ENPM673–Project 3 Spring 2022

Sumedh Koppula[1]
[1]University of Maryland, College Park, Maryland- USA
e-mail: sumedhrk@umd.edu, UID: 117386066

Problem 1: Calibration

Solution/ Pipeline used

1. First, I compared the two images in each dataset and selected a set of matching features.

2. I used a inbuilt function SIFT for feature matching and corner detection.
To calculate the descriptor, OpenCV provides two methods. Since we already found keypoints, we can call sift.compute() which computes the descriptors from the keypoints we have found.
cv2.xfeatures2d.SIFTcreate()
directly find keypoints and descriptors in a single step with the function,
kp1, des1 = siftimage.detectAndCompute(image0gray, None)

3. Estimated the Fundamental matrix using the features obtained in the previous step. I have used inbuilt SVD function to solve for the fundamental matrix.
8 Point Algorithm:

0. (Normalize points)

1. Construct the M x 9 matrix A

2. Find the SVD of ATA
When applying SVD to matrix A, the decomposition USV^T would be obtained with U and V orthonormal matrices and a diagonal matrix S that contains the singular values.Thus, the last column of V is the true solution.

3. Entries of F are the elements of column of V corresponding to the least singular value

4. (Enforce rank 2 constraint on F)

5. (Un-normalize F)

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

$$x_i x_i' f_{11} + x_i y_i' f_{21} + x_i f_{31} + y_i x_i' f_{12} + y_i y_i' f_{22} + y_i f_{32} + x_i' f_{13} + y_i' f_{23} + f_{33} = 0$$

Simplifying for m correspondences,

$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx_m' & x_my_m' & x_m & y_mx_m' & y_my_m' & y_m & x_m' & y_m' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

4. Estimated Essential matrix E from the Fundamental matrix F by accounting for the calibration parameters. I did implement the functions to estimate the Essential matrix and also to recover the rotation/transnational matrices.

$$E = K^T F K \tag{1}$$

5. Decomposing E into a translation T and rotation R
The 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. Since the E matrix is identified, the four camera pose configurations: (C1,R1),(C2,R2),(C3,R3) and (C4,R4). Thus, the camera pose can be written as:

$$P = KR[I_{3*3} - C] \tag{2}$$

$$E = UDV^T \tag{3}$$

$$R = UWV^T \tag{4}$$

```
E = cam1.T.dot(fundamentalmatrix).dot(cam0)
U,s,V = np.linalg.svd(E)
s = [1,1,0]
essentialmatrix = np.dot(U,np.dot(np.diag(s),V))
print("Essential Matrix: ", essentialmatrix)
```

For Dataset :1

Fundamental Matrix :

$$\begin{bmatrix} -4.56275283e-10 & 1.90802003e-07 & -1.242 \\ -1.77965602e-07 & 2.32876558e-08 & 1.8400 \\ 1.20963645e-04 & -1.86764325e-03 & 1.657 \end{bmatrix}$$

Essential Matrix:

$$\begin{bmatrix} -2.53306689e-05 & 1.97255562e-01 & 1.378 \\ -1.83033049e-01 & 1.14101894e-02 & 9.829 \\ 1.08129203e-02 & -9.80200670e-01 & 1.10767 \end{bmatrix}$$

Estimated Rotation:

$$\begin{bmatrix} 9.99892633e-01 & -8.93997715e-04 & -1.4626 \\ 7.26227916e-04 & 9.99933933e-01 & -1.1471 \\ 1.46359203e-02 & 1.145990670e-01 & 0.1974 \end{bmatrix}$$

Estimated translation: [0.98025513 0.01152605 0.19740069]

Similarly, I computed and followed the above pipeline for remaining datasets 2 and 3

