# Project 3
# ENPM673- project3
# mano srijan battula
# 118546490

## I. CALIBRATION
## A. Feature Detection and Matching:

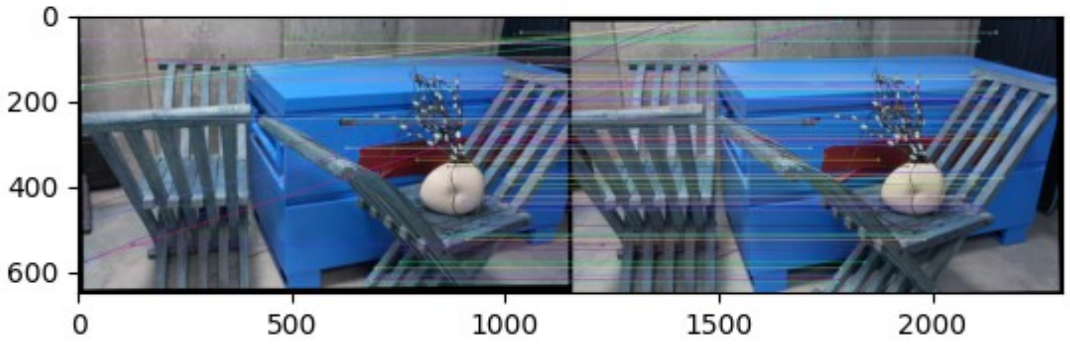
1) Make a grayscale conversion of the image.
2) To get the corner points, use the SIFT feature detector.
3) To acquire the matching corners, use the Brute-Force matcher.
The cv2.BFMatcher OpeenCV function can be used.
4) A list of DMatch objects is returned by the bf.match(des1,des2) line. The following attributes apply to this DMatch object: [3]
a) Distance between descriptors (DMatch.distance).
The smaller the number, the better.
b) DMatch.trainIdx - The descriptor's index in the train descriptors
c) DMatch.queryIdx - The descriptor's index in the query descriptors
d) DMatch.imgIdx - Image index for the train.
To acquire the first n best matches, sort the collected matches by distance parameter. The value of n is 100 in this case.
5) Figure 1 depicts the final result. There are a few incorrect matches obtained, as can be observed. These can have a significant impact on the outcome. RANSAC will be used to filter these incorrect feature pairings. The following parts will go over it in detail.

## B. Estimation of Fundamental Matrix and RANSAC:

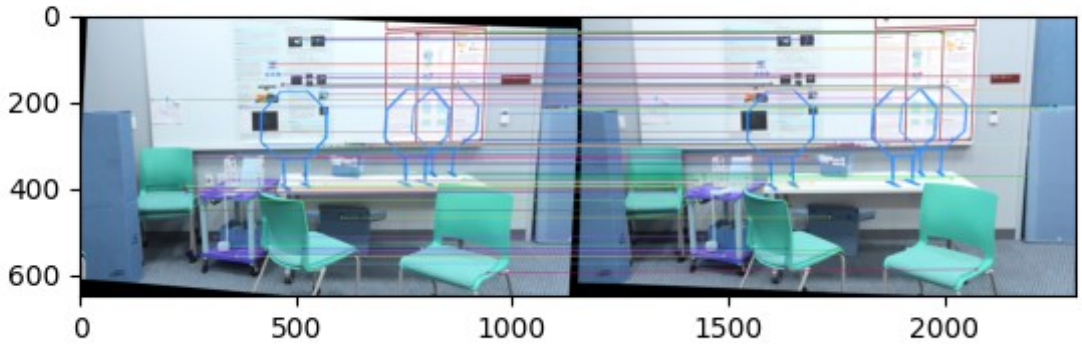
The 8-point technique is used to calculate the fundamental matrix.if the F matrix estimation is good. 1 should be near to 0, where x1 and x2 are image1 and image2 features, respectively. RANSAC can be used to filter outliers using these criteria. The RANSAC results are displayed. 2. It is clear that the number of outliers has decreased dramatically.

