ENPM673 Project 3

Prannoy Namala (pnamala)

April 2021

Introduction

This project requires us to compute the 3D details by examining the relative position between objects in two different images. For getting the information, the following steps are used.

- Computing the translation and rotation between the two images
- Rectifying the images
- Calculating disparity
- Developing the heatmap

Calculating the relative difference in position between both the cameras

Feature Matching

The first step in the process of computing 3D information is to get matching points between two images. There are multiple ways to get matching points on both images. For this project, we use ORB (Oriented FAST and Rotated BRIEF). The list of matching points from both the images are the output from this step.

Finding Fundamental Matrix

Using the matching points, we compute the Fundamental Matrix(F). This is computed by using the first 8 matching points. In this project, I also tried implementing RANSAC, but it didn't give me good enough question. The matrix F is obtained by computing the SVD of matrix A which has the rows in the following format. $[point_1[0] * point_2[0], point_1[1] * point_2[0], point_2[0], point_1[0] * point_2[1], point_1[1] * point_2[1], point_2[1], point_1[0], point_1[1], 1]$ By building a matrix with 8 rows of the same structure with 8 matches. By solving for the least solution of the matrix with 8 rows, we can get F matrix. The least solution is obtained by the last row of the V matrix in U,L,V decomposition of SVD.

Finding Essential Matrix

Using the above obtained F matrix Essential Matrix(E) can be computed. This is easily computed by using the extrinsic parameter matrix given in the ground truth document.

Getting rotation and transformation matrices

After obtaining E, we need to compute the transformation between both the images. There are two steps to the process

Linear Triangulation

After obtaining E, we need to find an estimated depth for all the points matched. The first step in achieving this is to find all the possible transformations possible. This is achieved by getting SVD of A. The minimum solution i.e last row of V will be the center and the using a matrix W where W is $\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, we can find two rotation matrices R by multiplying W and transpose of W with V. We get four pairs of R and T i.e [(r1, t), (r1, -t), (r2, t), (r2, -t)]. For each of these pairs, the 3D position of matches is obtained by linear triangulation. Since we have the information of camera calibration matrix, we can estimate the depth using the triangulation concept by using the camera calibration matrices. The output from this step is 4 sets of points which are used to validate the rotation and transformation.

Checking Disambiguity

The camera translation T and rotation R is valid only if for the points set obtained via linear triangulation X satisfy the condition $r3(XT) > 0$ for most points in T. Here, r3 is the Z- axis of rotation, which is 3rd column of R. We take the R,T and X which satisfy the most points in the above condition.

Rectifying the images

For computing depth, we need the images to be on the same plane. The first step is to compute epilines between the images. This is done by using the inbuilt function, computeCorrespondEpiLines(). Using this information, lines are drawn on both the images.The final step is to use inbuilt opencv function, stereoRectifyUncalibrated(). By using this function, we obtain two homography matrices for two images. These homography matrices are used to create new images where both of them are in parallel planes. The results are shown in figures 13, 4 and 5 for data set 1; 7, 9 and 10 for data set 2; 14, 14 and 15 for data set 3.