Problem 2: Rectification

Solution/ Pipeline used

1. I Applied perspective transformation to make sure that the epipolar lines are horizontal for both the images.

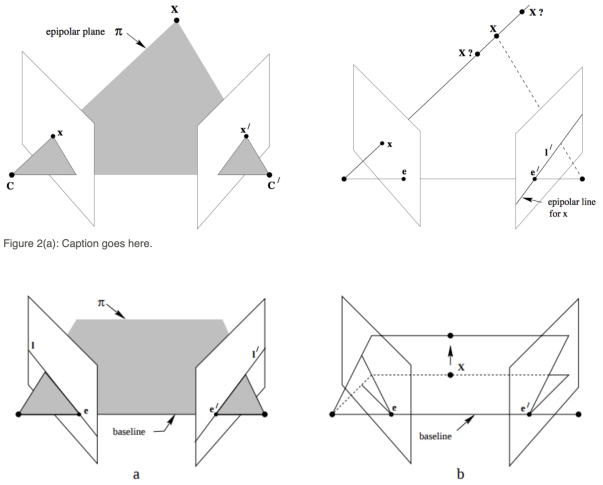

Figure 2(a): Caption goes here.

2. Warping the images and Transforming Images

3. I did compute the H1 and H2 the homography matrices for both left and right images that will rectify the images.
m, H1, H2 = cv2.stereoRectifyUncalibrated(np.float32(pair1), np.float32(pair2), fundamentalmatrix, imgSize=(w1, h1))

4. Plotted the epipolar lines on both images along with features points
lines2 = cv2.computeCorrespondEpilines(pair1rect.reshape(-1,1,2), 1,Frectified)
as shown in Figure 1 and 2 for Dataset 1, Figure 3 and 4 for Dataset 2 and Figure 5 and 6 for Dataset 3.

Problem 3: Correspondence

Solution/ Pipeline used

1. For obtained rectified epipolar lines, I had applied the matching windows concept using SSD
Sum of Squared-Differences (SSD) The SSD measure is sum of squared difference of pixel values in two patches. This matching cost is measured at a proposed disparity. If A,B are patches to compare, separated by disparity d, then SSD is defined as:

$$SSD(A,B) = \sum_{ij}(A_ij - B_ij)^2 \quad (5)$$

2. Then I had Calculated Disparity for the images

3. Later, I did Rescale the disparity to be from 0-255 and saved the resulting image.

4. Moreover, I saved the disparity as a gray scale and color image using heat map conversion. This can be seen in the figures 7, 8 and 9 for respective data sets.

