)` 內部實作，求出的 R 與 t 是從 points1 到 points2 的轉換。由於本次作業進行 visual odometry 要計算的是 camera 的轉換，因此正好是 points 的轉換的 inverse。亦即 camera C 在 opencv API 求出的 R, t 作用下會得到 $R^T(C - t)$ 而非 $RC + t$ 。
2. 本次作業助教提供 sample code 以同時 visualize 2D frames 以及 camera 的 3D 軌跡。由於 open3d visualizer 在預設情形下會以指出螢幕外的向量作為正 z 軸，因此在繪製 camera 軌跡時在 open3d 預設會不斷往螢幕外移動，正好與 opencv imshow 2d images 向內走的方向相反。為了 visualize 更符合直覺，在 open3d `add_geometry` 之前對構成 camera 的 `line_set` 進行一個 rotation，使其朝向螢幕內，即可得到與 2d image 方向較為一致的 visualization。

Loop Detection (Bonus)

reference: https://github.com/AayushKrChaudhary/VO_SLAM

demo影片: homework3-shiannn/loop_detection_demo_compress.mp4

1. 本次作業 Bonus 部份實作了 loop detection。使用的是 visual words matching 的方式。
2. 首先使用第一個 frame 抽取出的 500 個 ORB features 進行 kmeans clustering，計算出 M 個 centers。以這些 centers 作為 visual words 構成 dictionary。

3. 接著利用這 500 個 visual words 來描述每一個 frame。每個 frame 都能抽取出 500 個 ORB features，並且可以分別將 500 個 features 分配到 dictionary 中距離最近的 visual word。完成分配後可以得到一個直方圖，表示 500 個 features 在 M 個 centers 上的分佈。這個分佈是長度為 M 的向量， M 個 elements 總和為 500，可以作為描述這張圖的 representation。
4. 由於 visual odometry 是 online 進行，一邊拍攝新的 2D images 一邊進行計算。因此只使用第一個 frame 建構 dictionary。之後每個新拍攝的 2D frame 都使用同一個 dictionary 來抽取 representation。
5. 如果有一個新的 2D frame 的 representation 與過去抽取過的其中一個 representation 高度相似，則很可能代表拍攝到重複的場景，有 loop 的產生。本次作業選用 cosine similarity 以及 0.76 作為 threshold，並且令 $M = 450$ 作為 kmeans center 數。對於每一張新拍攝的 frame 都計算 representation 並與過去所有 frame 進行比較，similarity 超過 threshold 時即認定出現 loop 並 show 出 "It's Loop!"。