Figure 1: Unrectified image of im0 for data set 1 with epilines

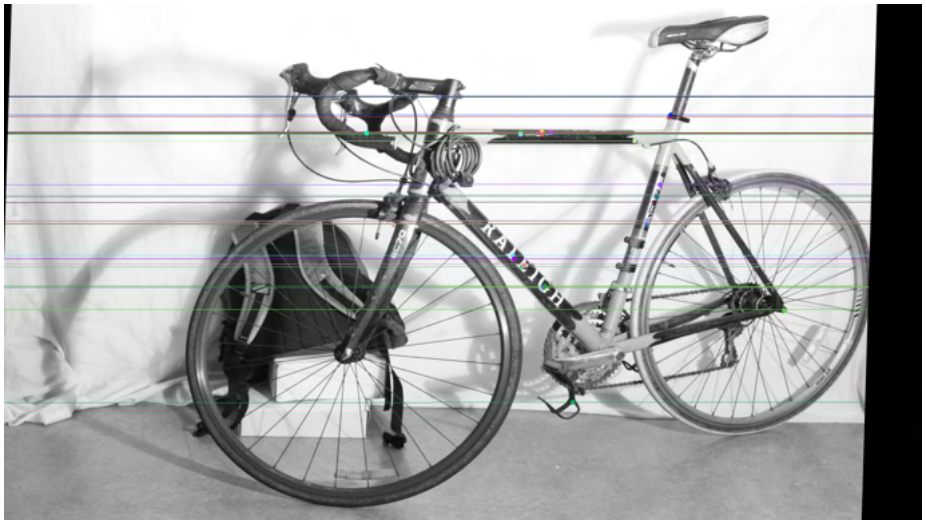

Figure 2: Stereo rectified image of im0 for data set 1

Figure 3: Unrectified image of im1 for data set 1 with epilines

Figure 4: Stereo rectified image of im1 for data set 1

Figure 5: Comparison of rectified images for data set 1

Figure 6: Unrectified image of im0 for data set 2 with epilines

Figure 7: Stereo rectified image of im0 for data set 2

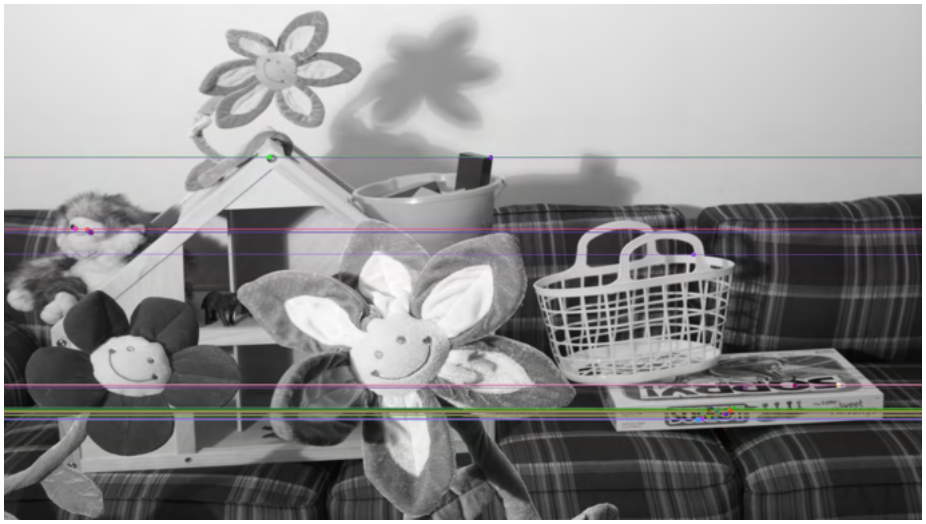

Figure 8: Unrectified image of im1 for data set 2 with epilines

Figure 9: Stereo rectified image of im1 for data set 2

