CV PROJECT

Camera Calibration and Fundamental Matrix Estimation with RANSAC

Mahendra Nandi
Sourav Karmakar
Guided By-Tamal Maharaj
MSc. COURSE ON BIG DATA ANALYTICS
Department of COMPUTER SCIENCE
Ramakrishna Mission Vivekananda Educational Research Institute BELUR

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our supervisor, Tamal Maharaj, Department of Computer Science as well as our HOD Swathy Prabhu Maharaj who gave the golden opportunity to do this wonderful project on the topic Hybrid Image , which also helped me in doing a lot of Research and we came to know about so many new things we are really thankful to them. Secondly we would also like to thank our friends who helped us a lot in finalizing this project within the limited time frame.

Abstract

The goal of this project is to introduce ourselves to camera and scene geometry. 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'll move on to estimating the fundamental matrix using point correspondences from ORB, which is an alternative to SIFT. 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

Contents

List of Figures

1. Camera Projection Matrix

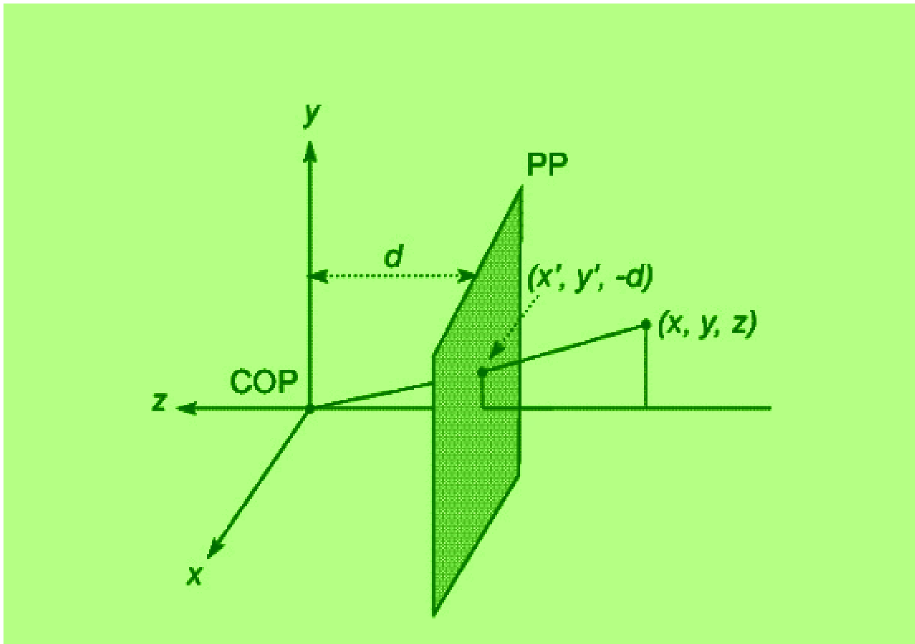


Fig. 1. projection: visualization

The goal initially is to compute the projection matrix that translates from 3D coordinates to 2D image coordinates. Our task is to find M (to be discussed below). We are going to set up a system of linear equations to find the least squares regression solution for these camera matrix parameters, given correspondences between 3D world points and 2D image points.We see two solutions to this problem, which both require rearranging the coefficients of the 3D and 2D point evidence:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u*s \\ v*s \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

This can be written as:

$u = \frac{m_{11}X+m_{12}Y+m_{13}Z+m_{14}}{m_{31}X+m_{32}Y+m_{33}Z+m_{34}}$

$or,\ (m_{31}X + m_{32}Y + m_{33}Z + m_{34})u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$

$or, 0 = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}uX - m_{32}uY - m_{33}uZ + m_{34}u$

