ENPM673 Project 5

Justin Albrecht & Brian Bock

Spring 2020

Table of Contents

1 Data Preparation

As we have in previous projects, we work with the video as individual frames, and iterate over all of the frames as we go. To start, we eliminate the first 22 frames of the video, which are overexposed and contain little useful data. This frame count was determined visually. We then use the `cv2.cv2Color()` function to convert each frame from it's original Bayer color space to a more traditional BGR. The images have some small lens distortion, which is removed by dewarping the frame via the provided `UndistortImage` function (Figure 1).

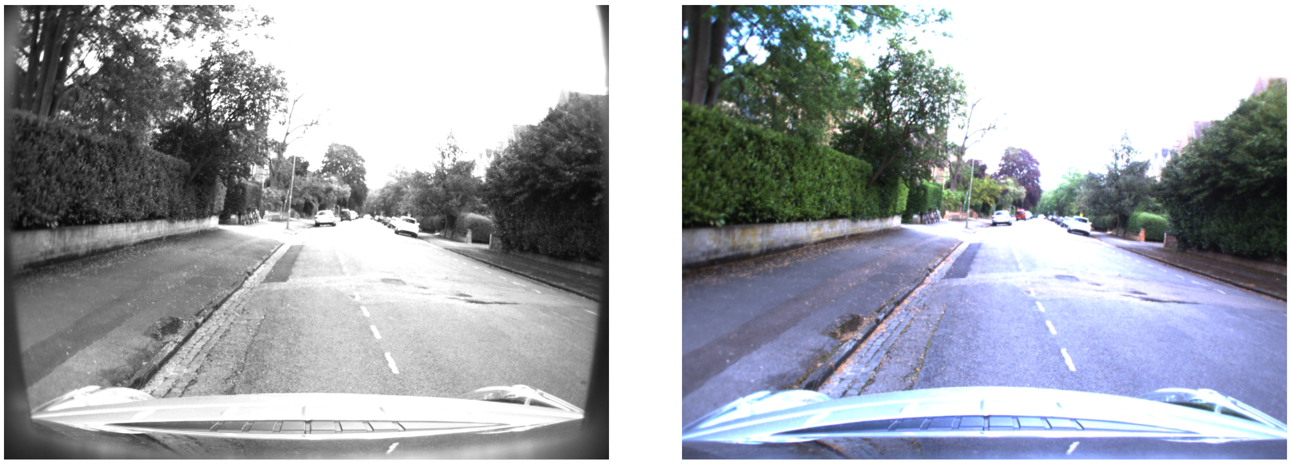
(a) Original Frame (b) Image converted to BGR color space and then undistorted

Figure 1: Two photos showing the initial image processing steps

2 Finding Key-points and matches

To find key points we used an algorithm called `SURF`. `SURF` stands for Speeded-Up Robust Features, it is similar to the `SURF` algorithm but introduces efficiencies to speed up the process. The `SURF` algorithm takes in an images and returns a list of key points and their descriptors.
To find matched features between two images we used an algorithm called `flann`. `Flann` uses the descriptors generated by running SURF on two separate images and tries to find find points that have similar descriptors. `Flann` returns a list of matches that we can use to find the location in the two original images where there are matching features. Using `SURF` and `flann` returns a large number of matches with a lot of false positives. Our video processing pipeline uses RANSAC in order to filter outliers in the data.
We define a vector $\boldsymbol{x}$ to contain our (x, y) coordinates:

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.1}$$

3 Fundamental Matrix

The fundamental matrix F for two matching points $\boldsymbol{x}$ and $\boldsymbol{x}'$ satisfies the following equation.

$$\boldsymbol{x}'^T F \boldsymbol{x} = 0 \tag{3.1}$$

We can estimate the fundamental matrix using Singular Value Decomposition (SVD).

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \tag{3.2}$$

We reshape F into f, a column vector:

$$f = \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} \tag{3.3}$$

We can decompose the above equation into a series of linear equations in the form:

$$A = \begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx_m' & x_my_m' & x_m & y_mx_m' & y_m & x_m' & y_m' & 1 \end{bmatrix} \tag{3.4}$$

$$A \cdot f = 0 \tag{3.5}$$

We need at least 8 correspondences to solve the system of equations. Using the eight points we can generate the matrix A, then use SVD to decompose the matrix into three matrices, U, Σ, and V.

$$A = U\Sigma V^T \tag{3.6}$$

The last column of V^T is the solution to f.

