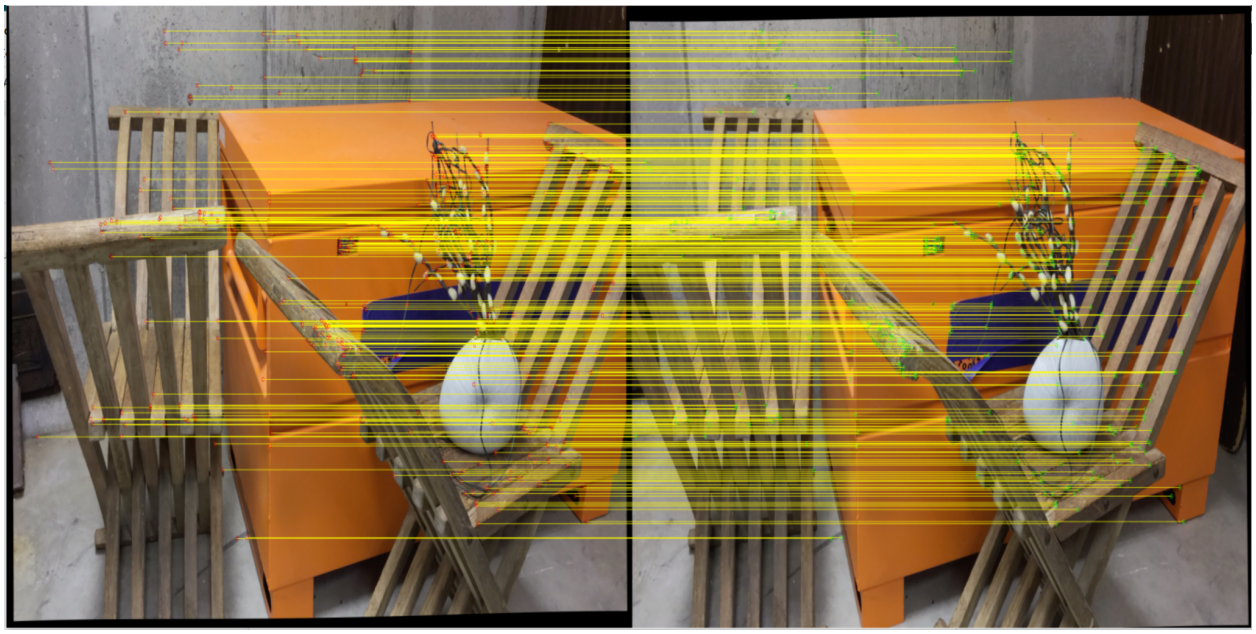
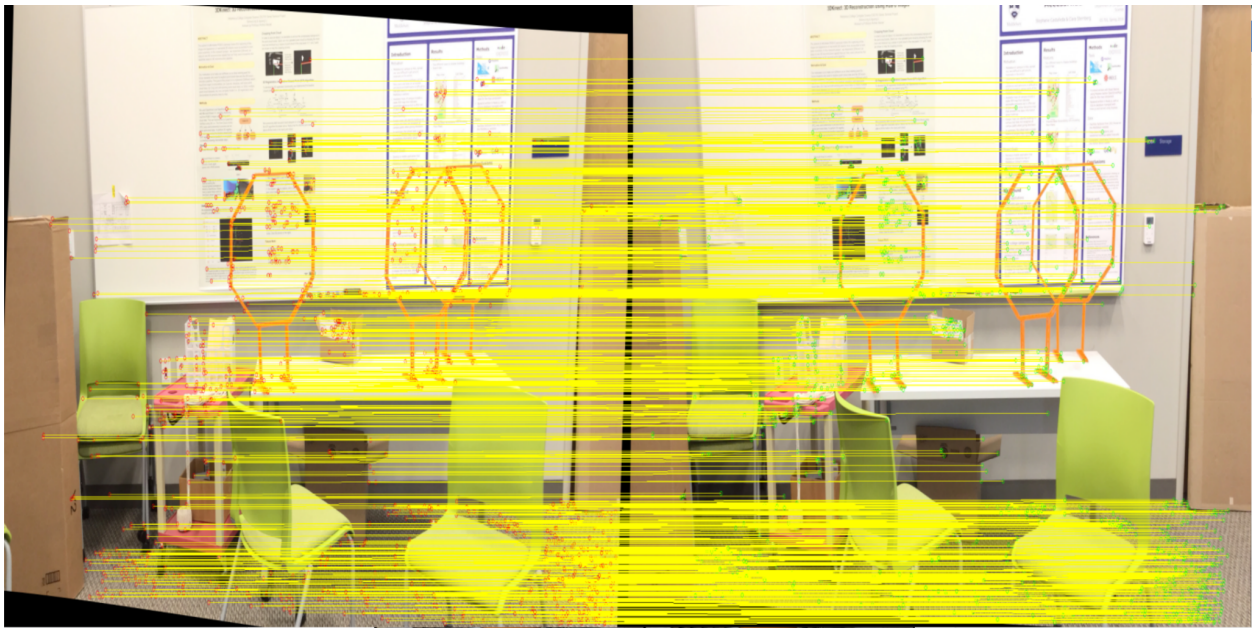
Project 3

Steps to solve the problem :

A. Feature extraction and matching

1. Convert the image to grayscale image.
2. Use SIFT feature descriptor to get the matching features.
3. Extract the matched features between two images by using BruteForce matcher.
4. Sort the matches by their distance and get the matched points in the source and destination image.
5. It can be seen there are wrong matches which need to be filtered using the RANSAC algorithm.





