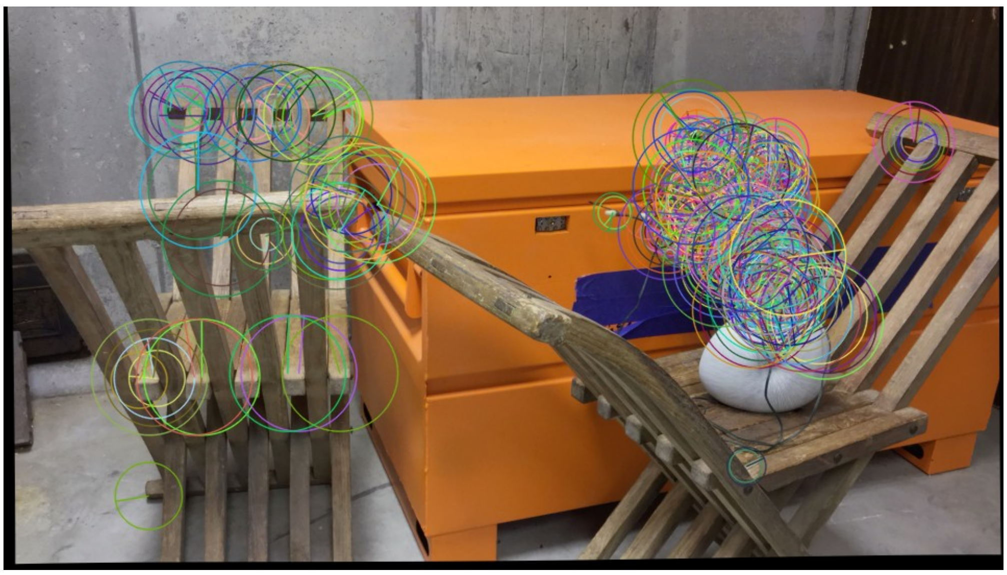
ENPM 673: Perception for Autonomous Robots
Project 3 – Report

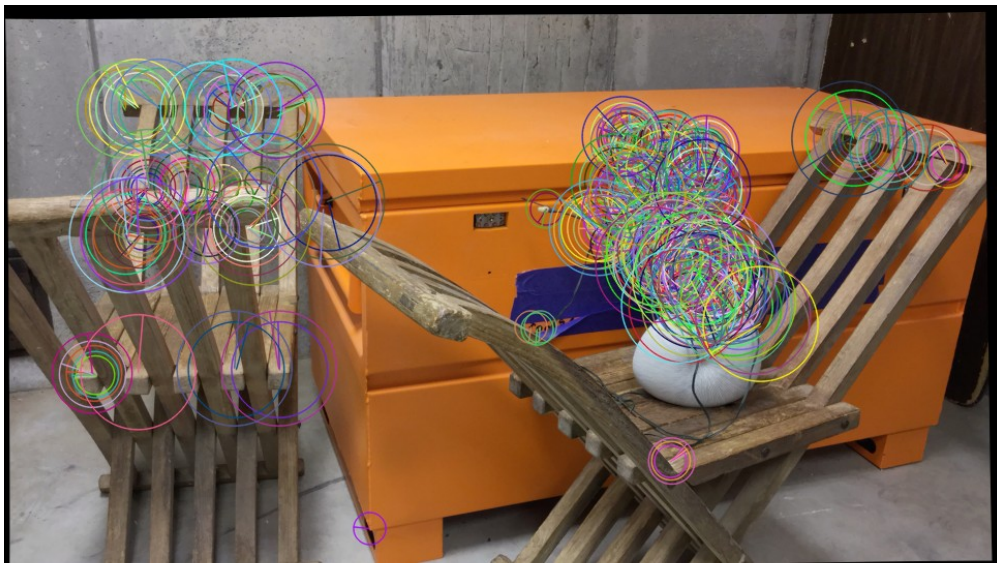
Hemanth Joseph Raj – 117518955

Stereo Vision System

Dataset 1 - Curule
Part 1 – Calibration
Pipeline towards the solution

1. Read two images separately
2. Rescale images to fit in screen
3. Use ORB detector to detect keypoints in both images. Create an orb object
4. Detect and compute key points and descriptors for both images


(Fig. 1 Keypoints in first image)


(Fig. 2 Keypoints in second image)

5. Match features on both images. Create a Brute force matcher and use k-nearest neighbours method to match points on both images
6. Now from the matched points we only choose good matches, ones which are distinct enough

