Assignment 2 Report

Epipolar Geometry and Visual Odometry

Puru Gupta(20171187) , Meher Shashwat Nigam(20171062)

Question 1 (python3 q1.py)

Given information :

- Two images of the same scene, taken from different view-points.

- The fundamental matrix (F)

```
[   [-1.29750186e-06   8.07894025e-07   1.84071967e-03],
    [3.54098411e-06    1.05620725e-06   -8.90168709e-03],
    [-3.29878312e-03   5.14822628e-03   1.00000000e+00]  ]
```

- A subset of the corresponding points in both the images that were used to estimate F.
- The convention for F we follow is (x'T F x) = 0, where x' is the location of the point in the second (right) image.

Part a :

Objective : For the points in the first image, plot their corresponding epipolar lines in the second image as shown. Repeat this for the first image.

We know that corresponding points in both images are connected by the equation as follows:

$$(x'^{T} F x) = 0$$

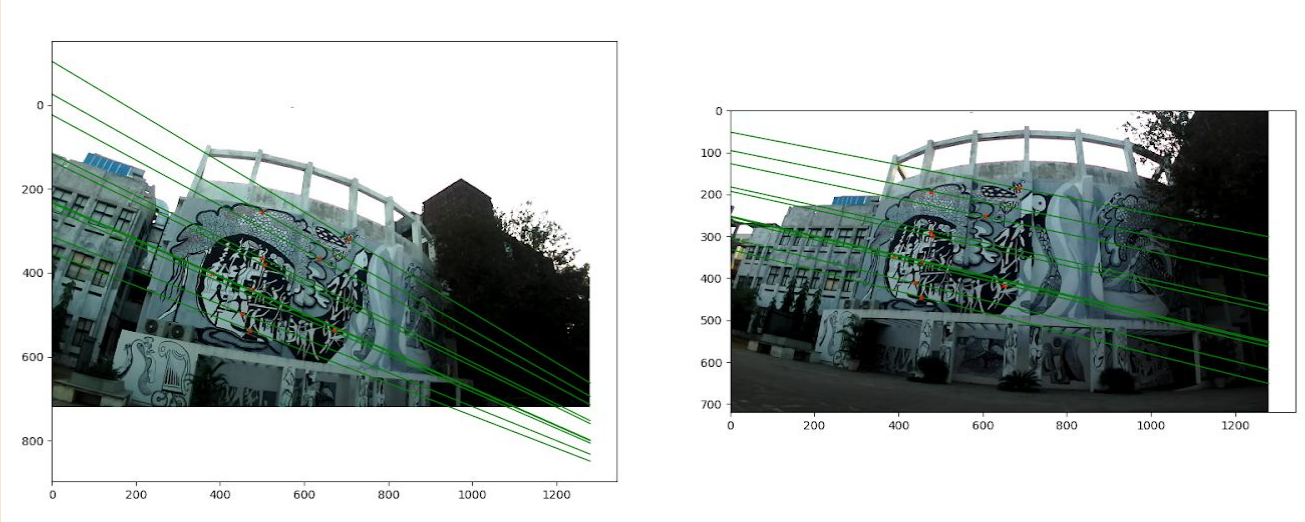
Here x' is a point in the second image and x is a point in the first image, F is the fundamental matrix.

Now **Fx** is the epipolar line associated with x and $\mathbf{F^{T}x'}$ is the epipolar line associated with x'.

We first concatenate an additional 1 at the end of the given image coordinates to go into the homogeneous coordinate frame.

Then we calculate the epipolar line equations associated with each point with the formula given above.

Following this we plot the given points and the corresponding epipolar line equations on the images as shown: **(output.png)**



Part b :

Objective : Computing the epipoles without using the epipolar line equations.

$\mathbf{Fe = 0}$ and $\mathbf{F^{T}e'} = 0$, where e and e' are the epipoles of the first and second image.

So e and e' belong to the left and right null space of the F matrix.

The last column of the V matrix (on taking SVD) is a good approximation for that as the corresponding lambda value is very low (order of 10^-15).