Matched Pairs in both the images

B. Estimating Fundamental Matrix

6. The Fundamental matrix is calculated using Eight-Point Algorithm.In RANSAC we select 8 random points in the left and right image and calculate the Fundamental matrix.I normalized the points before getting the fundamental matrix.Eight-Point Algorithm is not precise. Often, the distance between a point and its corresponding epipolar line will be very large, usually on the scale of 10+ pixels. Hence normalization is required.But without normalization I am getting the same results hence in the code I have commented normalization.

```
n=0;
for i = 1:M do
    // Choose 8 correspondences, x̂1 and x̂2 randomly
    F = EstimateFundmentalMatrix(x̂1, x̂2);
    S = ∅;
    for j = 1:N do
        if | x2j^T F x1j | < ε then
            S = S ∪ {j}
        end
    end
    if n <| S | then
        n =| S |;
        Sin = S
    end
end
```

7. The error is found by multiplying the fundamental matrix found using 8 points and each point in matched points of the left and right image.**(x1*F*x2)**

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

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

$$\begin{bmatrix} x_1 x_1' & x_1 y_1' & x_1 & y_1 x_1' & y_1 y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_m x_m' & x_m y_m' & x_m & y_m x_m' & y_m y_m' & y_m & x_m' & y_m' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

8. We have set a error threshold and if the error is less than the error threshold we take that matched point
9. This process is repeated for certain iterations and for each iteration we get a fundamental matrix.
10. We select that fundamental matrix for which we get the highest number of pairs of points(inliers) which have error below the threshold.Thus we get our Fundamental matrix.

```
[[-9.43106386e-09 -2.94874519e-05  2.25618574e-02]
 [ 2.90096517e-05 -5.28990505e-06  3.08267769e-01]
 [-2.22996236e-02 -3.05586398e-01  1.00000000e+00]]
```

Fundamental Matrix for Curule

```
[[ 5.09867994e-09 -1.59029910e-05  9.67205041e-03]
 [ 1.57096979e-05  6.14928822e-07  9.48687413e-02]
 [-9.52249133e-03 -9.68763872e-02  1.00000000e+00]]
```

Fundamental matrix for Pendulum

```
[[ 7.13141649e-09  7.48013485e-05 -5.20046368e-02]
 [-7.47389029e-05  1.73745020e-06  9.52294059e-01]
 [ 5.17174890e-02 -9.52131414e-01  1.00000000e+00]]
```

Fundamental matrix for octagon

C. Estimating Essential Matrix

11. We have got our Fundamental matrix from the previous step: the Essential matrix is calculated as where K is the camera intrinsic matrix.

$$\mathbf{E} = \mathbf{K}^{\mathbf{T}}\mathbf{F}\mathbf{K}$$

```
[[-3.44180082e-04 -1.51613483e-01  1.64109331e-02]
 [ 1.51323226e-01 -1.30024969e-02  9.88228431e-01]
 [-2.05902860e-02 -9.88178352e-01 -1.26603112e-02]]
```

Essential Matrix for curule

```
[[ 5.83889875e-04 -2.90026592e-01  1.09866418e-02]
 [ 2.90390683e-01  5.88521725e-03  9.56844110e-01]
 [-7.46263840e-03 -9.56957488e-01  5.30736148e-03]]
```

Essential Matrix for pendulum

```
[[ 3.49513041e-05  1.44720320e-01 -1.29332127e-02]
 [-1.44239676e-01  2.13696169e-03  9.89459911e-01]
 [ 1.23434773e-02 -9.89389442e-01  2.08904796e-03]]
```

Essential Matrix for octagon

12. Clearly, the essential matrix can be extracted from F and K. As in F matrix computation, the singular values of E are not necessarily (1,1,0) due to the noise in K. This can be corrected by reconstructing it with (1,1,0) singular values, i.e

$$. \mathbf{E} = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

D. Estimating Camera Poses

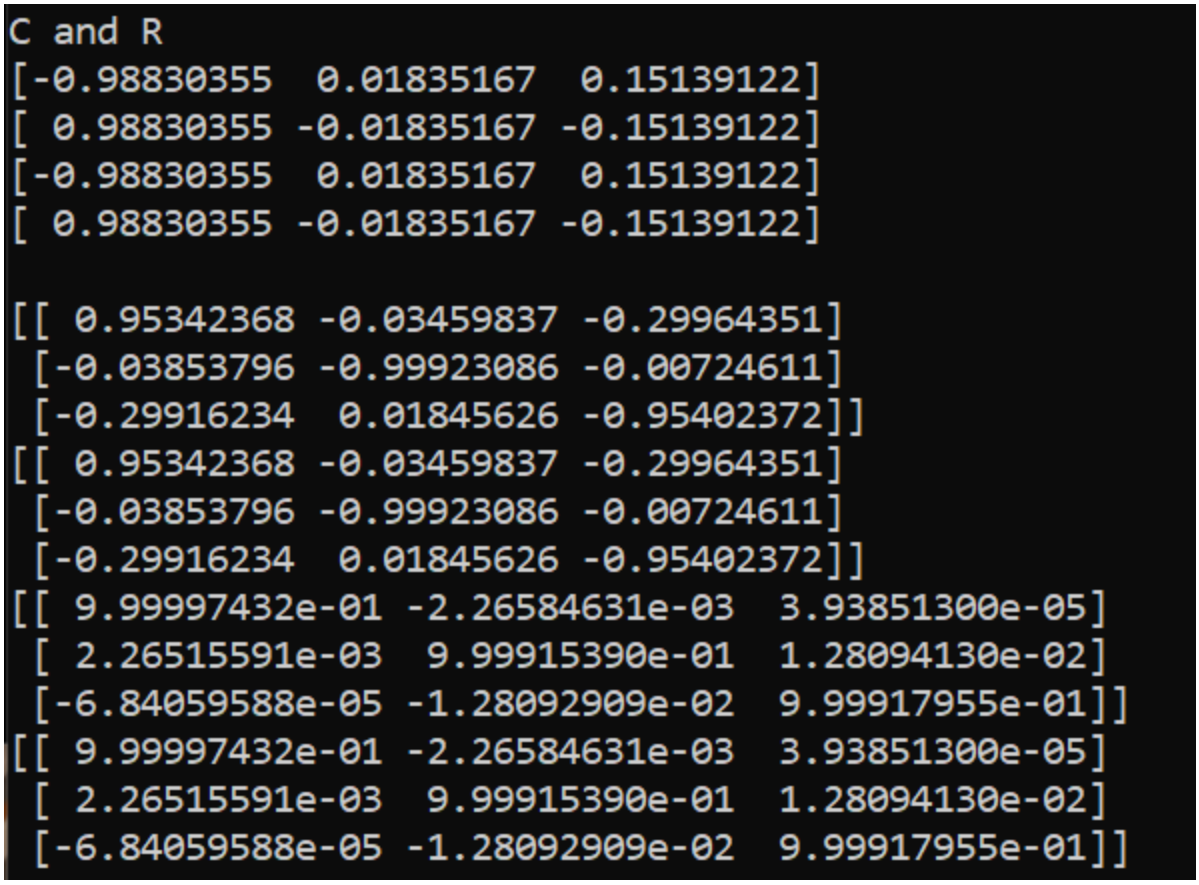
13. The camera pose consists of 6 degrees-of-freedom (DOF) Rotation (Roll, Pitch, Yaw) and Translation (X, Y, Z) of the camera with respect to the world. Since the E matrix is identified, the four camera pose configurations: (C1,R1),(C2,R2),(C3,R3) and (C4,R4)
    1. $C_1 = U(:,3)$ and $R_1 = UWV^T$
    2. $C_2 = -U(:,3)$ and $R_2 = UWV^T$
    3. $C_3 = U(:,3)$ and $R_3 = UW^TV^T$
    4. $C_4 = -U(:,3)$ and $R_4 = UW^TV^T$

    It is important to note that the $det(R) = 1$. **If** $det(R) = -1$, **the camera pose must be corrected *i.e.*** $C = -C$ **and** $R = -R$.