4 Inliers using RANSAC

To find which of our matches can be considered inliers, we use RANSAC (random sample consensus). The idea of this method is to use a heuristic that is formulated using a random sample of the data, then test to see how well the heuristic fits the rest of the data. This is done iteratively, updating the best heuristic with the current when it better explains the entire sample. This approach can be done for a finite number of iterations or until a large enough sample of the data is explained by the heuristic. For the matching we use the Fundamental Matrix as a heuristic. The fundamental matrix can be calculated using 8 sets of matches and is used to define the epipoles of the two images. The approach we use is as follows:

1. Select 8 random sets of points from the list of matches
2. Create an estimation of the fundamental matrix F from these points (See Section 3)
3. Find the value of $\boldsymbol{x}'^T F \boldsymbol{x}$ for each match. For a perfect match with no noise and the correct fundamental matrix, this value would be zero.
4. Check if the value is below a threshold, if it is add it to a list of inliers for the current loop
5. After all the values have been checked, see if the number of inliers for the current loop is greater than previous loops, if it is, this Fundamental Matrix better describes the data and therefore is the new F
6. Run for the specified number of iterations. We are currently using 50 iterations. More would potentially lead to a better result but makes the program run slower and for longer. For speed reasons we kept the number fairly low. The fundamental matrix F that has the most inliers is considered the best fit for the data.

5 Estimate Essential matrix

Define camera intrinsic matrix K:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

The values of f_x, f_y, c_x, and c_y are determined by the camera calibration and extracted from the provided function `ReadCameraModel`. f_x and f_y define the focal length of the camera in the x and y directions, respectively. The Essential matrix E is first defined as:

$$E = K^T F K \tag{5.2}$$

Due to noise, the singular values of E may not necessary be 1,1,0. This can be modified by using SVD to find the matrices U and V, then replacing Σ with:

$$\Sigma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.3}$$

The new essential matrix is now:

$$E = U\Sigma V^T \tag{5.4}$$

6 Extract 4 possible camera poses

A given essential matrix between two cameras defines four possible pose orientations between the two cameras. The configurations are:

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

$$C_1 = U[:,3] \qquad R_1 = UWV^T \tag{6.2}$$
$$C_2 = -U[:,3] \qquad R_2 = UWV^T \tag{6.3}$$
$$C_3 = U[:,3] \qquad R_3 = UW^TV^T \tag{6.4}$$
$$C_4 = -U[:,3] \qquad R_4 = UW^TV^T \tag{6.5}$$

$$det(R) = 1 \tag{6.6}$$

7 Cheirality Check

The cheirality check determines which of our camera poses satisfies the most points in the positive Z direction.

7.1 Triangulate 3D Points

Before we can find which points have a positive Z, we must use triangulation to determine where in 3D space the match point is. This can be done using linear least squares. We must first define the camera matrix P for each of the two cameras (frame i-1 and frame i)

$$P = KR\begin{bmatrix} I_{3\times 3} & -C \end{bmatrix} \tag{7.1}$$

For each frame we assume that the first camera is at the origin.

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{7.2}$$

Defining the camera matrix for the first camera as P and the camera matrix for the second camera as P'.

$$P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \qquad P' = \begin{bmatrix} p'_1 \\ p'_2 \\ p'_3 \end{bmatrix} \tag{7.3}$$

$$A = \begin{bmatrix} (y \cdot p_3) - p_2 \\ p_1 - (x \cdot p_3) \\ (y' \cdot p'_3) - p_2 \\ p'_1 - (x' \cdot p'_3) \end{bmatrix} \tag{7.4}$$

We can then use SVD again to solve for $\boldsymbol{X}$, the point in 3D world coordinates:

$$A \cdot \boldsymbol{X} = 0 \tag{7.5}$$

7.2 Depth Check

For each of the four possible poses we can find the corresponding points in world coordinates.
We check if

$$r_3(\boldsymbol{X} - C) > 0 \tag{7.6}$$

where r_3 is the last row of R. Whenever this is true we increment a counter. The combination of C, R, and X that produces the highest count is the correct camera pose.