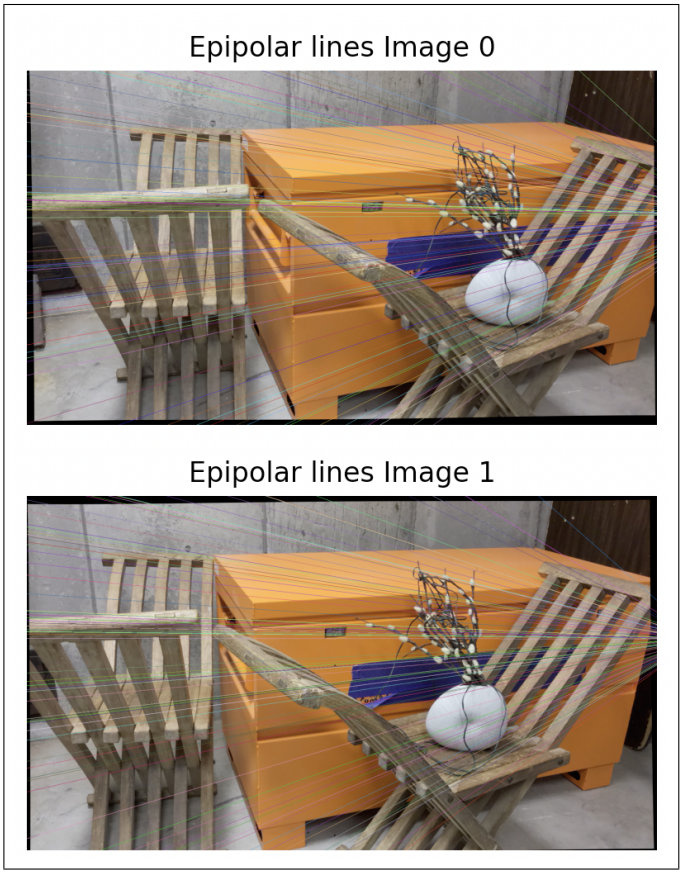

Fig. 1. Epipolar Lines for crule dataset

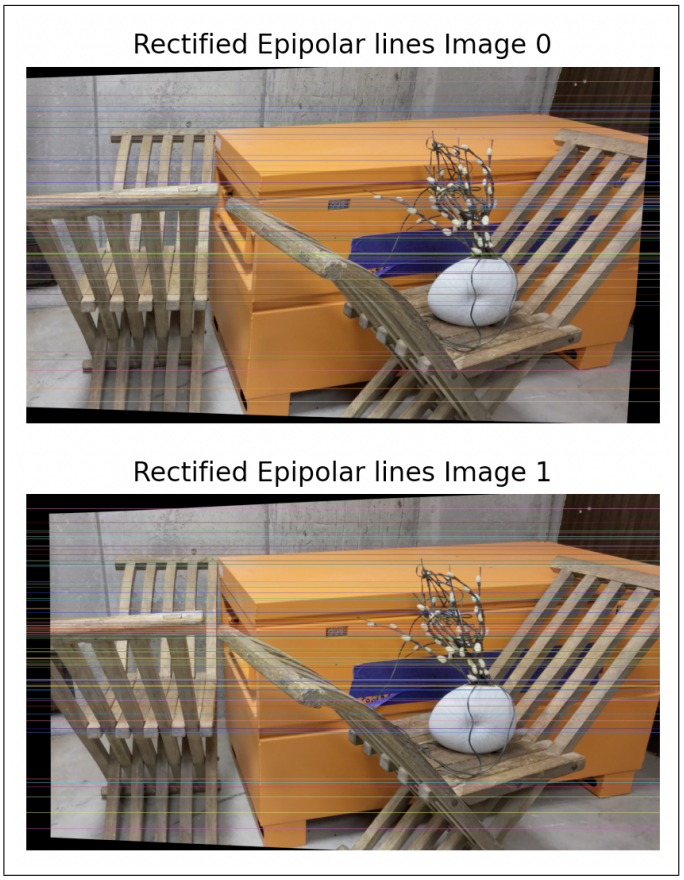

Fig. 2. Rectified Epipolar Lines for crule dataset

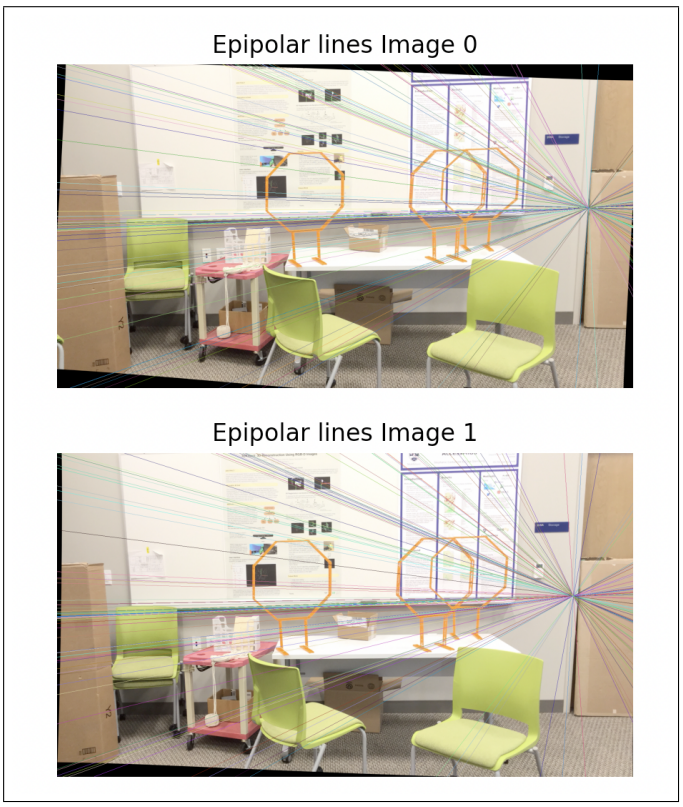

Fig. 3. Epipolar Lines for octagon dataset

Fig. 4. Rectified Epipolar Lines for octagon dataset

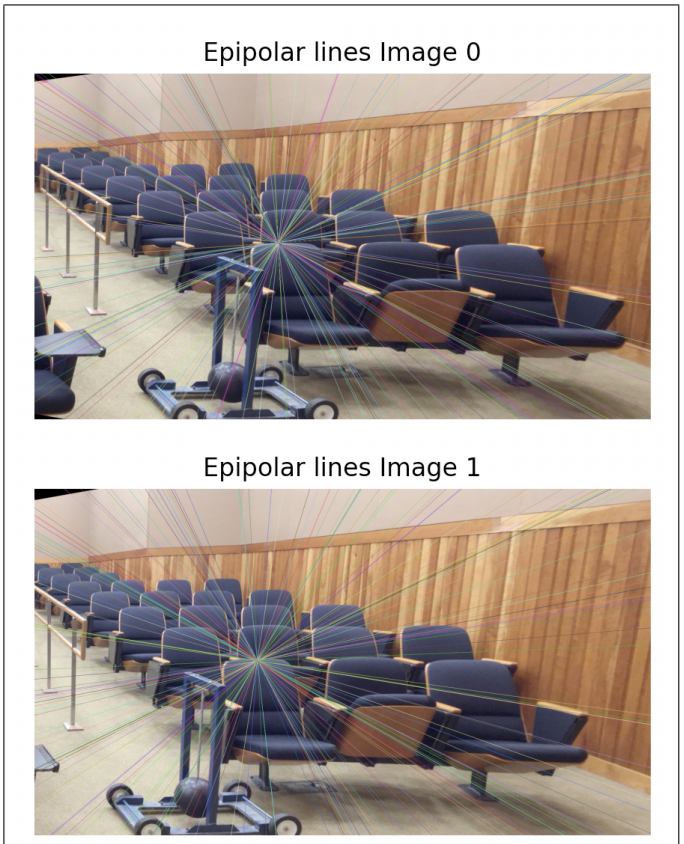

Fig. 5. Epipolar Lines for pendulum dataset