## C. Estimation of Essential Matrix

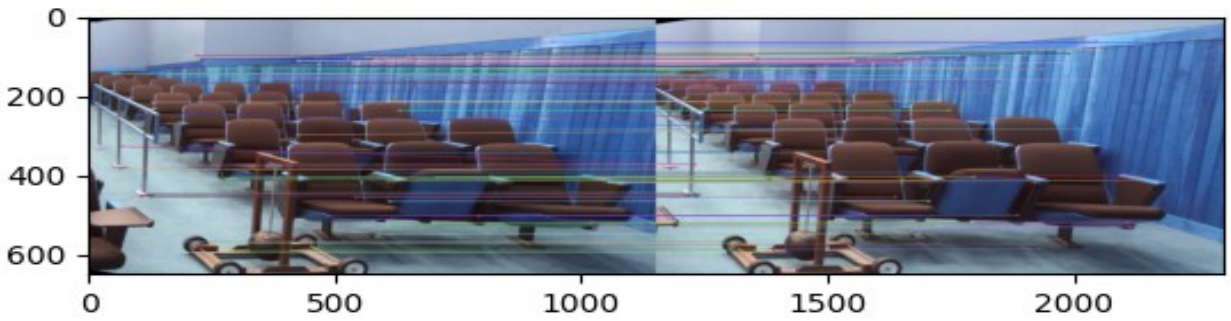
From the preceding section, we will create a Fundamental matrix. The cameras' inherent parameters are previously known (K1 and K2 ). The essential matrix is calculated as follows:




**(a)**




**(b)**




**(c)**

**fig:1 Sift features matched using Brute force matcher**

## D. Estimation of Camera Pose

The essential matrix E can be used to estimate camera pose(R and C) . We'll calculate camera 2's posture in relation to camera 1, which is considered to be at world origin. 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.

```
 R of camera Pose [[-0.55215682  0.31603544 -0.77152087]
 [-0.09008912 -0.94257075 -0.32162764]
 [-0.82885873 -0.10808326  0.54891822]]
C of camera Pose [ 0.4540647   0.13957074 -0.87996889]
 H1  [[ 2.91408276e-03 -1.64004267e-04 -1.33400745e+00]
 [ 6.28881171e-04  9.08527723e-04 -3.34803302e-01]
 [ 2.04079323e-06 -7.58032971e-07  1.65615054e-04]]
 H2  [[ 2.12240147e+00  2.12818316e-01 -7.15456381e+02]
 [ 5.34385438e-01  1.05859882e+00 -3.26792030e+02]
 [ 1.95727631e-03  1.96260818e-04 -1.90979657e-01]]
```

For problem1

```
 R of camera Pose [[ 0.99974098  0.00178933 -0.0226884 ]
 [-0.00214383  0.99987586 -0.01560975]
 [ 0.02265765  0.01565435  0.99962071]]
C of camera Pose [ 0.98806149 -0.06663895  0.13890195]
 H1  [[-3.40537227e-03  2.24332760e-04 -7.18739461e-02]
 [-9.31232238e-05 -3.54869050e-03  6.51950164e-02]
 [ 2.33411852e-07 -7.38672419e-08 -3.64724399e-03]]
 H2  [[ 9.53272946e-01 -4.67200946e-02  4.20520939e+01]
 [ 2.33418211e-02  1.00005629e+00 -1.34631276e+01]
 [-7.90420378e-05  3.87386582e-06  1.04427308e+00]]
```