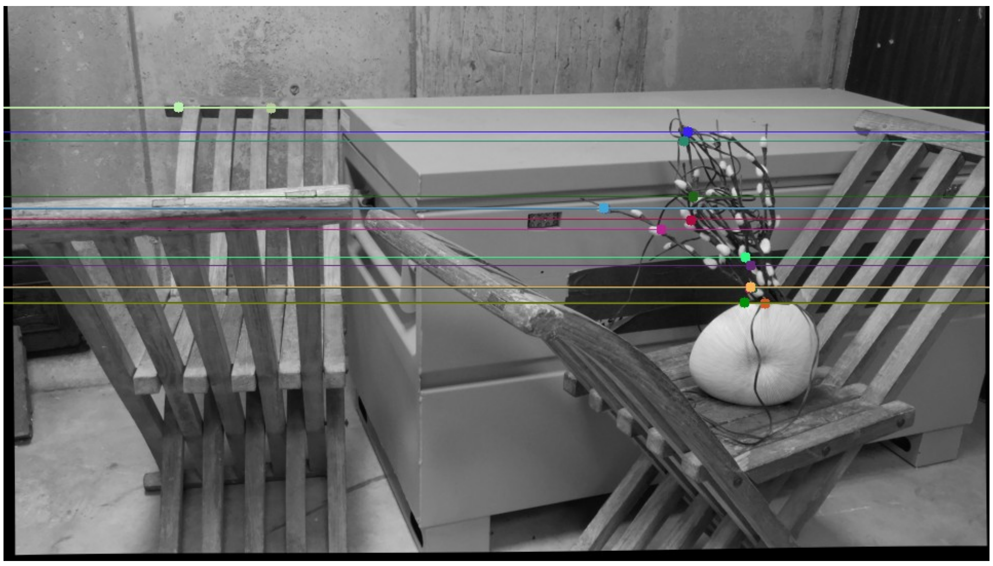
7. Now we need to calculate the pixel coordinates of the good matching points since the matching points are in the form of DObjects
8. From the resulting coordinate points choose only the first 8 points for each image as we need only 8 points to calculate our Fundamental matrix
9. Arrange these 8 points from each image into an array A and take its SVD
10. The last column of the V matrix from the SVD is the needed Fundamental matrix. We extract and reshape into an 3 by 3 matrix and therefore we get the Fundamental Matrix
11. Once we have the fundamental matrix we need to calculate the Essential Matrix. For that we need the intrinsic camera parameters, this is already provided to us
12. The essential matrix is the matrix multiplication of the three matrices in the particular order Transpose of intrinsic matrix * Fundamental matrix * Intrinsic matrix
13. Thus we obtain the Essential matrix for the images
14. Now we need to decompose the Essential matrix into corresponding pairs of translation and rotation matrices. We will get four such pairs
15. First we perform the SVD of Essential matrix. Define a new matrix W as follows
    [0, -1, 0]
    [1, 0, 0]
    [0, 0, 1]
16. We get the following four pairs of translation and rotation matrices
    C1 = U[:, 2]
    R1 = U * W * V transpose (U and V are from the SVD of essential matrix)

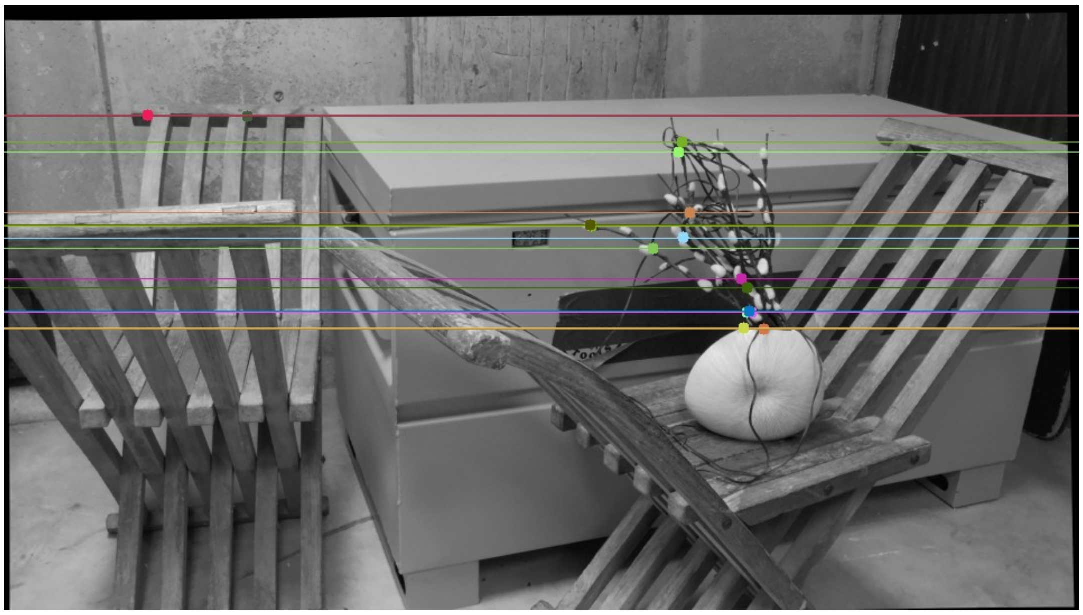
    C2 = - U[:, 2]
    R2 = U * W * V transpose

    C3 = U[:, 2]
    R3 = U * W transpose * V transpose

    C4 = - U[:, 2]
    R4 = U * W transpose * V transpose

The epilines for each image are as follows


(Fig. 3 Epilines in first image)


(Fig. 4 Epilines in second image)