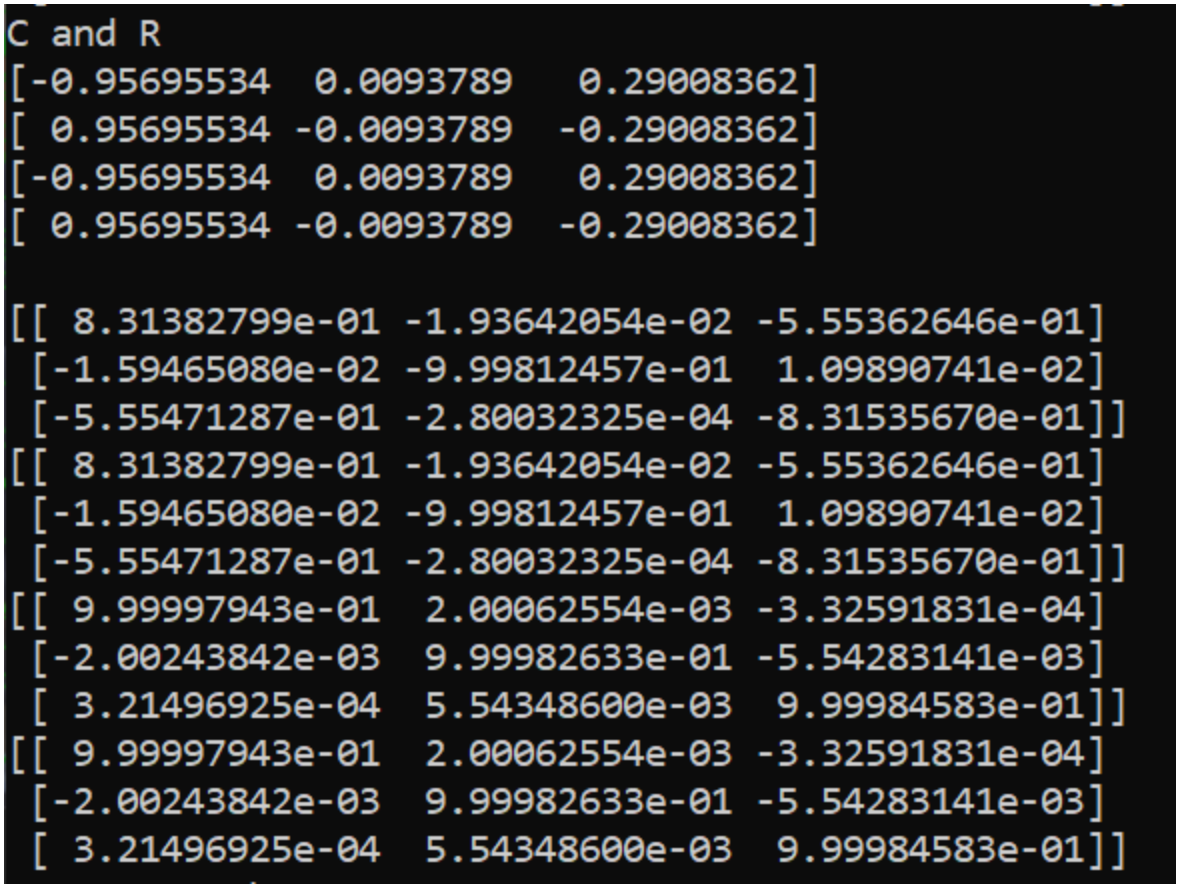
14. Once I got all the camera poses, I applied the cheirality check condition in which I triangulated the 3D points using linear least squares to check the sign of the depth Z in the camera coordinate system w.r.t. camera center. A 3D point X is in front of the camera if r3(X−C)>0 where r3 is the third row of the rotation matrix.

$$r_3(\mathbf{X} - \mathbf{C}) > 0$$

15. After the cheirality condition I got the most accurate camera pose.The cheirality condition was used in the code but it is commented as in rectification I am using **cv2.stereoRectifyUncalibrated** which does not require R and C.