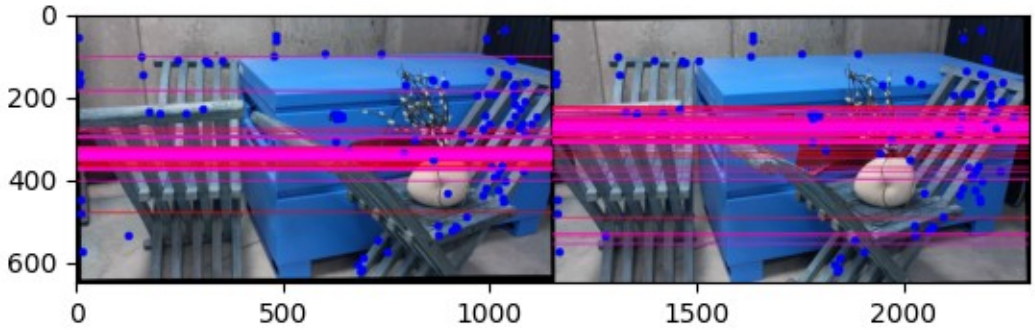
for problem 2

```
 R of camera Pose [[ 0.9995061  -0.02398288 -0.02030698]
 [ 0.02391448  0.99970751 -0.00360454]
 [ 0.02038749  0.00311713  0.99978729]]
C of camera Pose [-0.09949638  0.23724113 -0.96634213]
 H1  [[ 1.50682093e-03  2.90294020e-04 -4.92217519e-01]
 [ 1.38504720e-04  1.00777209e-03 -1.07084223e-01]
 [ 1.14699650e-06  2.39039829e-07  1.53881036e-04]]
 H2  [[ 1.67711223e+00  4.31854470e-01 -5.29937493e+02]
 [ 1.49280767e-01  1.07106040e+00 -1.09009290e+02]
 [ 1.23038629e-03  3.16823054e-04  1.88646828e-01]]
```

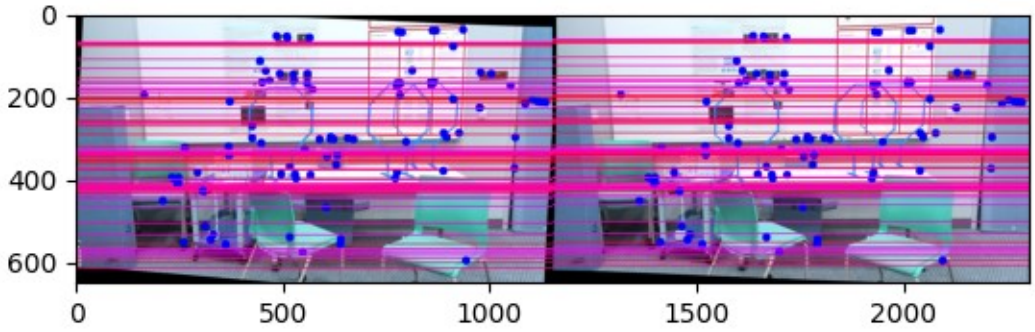
for problem 3

## II. RECTIFICATION

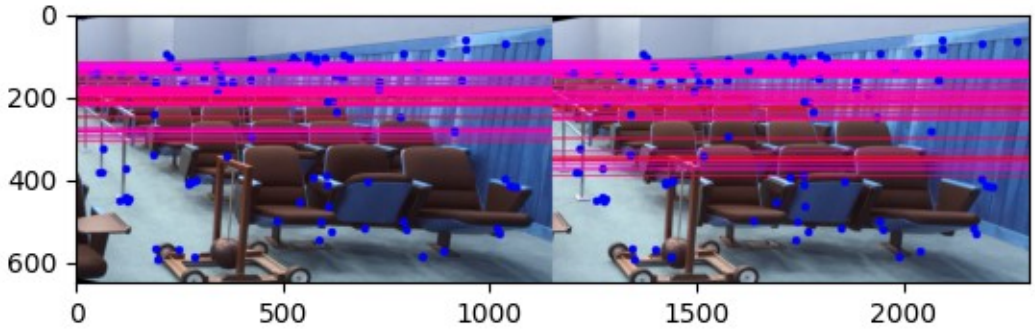
The epipolar lines for both images can be obtained using the fundamental matrix and feature points. For future computations to derive depth, the epipolar lines must be parallel. Re projecting picture planes onto a common plane parallel to the line between camera centers is one way to accomplish this. The OpenCV function cv2.stereoRectifyUncalibrated can be used to rectify the pictures using the matched image pairs and the calculated F matrix. Two homography matrices will be obtained, one for each image.