Part 2 – Rectification

Pipeline towards the solution

1. Using the cv.stereoRectifyUncalibrated function to get the homography matrices for both images

   The homography matrices are

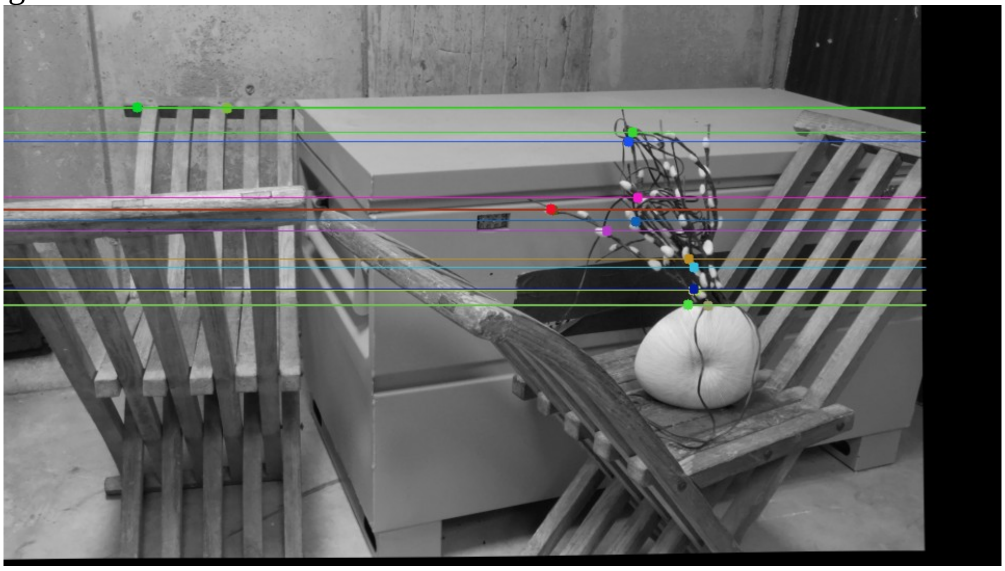
   H1 =

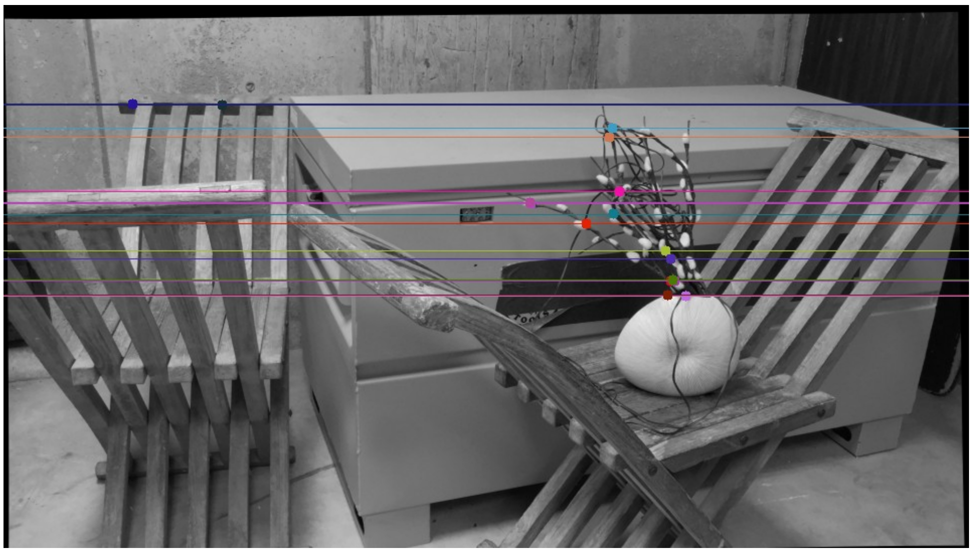
```
[[-6.79073699e-01 -2.55442303e-03  2.50387638e+01]
 [ 0.00000000e+00 -7.07110000e-01  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00 -7.07110000e-01]]
```

   H2 =

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

2. Warp the perspective of both the images so that the epilines are horizontal in the images

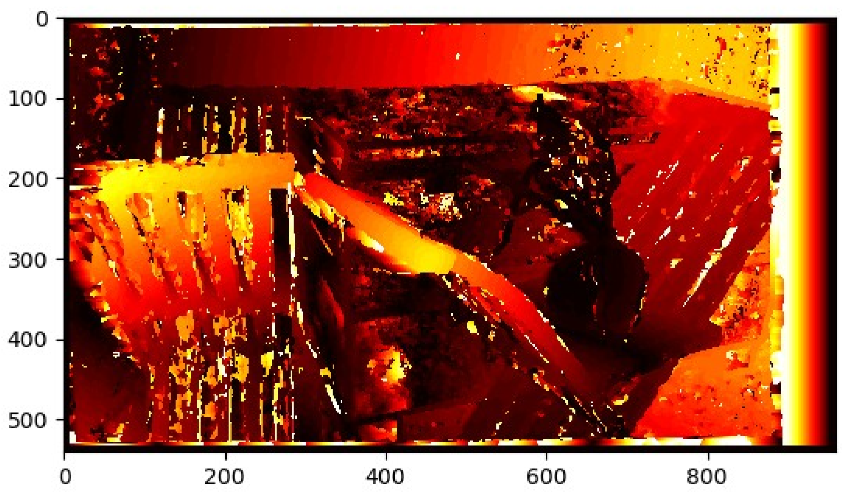

(Fig. 5 Stereo rectified image 1)



(Fig. 6 Stereo rectified image 2)