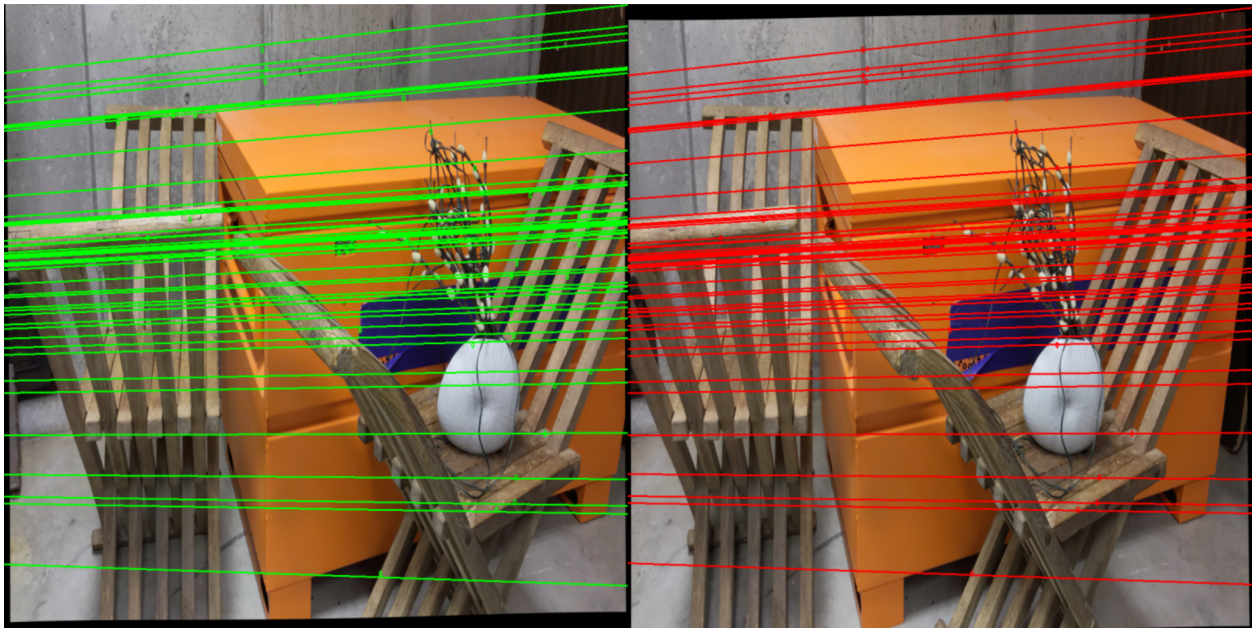
C and R Matrix for curule

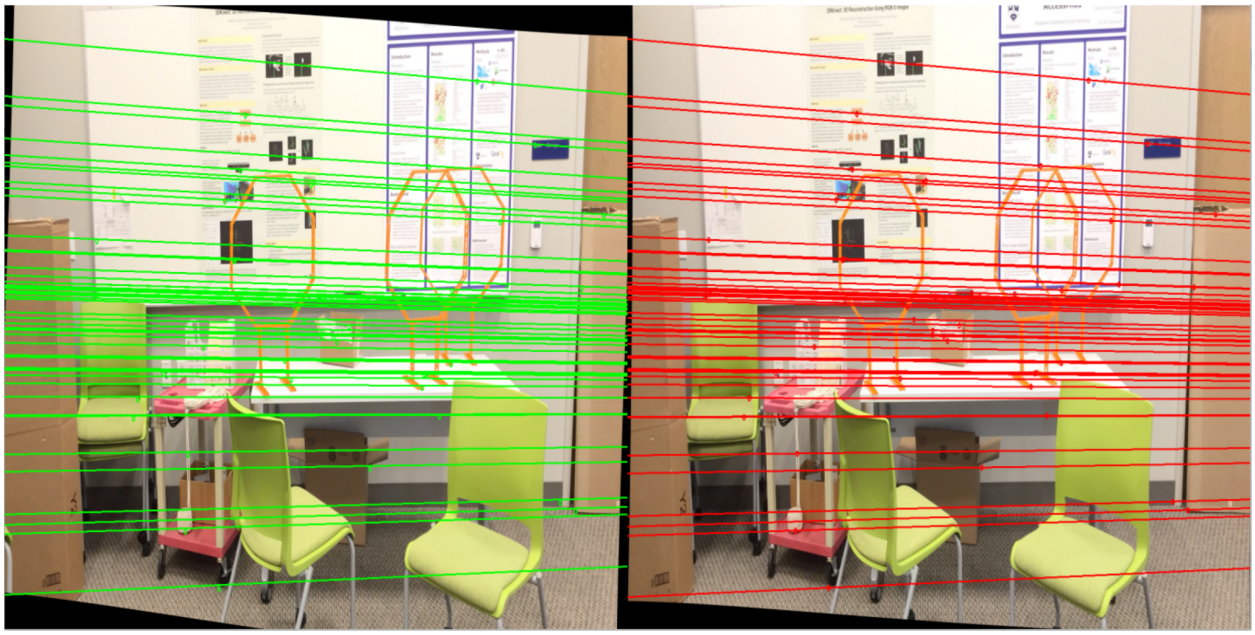


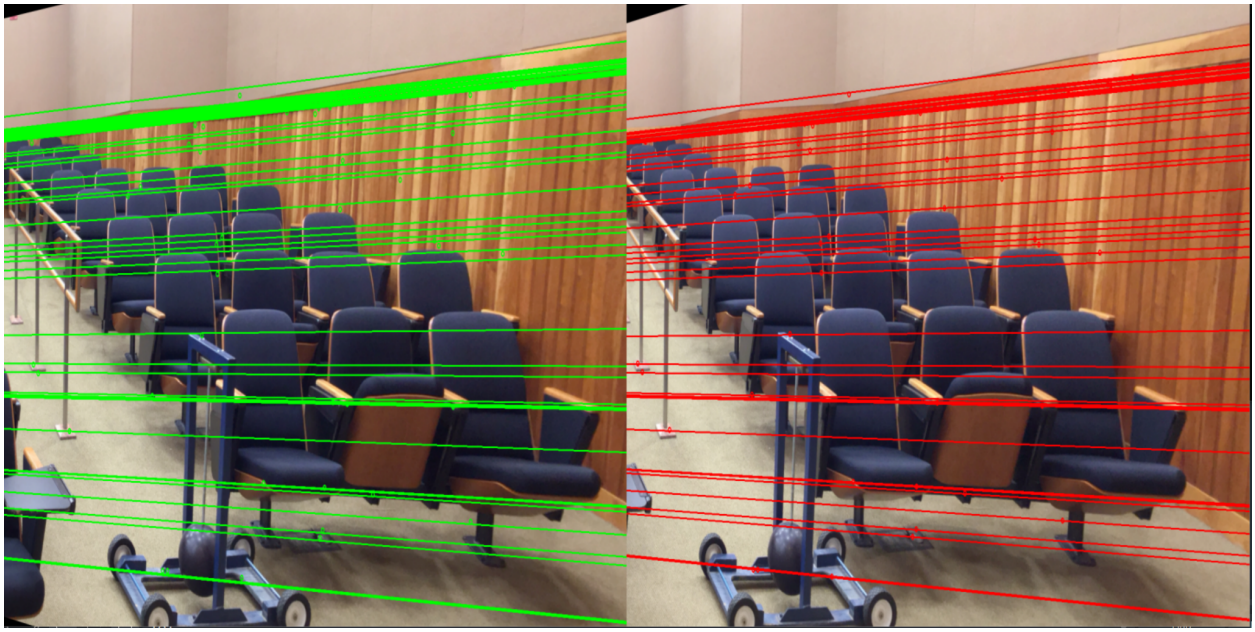
C and R Matrix for pendulum

E. Rectification

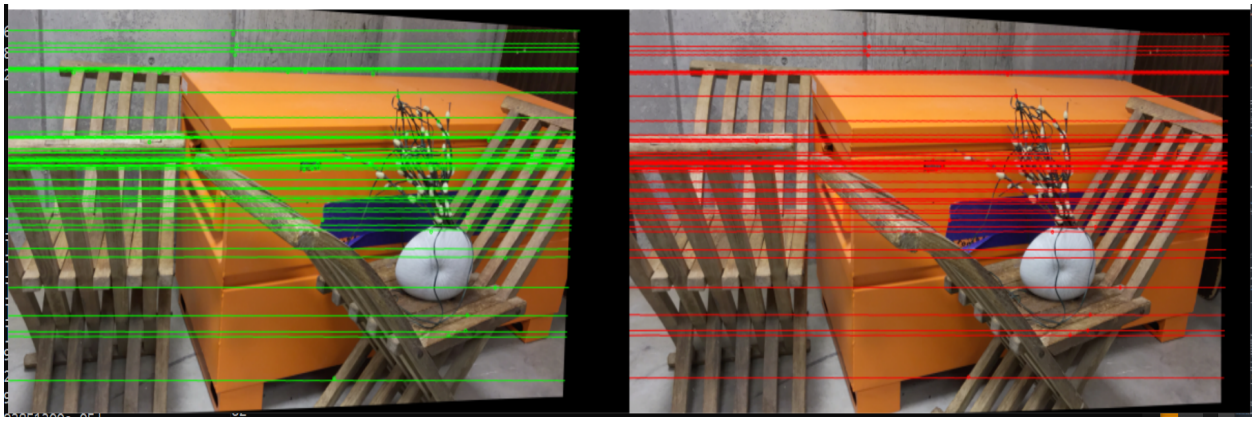
16. Using the fundamental matrix and the matched points in the left and right image I found out the direction of the epipolar line by multiplying the fundamental matrix with each matched point.
17. The epipolar lines that were found in the previous step need to be parallel to compute the depth otherwise the epipoles are not at infinity to make them at infinity we need to correct the image.For stereo systems the epipolar lines need to be parallel.
18. I have made use of OpenCV **cv2.stereoRectifyUncalibrated** to get the rectified image.The function returns the homography matrices for both the images which I used to warp the images to get the rectified image which will give me parallel epipolar lines.