Computing epipoles from the null spaces and normalizing :

epipole1 [[2.15916628e+03 1.18926810e+03 1.00000000e+00]]

epipole2 [[-5.13190964e+03 -9.48854631e+02 1.00000000e+00]]

Question 2 (python3 q2.py)

Output text file -> outputx.txt

Given information :

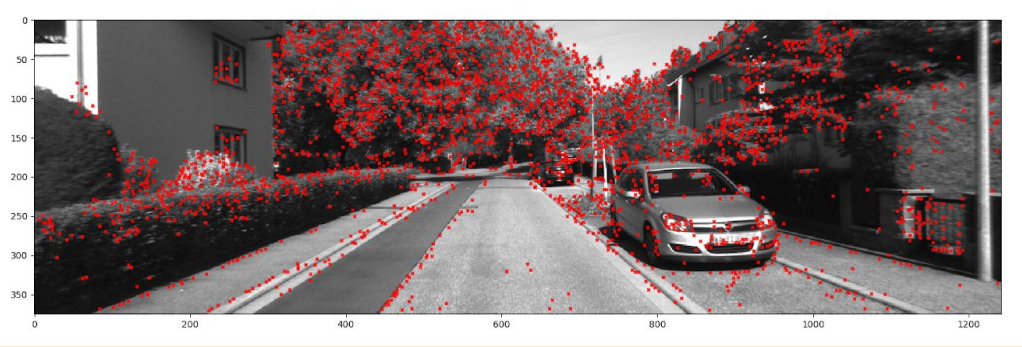
- a sequence of images from the KITTI dataset
- The ground truth pose of each frame (in row-major order) and the camera parameters are provided as well.

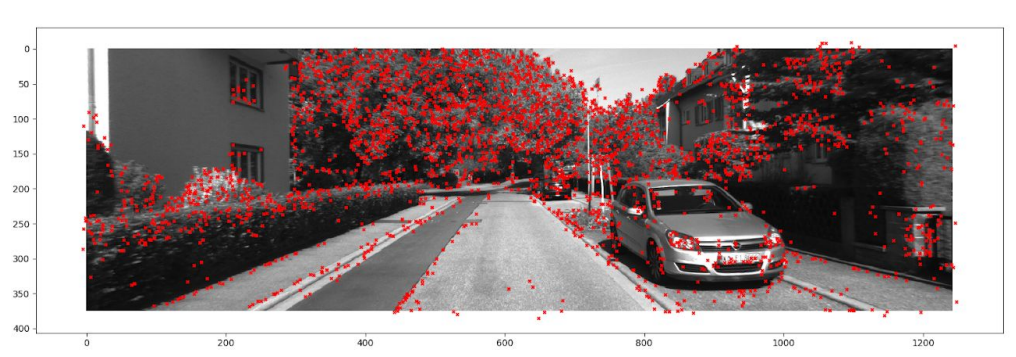
Algorithm used :

The following is an overview of the entire algorithm used:

1. **Find corresponding features between frames I_k, I_{k-1} :**

Finding corresponding features is done using **SIFT algorithm** (inbuilt openCV function used). This is used to find correspondences between consecutive images. When the number of feature correspondences goes below a certain threshold (which we have set to **2500**), we run SIFT again on that image and continue for the rest of the sequence.





2. **Using these feature correspondences, estimate the essential matrix between the two images within a RANSAC scheme.**

Then we run RANSAC (with number of iterations = **750** , threshold = **0.05)** on points of consecutive images and calculate the essential matrix for each pair of consecutive images.

3. **Decompose this essential matrix to obtain the relative rotation Rk and translation tk, and form the transformation Tk.**

Then we decompose this essential matrix to get the corresponding R,t and concatenate the R,t to get the transformation matrix T. Let's assume for going from C_k to C_{k+1} we get T_k.

4. **Scale the translation tk with the absolute or relative scale.**

We scale our obtained unit tk with the corresponding t from the ground truth provided.

5. **Concatenate the relative transformation by computing $C_k = C_{k-1}T_k$, where C_{k-1} is the previous pose of the camera in the world frame.**

We calculate a cumulative transformation matrix till the kth camera frame by multiplying all transformation matrices obtained till that frame. We reshape the first three rows of this transformation matrix into a row vector and store it in the text file given as input to EVO, for plotting trajectory.

