COMPUTER VISION

PROJECT-3

CAMERA CALIBRATION AND FUNDAMENTAL MATRIX CALCULATION USING RANSAC

Dipan Banik

May 9, 2021

1 INTRODUCTION :

In this project, we use the geometric relationships between images taken from multiple views to compute camera positions and estimate fundamental matrices for various scenes.Specifically we will estimate the camera projection matrix or calibration matrix, which maps 3D world coordinates to image coordinates, as well as the fundamental matrix, which relates points in one scene to epipolar lines in another. The camera projection matrix and the fundamental matrix can each be estimated using point correspondences. To estimate the projection matrix (camera calibration), the input is corresponding 3d and 2d points. To estimate the fundamental matrix the input is corresponding 2d points across two images. We will start out by estimating the projection matrix and the fundamental matrix for a scene with ground truth correspondences. Then we eill move on to estimating the fundamental matrix using point correspondences from ORB. By using RANSAC to find the fundamental matrix with the most inliers, we can filter away spurious matches and achieve near perfect point to point matching.

2 PROCESS :

2.1 Projection Matrix:

The goal is to compute the projection matrix that goes from world 3D coordinates to 2D image coordinates.Recall that using homogeneous coordinates the equation for moving from 3D world to 2D camera coordinates is

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \simeq M \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}.$$

Where M is camera projection matrix.

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix}$$

We can solve for M by setting up a homogeneous system -

$$\begin{bmatrix} \mathrm{X}_1 & \mathrm{Y}_1 & \mathrm{Z}_1 & 1 & 0 & 0 & 0 & 0 & \text{-u}_1X_1 & \text{-u}_1Y_1 & \text{-u}_1Z_1 \\ 0 & 0 & 0 & 0 & \mathrm{X}_1 & \mathrm{Y}_1 & \mathrm{Z}_1 & 1 & \text{-v}_1Z_1 & \text{-v}_1Y_1 & \text{-v}_1Z_1 \\ . & . & . & . & . & . & . & . & . & . & . \\ \mathrm{X}_n & \mathrm{Y}_n & \mathrm{Z}_n & 1 & 0 & 0 & 0 & 0 & \text{-u}_nX_n & \text{-u}_nY_n & \text{-u}_nZ_n \\ 0 & 0 & 0 & 0 & \mathrm{X}_n & \mathrm{Y}_n & \mathrm{Z}_n & 1 & \text{-v}_nZ_n & \text{-v}_nY_n & \text{-v}_nZ_n \end{bmatrix} * \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ u_n \\ v_n \end{bmatrix}$$

At this point, you're almost able to set up our linear regression to find the elements of the matrix M. There's only one problem, the matrix M is only defined up to a scale. So these equations have many different possible solutions, in particular M = all zeros is a solution which is not very helpful in our context. The way around this is to first fix a scale and then do the regression. M has 12 elements, but since the scaling term m_{34} is arbitrary, we only need to solve for 11 variables. We solved this solve this using least squares with np.linalg.lstsq function.

$$M = \begin{pmatrix} 0.45828095 & -0.29474332 & -0.01396452 & 0.00402529 \\ -0.05085792 & -0.05459096 & -0.54104038 & -0.05237589 \\ 0.10901111 & 0.17835024 & -0.04428027 & 0.59683007 \end{pmatrix}$$

2.2 Camera Center :

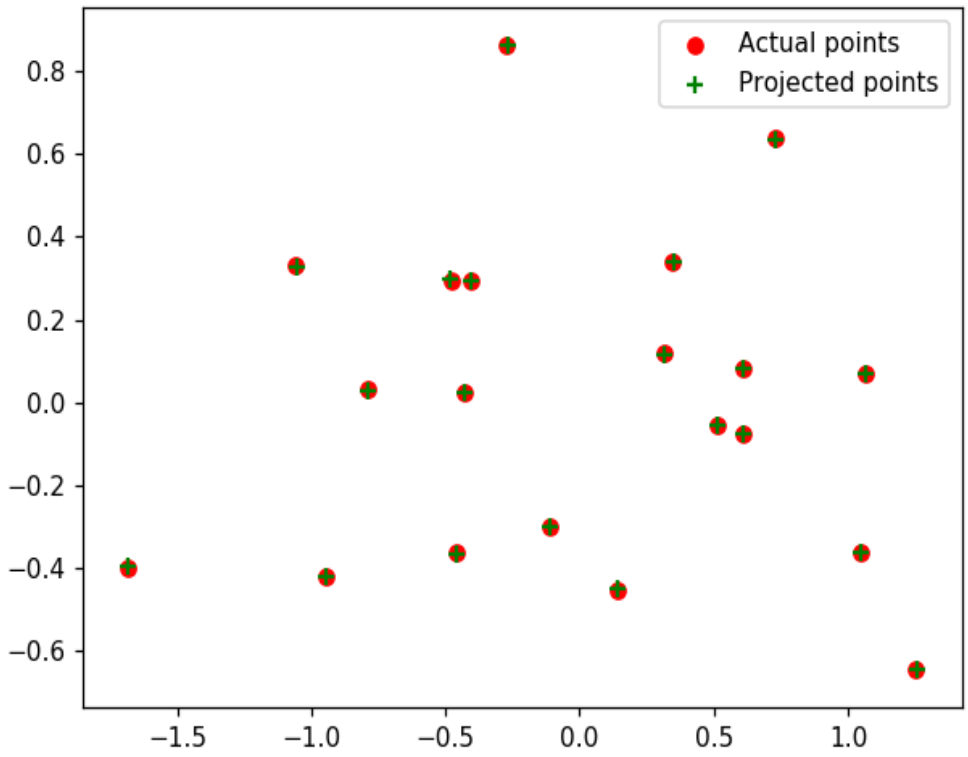
Writing M as the concatenation of a 3x3 matrix Q and a column matrix m4 as such

$$M = (Q|m_4)$$

we can compute the camera center by solving

$$\begin{aligned} C &= -Q^{-1}m_4 \\ &= -\begin{pmatrix} 0.45828095 & -0.29474332 & -0.01396452 \\ -0.05085792 & -0.05459096 & -0.54104038 \\ 0.10901111 & 0.17835024 & -0.04428027 \end{pmatrix}^{-1} \begin{pmatrix} 0.00402529 \\ -0.05237589 \\ 0.59683007 \end{pmatrix} \\ &= \begin{pmatrix} -1.51263977 \\ -2.35165965 \\ 0.28266502 \end{pmatrix} \end{aligned}$$