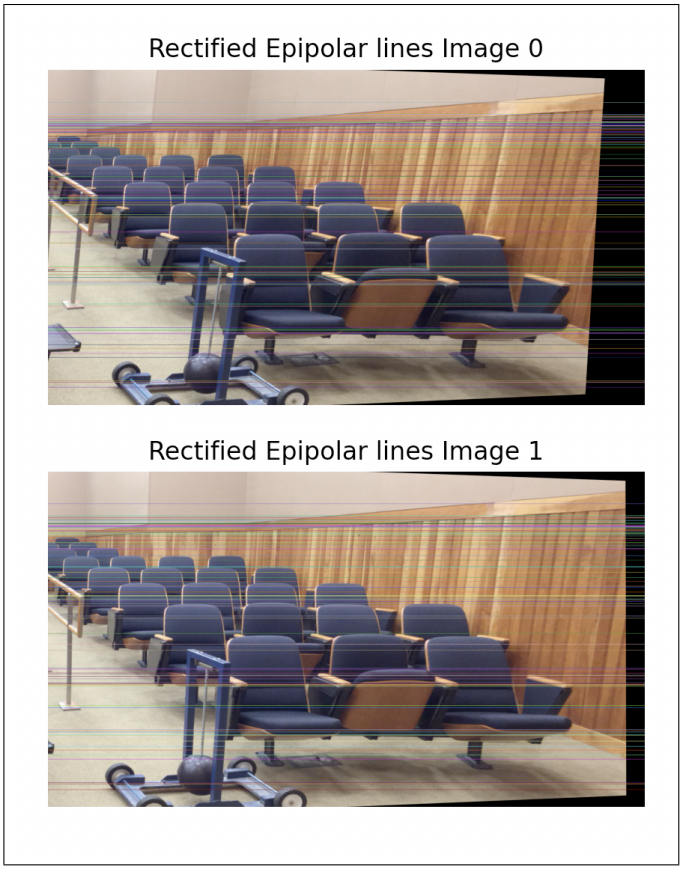

Fig. 6. Rectified Epipolar Lines for pendulum dataset

Note:

We follow the convention of search input in left image and search target in right image While searching for disparity value for a patch, it may happen that there are multiple disparity values with the minimum value of the similarity measure. In that case we need to pick the smallest disparity value.