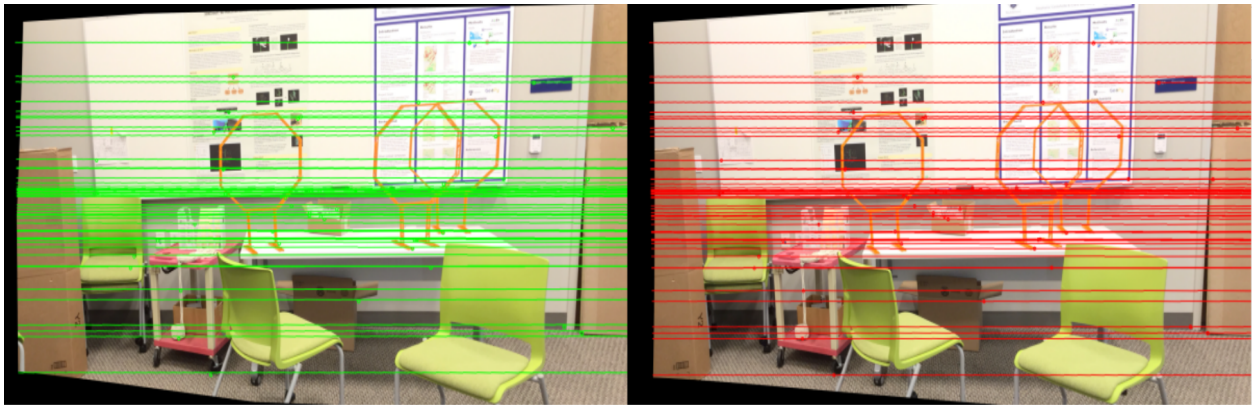






Epipole Lines in each image





Rectified Images with parallel epipole lines

```
---Homography---
[[ 1.18735341e+00 -8.71657309e-02 -1.93562689e+02]
 [ 7.22677230e-02  1.08077168e+00 -7.27424907e+01]
 [ 9.39695725e-05 -1.90130636e-05  1.00000000e+00]]
```

Homography for curule

```
[[ 8.48445433e-01  1.10589141e-02  6.65464797e+01]
 [-5.44895454e-02  9.23353574e-01  5.26511076e+01]
 [-7.86056346e-05 -1.02457145e-06  1.00000000e+00]]
```

Homography for octagon

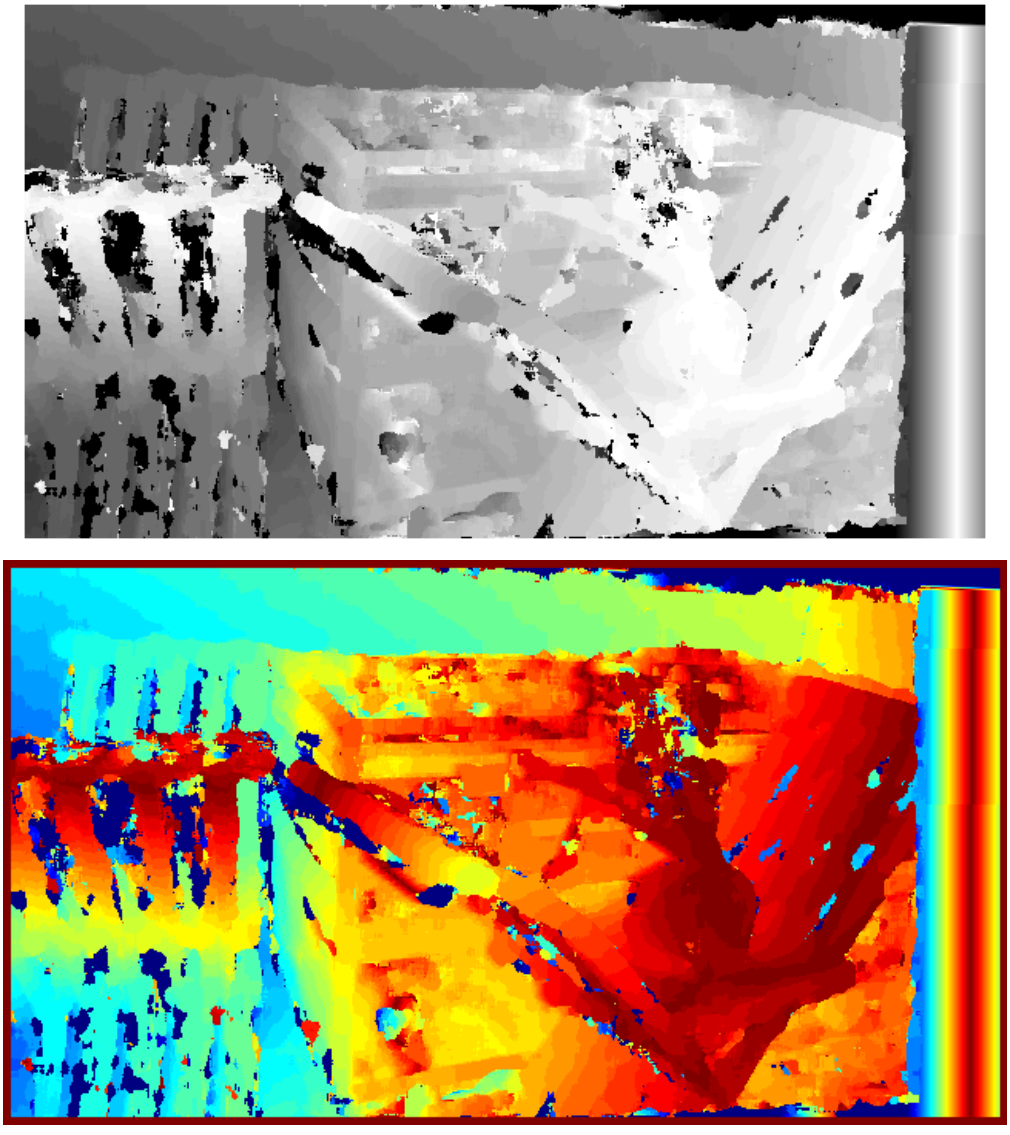
```
[[ 1.31518074e+00 -1.27478359e-02 -1.44790457e+02]
 [ 1.00118127e-01  1.15627062e+00 -9.56187233e+01]
 [ 1.64633757e-04 -1.59576859e-06  1.00000000e+00]]
```

