ENPM673: PERCEPTION FOR AUTONOMOUS ROBOTS

PROJECT 3: STEREO VISION

BHARADWAJ CHUKKALA
118341705

PROJECT DESCRIPTION:

In this project, we are going to implement the concept of Stereo Vision. We will be given 3 different datasets, each of them contains 2 images of the same scenario but taken from two different camera angles. By comparing the information about a scene from 2 vantage points, we can obtain the 3D information by examining the relative positions of objects.

The dataset used for this project is MiddleBury Stereo Dataset.

ANSWER:

Calibration:

1. *Feature Matching:*
   - Read the images from the dataset
   - Convert them to gray and rgb for further use.
   - Firstly, we need to obtain matching features of both the images in the data set. We can do that using cv2.SIFTcreate() . SIFT helps locate the local features in an image, commonly known as the 'keypoints' of the image. These keypoints are scale & rotation invariant that can be helpful. We can use cv2.detectandcompute to get the required parameters.
   - The next step would be to match the detected feature points from both the images using Brute-force Matcher. We can use cv2.BFmatcher() to match the features.
   - Sorting the matched points so that we can extract the points with low distance between the descriptors.
   - Here we choose the lower distance between the descriptors because we get greater efficiency and thus we can obtain the correspondences therefore.

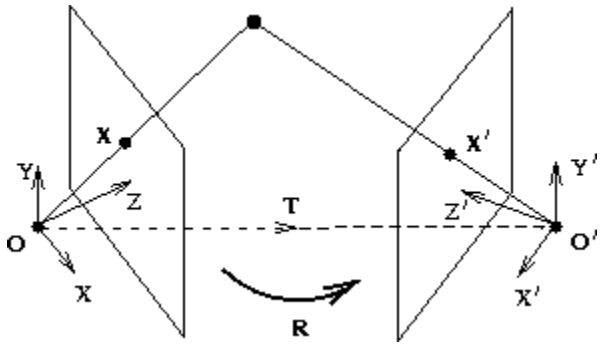

Fig 1: Epipolar Geometry

2. *Estimating the Fundamental Matrix:*
   - *What is the Fundamental Matrix?*

     The fundamental matrix will provide us with a relationship between any two images of the same scene taken from either the same camera placed at different points or different cameras at different places, the scene constrains where the projection of

points from the scene can occur in both images. A fundamental matrix is computed which gives the relationship between the feature correspondences of these stereo images.

- Now after getting the feature correspondences from feature matching, we can estimate the F matrix which describes the relationship between those correspondences, the fundamental matrix is only an algebraic representation of the epipolar geometry.
- To calculate the fundamental matrix, I used the 8-point Hartley algorithm. The eight-point Hartley algorithm has varied uses in the field of computer vision for estimating the fundamental and essential matrices used to compute stereo vision from matching set of corresponding points in a set of images.

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

$$\begin{bmatrix} u_1u_1' & u_1v_1' & u_1 & v_1u_1' & v_1v_1' & v_1 & u_1' & v_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_v' & u_nv_n' & u_n & v_nu_n' & v_nv_n' & v_n & u_n' & v_n' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ \vdots \\ f_{33} \end{bmatrix} = \mathbf{0}$$

- I normalized the values for X, X' which are the homogeneous point correspondences between the images where X = [u,v,1] and X'=[u',v',1]. To find the fundamental matrix we get the above form A.f = 0. Now using SVD, we compute the values of f. Here f is the 9 × 1 vector which is later reshaped to a 3 × 3 fundamental matrix F.
- For computing stereo vision for the images, The rank of the F matrix must be 2, However, it won't be 2 in many cases, and this is due to noise in the point correspondences of the images in the dataset, the estimated F matrix can be of rank 3.
- Hence, to ensure that the fundamental matrix has a rank 2 always constrained. We constrain it by having the last singular value of the estimated F as zero. The problem that arises when F has a full rank is it will have a null space, therefore epipoles wont exist for image planes.
- We need to get 8 best point correspondences from the sorted matches to compute the best fundamental matrix possible for which the terms Homogeneous equation should be close to 0 (minimized).
- To implement this, we can use RANSAC algorithm, this gives us the best point correspondences by removing the noise we get from all the other feature descriptors when we use SIFT to get matching features.

