3DCV Homework #2

r09944003 網媒所碩一 陳竣宇

Environment

> Python == 3.8
>
> OpenCV == 4.5.1
>
> Numpy == 1.19
>
> Open3d == 0.12.0
>
> Pandas == 1.2.4
>
> Scipy == 1.6.2

Execution

- Part 1
  - Q1-1 & Q1-3

```
python3 cameraPose.py
```

  - Q1-2

```
python3 cameraTrajectory.py
```

- Part 2

```
python3 plotCube.py
```

Problem 1

Q1-1:

- Pseudo code

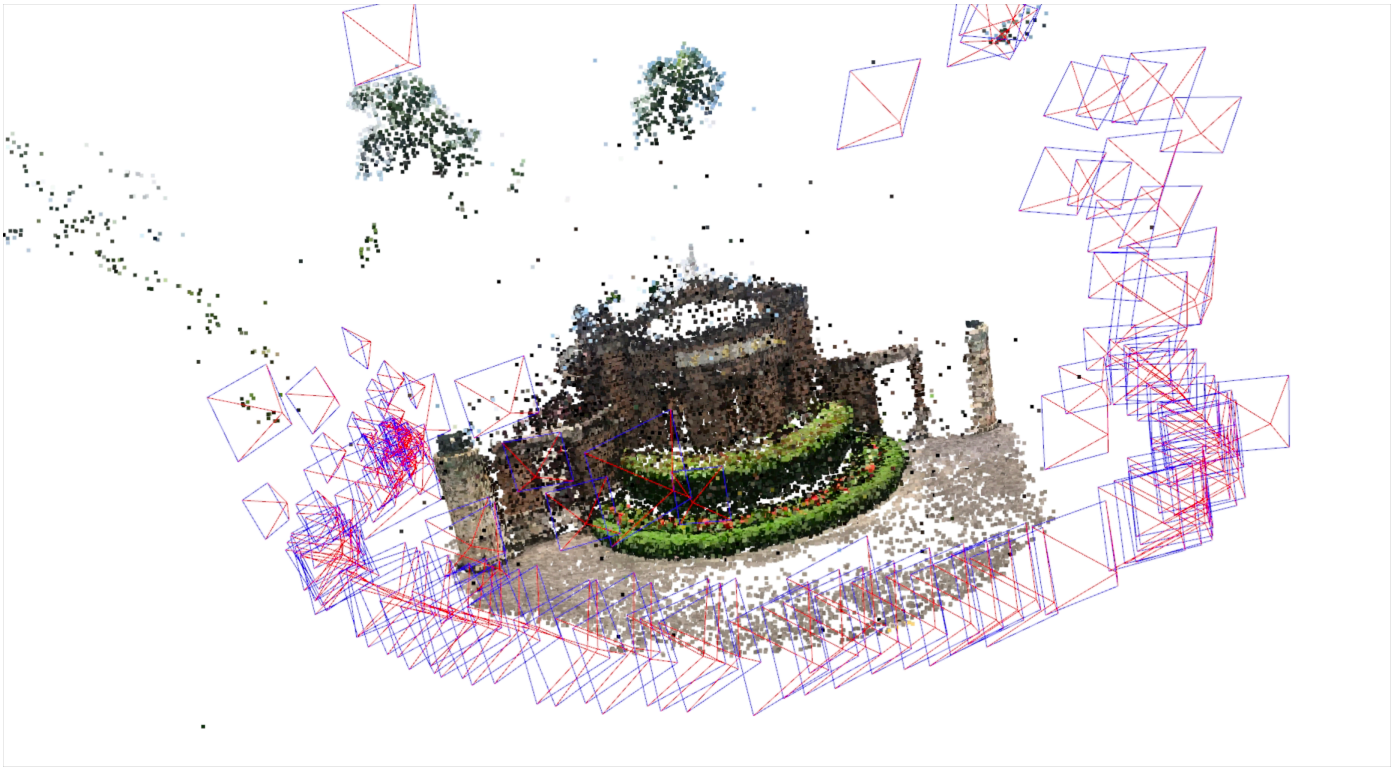
```
def solveP3P(points3D, points2D, cameraMatrix, distCoeffs, numValid):
    ## Sample correspondences
    numValid = 100
    indices = np.random.choice(len(points3D), 3+numValid)
    trainIdx, validIdx = indices[:3], indices[3:]
    x, u = points3D[trainIdx], points2D[trainIdx]
```

```
    ## Verifiy that world points are not colinear
    if np.linalg.norm(np.cross(x[1]-x[0], x[2]-x[0])) < 1e-5:
        print("Point A, B, C are colinear!")
        return False, []

    ## Compute angle cosines and distances then obtain lengths
    v = np.dot(np.linalg.inv(cameraMatrix), np.hstack((u, np.ones((u.shape[0], 1))))).T).T
    cosines = cosine(v[0], v[1]), cosine(v[0], v[2]), cosine(v[1], v[2])
    distances = np.linalg.norm(x[0] - x[1]), np.linalg.norm(x[0] - x[2]), np.linalg.norm(x[1] - x[2])
    lengths = solve_for_lengths(cosines, distances)
    if not lengths:
        return False, []

    ## Solve 3D-3D problem
    Rot, T = [], []
    Lambda = []
    for i in range(len(lengths)):
        ### Do trilateration to obtain T
        ans = trilaterate3D(x, lengths[i])
        for t in ans:
            l = np.linalg.norm(x[0] - t) / np.linalg.norm(v[0])
            if l < 0:
                continue

            ### Calculate the rotation matrix
            r = R_2vect(x[0] - t, l*v[0])
            if np.linalg.det(r) - 1 > 1e-5:
                continue

            Rot.append(r)
            T.append(t)
            Lambda.append(l)

    ## Utilize other correspondences to select the best solution
    best_R, best_T, best_error = 0, 0, float('inf')
    x_val, u_val = points3D[validIdx], points2D[validIdx]
    v_val = np.dot(np.linalg.inv(cameraMatrix), np.hstack((u_val, np.ones((u_val.shape[0], 1))))).T).T
    for i in range(len(Rot)):
        err = 0
        for j in range(len(x_val)):
            err += np.linalg.norm(Rot[i].dot(x_val[j]-T[i]) - Lambda[i] * v_val[j])
        error = err/len(x_val)
        if error < best_error:
            best_error = error
            best_R = Rot[i]
```

```
            best_T = T[i]

    return True, [best_error, best_R, best_T]


def p3psolver(query, model, cameraMatrix, distortion, numIter=100, numValid=100):
    kp_query, desc_query = query
    kp_model, desc_model = model

    bf = cv2.BFMatcher()
    matches = bf.knnMatch(desc_query, desc_model, k=2)

    gmatches = []
    for m,n in matches:
        if m.distance < 0.75*n.distance:
            gmatches.append(m)

    points2D = np.empty((0,2))
    points3D = np.empty((0,3))

    for mat in gmatches:
        query_idx = mat.queryIdx
        model_idx = mat.trainIdx
        points2D = np.vstack((points2D, kp_query[query_idx]))
        points3D = np.vstack((points3D, kp_model[model_idx]))

    ## Apply RANSAN algorithm
    best_R, best_T, best_error = 0, 0, float('inf')
    for i in range(numIter):
        isValid, rst = solveP3P(points3D, points2D, cameraMatrix, distCoeffs,
numValid)
        if isValid:
            err, R, T = rst
            if err < best_error:
                best_error = err
                best_R = R
                best_T = T

    return best_R, best_T