Homography for pendulum

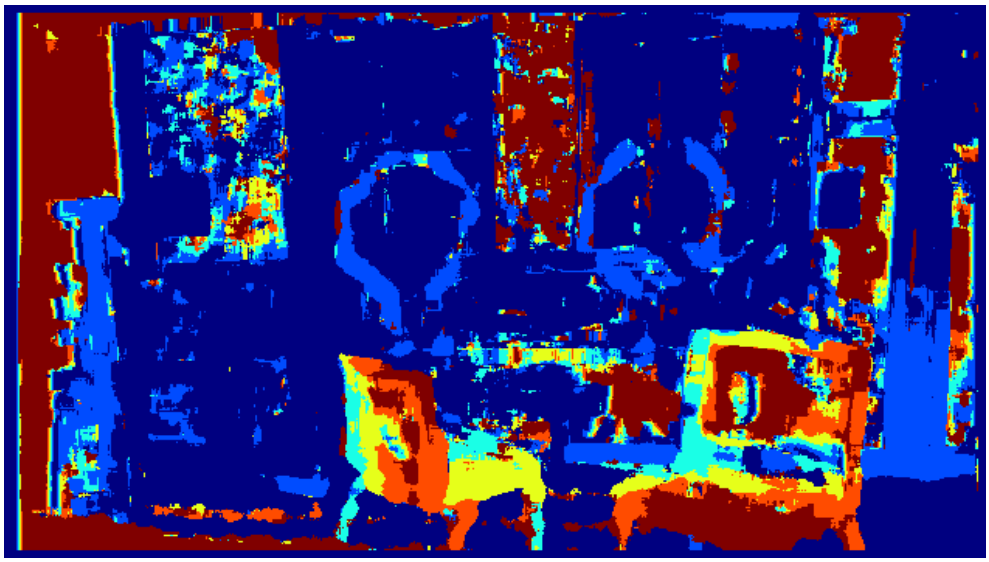
F. Correspondence

19. To get the disparity I created a window of size 11x11 and using this window I traversed this window over the entire image.
20. For each window I checked the corresponding line in the next image.The line in the next image was determined by checking that their y coordinates were the same.
21. I checked all the windows in the next image on the same parallel line of the previous image before and after the current window.I have defined a certain distance to check all the pixels before and after that point in the previous image.

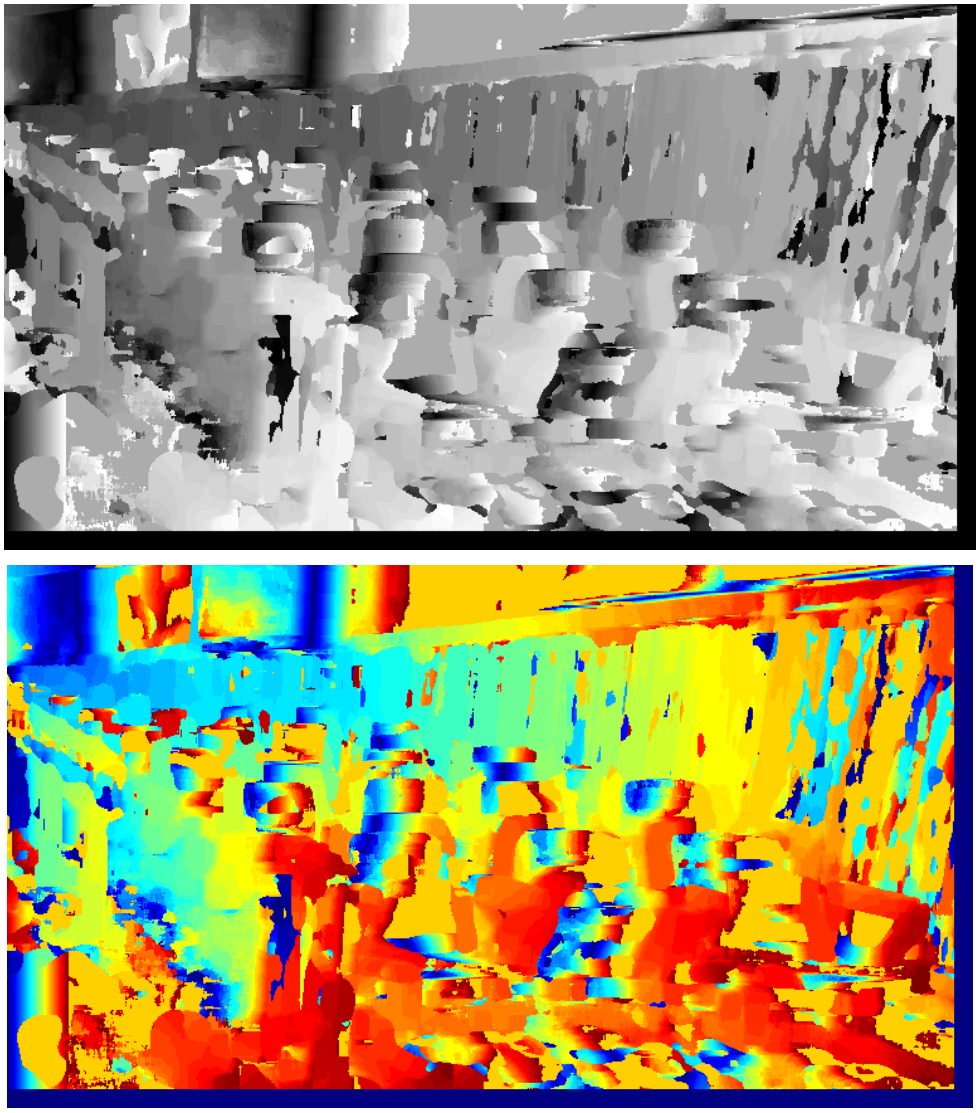
22. For every matched window in the next image I calculated SSD(Sum of square difference) with its corresponding image in the first image.For window which gives me minimum SSD I select that window and find the absolute difference between two windows.The difference between windows gives us the disparity for that pixel value.
23. I scaled down the disparity map between 0 to 255.I even normalized the image to get better results, while calculating disparities I got some disparity which were very large as compared to other disparities which altered my result to solve that I capped the disparities such that all disparities values at pixels above mean of disparity was made equal to mean of disparity.