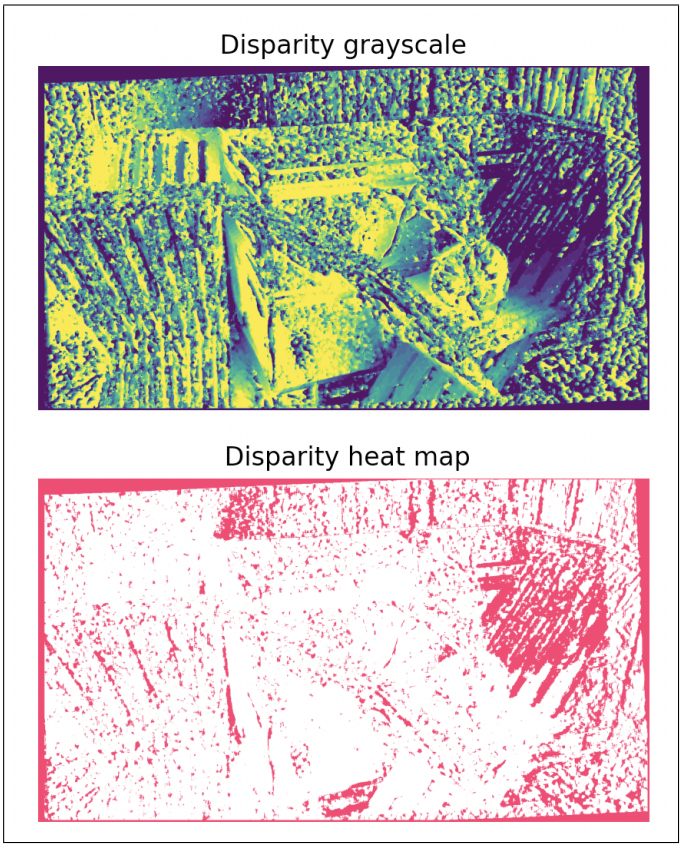

Fig. 7. Disparity vs heat map for crule dataset

Problem 4: Compute Depth Image

Solution

1. Using the disparity information, I had computed the depth information for each image pixel. The resulting depth image has the same dimensions of the disparity image but it has depth information instead.

2.Finally, I saved the depth image as a gray scale and color using heat map conversion which is showed in the figures 10, 11 and 12 respectively for respective data sets.