3. *Computing the Essential Matrix:*

- *What is an Essential Matrix?*

  The Essential Matrix is a 3 x 3 matrix that encodes epipolar geometry. Given a point in one image, multiplying by the essential matrix will tell us the epipolar line in the second view.

- We have been given the intrinsic parameters of the cameras in the calib.txt file (K1 and K2). The Essential matrix will estimate the relative camera poses between two views can be computed using the E matrix.

$$E = K_2^T \, F \, K_1$$

4. *Recovering the Camera Pose :*
   - We know that a camera pose consists of 6 degrees-of-freedom (DOF) Rotation (Roll, Pitch, Yaw) and Translation (X, Y, Z) of the camera with respect to the world coordinates. We can use the correspondences between points on the image plane and 3D points in the world to get the pose.
   - For recovering the Pose of the camera in the world, we use the essential matrix E, the essential matrix can be decomposed to 4 mathematically possible rotation and translation matrices.
   - To recover the camera 2 pose, will be estimating it with respect to camera 1 which can be assumed to be at the origin of the world coordinate system.
   - When we retrieve the Rotation and Translation from the essential matrix, we get four camera pose configurations based on the rotation and translation matrices so now to find the correct unique camera pose and given their triangulated points, we need to remove the ambiguity. This can be achieved by checking for the chirality condition. The cheirality condition say that point that lies in an image must lie in front of the camera producing that image. i.e. the reconstructed points must be in front of the cameras. So to check cheirality we triangulate the 3D points (given two camera poses) using linear least squares to check the sign of the depth Z in the camera coordinate system with respect to camera center.
   - The best camera configuration (Rotation, Translation, 3D points) is the one that produces the maximum number of points satisfying the cheirality condition.
   - Thus, by imposing cheirality condition we can distinguish between the points in front of the camera and thus find the right configuration.