$v = \frac{m_{21}X+m_{22}Y+m_{23}Z+m_{24}}{m_{31}X+m_{32}Y+m_{33}Z+m_{34}}$

$or,\ (m_{31}X+m_{32}Y+m_{33}Z+m_{34})v = m_{21}X+m_{22}Y+m_{23}Z+m_{24}$

$or, 0 = m_{11}X+m_{12}Y+m_{13}Z+m_{14}-m_{31}vX-m_{32}vY-m_{33}vZ+m_{34}v$

To fix the scale for the matrix, We set the last term in the matrix, which is m 34 , as 1. Then we solve the matrix based on the given point pairs. If we solve for M using all the points, we should receive a matrix that is a scaled equivalent of the following matrix:

$$\begin{bmatrix} 0.76785834 & -0.49384797 & -0.02339781 & 0.00674445 \\ -0.0852134 & -0.09146818 & -0.90652332 & -0.08775678 \\ 0.18265016 & 0.29882917 & -0.07419242 & 1. \end{bmatrix}$$

Once we have an accurate projection matrix M, it is possible to tease it apart into the more familiar and more useful matrix K of intrinsic parameters and matrix $[R|T]$ of extrinsic parameters. For this project we will only ask you to estimate one particular extrinsic parameter: the camera center in world coordinates. Let us define M as being made up of a 3x3 matrix called Q, with 4th column m 4 :
$M = (Q|m_4)$ The center of the camera C could be found by:
$C = Q^{-1} * m_4$. If we use the normalized 3D points and M given above, we would receive camera center: $C_{normA} = [-1.5126, -2.3517, 0.2827]$

2. Estimating Fundamental Matrix:

Recall that the definition of the fundamental matrix is:

$$\text{Af=} \begin{bmatrix} \mathrm{u}_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 & 1 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ u_nu'_n & u_nv'_n & u_n & v_nu'_n & v_nv'_n & v_n & u'_n & v'_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ . \\ . \\ . \\ f_{33} \end{bmatrix} \text{=0}$$

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \text{=0}$$

The fundamental matrix is sometimes defined as the transpose of the above matrix with the left and right image points swapped. Both are valid funda- mental matrices, but the

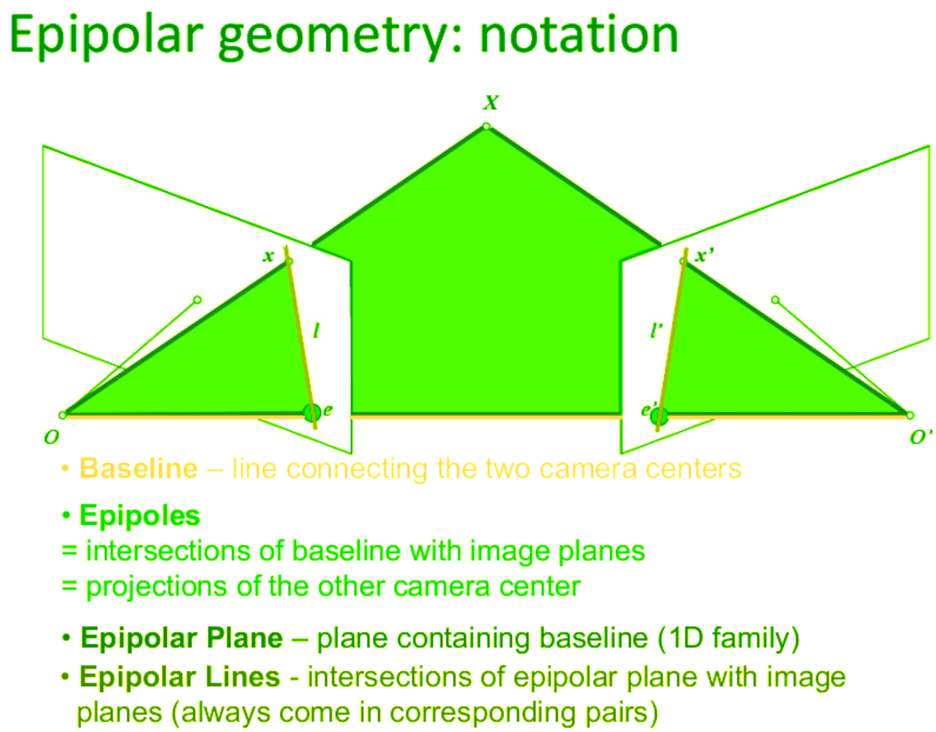


Fig. 2. Epipolar Geometry notation

visualization functions in the starter code assume3 you use the above form.
Another way of writing this matrix equations is:

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11}u + f_{12}v + f_{13} \\ f_{21}u + f_{22}v + f_{23} \\ f_{31}u + f_{32}v + f_{33} \end{bmatrix} = 0$$

We need to fix the scale and solve it. This leaves us able to solve the system with 8 point correspondences (or more, as we recover the best least squares fit to all points). The least squares estimate of F is full rank; however, the fundamental matrix is a rank 2 matrix. As such we must reduce its rank. In order to do this we can decompose F using singular value decomposition into the matrices $UDV^T = F$. We can then estimate a rank 2 matrix by setting the smallest singular value in D to zero thus generating D_2 . The fundamental matrix is then easily calculated as $F = UD_2V^T$. You can check your fundamental matrix estimation by plotting the epipolar lines using the plotting function provided in the starter code.

3. Recovering The Camera Center

Here in a single image it is shown how to recover the camera centre.As we need the camera centre , To get that we know $t = -RC$ so $C = -R^{-1}t$ where $t = K^{-1}m_4$, so , $C = -R^{-1}K^{-1}m_4 = -Q^{-1}m_4$ where $Q = KR$
Here K is intrinsic matrix, R is representing rotation and t representing translation

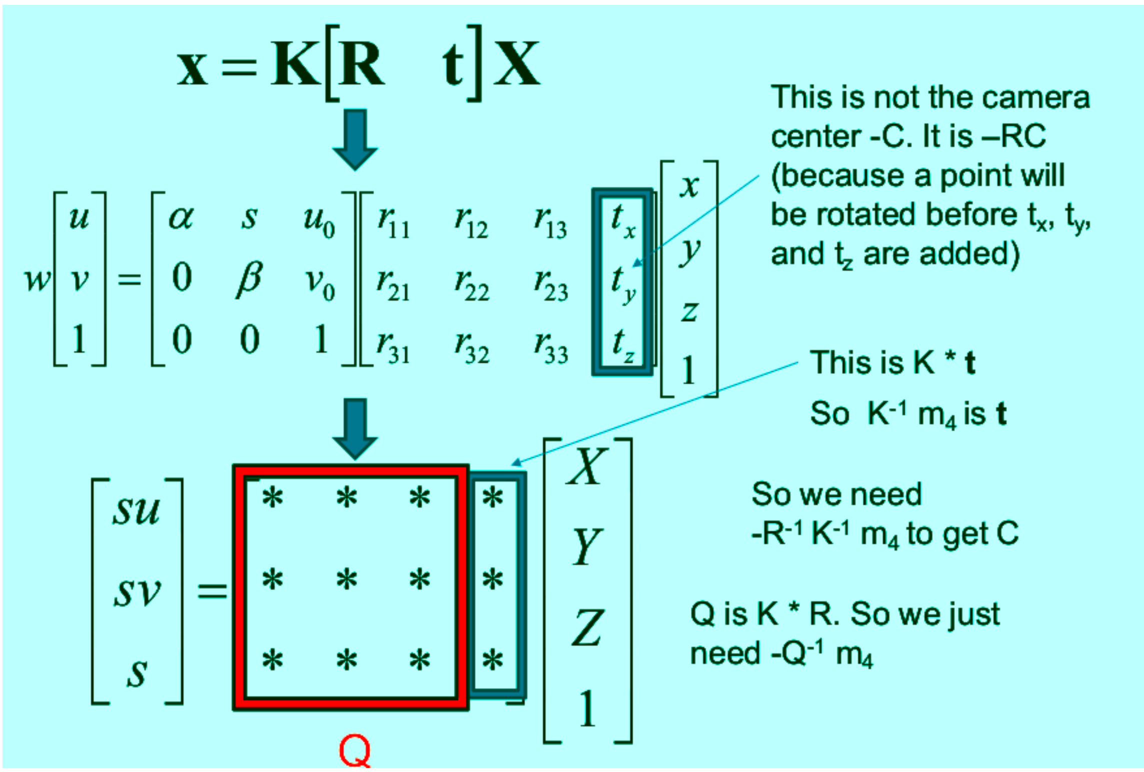


Fig. 3. Recovering Camera Centre

4. Fundamental Matrix with RANSAC

With parts 1 and 2 completed, we are almost ready to use epipolar geometry to improve our point correspondences from the feature matching in this post. However, even with an off the shelf SIFT pipeline that will presumably work much better than our earlier implementation, there is likely to be error and there is no easy way to determine the outliers. One method to properly fit a model, and the method used in this project, is RanSaC (Random Sample Consensus). For RanSaC, 8–12 sample correspondences are randomly chosen to calculate fit a fundamental matrix to. Then the number of inlier correspondences are counted for this randomly sampled fundamental matrix. This process is repeated thousands