8 Global Coordinates

Our computations so far get the rotation and translation between individual frames, with no regard for global coordinates. We need to convert these movements into a global frame to track the camera as it moves. To do so, we define a 4×4 transformation matrix T:

$$T = \begin{bmatrix} R & C \\ 000 & 1 \end{bmatrix} \tag{8.1}$$

With each frame, we generate a new T from our new C and R values.

$$T_{tot} = T_{tot} \cdot T_{new} \tag{8.2}$$

This T_{tot} defines the transformation from our origin (the first frame) to the most recent frame. The new (x, y, z) coordinates in the global frame can be found from the last column of T_{tot}:

$$T_{tot} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.3}$$

9 Visualizer

The visualizer function takes the previous frame and current frame, and stacks them horizontally. On this new combined image, it draws red dots on every matching point and then green lines between them. It generates a XZ and YZ plot, drawing the movement between the current and previous point. The two graphs are stacked horizontally, and then the combined matches image and graph image are stacked vertically, creating a four panel image containing the previous frame, next frame, XZ graph, and YZ graph (Figure 2). These four panel images are then written as the frames to an output video.

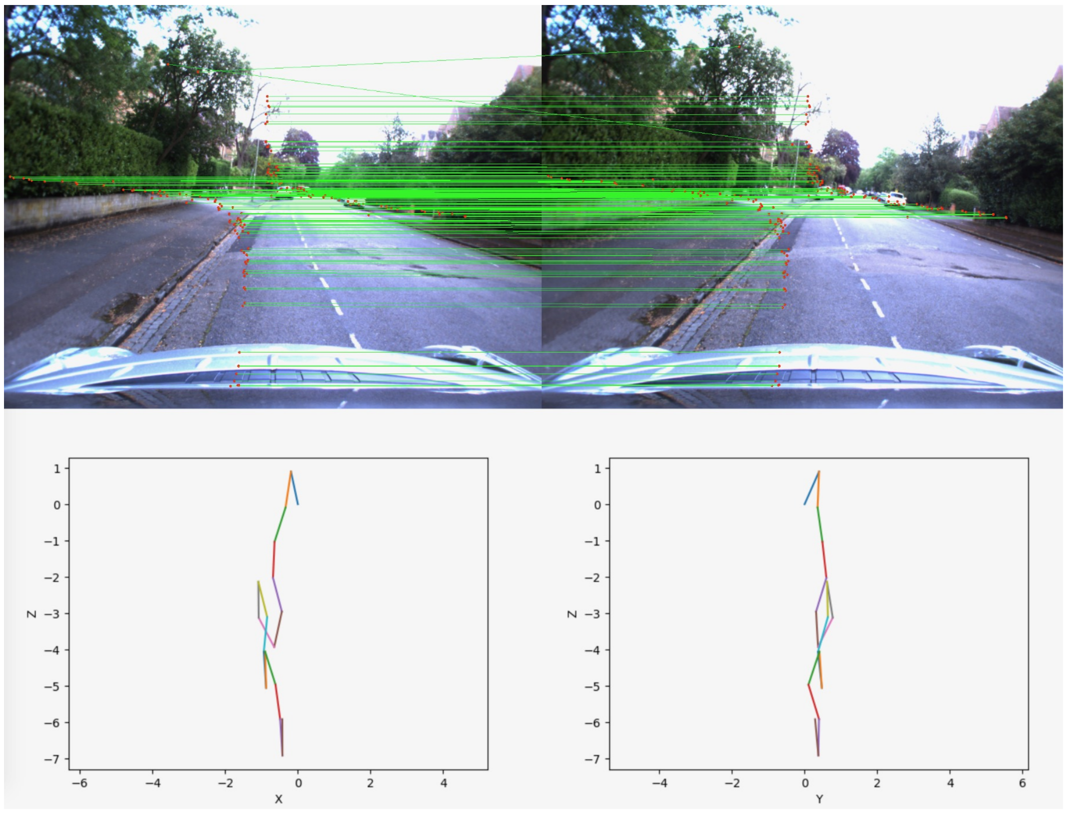

Figure 2: Four panel image exported to video

10 Videos

You can view our videos here: https://drive.google.com/drive/folders/15QiVtqNiMT6ksdWCpA5KJD32XfDCPCli

11 Code

You can view the code related to this project at https://github.com/jaybrecht/ENPM673_Project5