Project 3 Report

ENPM673 SPRING 2021

Divyam Garg

April 26th, 2021

Table of Contents

Problem 1 : Steps

- Feature matching was done using ORB feature matching inbuilt function of OpenCv which provides key points and descriptor vector of the matched feature.
- The reason to use ORB was the key features can be sorted on the base of degree of match which means the first key point after sorting will have the best match.
- By default, it gives 300 matches and one-third of them were chosen.
- Next step was calculating fundamental matrix.
- First A matrix was calculated by choosing 8 same random key points for both the images and putting in formula provided.

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

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

- This matrix is decomposed using SVD and last row of V which is eigen vectors of AT.A because it corresponds to the least eigen value. As we know,

$$\mathbf{x}'^T F\mathbf{x} = 0. \quad \text{---- 1}$$

- If X is an epipolar line, eigen value of F should be zero as it becomes

$$F\mathbf{x} = 0. \quad \text{---- 2}$$

  Where x is epipole hence from $\boxed{A\boldsymbol{x} = \lambda\boldsymbol{x}.}$ lambda = 0.

- For sanity check any random index is chosen and the left hand side of the 1 is checked with the calculated F.
- Since some points are not the best match even after sorting this formula is executed for 1000 iteration and the best fit F matrix is seeded.
- Once F matrix is calculated, which maps the key feature along the baseline (unlike homography which maps point on image 1 to point on image 2), Essential Matrix is calculated by the given intrinsic parameter of the camera.

- Using Essential matrix camera pose (Rotation and Translation) are calculated because of the equation $(\mathbf{y}')^\top \mathbf{E}\,\mathbf{y} = 0$ where y_dash and y correspond to the same point in 3D world.
- The rotation and translation are then estimated using SVD of E matrix where W (eigen value matrix is set to [0, -1, 0; 1 0 0; 0 0 1]
- R = np.matmul(np.matmul(U, np.linalg.inv(W)), VT)
- T = np.matmul(np.matmul(np.matmul(U, W), S), U.T)

Problem 2: Steps

- Once the fundamental matrix is calculated the epipolar lines are drawn using inbuilt function of OpenCv, cv.computeCorrespondEpilines and the observation is made the epipolar lines are not horizontal, Thus it is computationally expensive to equate each and every line to find corresponding points.
- Hence the images are rectified using cv.stereoRectifyUncalibrated which gives homography matrices allowing to warp the images which makes the epipolar lines horizontal.
- Important thing to be noted is that after warping the key feature indexes and fundamental matrix will be changed.
- The new key points are calculated by multiplying by homography matrix and for the fundamental matrix this equation is used

$$X2_{new} = H2 * X2\,, \qquad X1_{new} = H1 * X1, \qquad X2^T * F * X1 = 0$$

Hence, new F will be,

$$X2 = H2^{-1} * X2_{new}, \qquad X1 = H1^{-1} * X1_{new}\,, \qquad F_{new} = {H2^{-1}}^T * F * H1^{-1}$$

Problem 3: Steps

- Window matching concept is applied to calculate disparity between corresponding points which is basically calculating the offset between two matching points in both the images.
- Both images are converted to grayscale in order to match there pixel value.
- Obviously taking difference of a single pixel doesn't make sense as no. of pixels might have the same value but different location.
- Thus, a window is taken of certain size and the difference of both the sums are calculated.

- The index with least value is picked and normalised value is stored in a separate image as the disparity image.
- To avoid computational complexity instead of taking nested loops for window, numpy array is initialized to use inbuilt sum function , hence accelerating the process exponentially.
- Finally, the heatmap is plotted for the same.

Problem 4: Steps

- With given baseline and focal length information, Depth of each pixel is calculated with the equation of depth = B*f/disparity.
- Non normalized disparity image is used for this calculation.
- The catch with this method is that depth is multiplied by the scale of resizing for example, if resizing scale is 0.4, depth = 0.4 * (B*f/disparity).
- Finally “Jet” method of both depth and heat map is plotted as both color images and grayscale.

The output results are provided in the results folder submitted.

References :

https://cmsc733.github.io/2019/proj/p3/
https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj3/html/sdai30/index.html
https://www.youtube.com/watch?v=kxsvG4sSuvA
https://www.youtube.com/watch?v=O7B2vCsTpC0
https://pramod-atre.medium.com/disparity-map-computation-in-python-and-c-c8113c63d701
https://en.wikipedia.org/wiki/Essential_matrix
https://www.youtube.com/watch?v=QzYn0OPO0Yw
https://www.andreasjakl.com/how-to-apply-stereo-matching-to-generate-depth-maps-part-3/