of times and the fundamental matrix with the most inliers will be returned as the chosen model. As can be seen below, this method produces drastically better results when compared point correspondences from the feature matching in this post. However, this is not perfect. I found my implementation would perform well with 10,000–20,000 iterations, 8/9 samples, and a threshold between 0.003 and 0.009. With these parameters, I was able to achieve between 300–500 matches while staying under 2 minutes of runtime. It easy to say all attempts were above 90 % accuracy with a few errors per image, maybe a handful in the second image set (Gaudi). The Gaudi example would usually match a few points on the rightmost tree. I did not have time to test any filtering but I considered that some post-processing filtering could remove matches based on a few metrics of local patch or maybe a global translation metric.

5. Algorithm:

Algorithm 1: RANSAC

1: Select randomly the minimum number of points required to determine the model parameters.

2: Solve for the parameters of the model.

3: Determine how many points from the set of all points fit with a predefined tolerance .

4: If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold , re-estimate the model parameters using all the identified inliers and terminate.

5: Otherwise, repeat steps 1 through 4 (maximum of N times).

Algorithm 2: 8-point algorithm

1. Solve a system of homogeneous linear equations

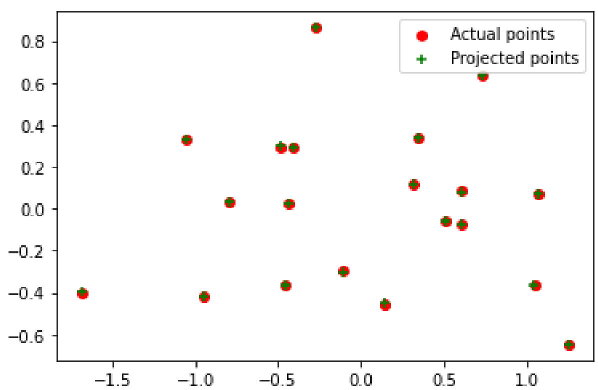
a. Write down the system of equations

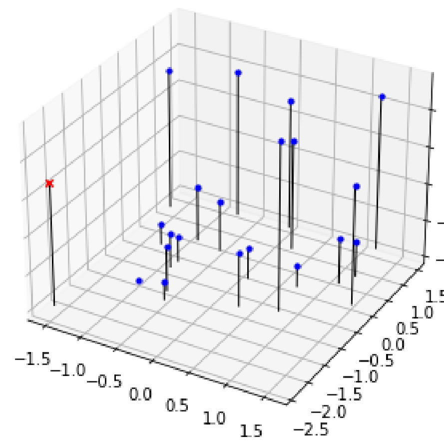
b. Solve f from Af=0 using SVD
Python:
$[U,D,V] = svd(A)$;
f = V(:, -1);
F = reshape(f, [3 3]);
2. Resolve det(F) = 0 constraint using SVD
Python:
$[U,D,V] = svd(F)$;
D[-1][-1] = 0;
F = U*D*V';

6. Result:

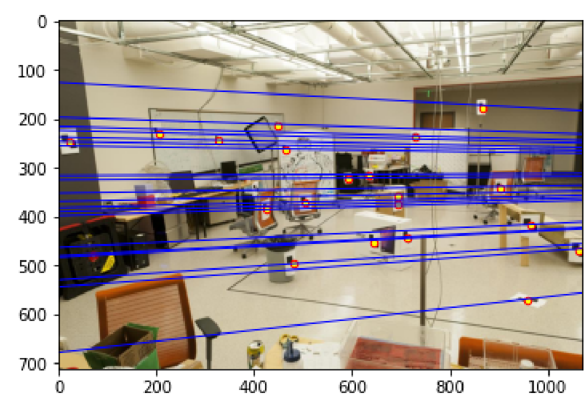
For part I we have expected output (matrix M and camera center C). These are numerical estimates so we won't be checking for exact numbers, just approximately correct locations. The projection matrix is