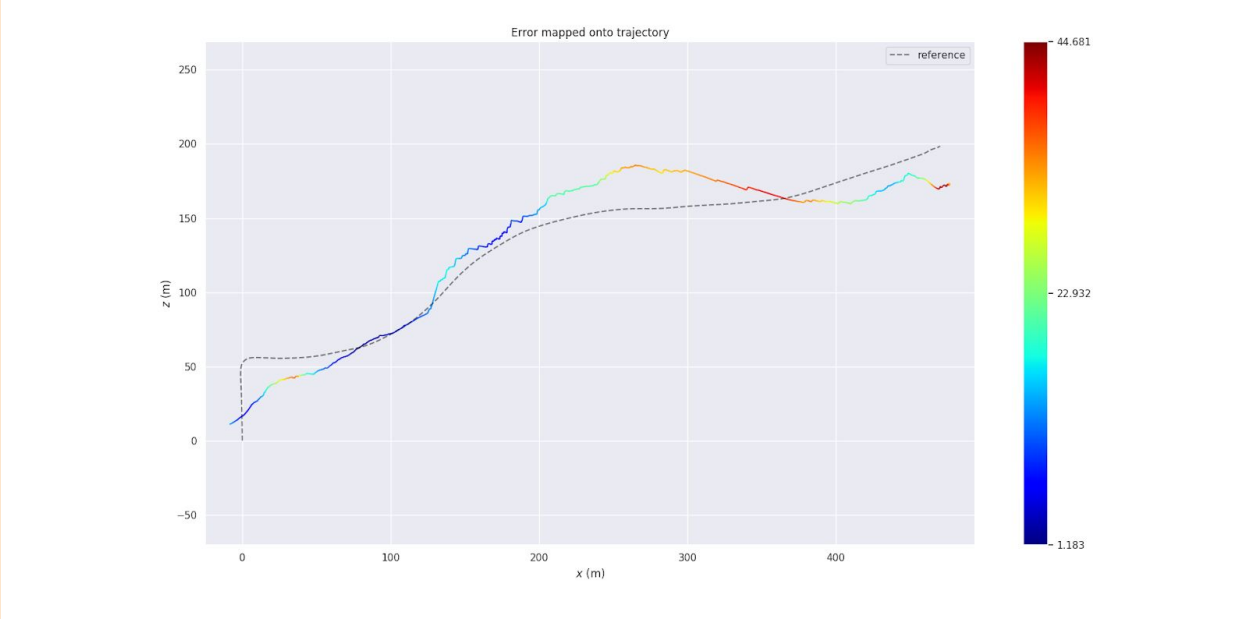
6. **Repeat steps 1 – 5 for the remaining pairs of frames.**

Performance Evaluation :

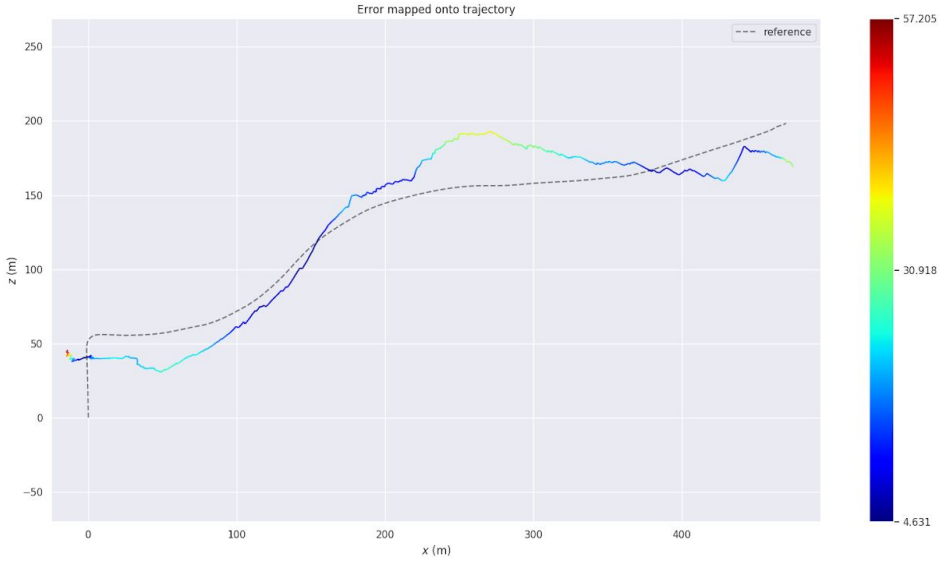
Our procedure gives around 35 to 40 Absolute Percentage error at max when compared to ground truth. It works at smooth curves but fails at sharp turns giving weird patterns.