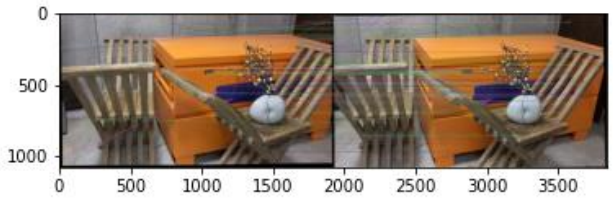

Fig2: Feature Matching (SIFT and BF matcher)

```
                                    Dataset (Curule)

The Fundamental Matrix:
 [[-3.35765676e-10  1.59350879e-07 -7.84593437e-05]
 [-1.37797616e-07  2.72997057e-08  2.01351881e-03]
 [ 7.29765436e-05 -2.05510016e-03  6.09454229e-03]]

The Essential Matrix:
 [[-6.18234591e-05  1.46637282e-01  2.743463444e-03]
 [-1.26907220e-01  1.19268621e-02  9.91832540e-01]
 [-3.456789992e-03 -9.89082339e-01  1.21166940e-02]]

Camera Pose: (Rotation Matrix)
 [[ 9.9682344e-01  7.963644443e-04 -2.01155608e-02]
 [-1.0567887e-03  9.96754253e-01 -1.21574498e-02]
 [ 2.01043757e-02  1.21759374e-02  9.99723742e-01]]

Camera Pose: (Translation Matrix)
 [ 0.99915237 -0.01453001  0.14882327]
```

```
                                    Dataset (Octagon)

The Fundamental Matrix:
 [[ 3.10618422e-10  3.87870889e-07 -3.36831558e-04]
 [-3.78926996e-07  6.24198152e-09  2.13102516e-03]
 [ 3.15056812e-04 -2.13072070e-03  1.86771156e-04]]

The Essential Matrix:
 [[ 5.3825774e-04  3.53194014e-01 -5.67640989e-02]
 [-3.38725452e-01  3.05072348e-03  9.39124091e-01]
 [ 5.56787086e-02 -9.39002758e-01  2.46312563e-03]]

Camera Pose: (Rotation Matrix)
 [[ 9.98888465e-01 -1.65439452e-03 -6.2356830e-04]
 [ 1.62173418e-03  9.99995138e-01 -2.66353086e-03]
 [ 6.63195855e-04  2.66245825e-03  9.99996236e-01]]

Camera Pose: (Translation Matrix)
 [0.97590404 0.06746045 0.34634865]
```

```
                                    Dataset (Pendulum)

The Fundamental Matrix:
 [[-2.04567392e-10  1.68006327e-07 -5.12238803e-05]
 [-1.61136784e-07  7.17997719e-09 -2.29804097e-03]
 [ 4.61857024e-05  2.29865298e-03 -2.13851369e-03]]

The Essential Matrix:
 [[-1.16789840e-04  1.298765441e-01  1.8892122e-02]
 [-1.23609785e-01  2.86912923e-03 -9.92155454e-01]
 [-1.85765482e-02  9.91517967e-01  2.79177281e-03]]

Camera Pose: (Rotation Matrix)
 [[ 9.99876308e-01  1.25743488e-04  5.03662399e-03]
 [-1.40397754e-04  9.99995758e-01  2.90929368e-03]
 [-5.03623680e-03 -2.90996389e-03  9.99983084e-01]]

Camera Pose: (Translation Matrix)
 [-0.97152449 -0.02843757  0.13860497]
```

Rectification:

- Essentially, Rectification of an image means, making it collimate with its axis. So here we have done a transformation which makes the pairs of conjugates epipolar lines formed in the images, to become collinear and parallel to the horizontal axis which is called the baseline of the image
- Here we perform image warping so that the input images align in such a way to each other (perspective transformation) that epipolar lines in both the images are horizontal and are passing through the same point in the image.
- Image rectification is done primarily so that the searching for corresponding points becomes a much easier task in the case of rectified images.
- Using the fundamental matrix and the feature points obtained from images through SIFT, we can construct the epipolar lines for both the images. The epipolar lines generally tend to meet at an optic center of the image and the point shifts when the camera s moved back and forth but for further computation, we will need to make the epipolar lines parallel to each other. We can do this by reprojecting image planes onto a common plane parallel to the line between camera centers.
- I have used the function cv2.stereorectifyUncalibrated() to perform the above task. This function provides a rectifying method that applies homographic transformation (we get the homography matrices from warping) to project the image plane of each camera onto a perfectly aligned virtual plane. This transformation is again computed from a set of matched point correspondences, matched images, and a fundamental matrix.
- After the rectification of images is done, the epipolar lines will be parallel. The next step after the images is warped is that we need to transform the feature points as well using the obtained homography matrices. These feature points constitute the epipolar points through which pass out epipolar lines. When the images get rectified the F matrix will get modified along with epipolar geometry.

DATASET 1: (CURULE)