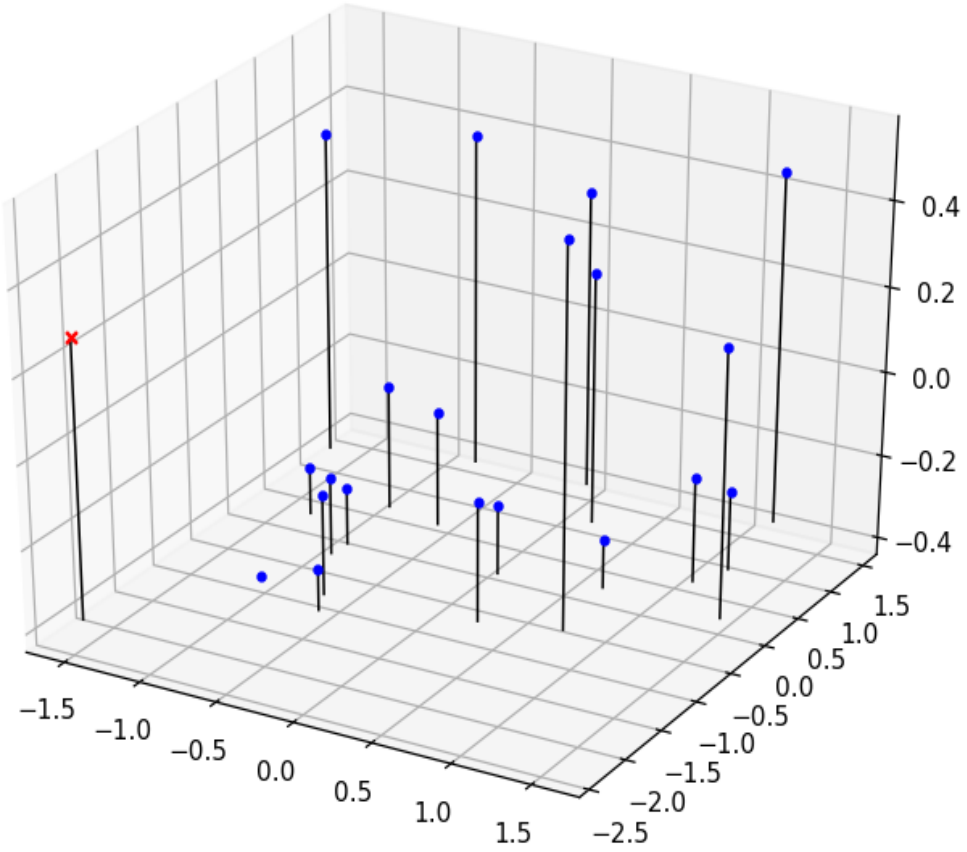
. The total residual is 0.044535.



2.3 Fundamental Matrix Estimation :

The next part of this project is estimating the mapping of points in one image to lines in another by means of the fundamental matrix.The normalized eight-point algorithm is used to compute the fundamental matrix given point correspondences x = (u, v) and x' = (u', v') in the left and right images, respectively. Each point correspondence generates one constraint on the fundamental matrix F and must satisfy the epipolar constraint equation

$$x'^{T} F x = 0.$$



Expanding the matrices out by multiplication, we obtain the following equation for n point correspondences

$$\begin{pmatrix} u'_1u_1 & u'_1v_1 & u'_1 & v'_1u_1 & v'_1v_1 & v'_1 & u_1 & v_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_nu_n & u'_nv_n & u'_n & v'_nu_n & v'_nv_n & v'_n & u_n & v_n & 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = \mathbf{0}$$

$$\Leftrightarrow \mathbf{Af} = 0.$$

where A is the n x 9 equation matrix, and f is a 9-element column vector containing the entries of the fundamental matrix F. From here, the least-squares solution f is easily computed by performing singular value decomposition (SVD) on the matrix $A = UDV^T$. It is well-known

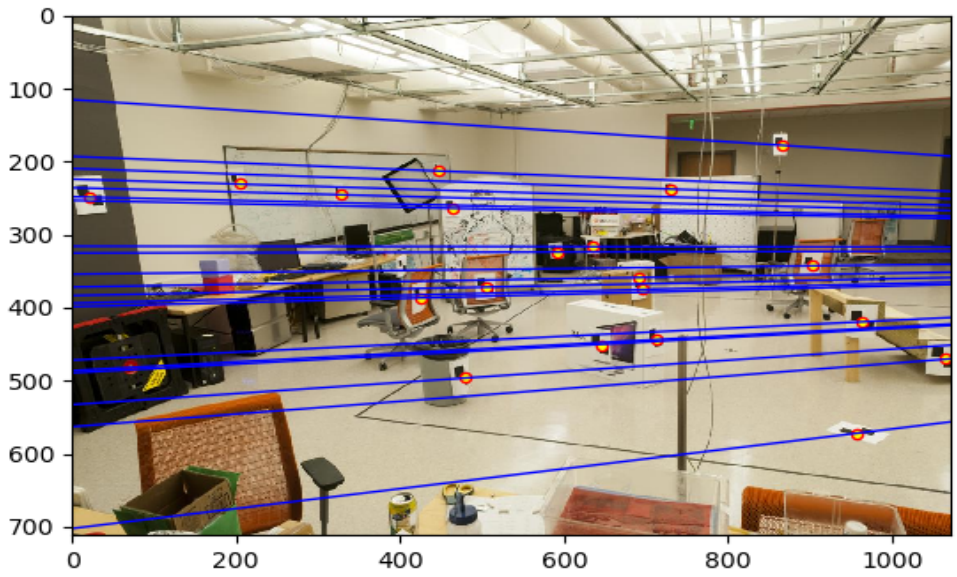
that the vector f that minimizes $||Af||$ such that $||f|| = 1$ can be found along the column of V corresponding to the least singular value. Next, we rearrange the 9 entries of f to create the 3x3 fundamental matrix F. Then, we perform SVD on F to obtain

