Perception for Autonomous Robots
ENPM673
Project 3

Name: Balaji Selvakumar
UID : 118545745
Date: 04/19/2022
email : balase22@umd.edu

STEP 1 : CALIBRATION

1.1 Feature extraction and Matching:

1. Here we create an SIFT feature detector, to generate max keypoints(10000) find keypoints and descriptors in the given image.

2. Then, we use a Brute Force/ FLANN based matcher to find the best matches in the keypoints between 2 input images based on hamming distance.

3. Finally, the matches are sorted based on distance and 50 best matches are selected from the total matches.

4. To fill the final list of feature points, points are extracted from the matches and the matches are plotted simultaneously on both images to visualize them.

1.2 Finding Fundamental Matrix :

Here we calculate the Fundamental matrix using the least squares method. We have 8 equations and we input exactly 8 points to form the A matrix from them.

2. We the take the Singular Value Decomposition of the F matrix to solve the homogeneous equation:

$$Ax = 0$$

3. After getting the A matrix,we find the lowest singular value using the last column of the V_t matrix.
4. Due to noise in the correspondences, the estimated F matrix can be of rank 3 so we downgrade the rank of the F matrix to 2, by making the last column of the V_t matrix of SVD decomposition of F

1.3 Finding Inliers using RANSAC :

Here, the RANSAC algorithm is applied to choose the best F matrix. 8 random points are sampled and fed into the calculate_F_matrix function iteratively and we calculate the error in estimating the points.

Comparing this with a predefined threshold, we can classify the inlier points and choose the Best F matrix based on the max_inliers criteria.