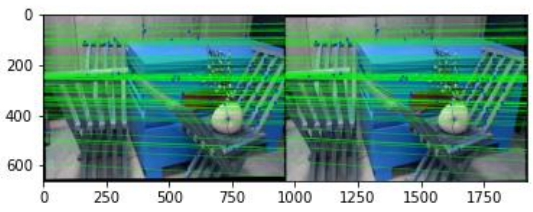

Fig3: Epipolar Lines

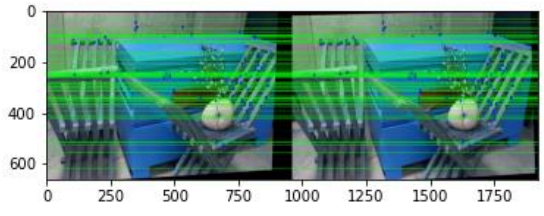

Fig4: Epipolar lines (After Rectification)

```
                                    Dataset (Curule)
 Homography Matrix image 1:
 [[-1.66266140e-03  8.82778100e-05  3.43145110e-02]
 [ 9.21925480e-05 -1.79348199e-03 -8.91689876e-02]
 [ 1.31257842e-07 -1.97003639e-08 -1.90349335e-03]]
Homography Matrix image 2:
  [[ 9.21185419e-01  9.47321036e-03  7.05464643e+01]
 [-5.45866342e-02  9.99491523e-01  5.26777467e+01]
 [-8.20434457e-05 -8.43711596e-07  1.07921731e+00]]
```

DATASET 2: (OCTAGON)

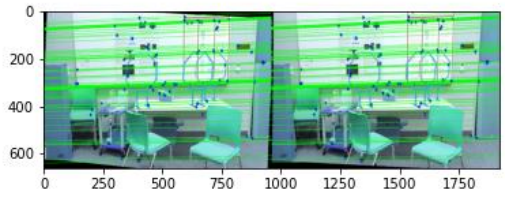

Fig5: Epipolar Lines

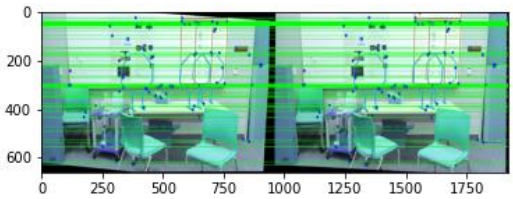

Fig6: Epipolar Lines (Rectified)

```
                                    Dataset (Octagon)
Homography Matrix image 1:
 [[ 2.07910135e-03  4.16145657e-05 -3.29577777e-01]
 [ 9.04757409e-05  1.84675985e-03 -9.72012217e-02]
 [ 2.47752000e-07  7.60107574e-09  1.58835022e-03]]
Homography Matrix image 2:
  [[ 1.12828044e+00  2.71663157e-02 -1.37819036e+02]
 [ 4.82500773e-02  1.00145157e+00 -4.71039228e+01]
 [ 1.33927274e-04  3.22465096e-06  8.69688505e-01]]
```

DATASET 3: (PENDULUM)

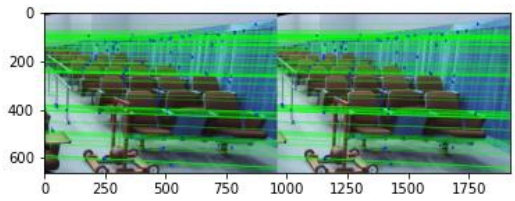

Fig 7: Epipolar Lines

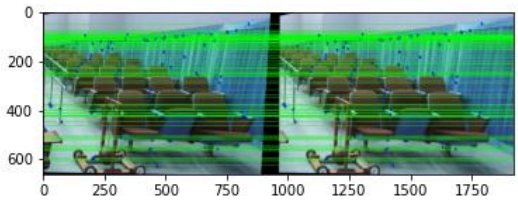

Fig8: Epipolar Lines (Rectified)

```
                                    Dataset (Pendulum)
 Homography Matrix image 1:
 [[-2.06122424e-03  9.21640711e-05  1.94888035e-01]
 [-4.45221032e-05 -1.97477144e-03  4.86438257e-02]
 [-1.47672402e-07  3.82204630e-09 -1.82497888e-03]]
Homography Matrix image 2:
  [[ 1.07270351e+00  1.89874373e-02 -8.00485877e+01]
 [ 2.32860498e-02  1.00056882e+00 -2.26617696e+01]
 [ 7.58959687e-05  1.34340004e-06  9.26414434e-01]]
```