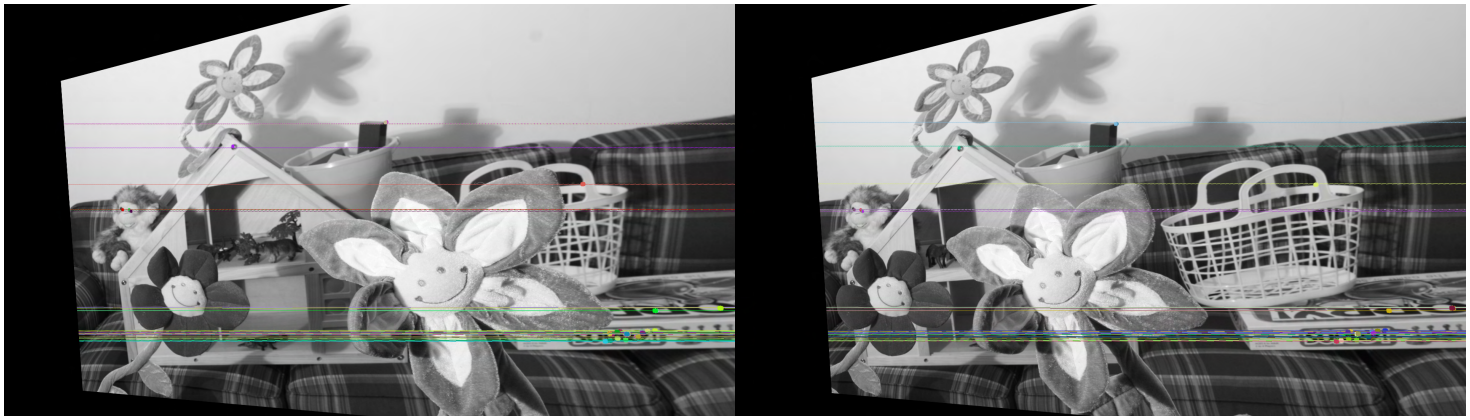

Figure 10: Comparison of rectified images for data set 2

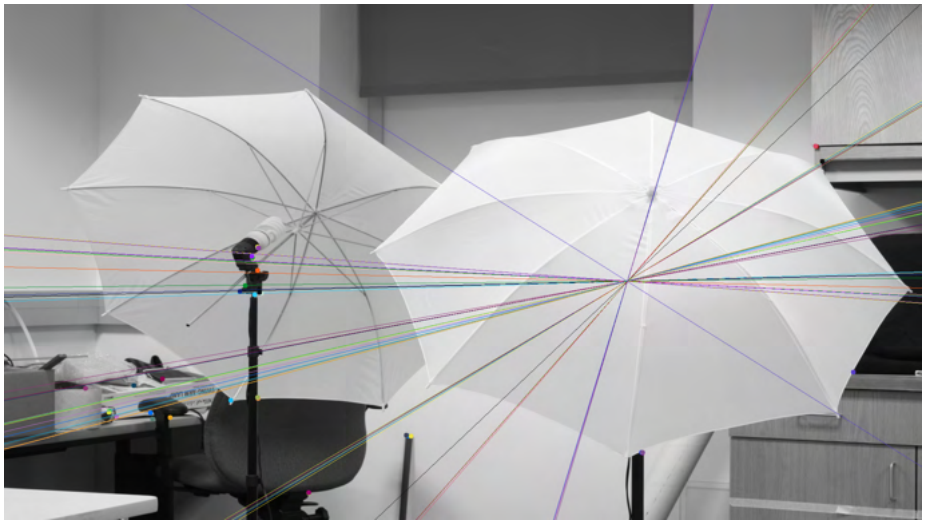

Figure 11: Unrectified image of im0 for data set 3 with epilines

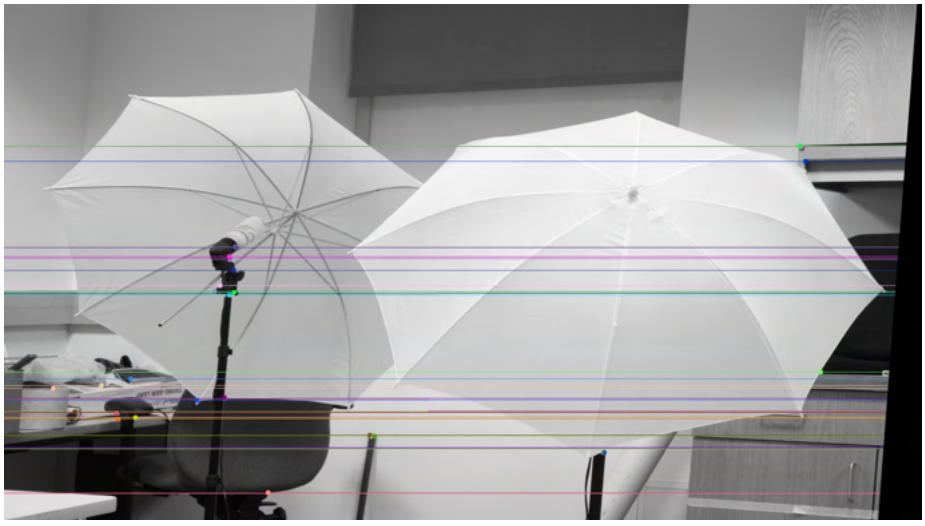

Figure 12: Stereo rectified image of im0 for data set 3

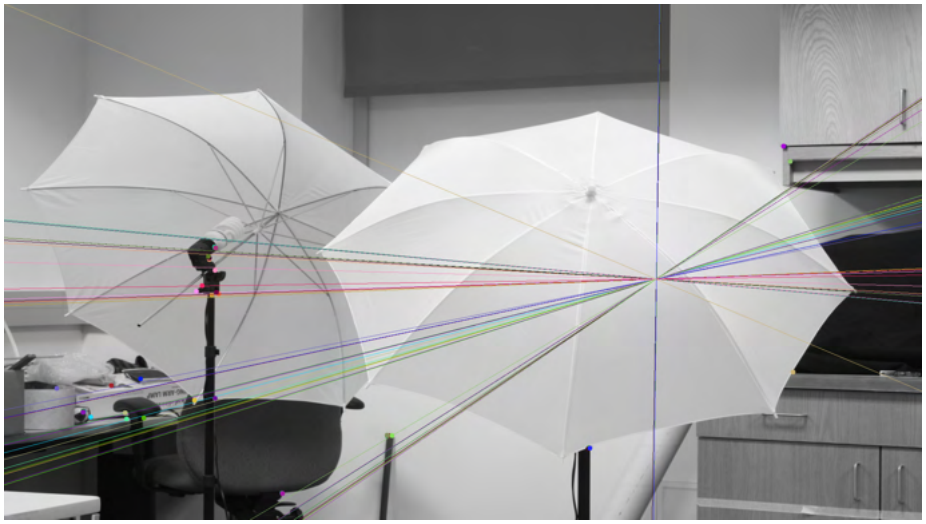

Figure 13: Unrectified image of im1 for data set 3 with epilines

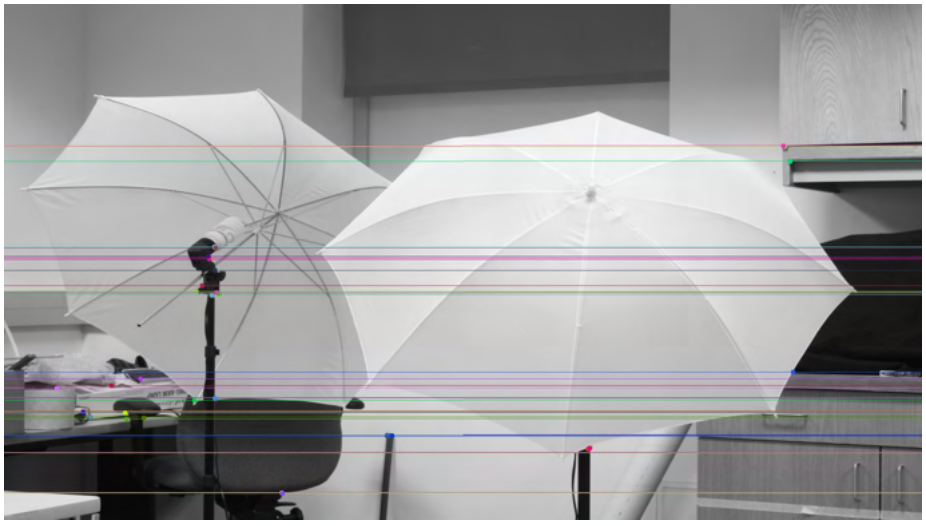

Figure 14: Stereo rectified image of im1 for data set 3

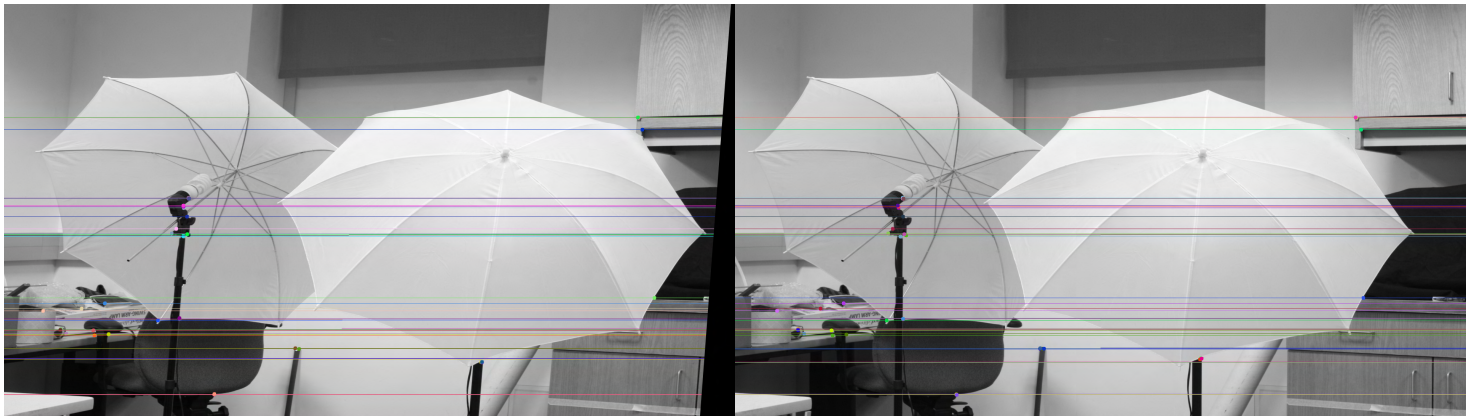

Figure 15: Comparison of rectified images for data set 3

Compute Disparity and Depth map

In the previous section, we got two images which are on the same plane. For computing the depth, we need to calculate the difference between the x co-ordinates of matching points of the images in the parallel plane. For finding disparity, the following steps are followed.

- The first image is split into windows of size 21.
- Each window is individually mapped in the second image to find the matching point. To achieve this, we use cv2.matchTemplate() function. This function matches points based on SSD.
- The matching point obtained from previous step is then used to compute disparity. The difference between the x coordinate of both the point and match point will give us the value of disparity.
- Using the value of disparity, depth can be calculated using the formula $depth = focal_length * baseline/disparity$.
- For each window, the value of depth map and disparity are input in a image matrix of the same to obtain the depth and disparity map.

The following images are depth and disparity maps for all the data sets are as follows(figures 16, 17, 18, 19, 20, 21). As you can see, for a window size of 21, the images look patchy and sometimes the depth values look mixed in the maps. This clearly seen in the disparity map of data set 1 for rear wheel.

