3DCV Homework 3: Visual-Odemetry

Department: CSIE Student ID: d11922016 Name: Jia-Wei, Liao
YouTube: https://youtu.be/eqMYXd52ooY

November 13, 2023

1 Camera Calibration

We utilize the script `camera_calibration.py` to select four images from `calib_video.avi` for camera calibration, resulting in the determination of the camera's intrinsic parameters. The derived camera intrinsic matrix K is as follows:

$$K = \begin{bmatrix} 441.8938 & 0 & 301.7754 \\ 0 & 438.8981 & 188.5821 \\ 0 & 0 & 1 \end{bmatrix}$$

The distortion coefficients are given by:

$$\begin{bmatrix} 0.08670775 & -0.3798425 & -0.00111281 & -0.00170044 & 0.45260974 \end{bmatrix}$$

2 Feature Matching

We first convert the image to grayscale then employ the ORB (Oriented FAST and Rotated BRIEF) algorithm to detect features within the image. Upon detecting the features, we proceed to match these features between two adjacent frames. By calculating the Hamming distance, we are able to identify and pair the matched feature points between the two frames. Finally, we rank the feature points based on their distance and select the top 85% for subsequent calculations. Figure 1 illustrates image frame corresponding to the identified feature points.

Figure 1: Feature point.

3 Pose from Epipolar Geometry

Algorithm 1 Estimate Pose from Epipolar Geometry

1: **Input:** A sequence of frame images I_k, the number of frames F, with P_k denoting feature points corresponding to frame k.
2: Capture the initial frame I_0.
3: **for** each frame $k = 0$ to $F - 1$ **do**
4: Acquire the subsequent frame I_{k+1}.
5: Perform feature extraction and matching between frames I_k and I_{k+1}.
6: Compute the essential matrix $E_{k,k+1}$ using matched feature points P_k and P_{k+1}.
7: Decompose $E_{k,k+1}$ into rotation R^k_{k+1} and translation t^k_{k+1}, forming the relative pose matrix

$$\xi^k_{k+1} = \begin{bmatrix} R^k_{k+1} & t^k_{k+1} \\ O_{3\times 1} & 1 \end{bmatrix}.$$

8: Compute $\|t^k_{k+1}\|_2$ from $\|t^{k-1}_k\|_2$ and rescale t^k_{k+1} accordingly.
9: Determine the pose of camera at frame $k+1$ relative to the first camera using the equation $\xi^0_{k+1} = \xi^0_k, \xi^k_{k+1}$.
10: **end for**
11: **return** The computed pose ξ^0_{k+1}.

The pose ξ^0_{k+1} is obtained by iteratively chaining the relative poses from one frame to the next. This iterative process is mathematically described as follows:

$$\begin{aligned}
\xi^0_{k+1} &= \xi^0_k \, \xi^k_{k+1} \\
&= \begin{bmatrix} R^0_k & t^0_k \\ O_{3\times 1} & 1 \end{bmatrix} \begin{bmatrix} R^k_{k+1} & t^k_{k+1} \\ O_{3\times 1} & 1 \end{bmatrix} \\
&= \begin{bmatrix} R^0_k R^k_{k+1} & R^0_k t^k_{k+1} + t^0_k \\ O_{3\times 1} & 1 \end{bmatrix}
\end{aligned}$$

Therefore, we have

$$\begin{cases} R^0_{k+1} &= R^0_k R^k_{k+1} \\ t^0_{k+1} &= R^0_k t^k_{k+1} + t^0_k. \end{cases}$$

4 Visualization Results

We apply the derived rotation matrices R^0_{k+1} and translation vectors t^0_{k+1} to a predefined set of points: $(0, 0, 0)$, $(0, 0, 1)$, $(0, 269, 1)$, $(842, 269, 1)$, and $(842, 0, 1)$. This application transforms these points into the world coordinate system, enabling us to map the camera's position and orientation in a global reference frame. By performing this transformation across all frames, we are able to reconstruct the camera's trajectory in the world coordinate system. The camera's trajectory is shown in Figure 2.

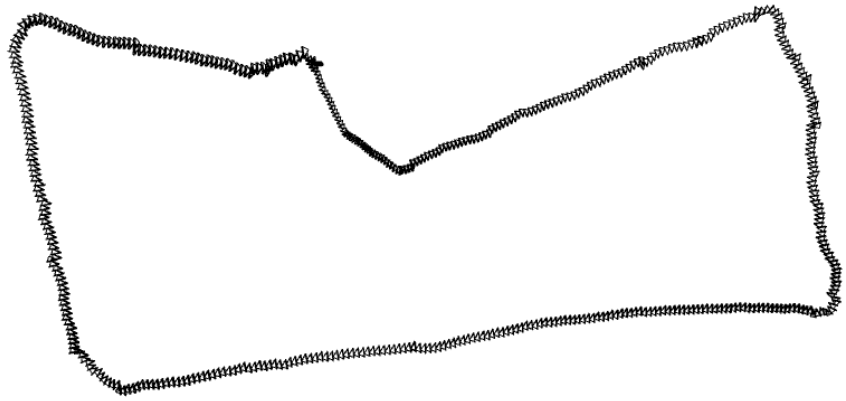

Figure 2: Camera's trajectory.

Appendix

Device

All the experiment were conducted on a PC running Ubuntu 20.04.2 with a 12th Gen Intel(R) Core(TM) i7-12700 CPU.

Environment

To set the environment, you can run this command:

```
pip install -r configs/requirements.txt
```

The packages and their respective versions are displayed as follows:

```
numpy==1.24.4
pandas==2.0.3
scipy==1.10.1
opencv-python==4.8.0.76
open3d==0.17.0
gdown==4.7.1
```

Reproduce

To get the camera calibration, you can run the command:

```
python camera_calibration.py calib_video.avi
```

To reproduce the result, you can run the command:

```
python main.py
```