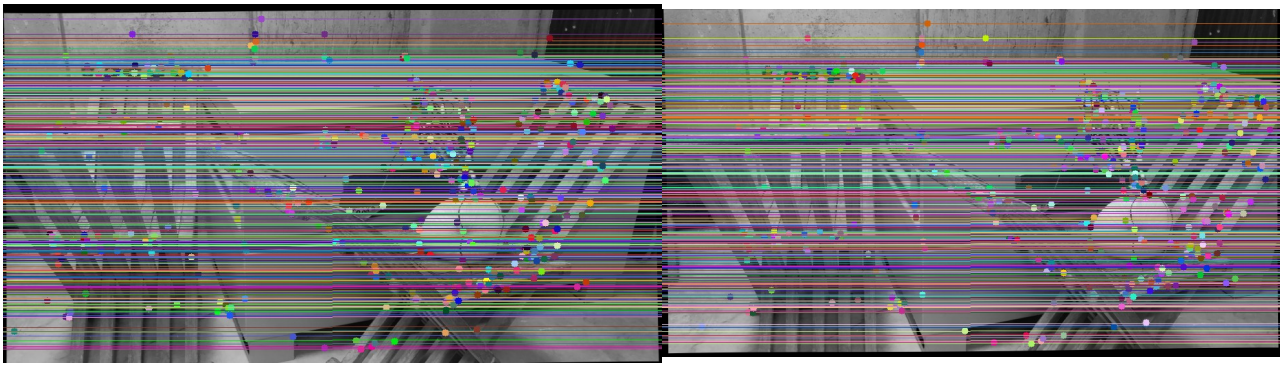


Fig 1 :Curule Epilines

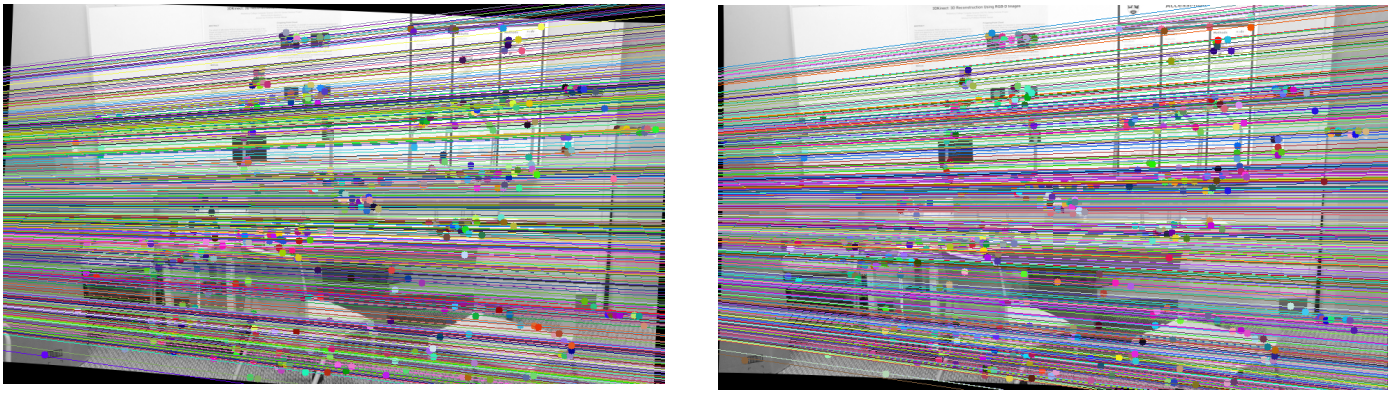


Fig 2: Octagon Epilines

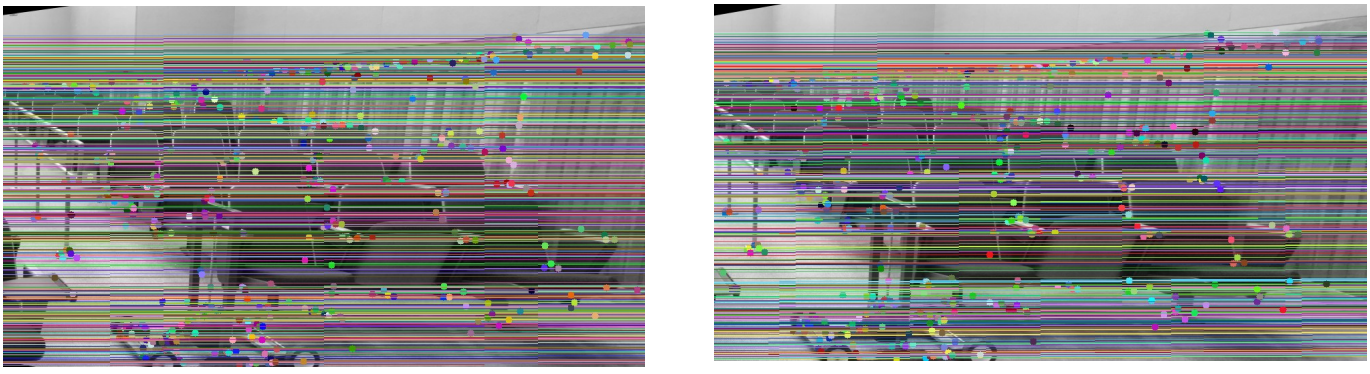


Fig 3: pendulum Epilines

1.4: Calculate_E_matrix:

Here, the Essential matrix is calculated using the Fundamental matrix and the intrinsic camera parameters.

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

1.5 Estimation of Camera Pose:

The essential matrix E can be used to estimate camera pose(R and C) as explained on page 258 of [6]. We'll calculate camera 2's posture in relation to camera 1, which is considered to be at world origin. We'll obtain four answers, two of which will be twisted pairs and one of which will be flipped 180 degrees. Please see fig.?? for more information. The Z value of the 3D points must be positive for a proper pair of camera poses. I calculated the 3D points for each set of R and C and chose the R and C set with the highest positive Z values for both cameras. Refer to fig. 4 for an example of a 3D point plot.

