3DCV&DL HW3

R11922196 林佑鑫

Briefly explain my method in each step:

Camera Calibration:

使用助教提供的 camerat_calibration.py，在多張圖上抓出棋盤格 corner 以及相對應的 3D corner 座標(自己定義)對其做校正，得到內部參數與形變參數。

K:

```
[[519.57919693   0.         316.83658406]
 [  0.         520.46700104 181.96843351]
 [  0.           0.           1.        ]]
```

dist:

```
[[ 1.29384673e-01 -9.44525722e-01 -1.79174961e-03  1.02536311e-03
   1.76098660e+00]]
```

之後每張照片讀取進來後都會先做 distortion，先去除鏡頭形變影響。

Feature Matching:

去形變後的影像轉成灰階，之後用 orb 抓 feature，再對相鄰兩張 image 根據 Hamming distance 做 match，得到相鄰圖片的 feature matching。

Pose from Epipolar Geometry(pseudo codes with comments):

```
# Define matching method and the original pre_R, pre_t
def pre_R = 3X3 identity matrix
def pre_t = 3X1 zero vector
# Read first image
img1 = all_images[0]

# Iterate through all the images
for image in all_images[1:]:
    # Step 1. Capture new frame img_k+1
    img2 = image

    # Step 2. Extact and match features between img_k and img_k+1
    point1, point2 = ORB_Feature_Matching(img1, img2)
```

```
        # Step 3. Esimate the essential matrix E_k,k+1
        E = Estimate_Essential(point1, point2)

        # Step 4. Decompose the E_k,k+1 into R_k,k+1 and t_k,k+1 to get the relative pose.
        R, t, triangulatedPoints = Recover_Pose(E)

        # Step 5. Calculate the pose of camera k+1 relative to the first camera.
        R = R * pre_R
        t = pre_t + R * t

        # Step 6. Calculate the consistent scale of t_k,k+1 by 2 correspondences at the
adjacent images.
        if image is not all_images[1]:
            select pre_point1, pre_point2 at pre_trianguledPoints
            select cur_point1, cur_point2 at triangulatedPoints
            scale = norm(cur_point1 – cur_point2) / norm(pre_point1 – pre_point2) \
                    * norm(pre_t) / norm(t)
            return (R, t * scale)
        else:
            # Set t_0,1 as default scale of t.
            return (R, t)

        # Update image1 and pass R, t, triangulatedPoints for next iteration.
        pre_triangulatedPoints = triangulatedPoints
        pre_R = R
        pre_t = t
        img1 = img2
```

Result Visualization:

將圖上四個點和原點用內參的反矩陣投影回三維空間中，用這五個點表示相機的焦點和 pose 對每一個時間點，用前一步得到的 R, t 對五個點做 rotation, translation。得到他們相對於初始位置，最後用 Homework2 的 open3d lineset 方法將 pose 呈現在三維空間中。

Youtube link:

https://youtu.be/X9Y1bv8o08s

Please tell us how to excute your codes, including the package used and the envirment:

Hardware:

1. CPU: i7-6700k
2. GPU: Nvidia GTX 1080

Environment:

Windows 10 21H2 / Python 3.10

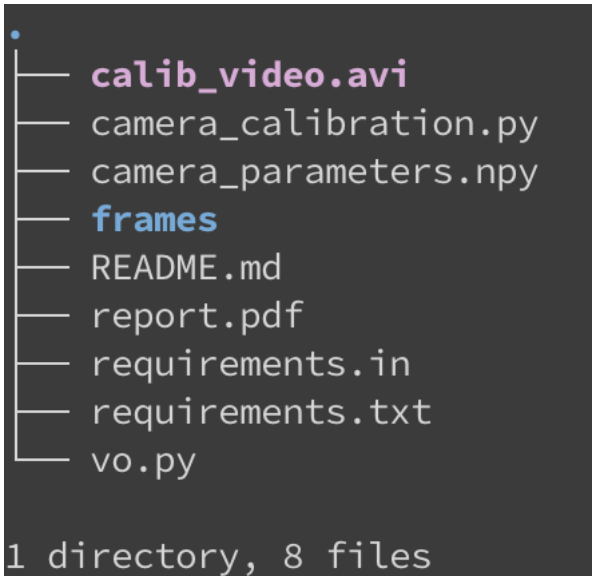
Packages:

1. numpy
2. opencv-python==4.5.1.48
3. open3d

環境配置是用 pip-compile 取得上面三個 package 所需的相依套件，

可直接用 pip install -r requirements.txt 安裝

File structures:



Commands:

Environment:

pip install -r requirements.txt

Reproduce:

python3 vo.py frames