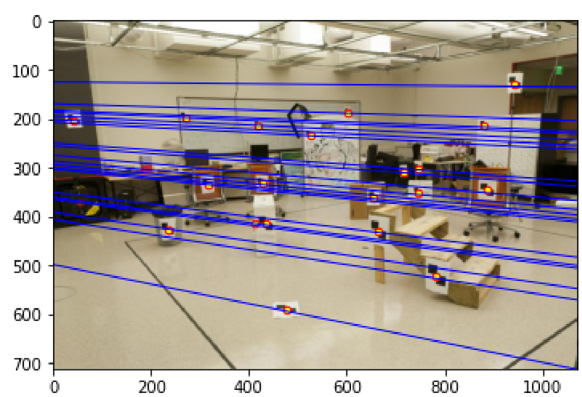
$$\begin{bmatrix} 0.76785834 & -0.49384797 & -0.02339781 & 0.00674445 \\ -0.0852134 & -0.09146818 & -0.90652332 & -0.08775678 \\ 0.18265016 & 0.29882917 & -0.07419242 & 1. \end{bmatrix}$$





For part II we evaluated based on our estimate of the fundamental matrix and by testing how good our estimate of the fundamental matrix is by drawing the epipolar lines on one image which correspond to a point in the other image. we see all of the epipolar lines crossing through the corresponding point in the other image.





7. Useful Functions

• np.linalg.svd(): This function returns the singular value decomposition of a matrix. Useful for solving the linear systems of equations you build and reducing the rank of the fundamental matrix.

• np.linalg.solve(): This function returns the solution of matrix x from equation of type Ax=b.

• np.random.randint(): Let's you pick integers from a range. Useful for RANSAC.

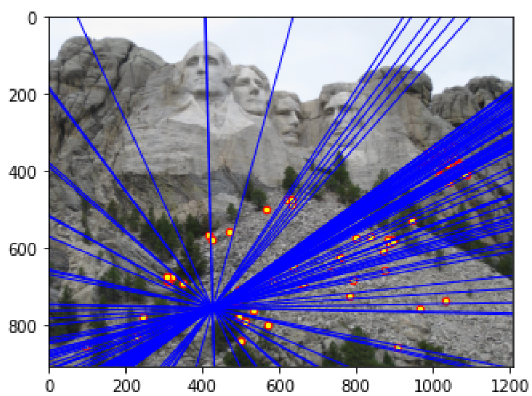
• np.argsort(): It's sort an array in ascending order and returns the indices of that sorted array.

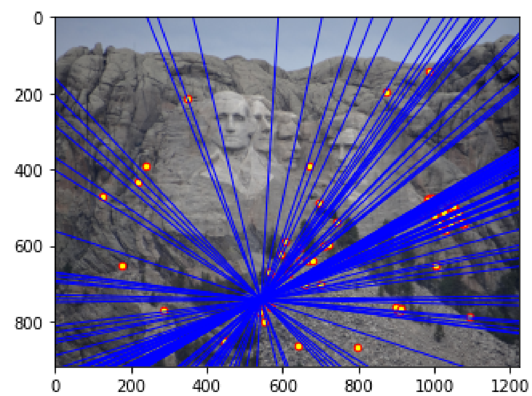
• np.insert(): It helps to add a new column or row of a matrix.

• enumerate(): It helps to get the indices of the zipped objects.