STEP_2: RECTIFICATION :

Using the fundamental matrix and the feature points, we can obtain the epipolar lines for both the images. Refer fig. 5. The epipolar lines need to be parallel for further computations to obtain depth. This can be done by reprojecting image planes onto a common plane parallel to the line between camera centers[1]. Using the matched image pairs and the estimated F matrix, the images can be rectified using the OpenCV function cv2.stereoRectifyUncalibrated. We will obtain two homography matrices, one for each image

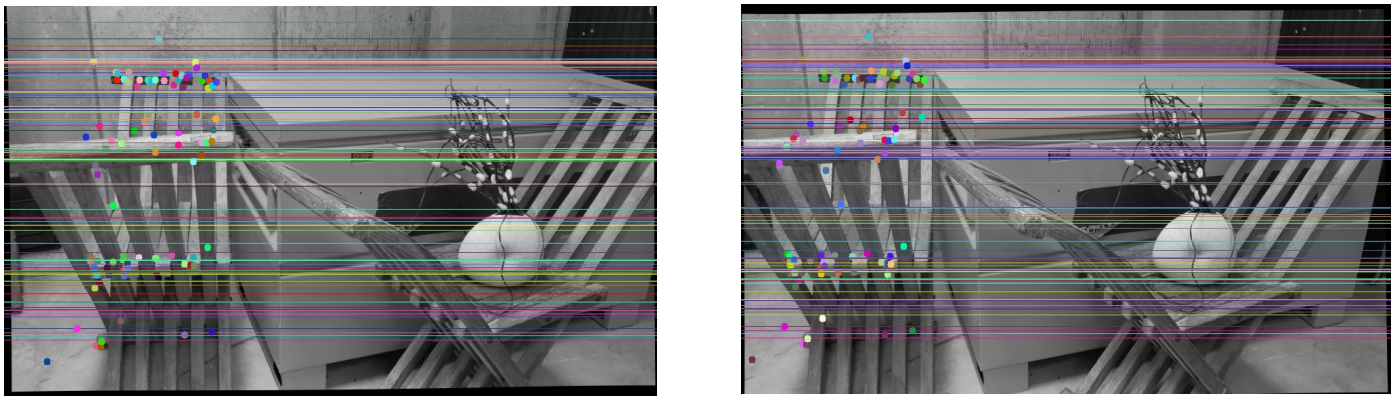


Fig 4: Rectified dataset 1

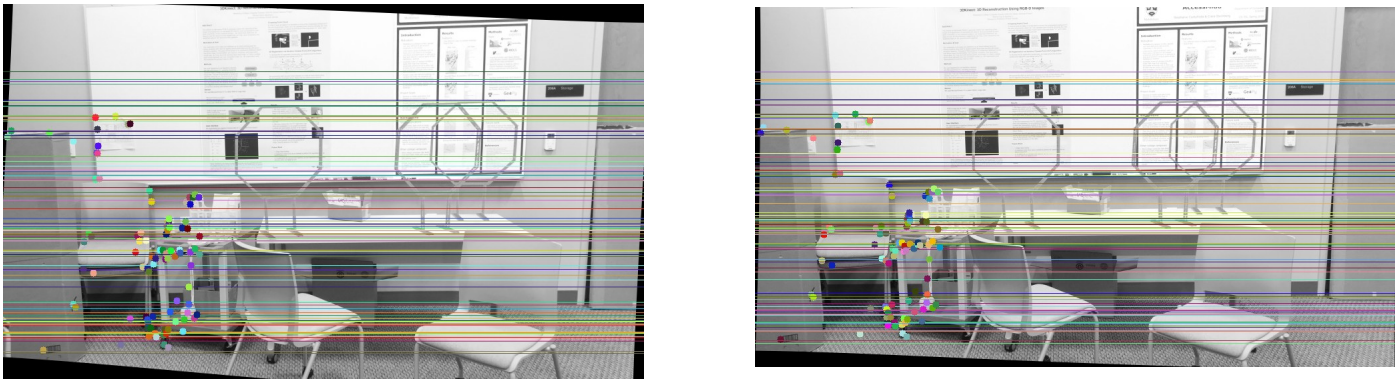


Fig 5: Rectified dataset 2

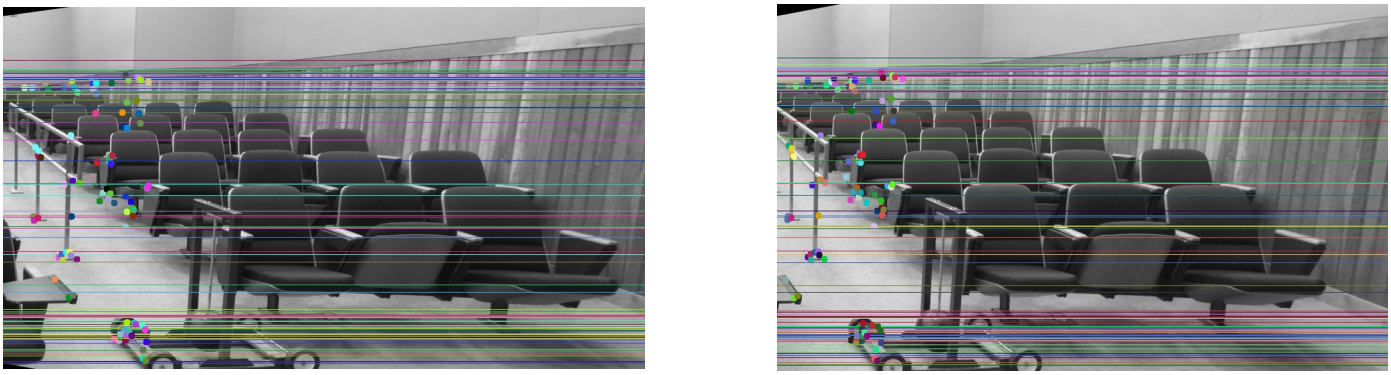


Fig6 : Rectified dataset 3

After rectification, the epipolar lines will be parallel. Refer fig. 6. After the images are warped, we need to transform the feature points as well using the obtained homography matrices. Also, the F matrix will get modified as the epipolar geometry has changed.

3. CORRESPONDENCE: (DISPARITY)

For every pixel in image 1, we try to find a corresponding match along the epipolar line in image 2. We will consider a window of a predefined size for this purpose, so this method is called block matching. Essentially, we will be taking a small region of pixels in the left image, and searching for the closest matching region of pixels in the right. image[2]. There are three block comparison methods: 1) Sum of Absolute Differences (SAD) 2) Sum of Squared Differences (SSD) 3) Normalized Cross-Correlation (NCC) Normalized Cross-Correlation (NCC) gives the best results but takes a lot of computation time. Sum of Absolute Differences (SAD) takes the least computation time, but the results are not that great. Sum of Squared Differences (SSD) takes a moderate computation time and the results are decent. I have used Sum of Absolute Differences (SSD). A simple approach to implement the block matching algorithm would be to use nested for loops. However, the approach is not efficient.

It is noticed that there were black line regions where the epipolar lines are not available, since the sift features were not detected anywhere these regions. Thus, this method of scanning horizontally along only the available epipolar lines is bound to introduce irregular heatmaps. To account for this irregularity, I performed the block matching along every pixel along the left and right rectified images and constructed my disparity map. This results are the final disparity maps and are shown below