Part 3 & 4 – Correspondence and Depth computation
Pipeline towards the solution

1. Convert the images into arrays
2. If shape isn’t same raise an error to handle the exception
3. Create a zeros of disparity map
4. Since disparity calculation takes time, tqdm is used to monitor progress
5. Iterate over each pixel to calculate disparity
6. Choose a small block in left image
7. Compare it with a block from the right image
8. Get a search range for right image
9. Iterate through the range for right image
10. Calculate the disparity and depth



(Fig. 7 Stereo rectified image 2)

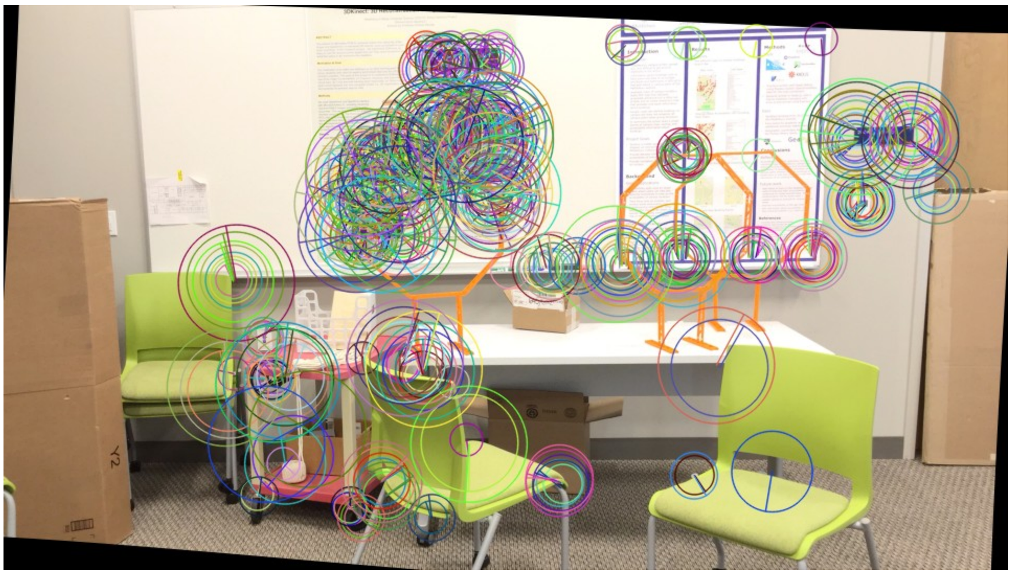
Dataset 2 – Octagon

For this dataset also, follow the same steps as the first dataset for each part.

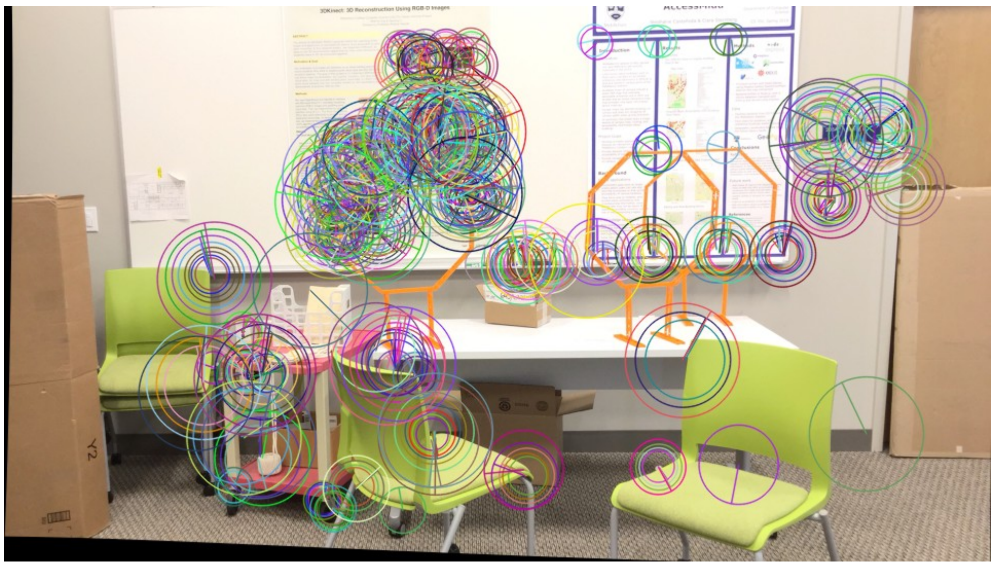
Differences to be made:

1. Change the camera intrinsic matrix
2. When rounding Fundamental Matrix, do np.round (the default no. of decimals)

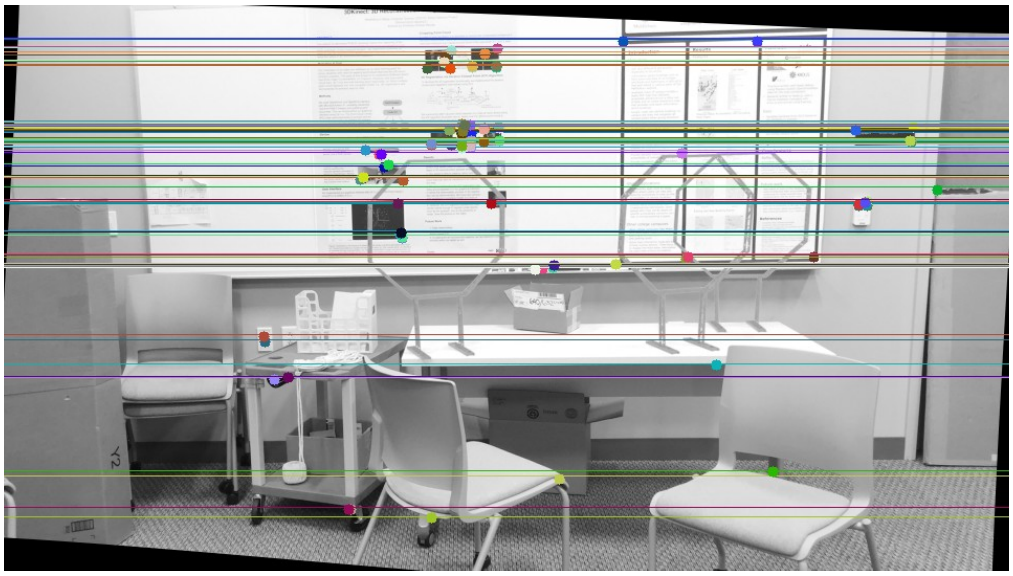
Output images for dataset 2:

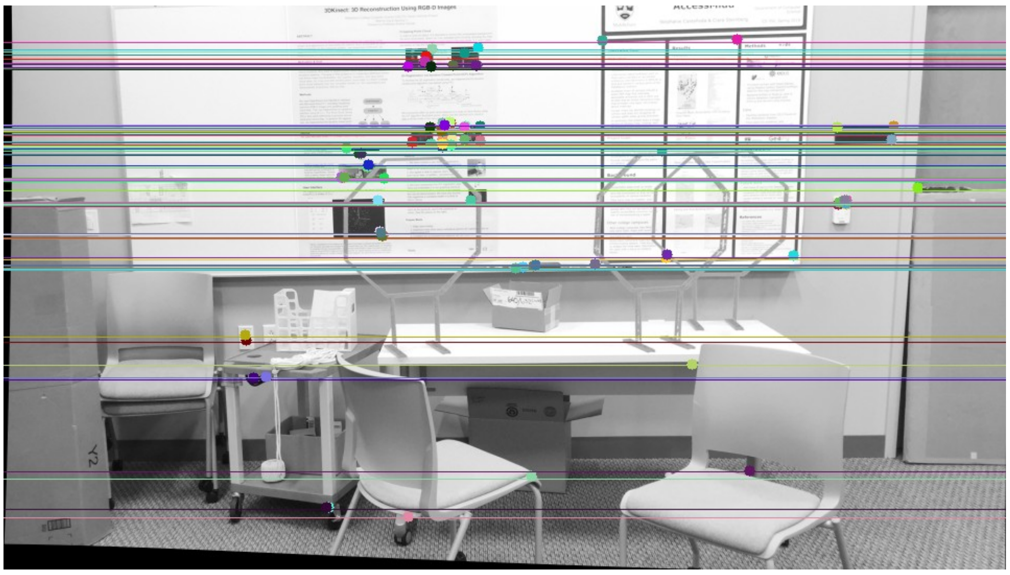


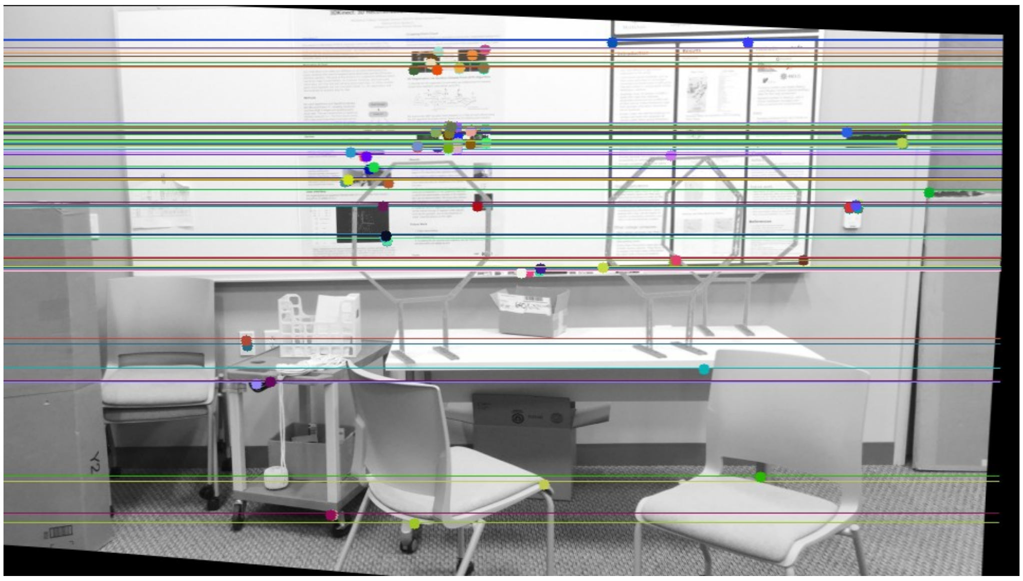
(Fig. 9 Keypoints in first image)