**(a)**



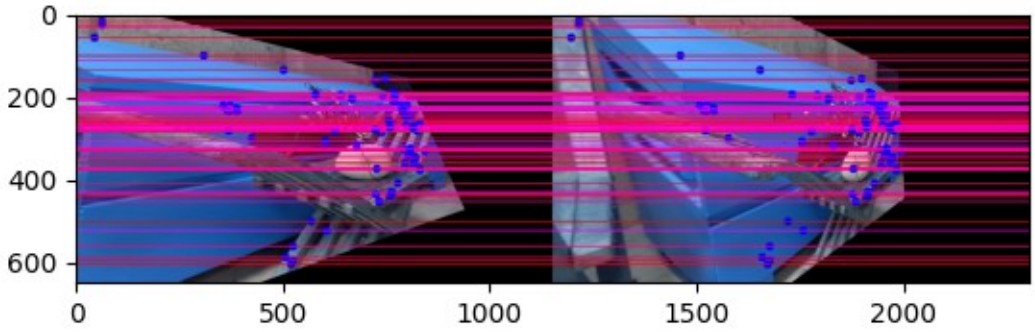

**(b)**



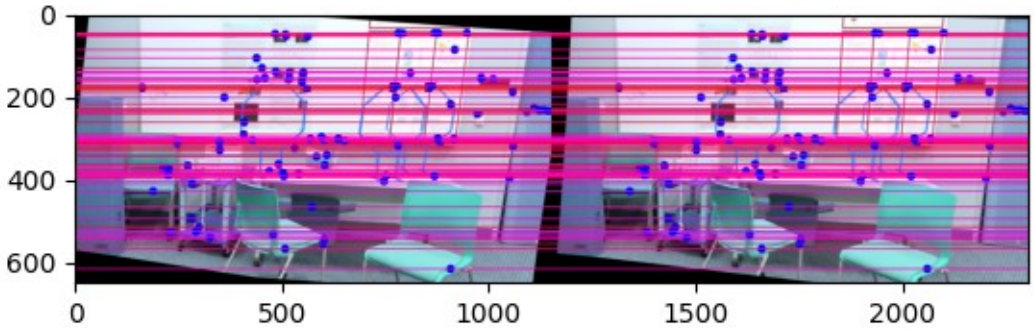

**(c)**

**fig 2: epipolar lines for unrectified images**

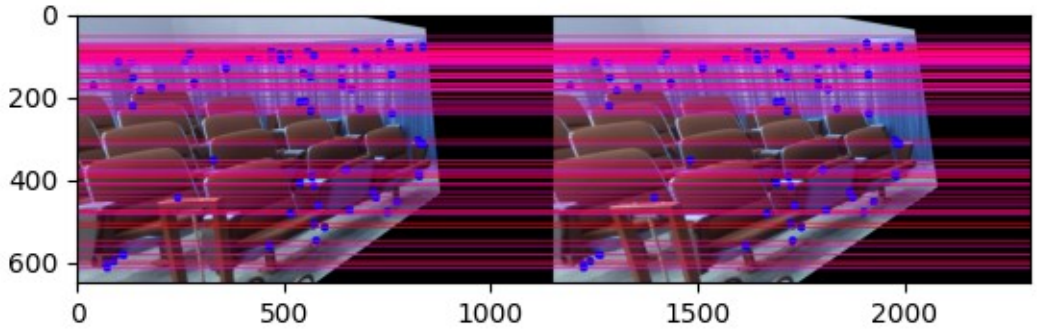
The epipolar lines will be parallel after rectification. We must also convert the feature points using the generated homography matrices once the images have been distorted. In addition, as the epipolar geometry changes, the F matrix will vary. H2 and H 1 will be the results.