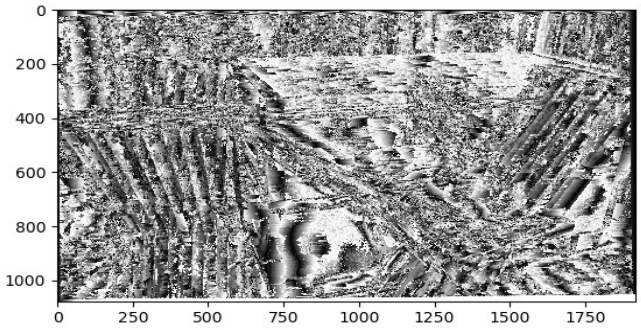


Fig:7 Disparity dataset 1

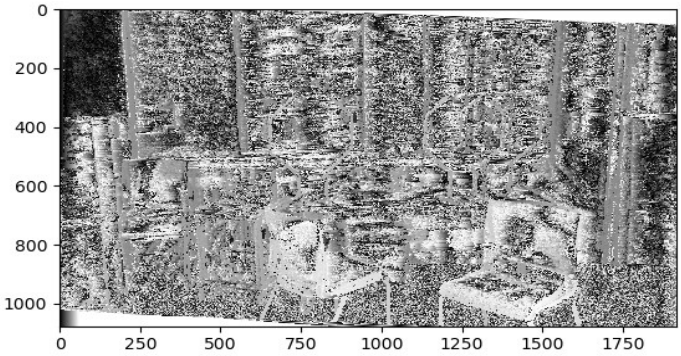


Fig 8: Disparity dataset 2

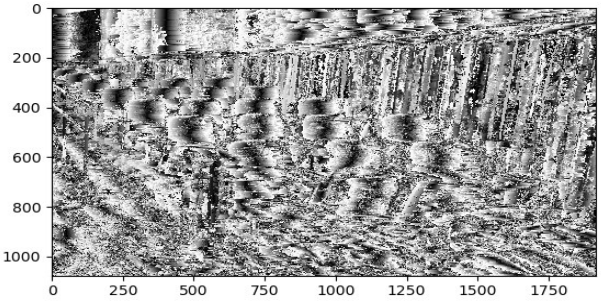


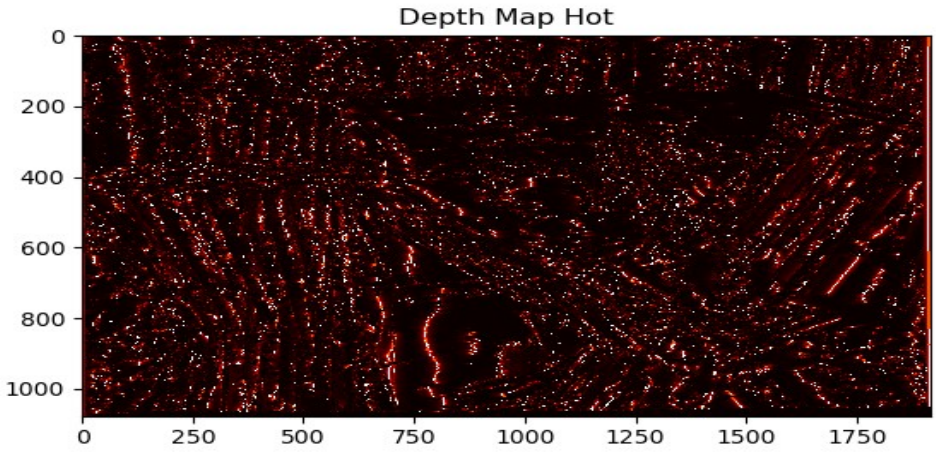
Fig 9: Disparity dataset 3

4. COMPUTE DEPTH IMAGE

Objective: Using the disparity map from previous step compute the depth information for each pixel using the below formula:

$$depth = (baseline * f) / \text{disparity}$$

The parameters baseline and f are already given. The computed depth maps are shown below