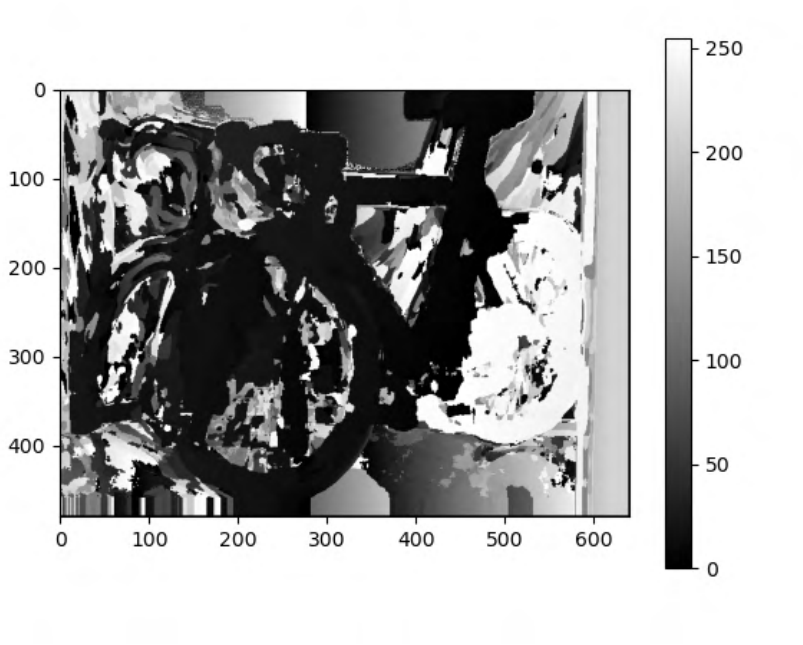

Figure 16: Disparity map for Dataset 1

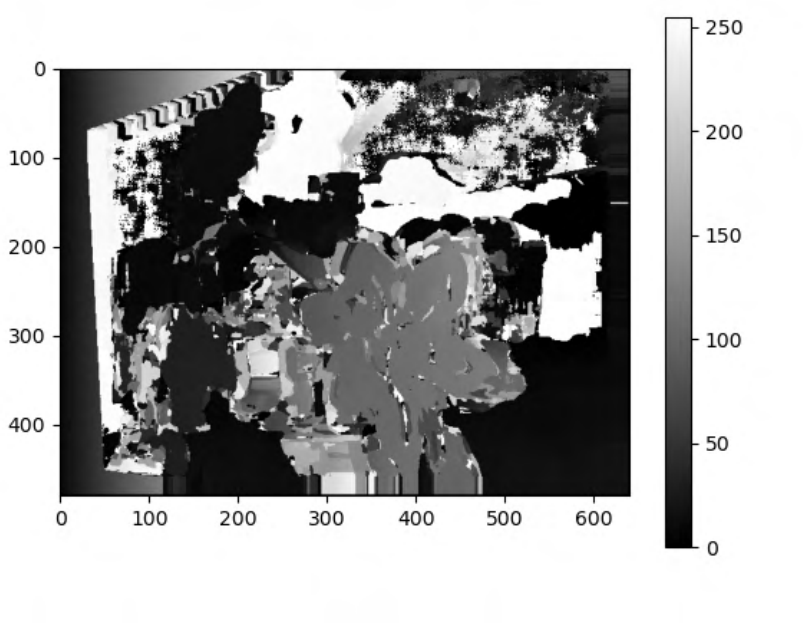

Figure 17: Disparity map for Dataset 2

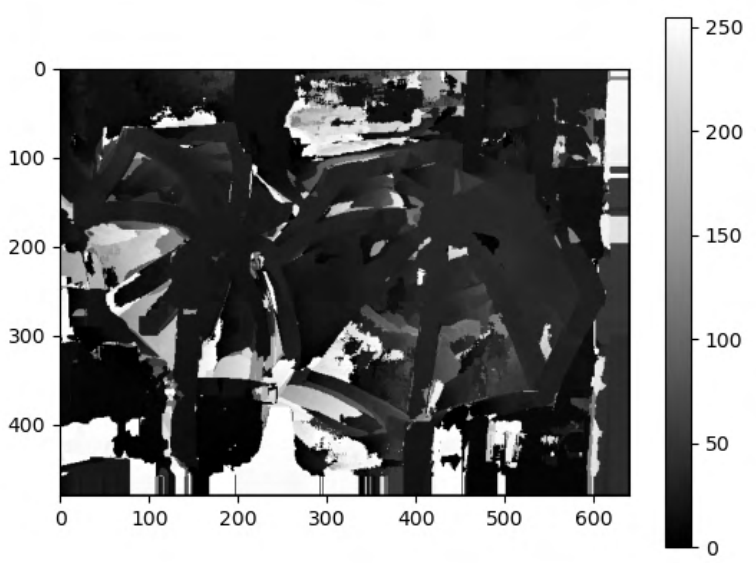

Figure 18: Disparity map for Dataset 3