Disparity Map curule Gray and Heat Map



Disparity Map for octagon Gray and Heat Map

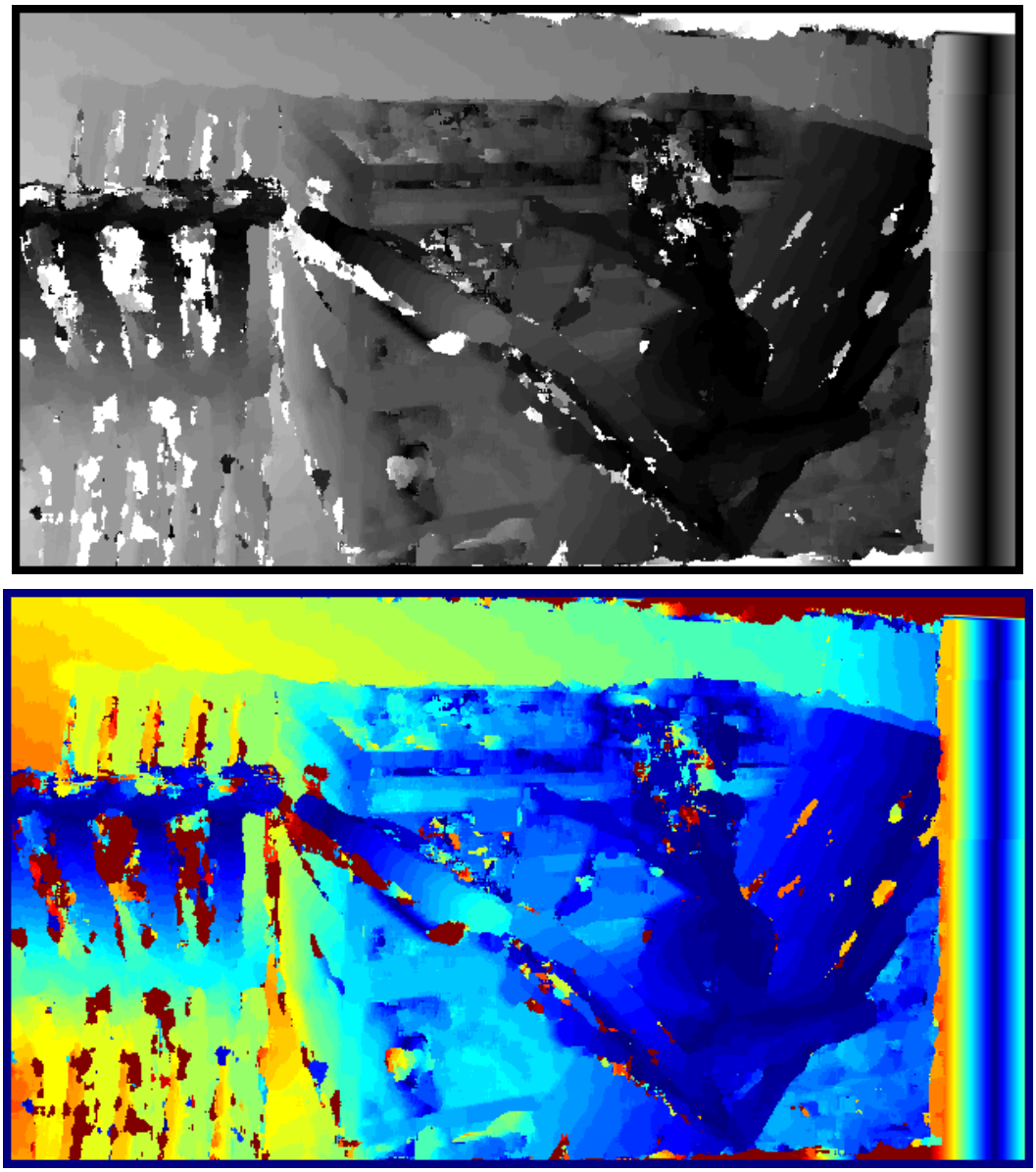


Disparity Map for pendulum Gray and Heat Map

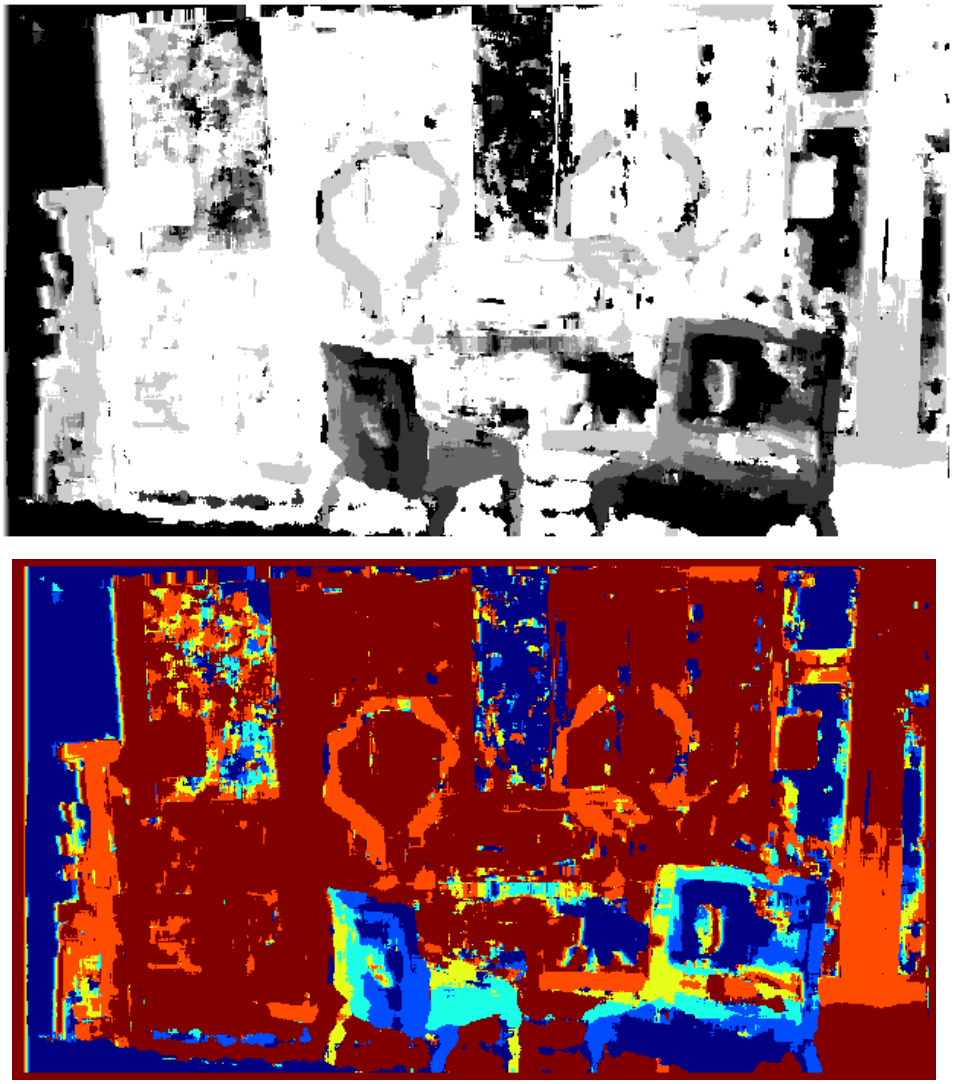
G. Depth Map

24. The depth map can be calculated with