Correspondence:

- In this part of the stereo vision problem, we find the corresponding matching region in each of the images. From the matching features that we have obtained in the above parts of the problem. We try to find the matching parts of the scene encapsulated in each of the images.
- For every point in image 1, we try to find a corresponding match along the epipolar line in image 2. Epipolar geometry gives us the way to not search the whole image but reduce the search area to a line. So, we will consider a window of a predefined size which we slide over every point in image 1 to find a closest corresponding match in image 2.This method is called block matching.
- I have used the method called SAD (Sum of absolute differences) to implement the block matching to find corresponding regions in the image.
- I have taken different window sizes to do block matching for each dataset. I initially tried implementing with the same window size to all the images in the data set, but the results were very not satisfactory. Therefore, I tuned the window size according to each dataset to obtain a little bit better heat maps for disparity.
- For block matching, I chose a window size and slid it across the epipolar line along every matching pixel coordinate index. I did this for both the images and after I computed the SAD between both the windowing regions. After doing this I could retrieved the matching correspondence on the epipolar line with the least SAD value. The matched values in both images are actually the same point in the world scene viewed from a different point.

- Through this I calculated disparity for all windowed regions along the corresponding epipolar lines from both the images.
- The SAD approach is time consuming, and it depends on the homogeneity of the images being used for block matching. If the window size is less, the blocks will have small variance values and are more in value in the current frame. The smaller size of the window and less variance ensure that this algorithm takes far fewer computations for best block searching.

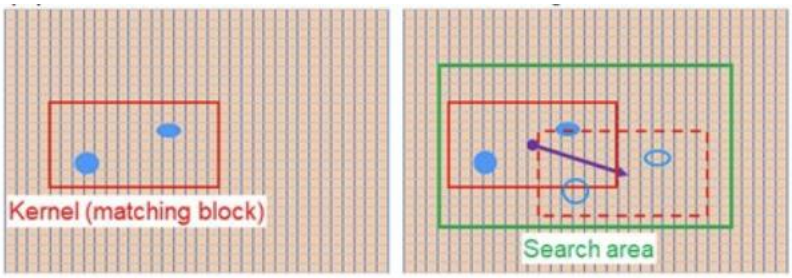

Fig9: Block Matching

- The SAD is given by the formula:

$$s = \sum_{(u,v)\in \mathbf{I}} |\mathbf{I}_1[u,v] - \mathbf{I}_2[u,v]|$$

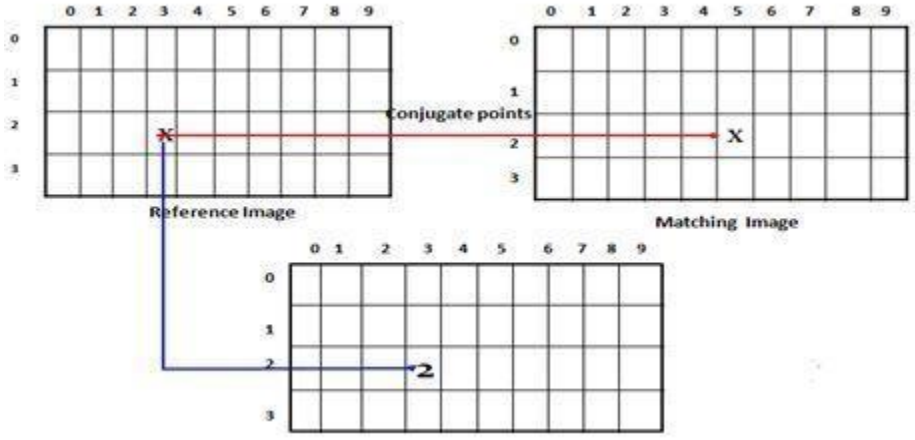

Fig: Calculate the disparity using SAD

Dataset 1	
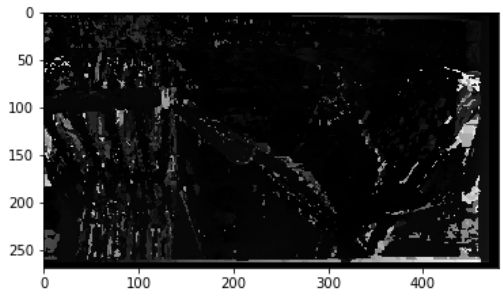	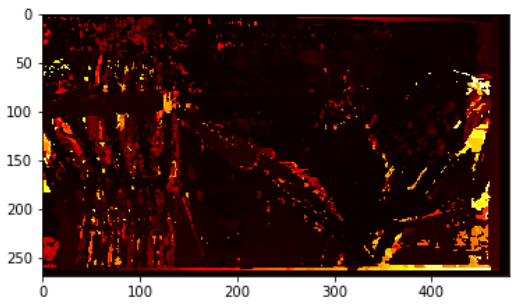
Fig10: Disparity image gray	Fig11: Disparity Image heat
Dataset 2	
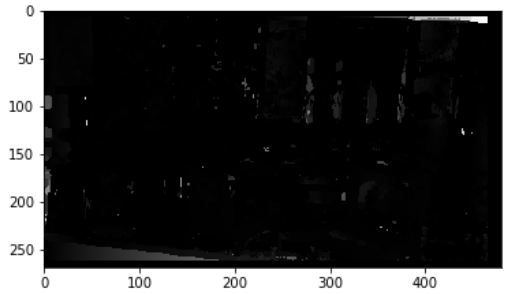	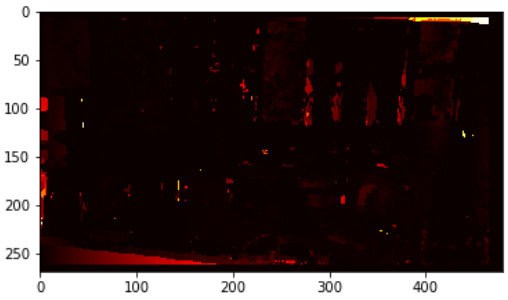
Fig12: Disparity image gray	Fig13: Disparity Image heat
Dataset 3	
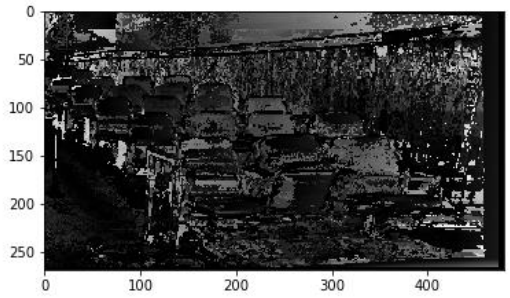	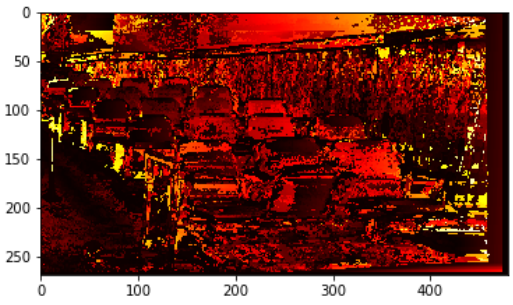
Fig14: Disparity image gray	Fig15: Disparity Image heat

Depth Calculation:

- In the calib.txt file that was given to us, we were given a set of camera and the image parameters for computing stereo vision
- After finding the disparity of the images, we can find the depth of the image using the disparity information and the information provided. We use the focal length and baseline values to calculate the depth.

$$depth = \frac{focal\ length * baseline}{disparity}$$

- For each image date set, when I tried calculating the depth, I was getting absurd results, therefore I used a thresholding parameter to tune the depth values above a certain value to get the depth images.
- I tried using the same thresholding value for all the data sets but that wasn't giving satisfactory results as well. Therefore I have tuned the thresholding parameter individually for each dataset and obtained a somewhat better result.