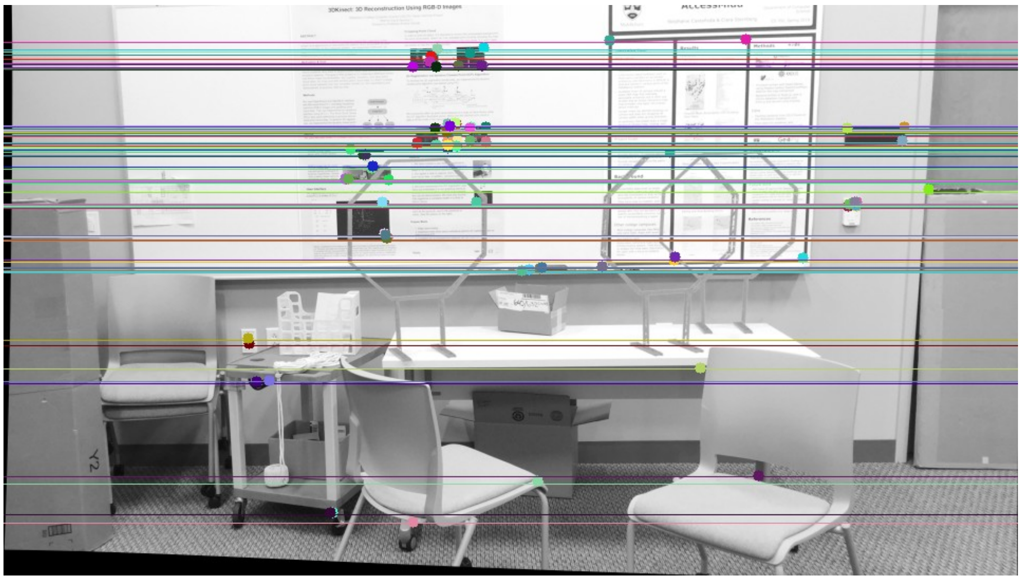


(Fig. 10 Keypoints in second image)


(Fig. 11 Epilines in first image)


(Fig. 12 Epilines in second image)


(Fig. 13 Stereo rectified image 1)


(Fig. 14 Stereo rectified image 2)

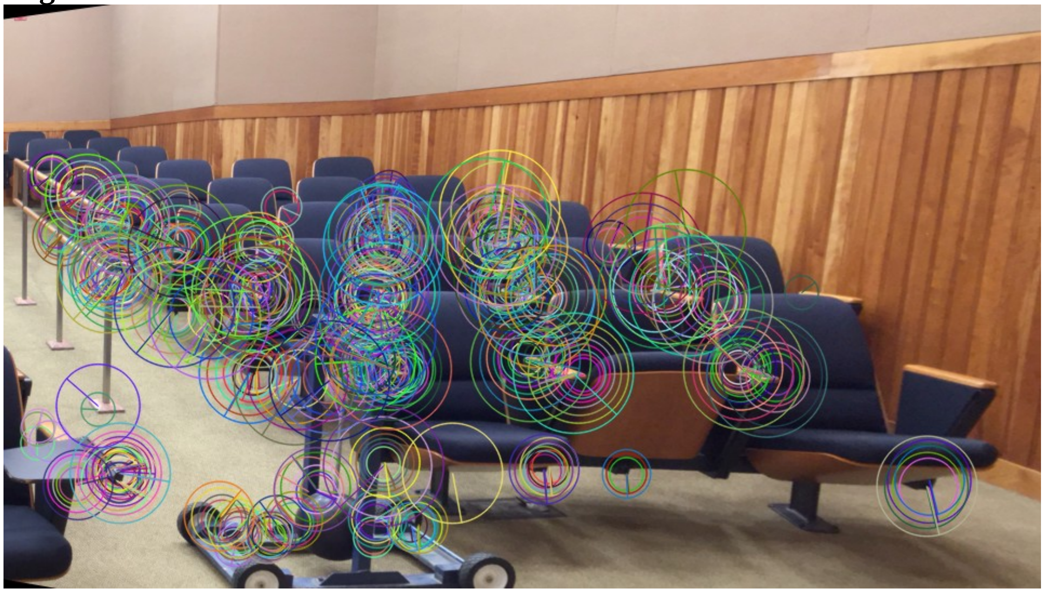
Dataset 3 – Pendulum

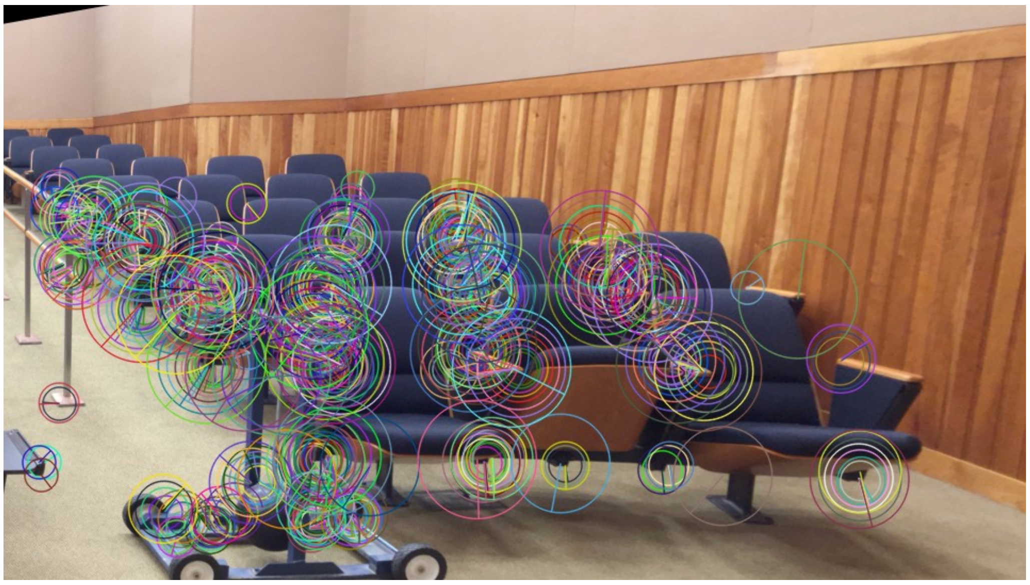
For this dataset also, follow the same steps as the first dataset for each part.

