CS 585: Challenge Report

Aditya Agrawal U29099443

Steps

1. Initial Variables: - K is defined as the camera matrix. Initial Pose is defined as $[I|0]$. So initial value in predictions is [0,0,0]. D is a variable used for calculating R and t from E.

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \; D = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \; Initial\ Pose = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2. The first two frames are selected.
3. Feature Selection: - First, the FAST (Features from Accelerated Segment Test) algorithm (implemented using cv2.FastFeatureDetector_create.detect()) is used to find the best features in the first frame. The intensity threshold is set at 10 and non-max suppression is applied.
4. Next, The Lucas-Kanade Optical Flow between the first and second frame is calculated using cv2.calcOpticalFlowPyrLK() with features from first frames as input. The variable st returned is used to select the matching features in both frames. Hence the best matching features are found.
5. Essential Matrix: - The essential Matrix between the two frames is calculated using cv2.findEssentialMat(). To ensure rank 2 constraint of Essential Matrix, it is decomposed in U, S, V^T using SVD and recomputed as E= U@diag([1,1,0])@V^T. I tried to implement the 8-point algorithm method but did not good results. It is described in "Methods tried Section".
6. Rotation and translation: - The Rotation (R) and translation (t) between the two frames is calculated using cv2.recoverPose(). I tried to implement my own function to calculate this, but it did not give good results. It is described in detail in "Methods tried Section".
7. Pose for second camera: - Let the pose of the first camera be $[R_{old}|t_{old}]$ and the pose of the second camera be $[R_{new}|t_{new}]$. Then the second pose is calculated as: -

$$[R_{new}|t_{new}] = [R_{old}|t_{old}] \begin{bmatrix} R & scale * t \\ \mathbf{0^T} & 1 \end{bmatrix}$$

$$R_{new} = R_{old}@R$$

$$t_{new} = scale * (R_{old}@t) + t_{old}$$

   where scale is calculated using self.get_scale(frame_no).

8. For the path array (predictions), t_{new} is added to the array.
9. Steps 3 to 8 are repeated for all consecutive pairs of frames till the n-1th and nth frames.

With this method, I got an MSE of 2.1259 on the given dataset and a score of 121.26 on the Gradescope dataset.

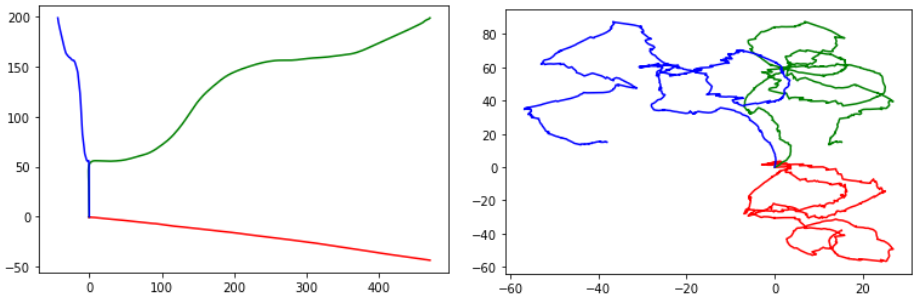
Methods tried

1. Essential Matrix: - I tried to implement the normalized 8-point algorithm to find the Fundamental Matrix.
    a. To normalize the points, the transform matrix is calculated for each image separately as

$$T = \begin{pmatrix} s & 0 & -st_y \\ 0 & s & -st_x \\ 0 & 0 & 1 \end{pmatrix}$$

    where (t_x, t_y) is the centroid of all points and s is calculated as $\frac{\sqrt{2}}{\sqrt{\frac{1}{N}\sum(points-centroid)^2}}$. The points in both images are multiplied by their respective transformation matrix.
    b. The 8-point algorithm is implemented as A.append([x_2x, x_2y, x_2, y_2x, y_2y, y_2,x,y,1]) for all matches, then SVD decomposition (USV^T) of A is done, F is the last column of V.
    c. F is decomposed again, the last eigenvalue is made 0 to enforce rank 2 constraint, and F=U@diag([S_1,S_2,0])@V^T. It is normalized by dividing with last value in matrix.
    d. Finally F is multiplied by both transformation matrix to undo transformation. F=T_b^T@F@T_a.
    e. The essential matrix is calculated as E=K^T@F@K. To ensure rank 2 constraint of Essential Matrix, it is decomposed in U, S, V^T using SVD and recomputed as E= U@diag([1,1,0])@V^T

    When I run this method, I get a zig-zag pattern in the predicted path, an MSE score of 340 and a Gradescope score of -330 (approx.). On the left is ground truth path. On the right is path due to this method.



2. Rotation and translation: - For this I followed the method highlighted in Wikipedia and "An Efficient Solution to the Five-Point Relative Pose Problem" research paper.
    a. Essential matrix is decomposed into U,S,V^T using SVD. Is |U|<0 or |V|<0, the respective matrices are multiplied by -1 to make their determinants positive.
    b. Then four values are calculated.
        i. $[t_A]_x$ = U@D^T@diag([1,1,0])@U^T
        ii. $[t_B]_x$ = U@D@diag([1,1,0])@U^T
        iii. R_A= U@D@ V^T

iv. $R_B = U@D^T@V^T$

Here $[t]_x = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$

c. The points in both images are pre-multiplied by K^{-1}.

d. Using $P_A=[R_A|t_A]$ and $P_I=[I|0]$ as the two projection matrices, triangulation is done on a single feature match pair (pre-multiplied), to generate a 3D triangulated homogeneous point Q.

e. Define $H_t = \begin{bmatrix} I & 0 \\ -2 * last\ row\ of\ V^T & -1 \end{bmatrix}. c_1 = Q_3 * Q_4.\ c_2 = (P_A@Q)_3 Q_4.\ \ c_3 = Q_3(H_t@Q)_4.$

f. If $c_1 > 0\ and\ c_2 > 0$, then R_A and t_A are returned as rotation and translation. If $c_1 < 0\ and\ c_2 < 0$, then R_A and t_B are returned. If $c_1 c_2 < 0$, then we check c_3 is checked. If $c_3 > 0$ then R_B and t_A otherwise R_B and t_B.

Using this method (but cv2 for essential matrix) I get an MSE score of 2.1259 on the given dataset, which is exactly the score by using cv2.recoverPose(). The predicted path also matches closely to the ground truth. However, on the Gradescope dataset, I get a score of -18.

The code for both alternate implementation of Essential & Rotation, translation are present in the submitted Odometry.py file, they are just not called since they do not cross the baseline.