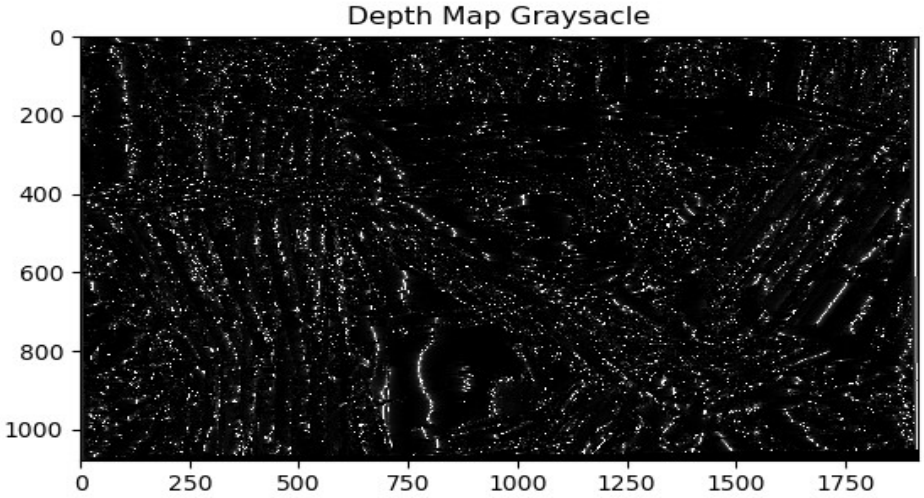
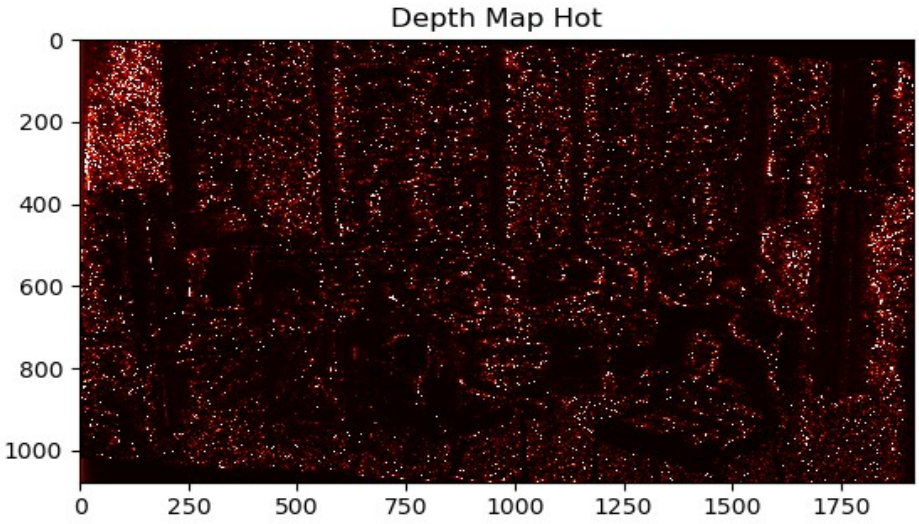


Fig 10:Depth map for dataset 1



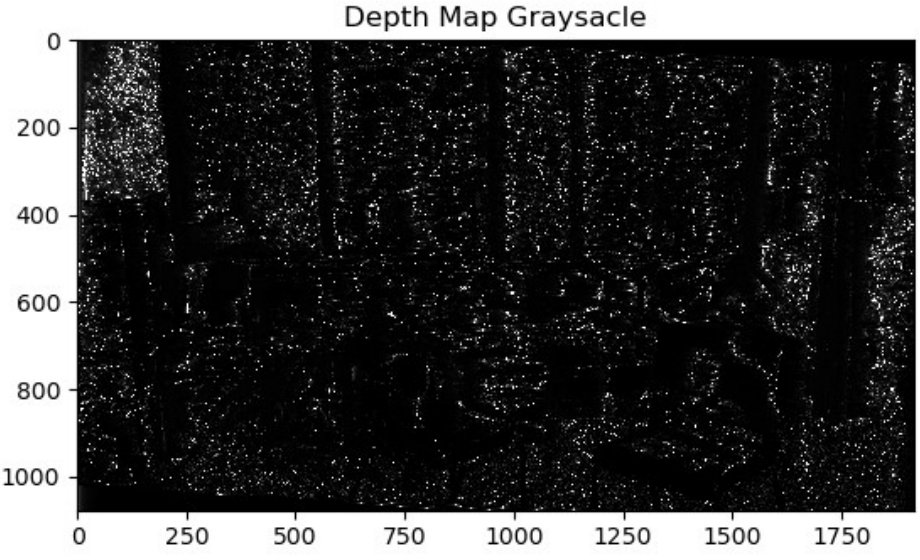
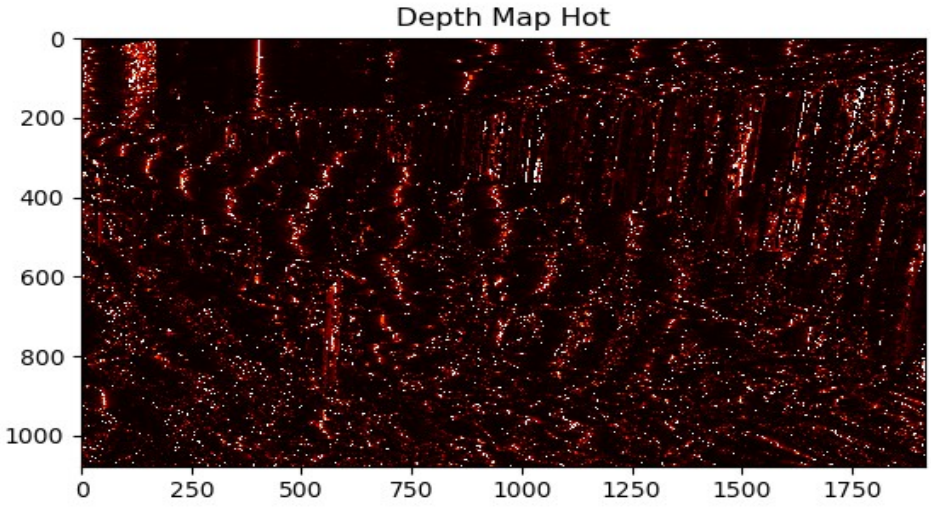