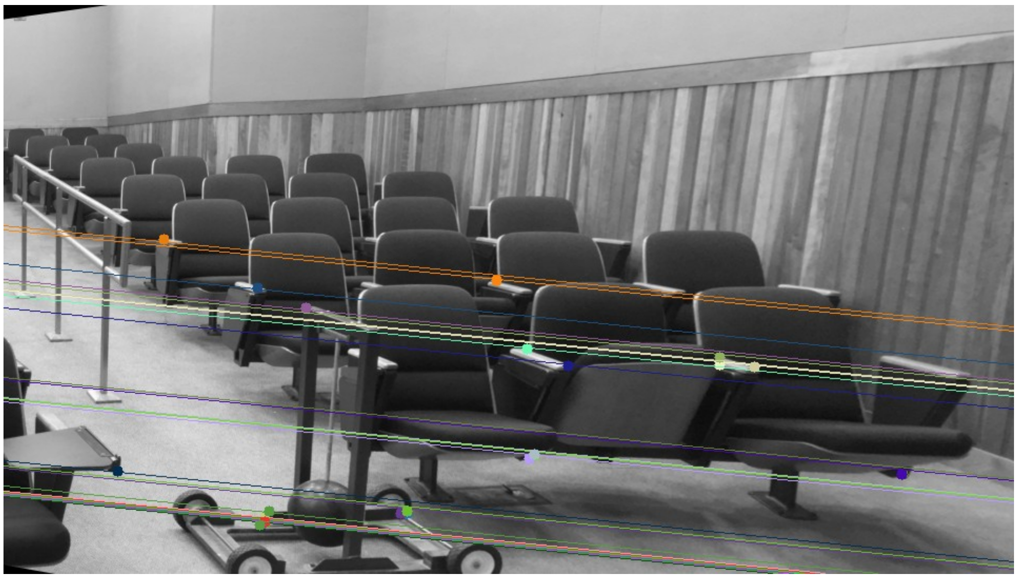
Differences to be made:

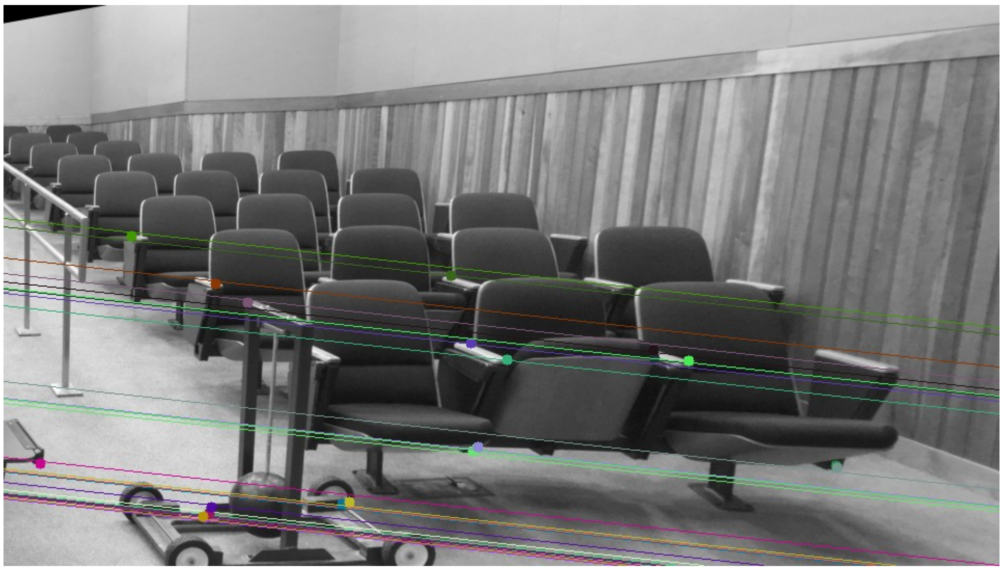
1. Change the camera intrinsic matrix
2. When rounding Fundamental Matrix, do np.round upto three decimal values