$$F = U_f D_f {V_f}^T$$

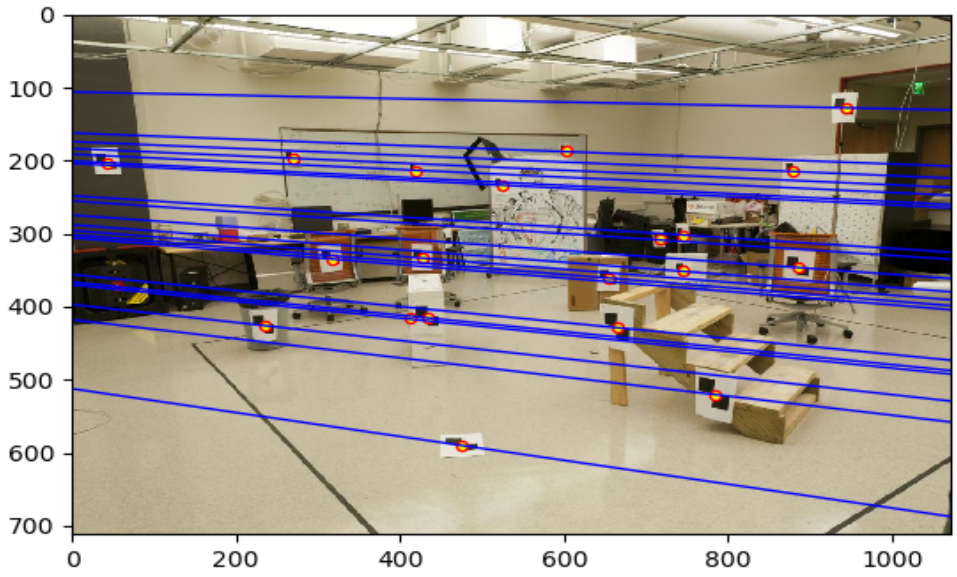
We set the smallest singular value of Df to 0 to create matrix D'f, thus reducing the rank of the matrix from 3 to 2, and from there we can recompute the rank-2 fundamental matrix as

$$F = U_f D_f {V_f}^T.$$

Using the version of the eight-point algorithm without prior coordinate normalization to compute the fundamental matrix on the laboratory image pair, we obtain the following fundamental matrix F (truncated to 4 decimal places) and the epipolar lines for both images:



$$F = \begin{pmatrix} -1.17248591e^{-07} & 1.60824663e^{-06} & -4.01980786e^{-04} \\ 1.11212887e^{-06} & -2.73443755e^{-07} & 3.23319884e^{-03} \\ -2.36400817e^{-05} & -4.44404958e{-}^{03} & 1.03455561e^{-01} \end{pmatrix}$$



2.4 Fundamental Matrix with RANSAC :

The starter code uses ground truth matches (along with a flag to perform feature point matching with the ORB descriptor) for an image pair. Optionally, we will perturb these matches with noise using your functions we just wrote. We'll use these possible point correspondences and RANSAC to try and find a good fundamental matrix. We will iteratively choose a random set of point correspondences (we took 8 points), solve for the fundamental matrix using the function we wrote for part II, and then count the number of inliers. Inliers in this context will be point correspondences that "agree" with the estimated fundamental matrix. We run our code on these following images:-

2.4.1 Mount Rushmore:

This pair is easy, and most of the initial matches are correct. The base fundamental matrix estimation without coordinate normalization will work fine with RANSAC.