```

- Explaintion
  - 參考上課投影片的做法實作 P3P with Ransac，大致的流程如下
    - sample出3個對應的2D-3D correspondences
    - 使用公式 $v = K^{-1}u$ 得出u在3D空間中的位置v，並利用這些資訊算出3個夾角的cosine值
    - 用投影片公式算出x1, x2, x3到T點的長度，接著就能利用這些資訊求得 **R, T**
    - 因為會有多個解，所以使用額外的correspondence來驗證並確認最後的唯一解
    - 最後還有apply ransac來得到全局最佳解

Q1-2:



- Discussion
  - 從visualization的圖大致上可以看出camera的移動軌跡是沿著主要的3d point cloud model繞半圈，但是也可以明顯可以看出outlier的存在
  - 我認為一個合適的outlier rejection algotirhm有很大的機會能夠幫助改善performace
  - 用deep-based的method來estimate camera pose也不失為一個改善方法

Q1-3:

> Median pose error is **[R = 0.173, T = 3.646]**

- Discussion
  - 由於受到outlier的影響我的median pose error並沒有非常完美符合ground-truth的旋轉和平移，尤其觀察到translation的部分又更明顯了一些
  - 這個task使用助教要求計算median pose error來衡量localization的結果而不是一般常用的mean error，原因我認為是因為performace會嚴重被outlier影響所以算mean的話會導致誤差非常大，取median反而更能看出定位的準確度

Problem 2

Q2-1:

- Implementation
  - 把每個valid_img搜集起來並由後到前排序
  - 每個frame從ground-truth去取得其外參
  - 用事先存起來cude的3D座標配合外參、內參算出每個面上點的2D座標後用原本的3D depth information做排序並儲存
  - 用 `cv2.circle` 在img上根據 **Painter's Algorithm** 畫點，每個邊是取8個點
  - 將每個frame concat起來就得到最終的AR video
- Observation
  - 一開始算出點後要畫圖是直接去更改img上的pixel value，然而發現看不出任何更動，之後才發現只有一些pixel的變動太細微以致於難以發現
  - 用內參和外參把3D點轉到2D點的公式必須要寫好不然很容易出問題，例如做完內積之後要把前兩維的[x, y]座標除以最後一維的scale等等