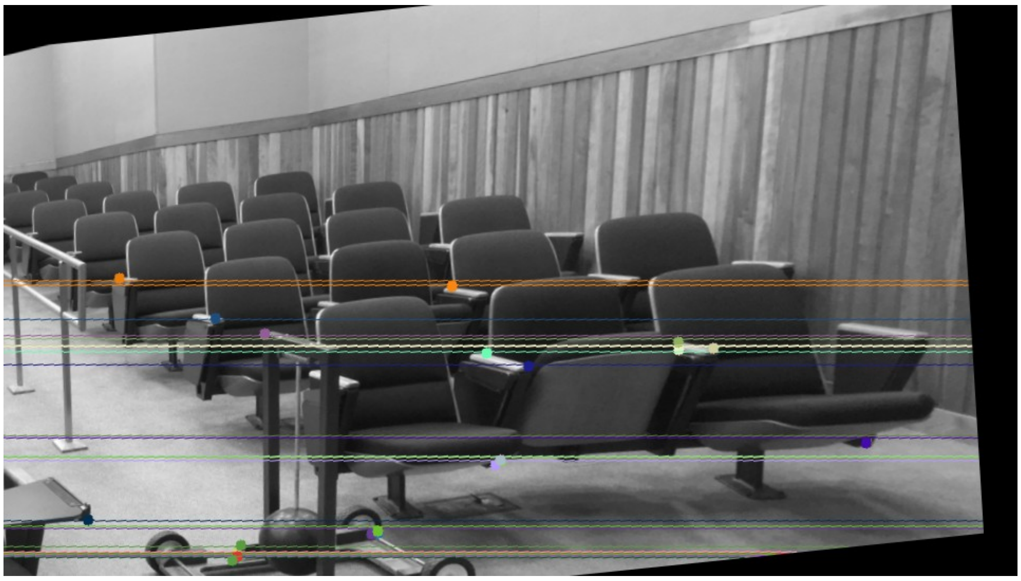
Output images for dataset 3:


(Fig. 15 Keypoints in first image)

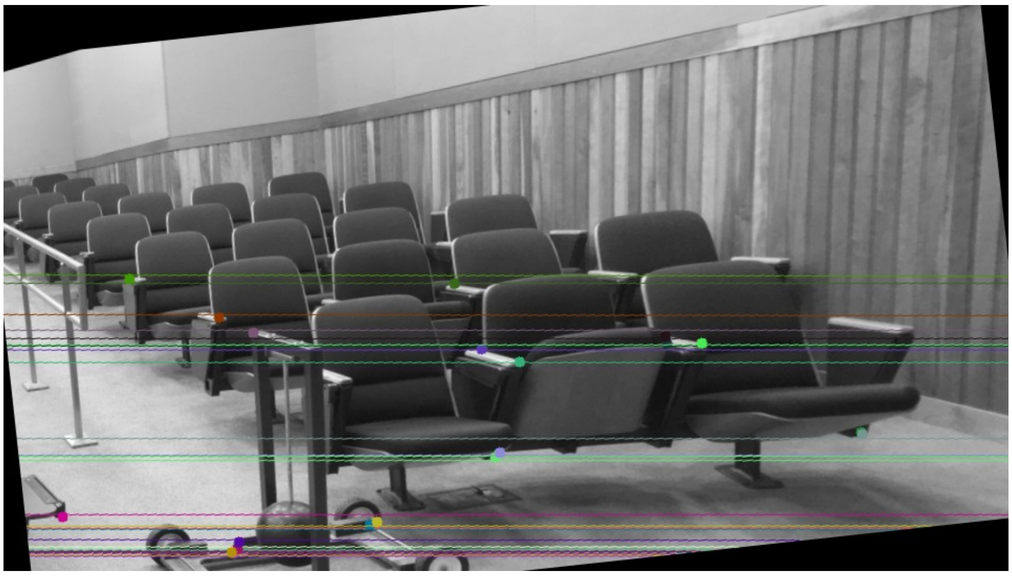

(Fig. 16 Keypoints in second image)


(Fig. 17 Epilines in first image)


(Fig. 18 Epilines in second image)



(Fig. 19 Stereo rectified image 1)



(Fig. 20 Stereo rectified image 2)

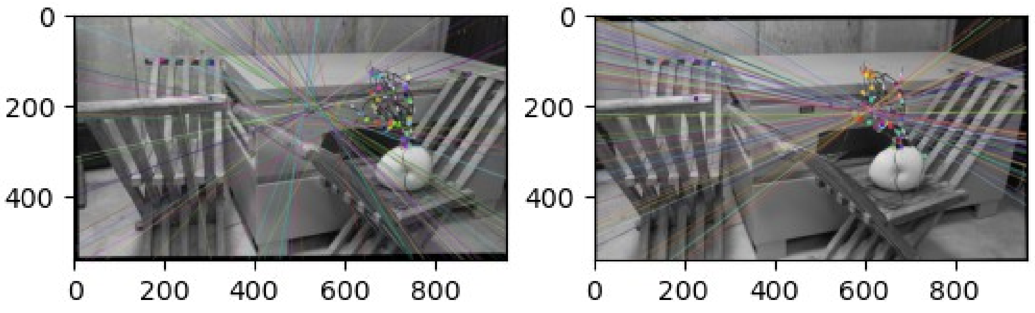
Problems Faced and Solution Implemented

I faced an issue where my epipole was falling inside my image and I wasn't getting the results I had expected. After going through eash line I found out the problem was with the bruteforce matcher method before getting the matching points. I was using the cv.NORM_HAMMING for matching which gave this problem.

```
bf = cv.BFMatcher_create(cv.NORM_HAMMING)
```

I solved this by using a simple brute force matcher.

```
bf = cv.BFMatcher()
```



(Fig. 21 Intersecting epipoles due to wrong bf matcher)