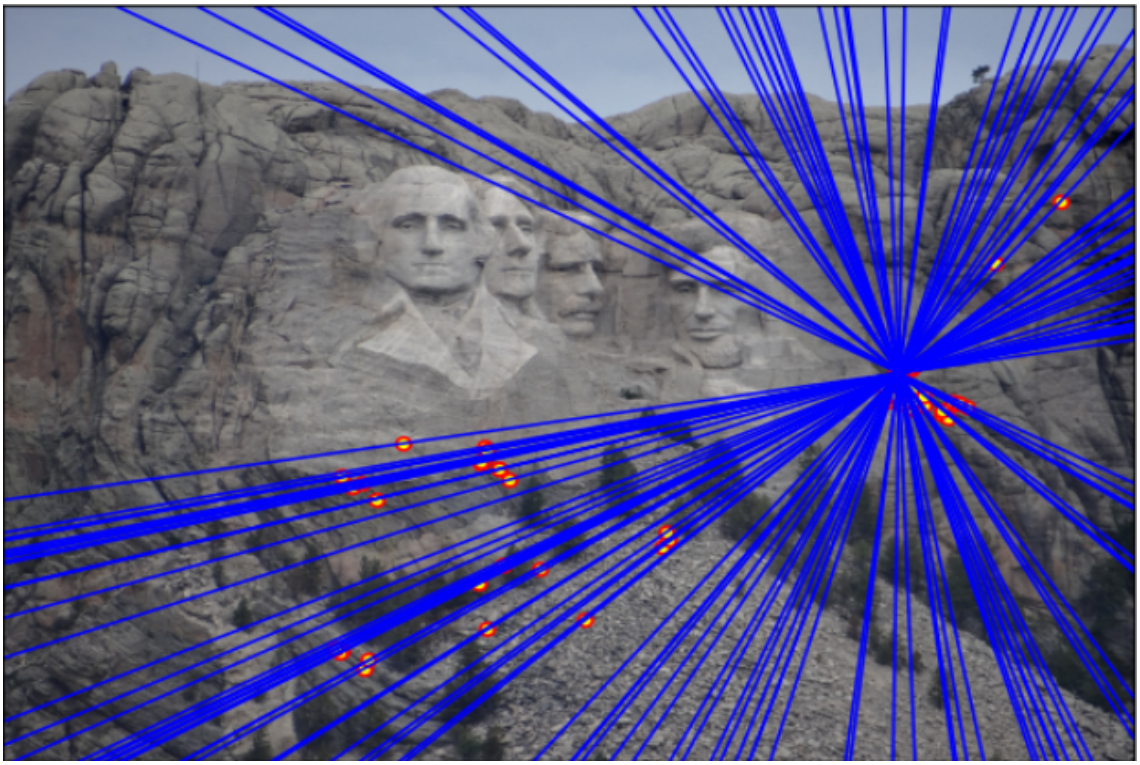


Figure 1: Epipolar Lines in first Image

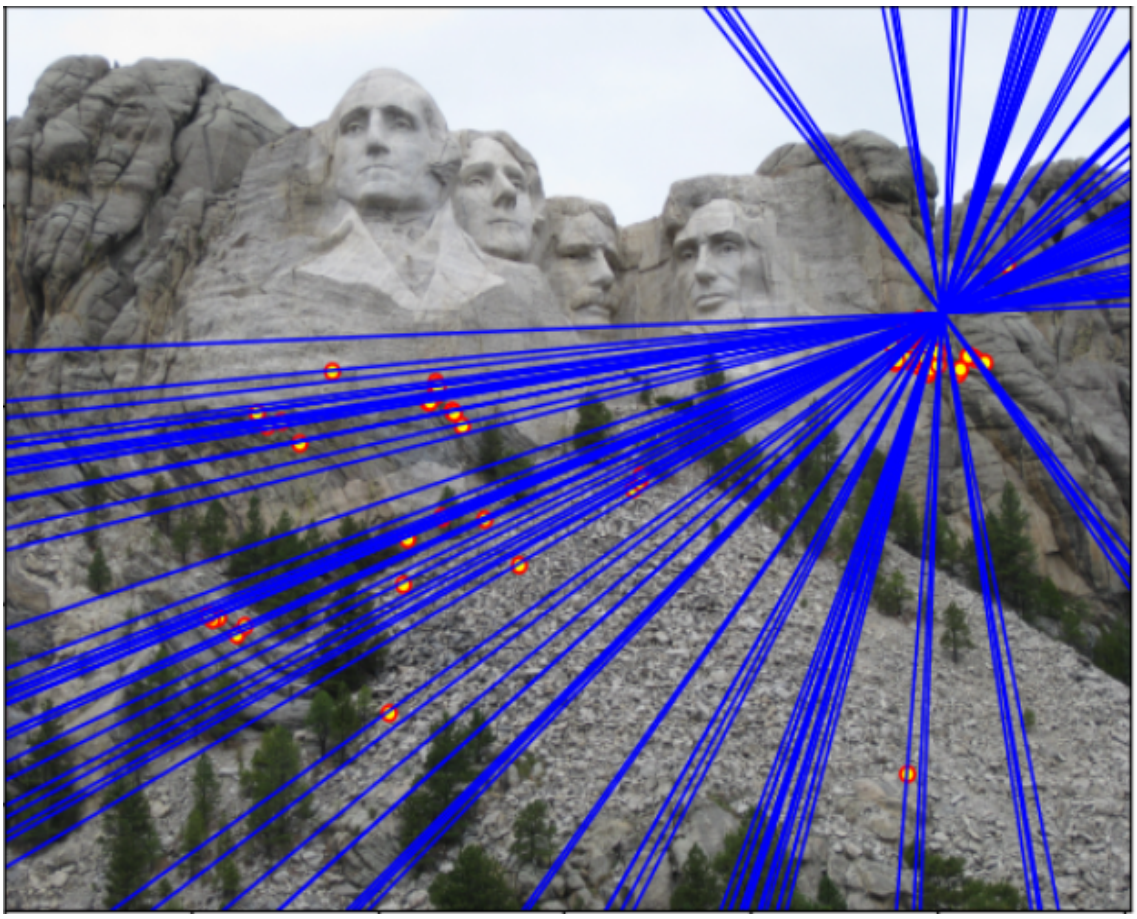


Figure 2: Epipolar Lines in second Image

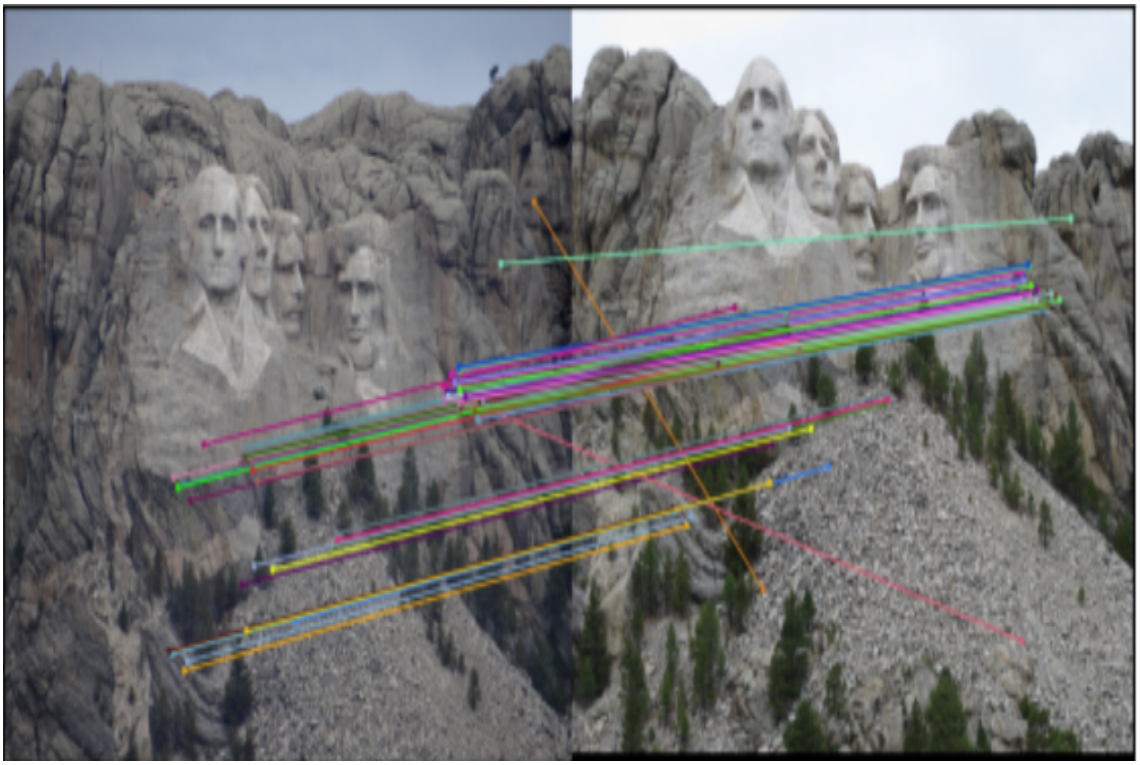


Figure 3: match two images