**(a)**




**(b)**




**(c)**

**fig 3: epipolar lines for rectified images**

-

## III. CORRESPONDENCE

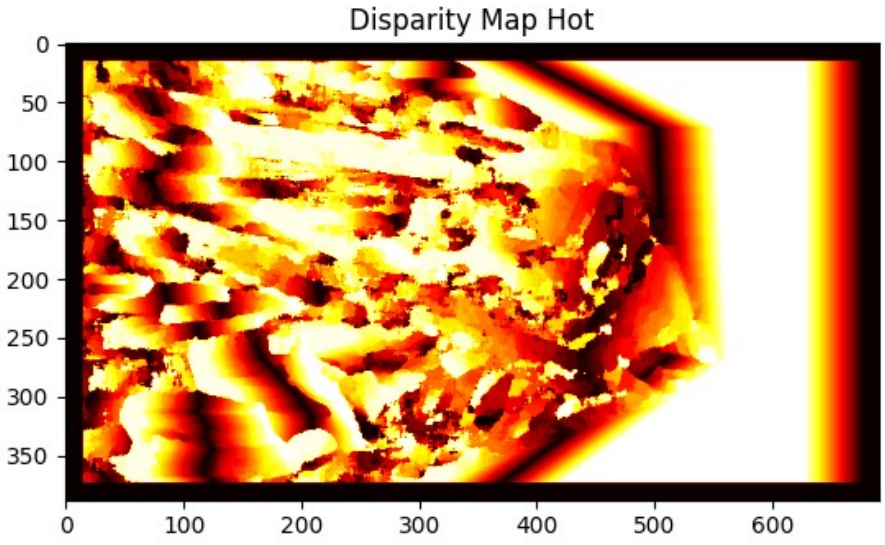
We try to identify a corresponding match along the epipolar line in image 2 for each pixel in image 1. This method is termed block matching since we will consider a window of a preset size for this purpose. Essentially, we'll search for the closest matching region of pixels in the right image using a small region of pixels from the left image. I used Sum of Squared Differences (SSD) Sum of Squared Differences (SSD) takes a reasonable amount of time to compute and produces acceptable results. The usage of nested for loops is a straightforward way to build the block matching method. However, the strategy is ineffective. In the following section,

## IV. COMPUTE DEPTH IMAGE

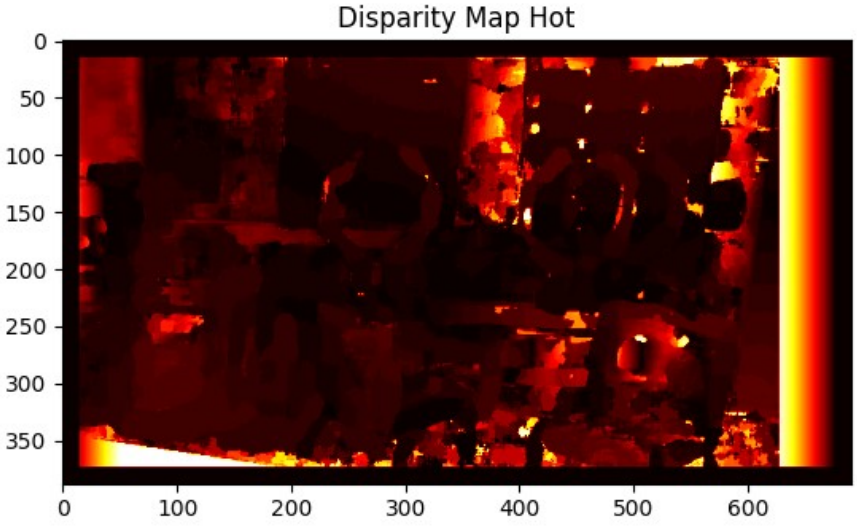
### A. Disparity Map

After we've discovered the matching pixel location, we can calculate the disparity by taking the absolute difference between the source and the matched pixel location.