the help of a disparity map if we know the baseline and focal length.

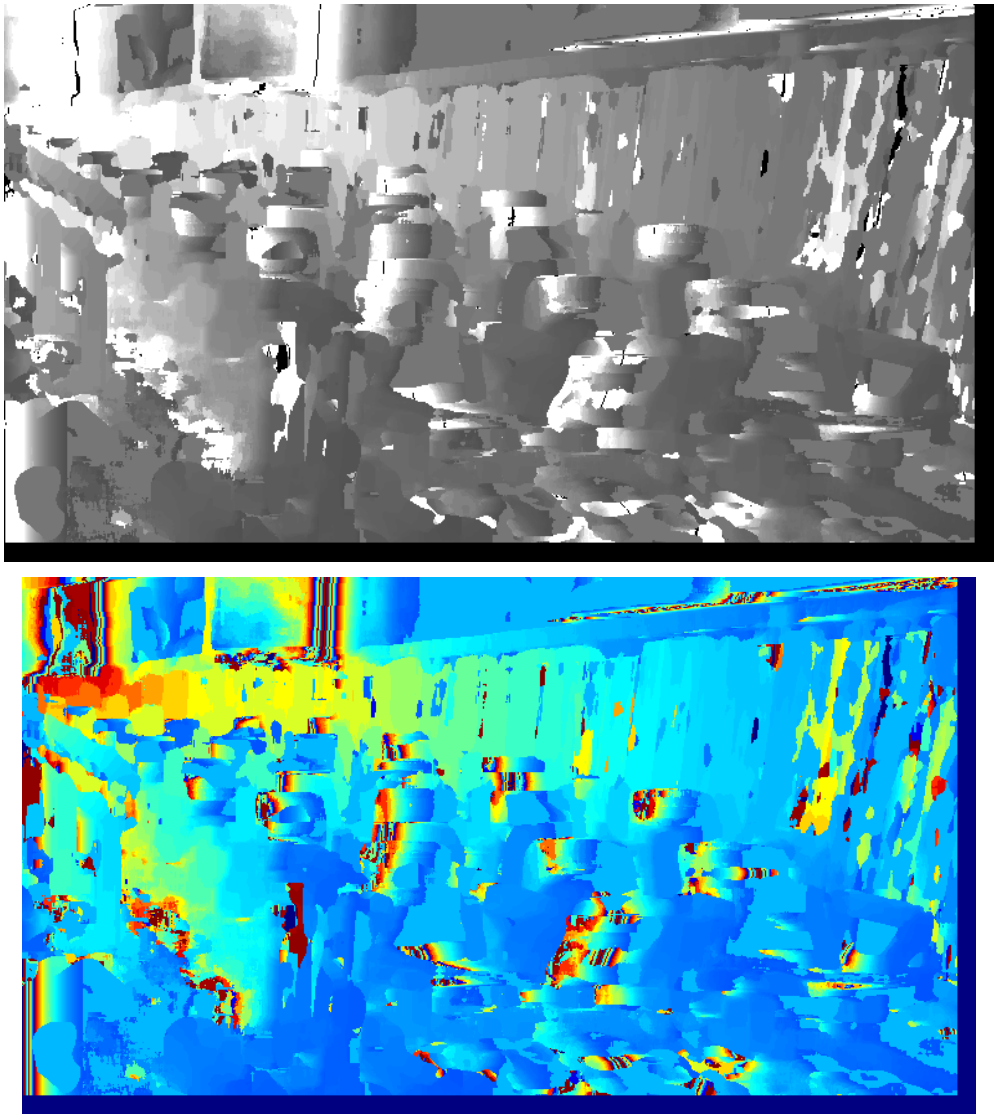
Depth = (Baseline * focal length) / disparity)



Depth Map Curule Gray and Heat Map



Depth Map Octagon Gray and Heat Map



Depth Map pendulum Gray and Heat Map