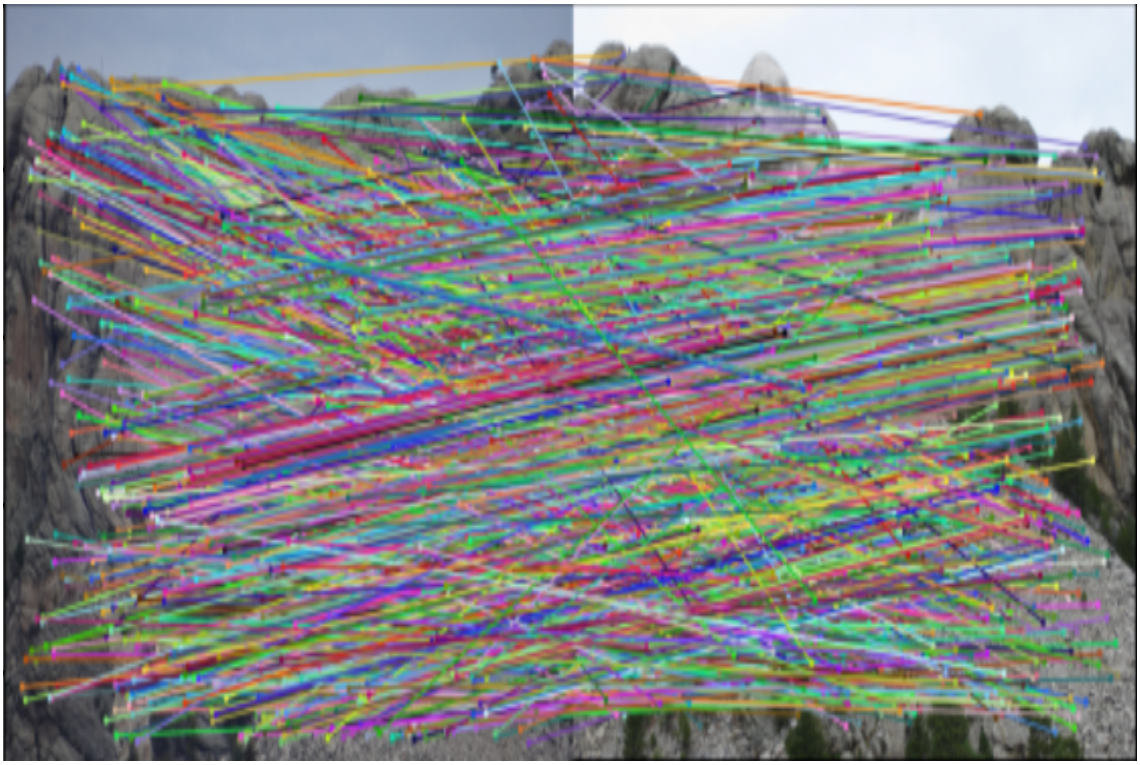


Figure 4: Matching Using ORB

2.4.2 Notre Dame:

This pair is difficult because the keypoints are largely on the same plane. Still, even an inaccurate fundamental matrix can do a pretty good job of filtering spurious matches.