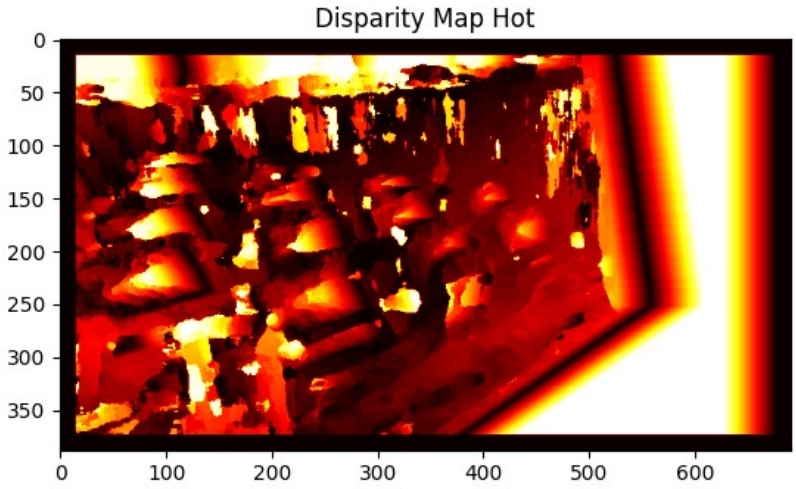
**DISPARITY MAP(HOT):**




For problem(1):

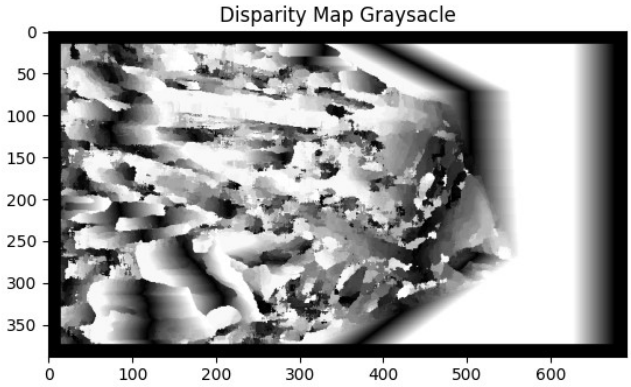


For problem(2):

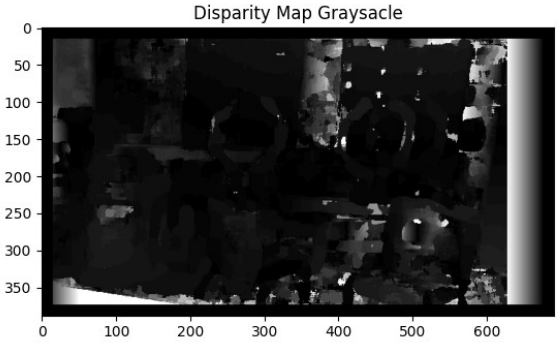



For problem(3):

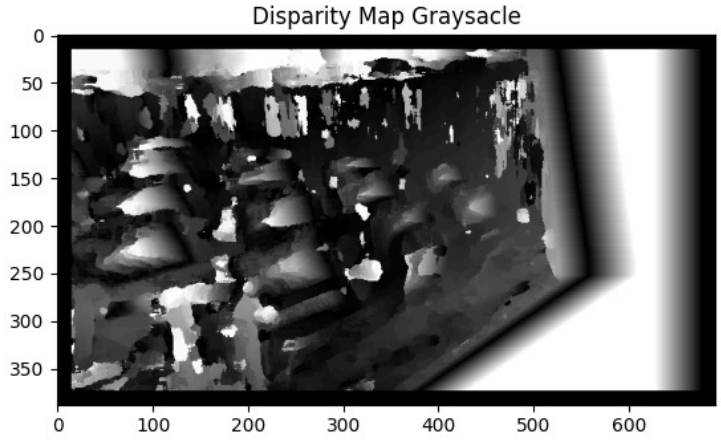
## DISPARITY MAP(GRAYSCALE):