8. Different examples

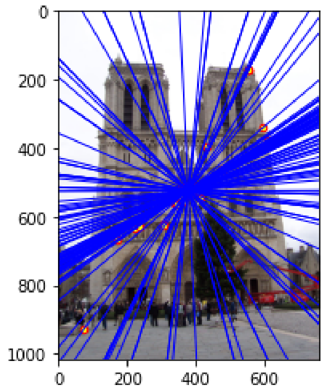
Mount Rushmore

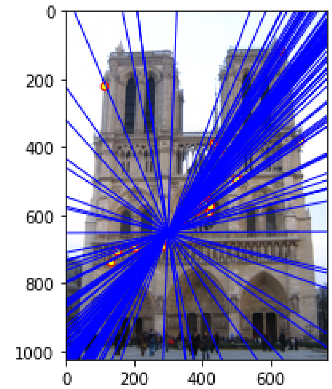




This pair is easy, and most of the initial matches are correct. The base fundamental matrix estimation without coordinate normalization will work fine with RANSAC.

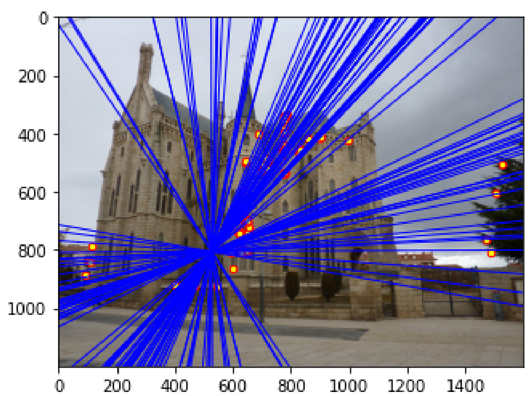
Notre Dame

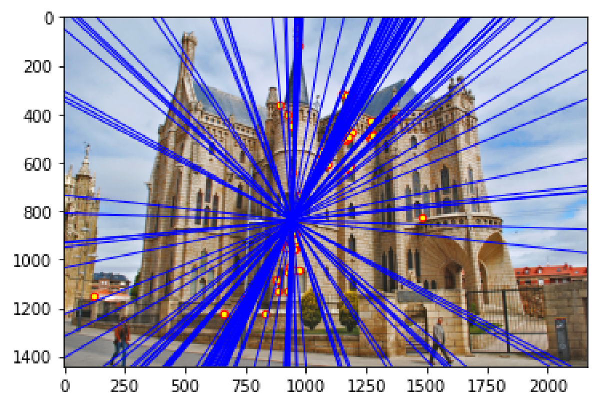




This pair is difficult because the keypoints are largely on the same plane. Still, even an inaccurate fundamental matrix can do a pretty good job of filtering spurious matches.

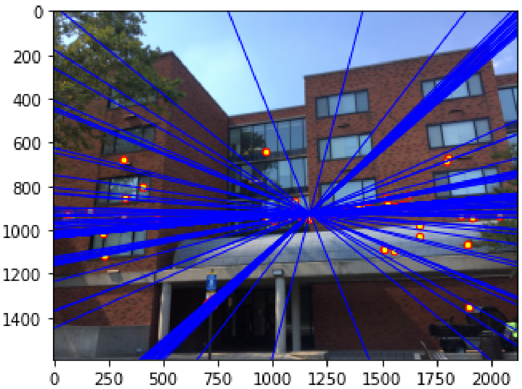
Gaudi

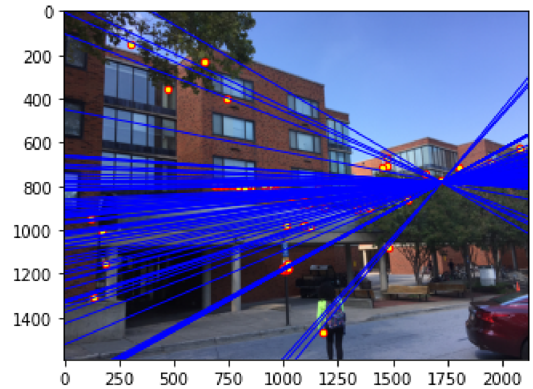




This pair is difficult and doesn't find many correct matches unless you run at high resolution, but that will lead to tens of thousands of ORB features, which will be somewhat slow to process. Normalizing the coordinates seems to make this pair work much better.

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.

9. Conclusion:

The objective of this project was to improve upon image matching by leveraging epipolar geometry of a stereo image pair.**??**