Dataset 1	
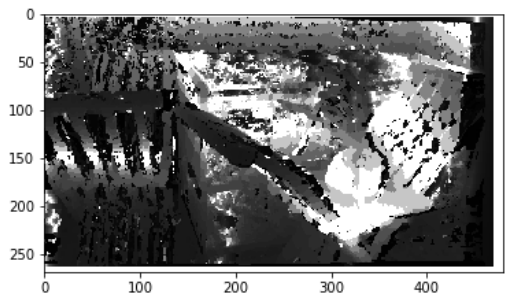	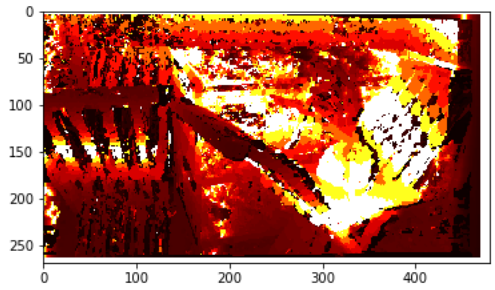
Fig16: Depth image gray	Fig17: Depth Image heat
Dataset 2	

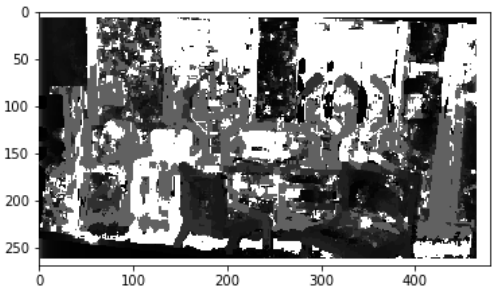

Fig18: Depth image gray

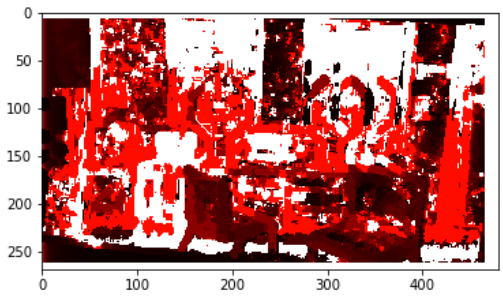

Fig19: Depth Image heat

Dataset 3

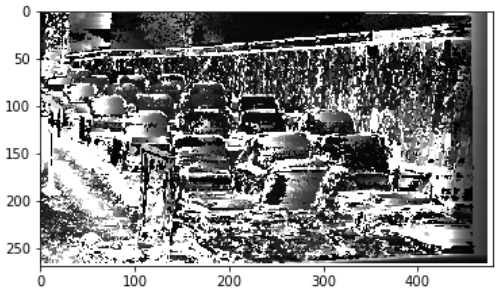

Fig20: Depth image gray

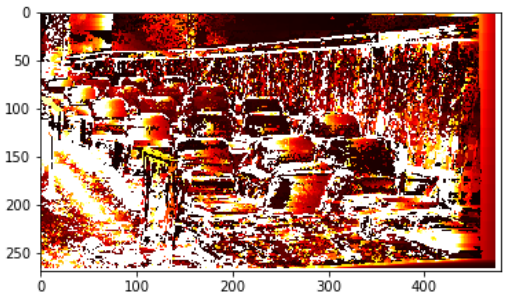

Fig21: Depth Image heat

References:

1. https://www.cs.cmu.edu/~16385/s17/Slides/12.2_Essential_Matrix.pdf

2. https://cmsc733.github.io/2022/proj/p3/#estfundmatrix

Drive link to outputs:

https://drive.google.com/drive/u/1/folders/1iicf6SVcWrGKDhqoB2W6d_XdmqmuklvB