CS 342: Computer Vision

Project 3: Camera Calibration and Fundamental Matrix Estimation with RANSAC

Sourav Bhattacharjee
Sawan Aich
Ramakrishna Mission Vivekananda Educational and Research Institute
Department of computer science
MSc, Big Data Analytics

Table of Contents

1. Camera projection matrix

A camera makes a mapping between the 3D world and a 2D image. It projects the 3D world to 2D. The equation of the homogeneous image (3x1), camera matrix (3x4) and a homogeneous world point (4x1) is written as,

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

Another way of writing this equation is:

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

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

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

$$\implies 0 = m_{21}X + m_{22}Y + m_{23}Z + m24 - m_{31}vX - m_{32}vY - m_{33}vZ - m_{34}u$$

We will be able to set up our linear regression to find the elements of the matrix M. The problem with this procedure is that matrix M is only defined up to a scale. In some case it may be that the value of the matrix M will be 0, i.e., all the elements of the matrix M will be 0, which is not very helpful in our context. The way around this is to first fix a scale and do the regression. There are several options for this, we can fix the last element to 1 then find the remaining coefficients, or we can use the singular value decomposition to directly solve the constrained optimization problem:

$$min \left|\left|Ax\right|\right| such\ that \left|\left|x\right|\right| = 1$$

For this part, we used the function of the starter code named as calculateprojectionmatrix()which took two points' list i.e., point's list corresponding to 2D and point's list corresponding to the 3D coordinate system. All we needed to do is we used the least squares solution to solve for M using np.linalg.inv()function. We then returned the M matrix by adjusting the parameters accordingly. This is actually done by solving a non-homogeneous system of linear equation,by solving it through least squares, given in the following:

$$\begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 \\ -u_1Z_1 & -u_1 & & & & & & & & \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 \\ -v_1Z_1 & -v_1 & & & & & & & & \\ & & & \cdot & & & & \cdot & & \\ & & & \cdot & & & & \cdot & & \\ & & & \cdot & & & & \cdot & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n \\ -u_nZ_n & -u_n & & & & & & & & \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n \\ -v_nZ_n & -v_n & & & & & & & & \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

By doing this we got a projection matrix of:

$$\begin{bmatrix} 0.45827554 & -0.29474236 & -0.01395748 & 0.0040258 \\ -0.05085589 & -0.0545847 & -0.54105993 & -0.05237592 \\ 0.10900958 & 0.17834549 & -0.04426782 & 0.5968205 \end{bmatrix}$$

with a residual of 0.044549.
We can further use this step to find the camera centre.

Fig. 1. Image on left is actual points and the projected points estimated from the project matrix M, Image on right is the estimated location of the camera

2. Fundamental Matrix Estimation

We have used a function named estimatefundamentalmatrix() to feed in two points, i.e.,pointsa and pointsb.The first list of points contains the 2D points in image A and the second one contains a list of 2D points in image B. This function returns 3x3 shaped fundamental matrix. We have used the 8 point algorithm for estimating the fundamental matrix. We have

used least squares solution using SVD of equations from 8 pairs of correspondences. It can be written as

$$\begin{pmatrix} u_1u_1' & u_1v_1' & u_1 & v_1u_1' & v_1v_1' & v_1 & u_1' & v_1' & 1 \\ & & & & \cdot & & \cdot & & \cdot \\ \cdot & & & & & & & & \\ & & & & \cdot & & \cdot & & \cdot \\ \cdot & & & & & & & & \\ & & & & \cdot & & \cdot & & \cdot \\ \cdot & & & & & & & & \\ u_nu_n' & u_nv_n' & u_n & v_nu_n' & v_nv_n' & v_n & u_n' & v_n' & 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ \cdot \\ \cdot \\ f_{33} \end{pmatrix} = 0$$

We have used np.linalg.svd() to solve for F. The results of the images are shown in Fig

Fig. 2. Epipolar lines on two images

3. Fundamental Matrix Estimation with RANSAC

We found the best fundamental matrix estimation using RANSAC on potentially matching points. Our RANSAC loop contains call to the previous $functionestimate_fundamental_matrix()$. For uncluttered visualization of points, we needed to return more than 100 points for either left or right images

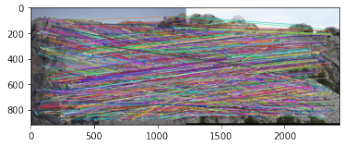

Fig. 3. Feature matches by running ORB

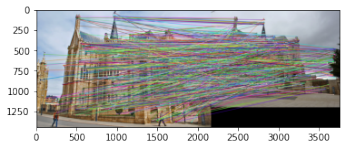

Fig. 4. Feature matches by running ORB

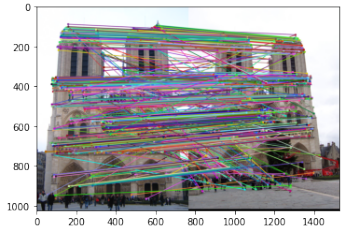

Fig. 5. Feature matches by running ORB

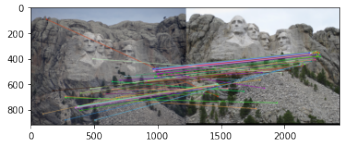

Fig. 6. Feature matches on both the images of Mount Rushmore

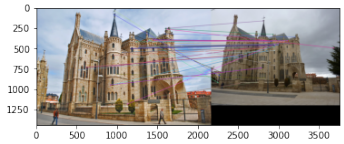

Fig. 7. Feature matches on both the images of Gaudi

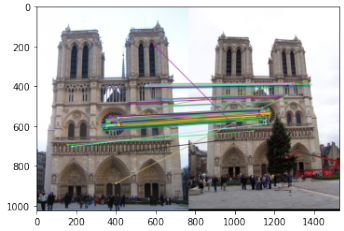

Fig. 8. Feature matches on both the images of Notredam

Fig. 9. Fundamental matrix estimation for Mount Rushmore

Fig. 10. Fundamental matrix estimation for Gaudi

Fig. 11. Fundamental matrix estimation for Notredam

4. Conclusion

We found ways to apply theoretically 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 computationally 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 reconstructions and other exciting stuffs like point cloud reconstructions etc. in the future.