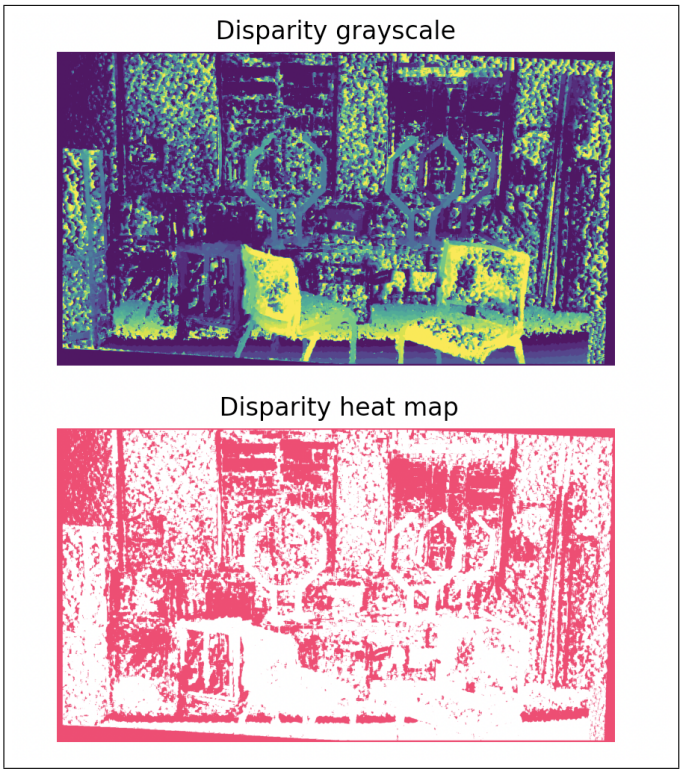

Fig. 8. Disparity vs heat map for octagon dataset

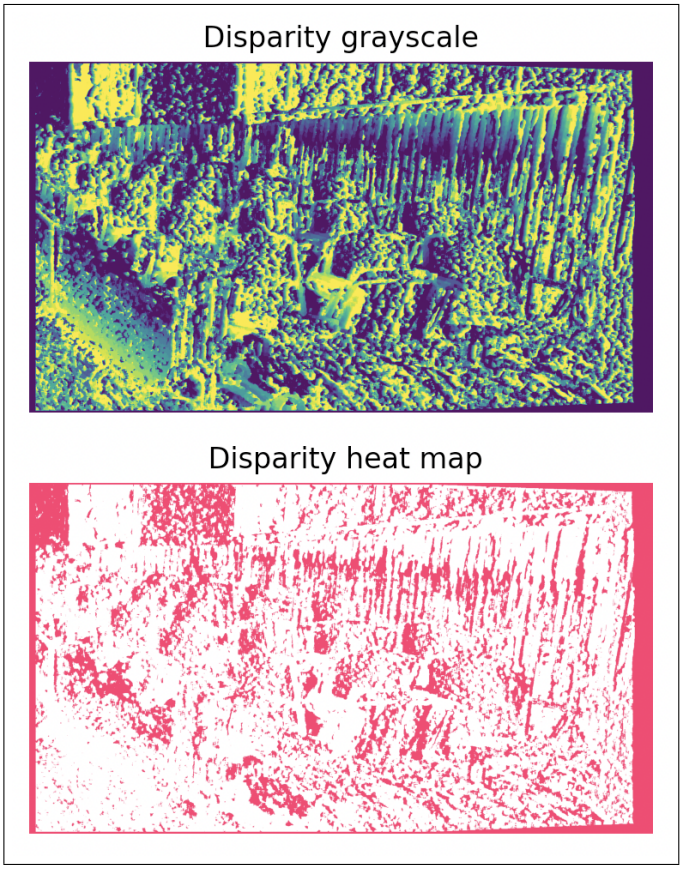

Fig. 9. Disparity vs heatmap for pendulum dataset

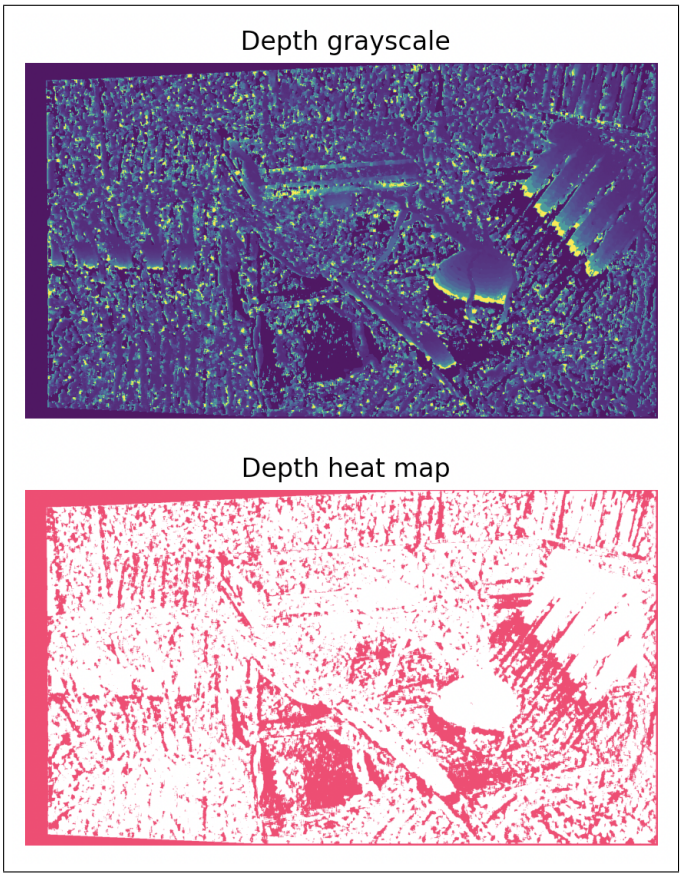