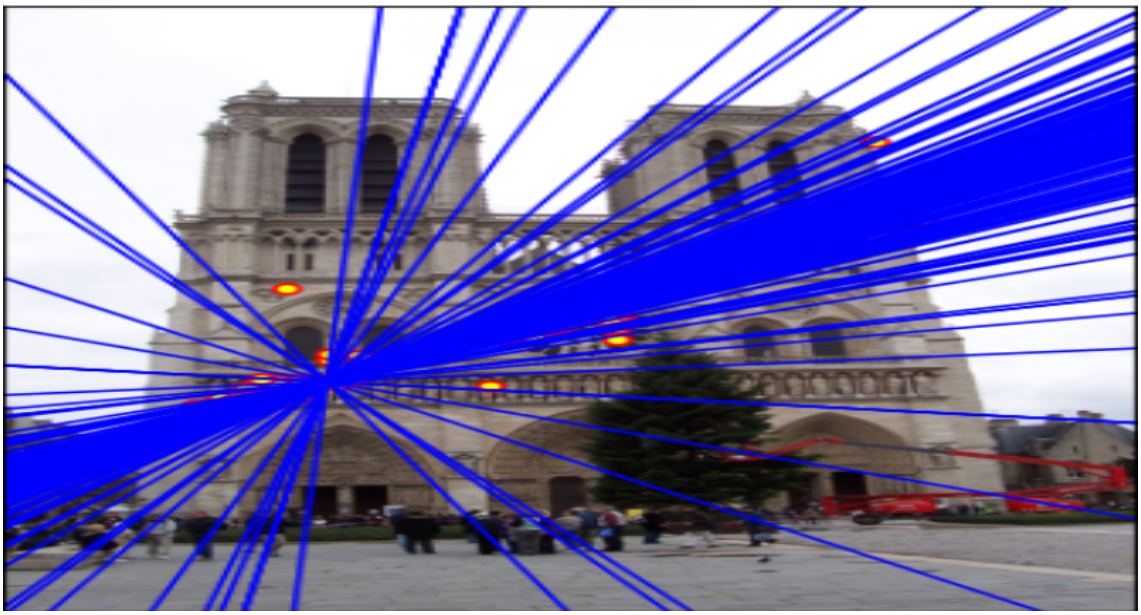


Figure 5: Epipolar Lines in first Image

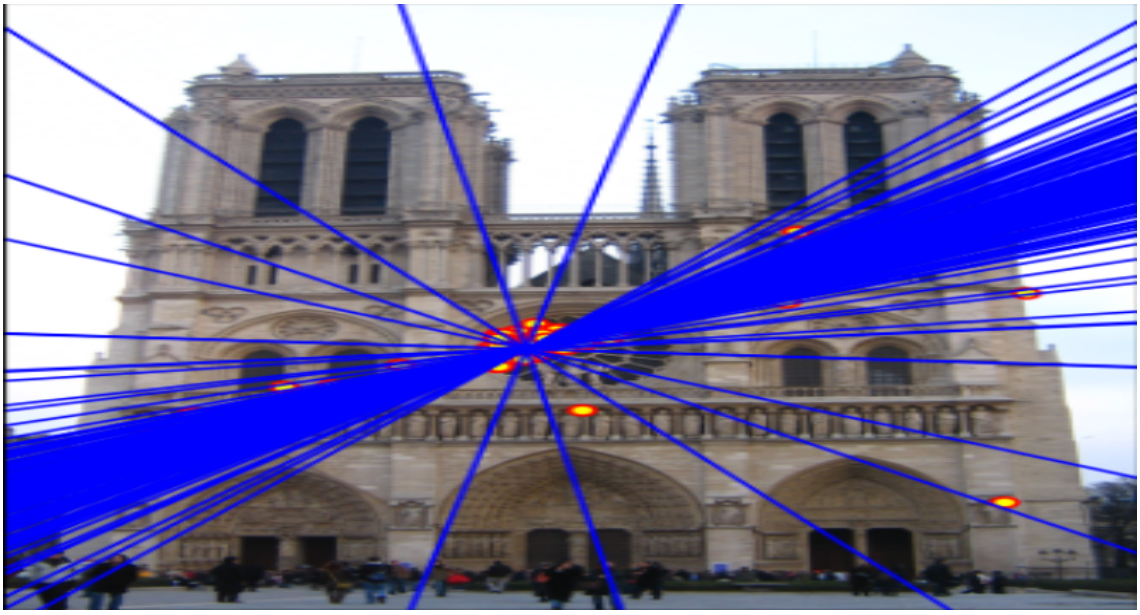


Figure 6: Epipolar Lines in second Image

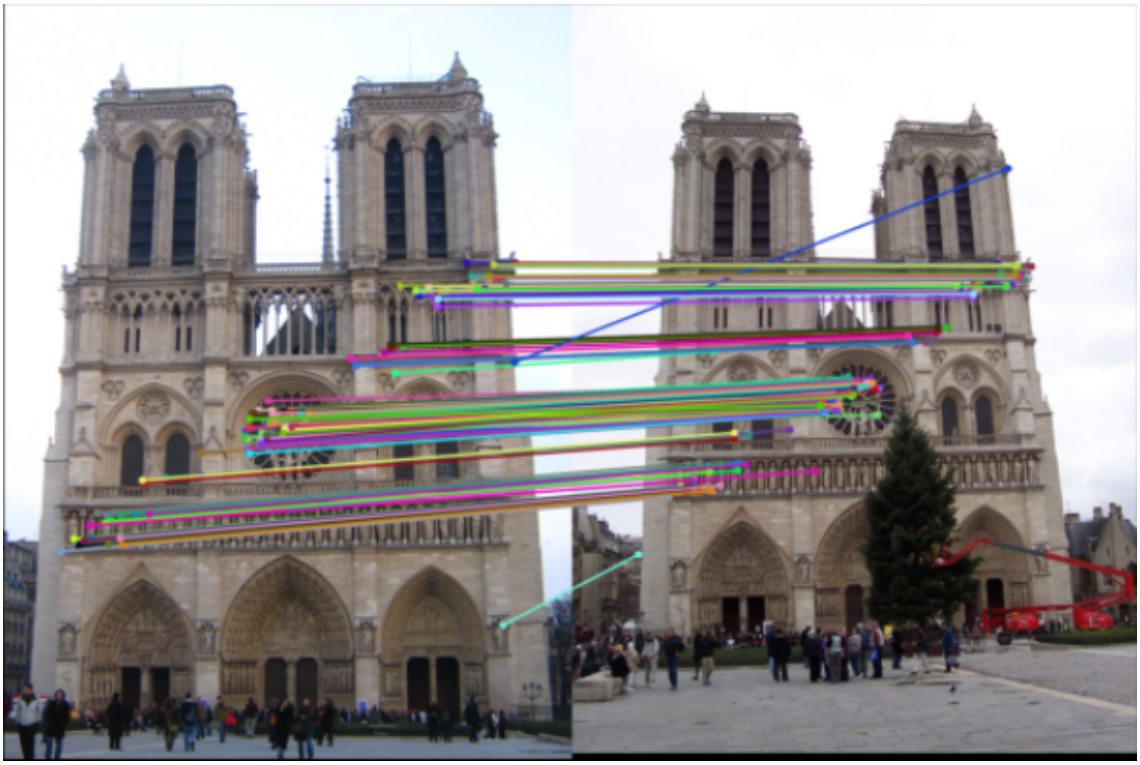


Figure 7: Match Two Images

Figure 8: Matching Using ORB

2.4.3 Gaudi:

This pair is difficult and doesn't find many correct matches unless you run at high resolution, but that will lead to tens of thousands of ORB features, which will be somewhat slow to process. Normalizing the coordinates seems to make this pair work much better.