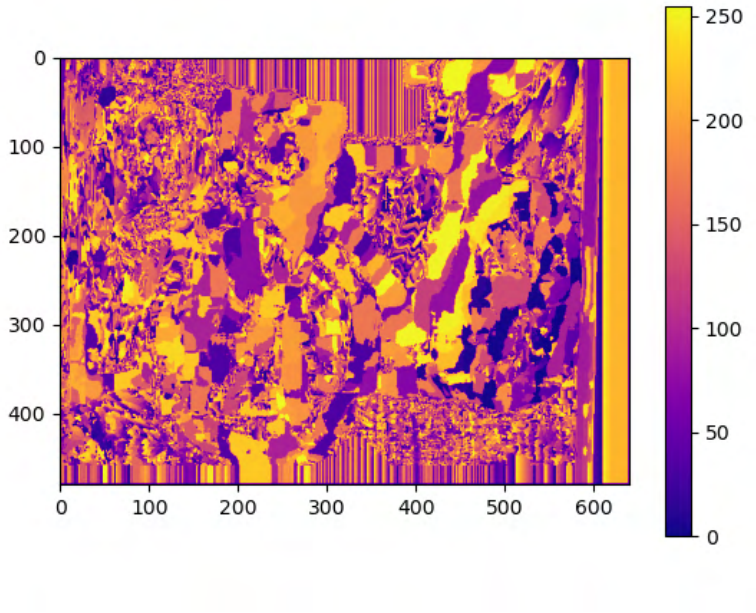

Figure 19: Depth map for Dataset 1

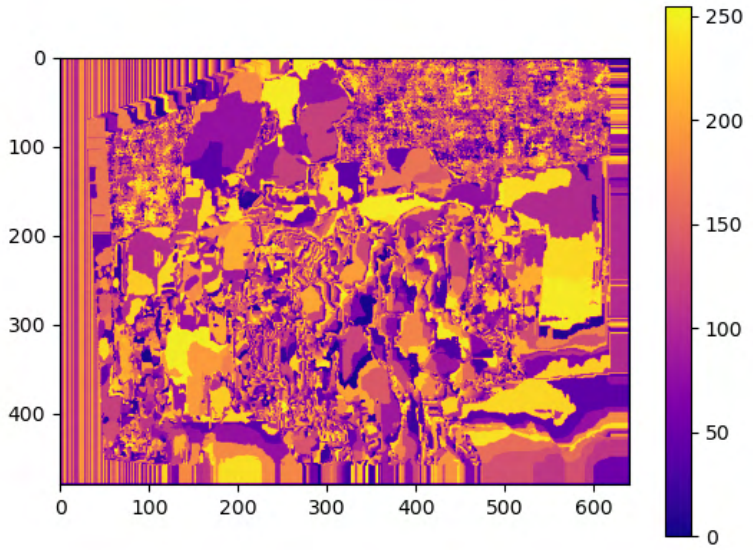

Figure 20: Depth map for Dataset 2

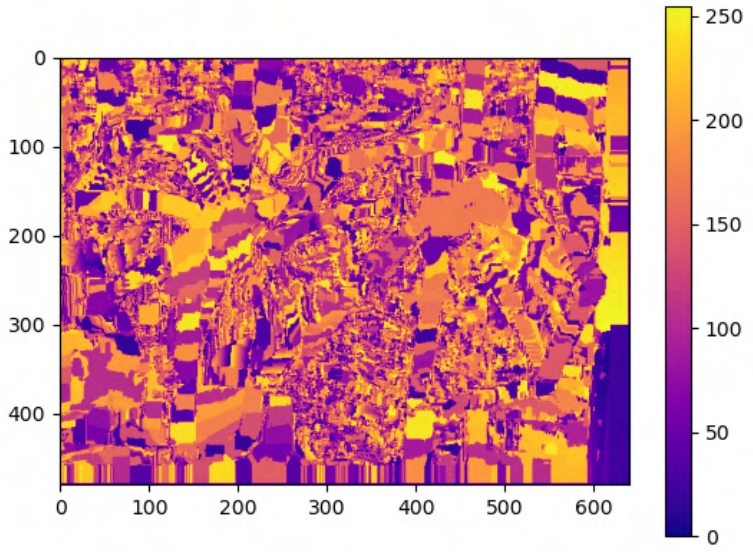

Figure 21: Depth map for Dataset 3