Fig. 10. Depth vs Heat for crule dataset

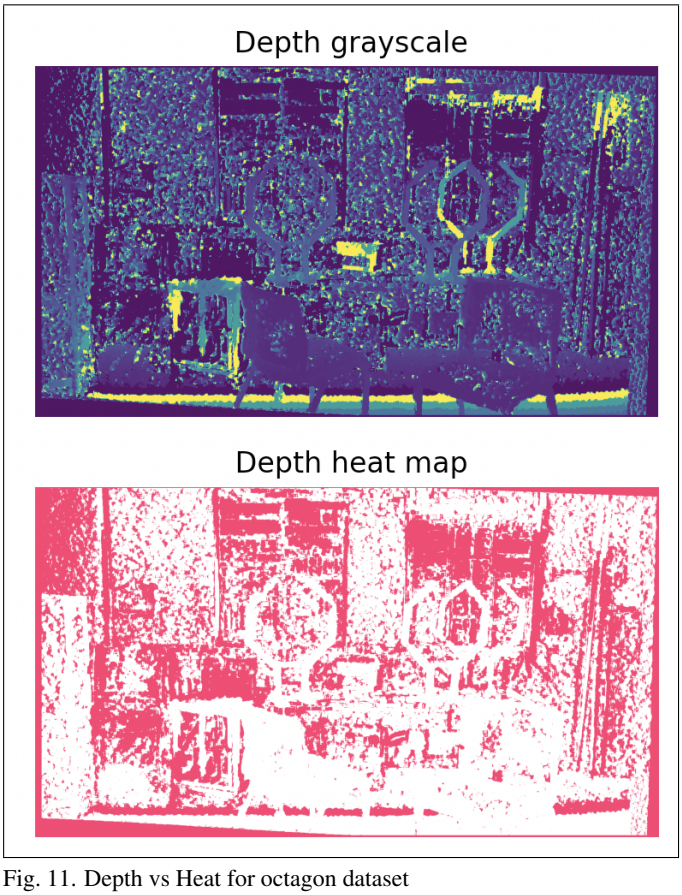

Fig. 11. Depth vs Heat for octagon dataset

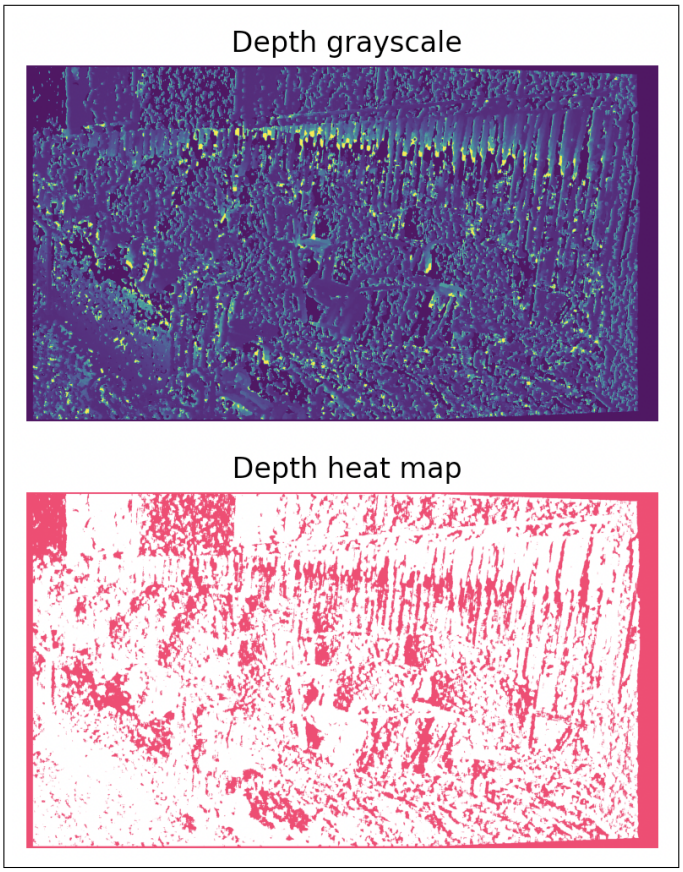

Fig. 12. Depth vs Heat for pendulum dataset