For problem(1)



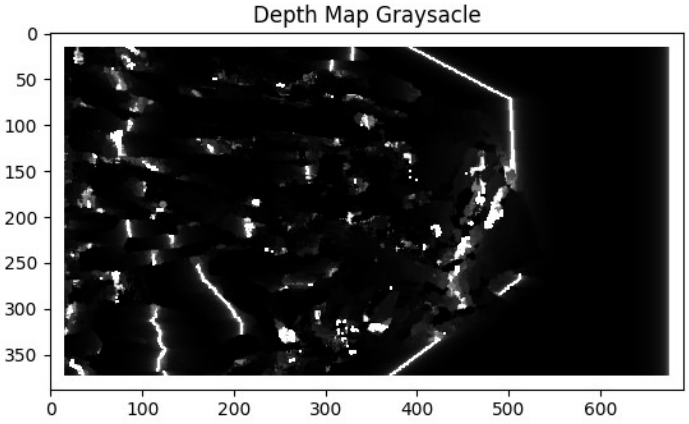

For problem(2):



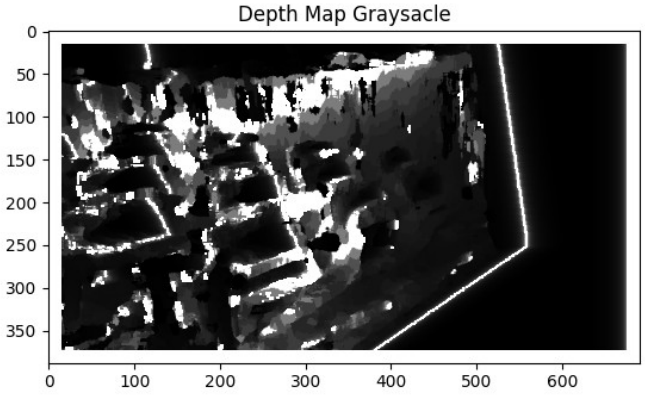

For problem(3):