Fig 11: Depth map for dataset 2



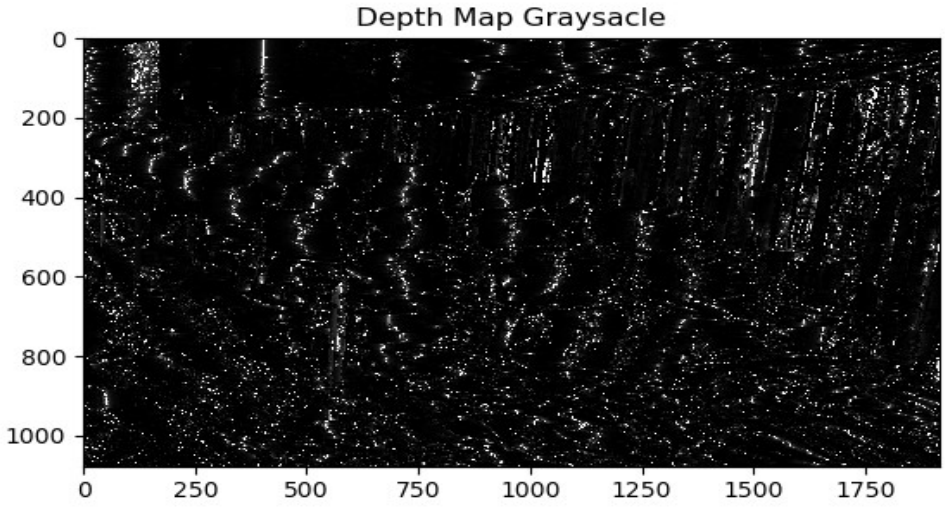


Fig 12: Depth map for dataset 3

The output files can be accessed from this link : https://drive.google.com/drive/folders/1WJGFSUIlekWHneEhVCX7V5PK7-hwE1EU?usp=sharing

References

1. https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html
2. https://cmsc733.github.io/2019/proj/p3/#estE
3. https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj3/html/sdai30/index.html
4. https://stackoverflow.com/questions/30716610/how-to-get-pixel-coordinates-from-feature-matching-in-opencv-python
5. https://stackoverflow.com/questions/27856965/stereo-disparity-map-generation
6. https://stackoverflow.com/questions/36172913/opencv-depth-map-from-uncalibrated-stereo-system
7. https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj3/html/sdai30/index.html
8. https://pramod-atre.medium.com/disparity-map-computation-in-python-and-c-c8113c63d701
9. https://stackoverflow.com/questions/46689428/convert-np-array-of-type-float64-to-type-uint8-scaling-values/46689933
10. https://stackoverflow.com/questions/59478962/how-to-convert-a-grayscale-image-to-heatmap-image-with-python-opencv
11. https://www.youtube.com/watch?v=KOSS24P3_fY