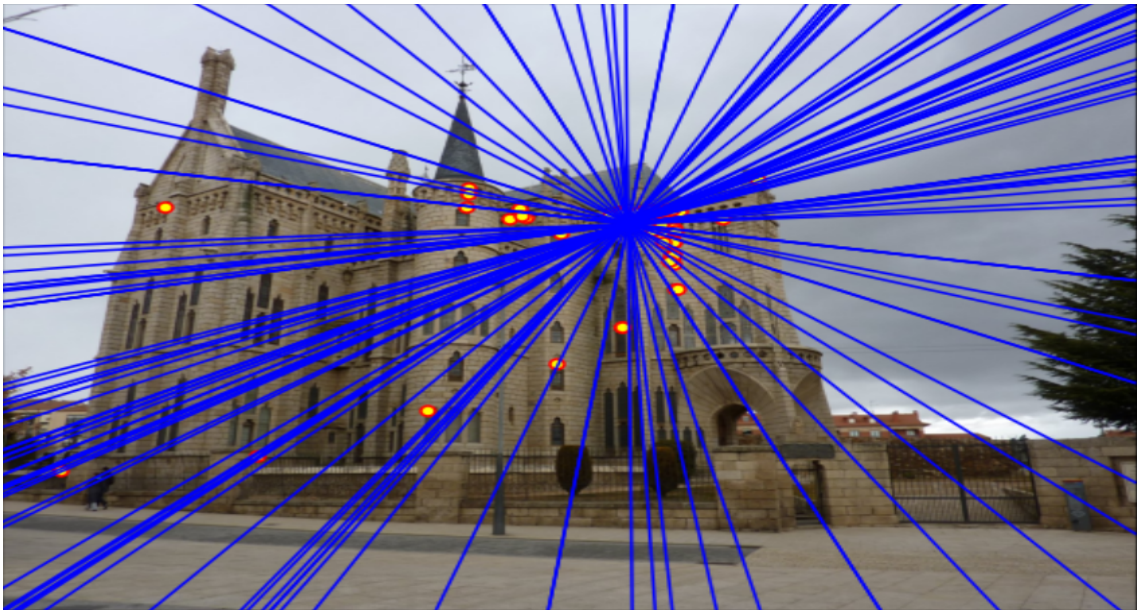


Figure 9: Epipolar Lines in first Image

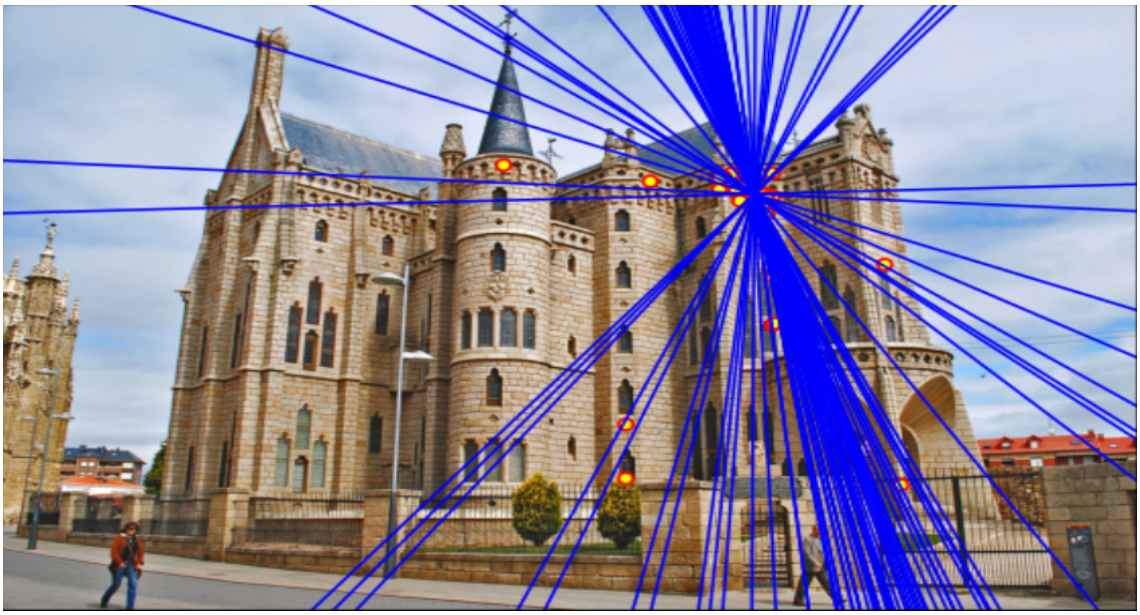


Figure 10: Epipolar Lines in second Image

Figure 11: Match Two Images

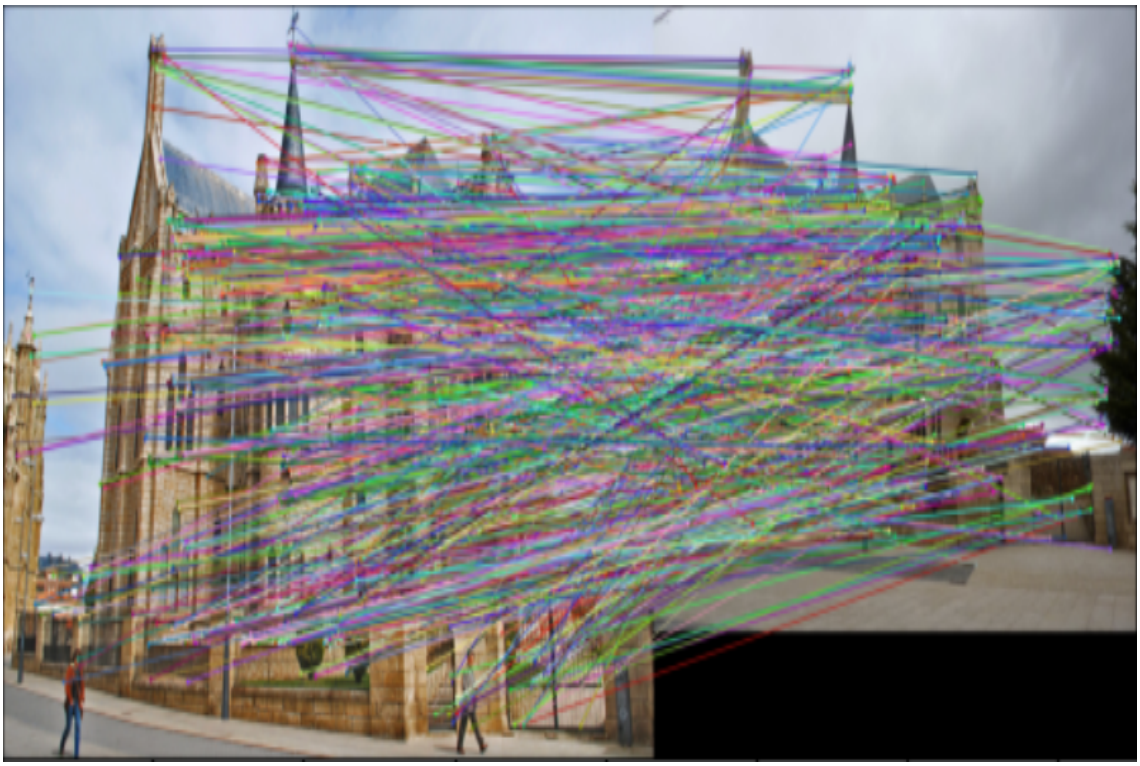


Figure 12: Matching Using ORB

2.4.4 Woodruff:

This pair has a clearer relationship between the cameras (they are converging and have a wide baseline between them). The estimated fundamental matrix is less ambiguous and you should get epipolar lines qualitatively similar to part 2 of the project.

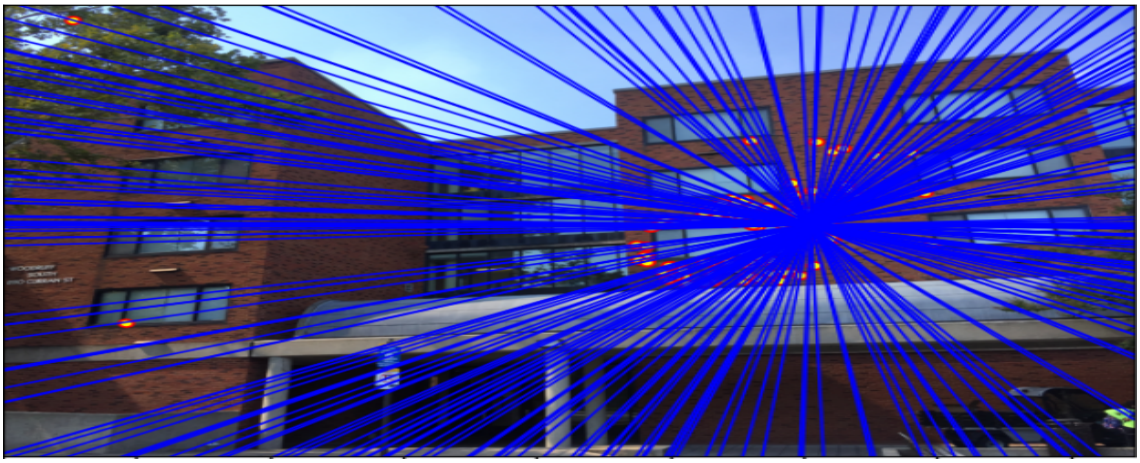


Figure 13: Epipolar Lines in first Image

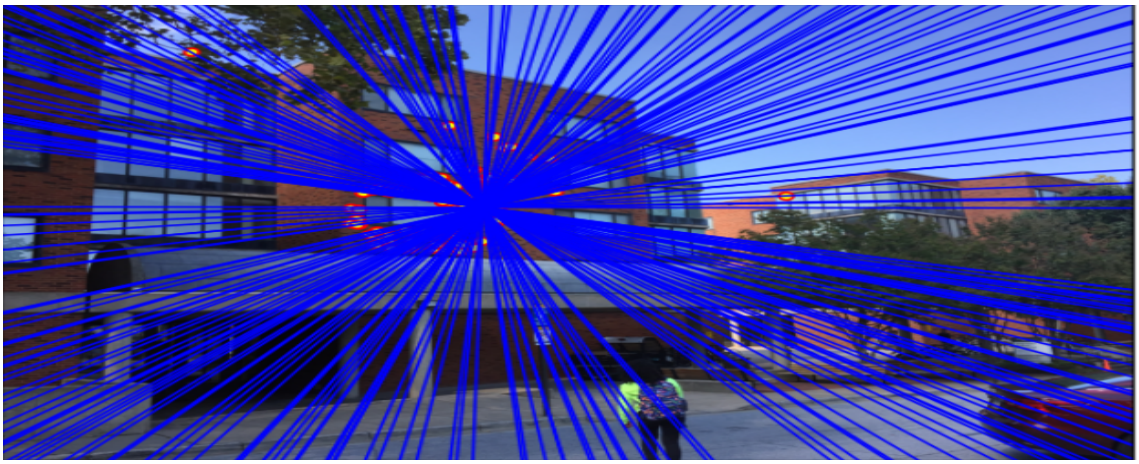


Figure 14: Epipolar Lines in second Image

Figure 15: Match Two Images

Figure 16: Matching Using ORB

3 CONCLUSION :

We found ways to apply theoritically computable epipolar lines to real world applications. This type of applications can be used for finding any point on an image by performing a linear search, saving time and computational power to process the whole image. RANSAC works pretty fine in most cases, but since it is performing randomly, it may not find the best or optimal match everytime. The best way to do this is indeed very compuatationally expensive but, indeed the local minimum of the best points might be similar to the global optimum, just as machine learning problems. So, we can increase this part by further modifying these things for 3D resonstructions and other exciting stuffs like point cloud reconstructions etc. in the future.