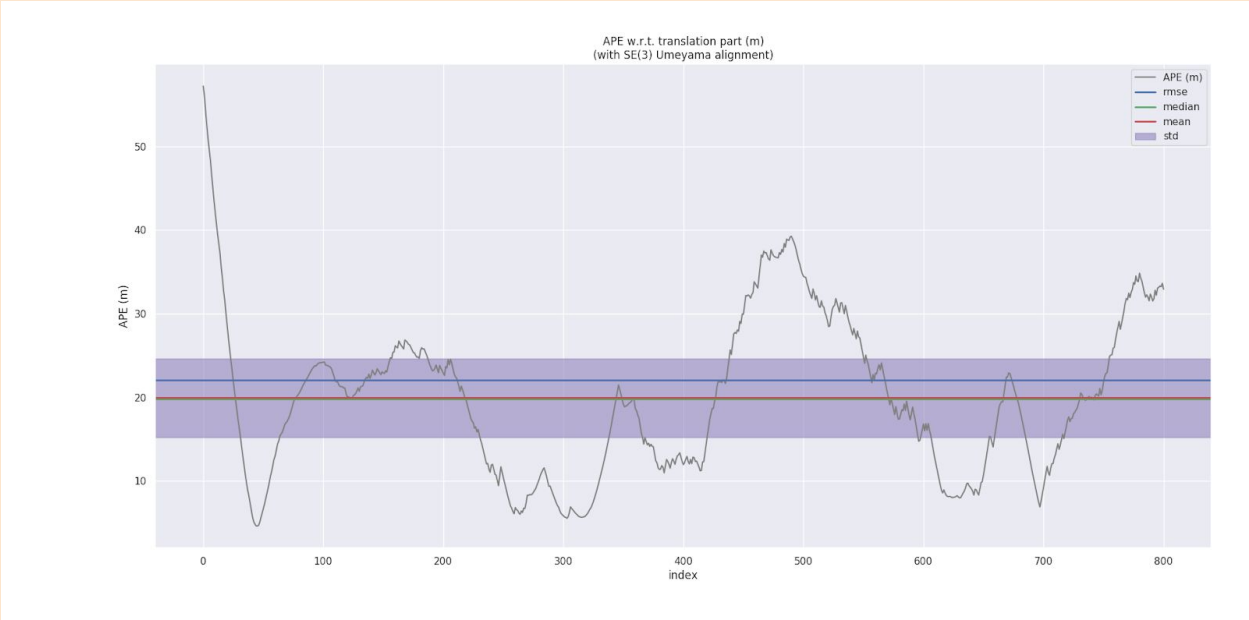
- Significant rotational bias on some estimated trajectory segments due to non-planarity of the road environment in those segments.
- Unavoidable visual odometry drift and deviation due to road humps that violate the planar motion assumption.
- Fails at sharp turns, due to steep decrease in the number of corresponding features in consecutive frames.

For different values of RANSAC threshold, iterations we obtain these outputs as compared to ground truth:




Error mapped onto trajectory
reference
250
200
150
100
50
0
−50
z (m)
0
100
200
300
400
x (m)
34.500
17.584
0.668


Error mapped onto trajectory
reference
250
200
150
100
50
0
−50
z (m)
0
100
200
300
400
x (m)
57.205
30.918
4.631



[Bonus] Other ways of calculating absolute scale :

By sighting a distant object from 2 different locations/ two different frames and knowing the distance between those locations (called the line of position), we can determine the distance to the object.

Assuming that an object's length is b, based on the 3D reconstruction, the scale factor is defined as s = b'/b, where b' is the ground true length of this object. Absolute scale estimation aims at recovering this coefficient s.

Camera height is commonly used for scale estimation. Usually, the camera is fixed on a platform and its height (the distance from camera principle center to the ground plane) is unchanged during a certain amount of time. Assuming the ground surface right in front of the camera is flat, the scale can be recovered according to this height information.

GROUND PLANE MODEL

The camera coordinate at the first frame is assumed to coincide with the world coordinate. Any 3D point $X = (X, Y, Z)^T$ belonging to ground plane follows the following constraint $n^TX = h$, where n ($|n| = 1$), is the normal of ground plane and h is the distance from camera center to ground (given to be 1.65m). Now that we have the relative motion [R | t] using 8-points algorithm, in which R is rotation matrix and t is the translation up to a scale between consecutive image frames. The 3D

points X can be triangulated based on 2D features matching and relative motion, and then the scale can be estimated from the depth.