## B. Depth Map

### DEPTH MAP(GREY):



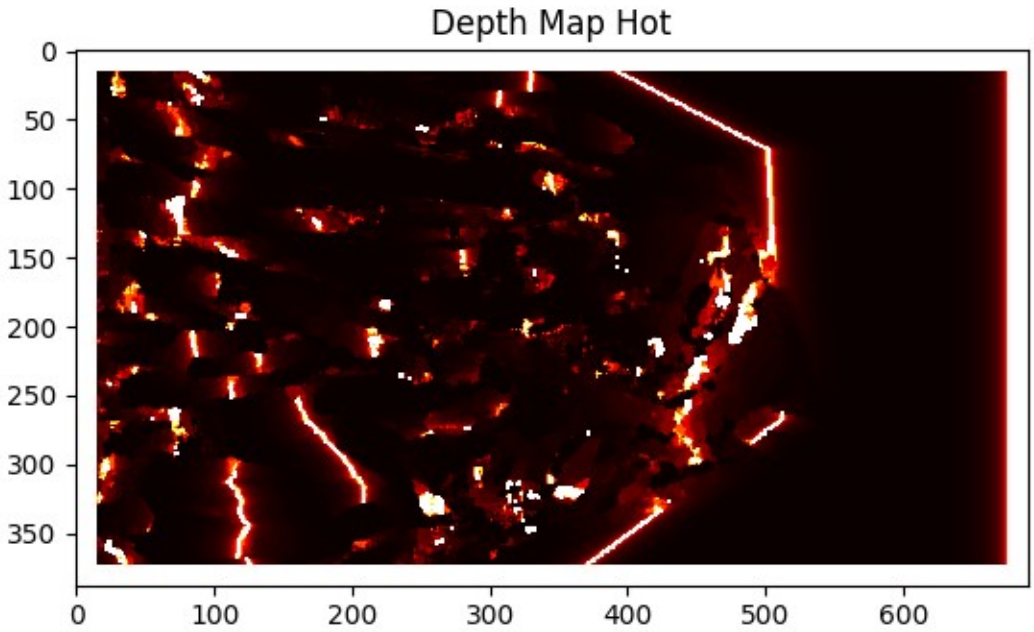

For problem(1)

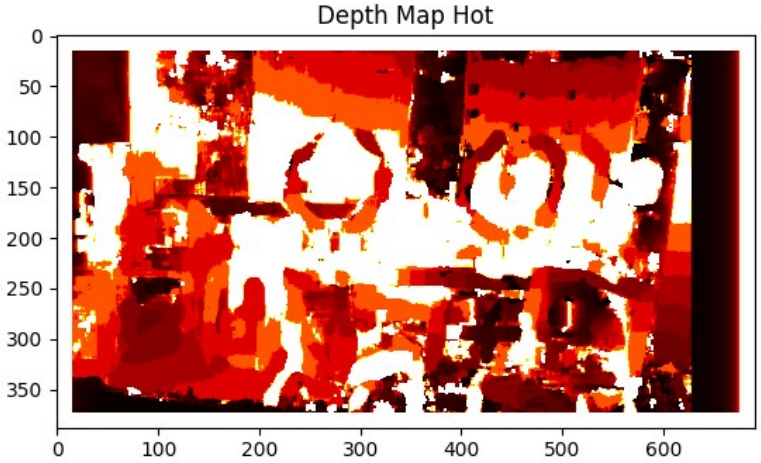


For problem(2):

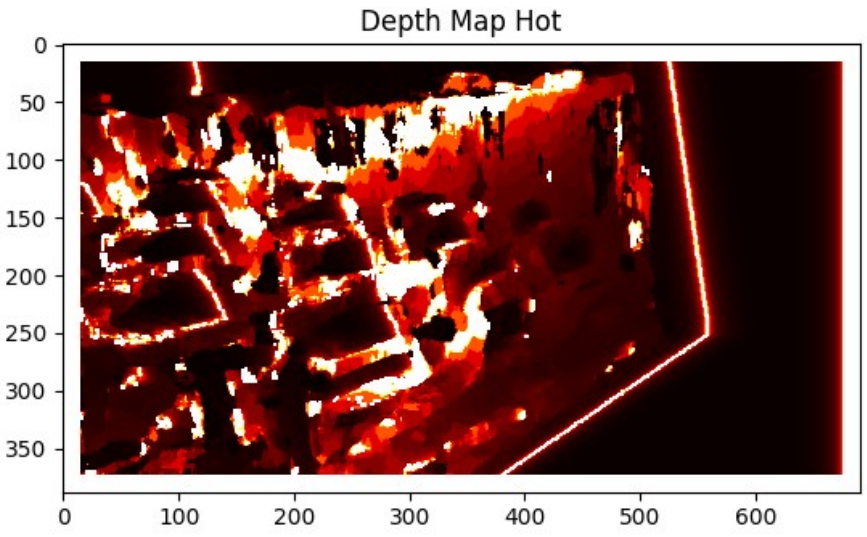
**DEPTH MAP(HOT):**

For problem(1)



For problem(2):




For problem(3):

Drive link to output:
https://drive.google.com/drive/folders/1klBicmZIbdAdg4wgxrt54jcuTUmu7DSI?usp=sharing