ENPM 673 :

Param Dave
Masters in Robotics
University of Maryland , College Park
Email: pdave1@umd.edu

I. Calibration

First, we need to compare the two images in each dataset and select a set of matching features. I used sift to select all the features and matched them using brute force method. After selecting those features and separating corresponding points in two lists I used 8-point algorithm to compute the Fundamental matrix. As we use 8 points out to image randomly, Fundamental Matrix is prone to errors.We put a rank = 2 constraint on fundamental matrix So we use ransac algorithm to calculate the Fundamental matrix. We select the fundamental matrix with the least epipolar constraint error as shown in equation 1. I normalized the points to get better results but everytime due to randomness in ransac , the result is different which causes some errors in warping.

$$x'^T F x = 0 \tag{1}$$

We select fundamental matrix for which maximum number of points satisfy the constraint where the error is less than 0.001. We use 2000 iterations. The Essential matrix is calculated using equation 2 where k_1 is camera internal matrix of left image and k_2 is camera internal matrix of right image.

$$E = k_2^T F k_1 \tag{2}$$

We extract the rotation and translation matrices by decomposing E and doing some calculations and explained in the reference article. Out of four possible values we extract only the one we need by using linear interpolation and condition where depth is positive.

II. Rectification

The estimated fundamental matrix from the previous step can use used to compute the epipolar lines in the image using equation 3.

$$Fx = l \tag{3}$$

The epipolar lines plotted in the image are shown in figures Having computed the pose of the second camera, I can utilise this pose to perform the rectification of the left and right images. I tried to use the cv2.stereoRectify() function to perform this task, but I was unable to successfully get the right rectified image, since it was not able to recover the rectified image with the correct scale, that is required for further processing. Hence, I computed the homography matrices H1 and H2 for the left and right camera images respectively . I used the inbuilt function cv2.stereoRectifyUncalib() which takes the estimated fundamental matrix and point correspondences as inputs. These homographies are shown in figure . These homographies can be used for further rectification. The epipolar lines are rectification and are shown in figure.

```
Value of F using custom algorithm method is: [[ 2.42359443e-09  1.12868032e-07  1.41017779e-04]
 [-9.06671709e-08 -2.50159736e-08  1.99269613e-03]
 [-1.55512572e-04 -2.02374471e-03 -1.21314783e-02]]
Value of F using inbuilt method is:  [[-6.55241130e-09  1.70198010e-05 -7.86151641e-03]
 [-1.60688832e-05 -1.87581139e-06  2.73378504e-01]
 [ 7.52745952e-03 -2.74644254e-01  1.00000000e+00]]
E forcurule is : [[ 0.00201612  0.10232912  0.10828532]
 [-0.08301057 -0.01799375  0.99068288]
 [-0.10389034 -0.98898913 -0.01546564]]
H1 forcurule is : [[ 1.89094815e-03  1.57897489e-04  3.60394507e-01]
 [ 1.61915704e-04  2.02596801e-03 -1.27806798e-01]
 [-9.47626197e-08 -1.35838377e-08  2.10445899e-03]]
H2 forcurule is : [[ 9.37488521e-01 -1.00140435e-01  1.14086855e+02]
 [ 7.42327032e-02  9.97759465e-01 -7.00535061e+01]
 [-5.92237727e-05  6.32615145e-06  1.05343870e+00]]
```

Fig. 1: Matrices of dataset 1

```
Value of F using custom algorithm method is: [[ 4.23067918e-11  7.53071829e-08 -3.43931542e-05]
 [-7.45271007e-08  3.30315123e-09 -1.70954805e-03]
 [ 3.51216220e-05  1.69916328e-03  7.01742952e-03]]
Value of F using inbuilt method is:  [[-7.75272161e-07  6.15081083e-03 -2.71929080e+00]
 [-6.18068358e-03 -4.23882185e-05  2.49084072e+01]
 [ 2.75104939e+00 -2.50321517e+01  1.00000000e+00]]
E foroctagon is : [[ 4.66028149e-05  7.42730078e-02  3.68890415e-03]
 [-7.32451440e-02  2.18843388e-03 -9.97305320e-01]
 [-2.85474877e-03  9.97230996e-01  2.13504182e-03]]
H1 foroctagon is : [[ 1.77024101e-03  1.62038293e-05 -6.63360579e-03]
 [ 3.37350485e-05  1.69970008e-03 -2.50118513e-02]
 [ 7.15021874e-08 -2.91878110e-09  1.64004321e-03]]
H2 foroctagon is : [[ 1.04076961e+00  3.60550900e-03 -4.10857967e+01]
 [ 1.94720275e-02  1.00007346e+00 -1.87328132e+01]
 [ 4.24745902e-05  1.47143533e-07  9.59144936e-01]]
```

Fig. 2: Matrices of dataset 2

III. Correspondence

Using the rectified left and right images, for each corresponding epipolar line e_1 , e_2 I performed block matching between left and right images with the epipolar line as my search space. But i got black patches as the sift points are not uniformly distributed thus the method fails. Generally the block matching operations yields the disparity map which is $|x_l$ - $x_r|$ where for a point (x_l , y_l) in left image x_r is x coordinate of same point in right image which has an inversely proportional relationship with the depth map to be estimated.

```
Value of F using custom algorithm method is: [[-6.06835171e-12 -3.10765107e-07  1.44585927e-04]
 [ 3.13919751e-07 -2.00326309e-08 -2.06824932e-03]
 [-1.45756637e-04  2.13794237e-03 -2.97794692e-02]]
Value of F using inbuilt method is:  [[-6.40419364e-08  1.85334998e-05 -3.80021422e-02]
 [-2.01985046e-05  1.83585983e-06 -4.68345979e-01]
 [ 4.01936819e-02  4.68695513e-01  1.00000000e+00]]
E forpendulum is : [[-3.87817724e-04 -2.33142736e-01 -1.01771572e-02]
 [ 2.40259951e-01 -7.37781417e-03 -9.70608436e-01]
 [ 1.38149242e-02  9.72339173e-01 -6.81232607e-03]]
H1 forpendulum is : [[-1.75764854e-03 -1.71304014e-04 -3.50047035e-01]
 [ 1.69880307e-04 -2.14670300e-03 -1.25799705e-01]
 [ 3.65865806e-07 -2.79939041e-08 -2.40833294e-03]]
H2 forpendulum is : [[ 8.36947288e-01 -1.07368047e-02  1.62328478e+02]
 [-7.88433943e-02  1.00109373e+00  7.50990458e+01]
 [-1.69760871e-04  2.17778269e-06  1.16179443e+00]]
```

Fig. 3: Matrices of dataset 3

Instead of searching through the entire width of the right image, I limited my searching range in the right image, to a certain number of windows, with respect to the location x_l. A window of nxn is slided in right image relative to position of point in left image. sum of squared difference(SSD) is calculated and the point with minimum SSD is taken as point. This drastically reduced the computational time, by avoiding redundant SSD computations. But it gives some error related to calculating depth where same color is present in a uniform area aroud our selected point. SSD fails to give correct depth for walls and other big object with only one color.

IV. Compute Depth Image

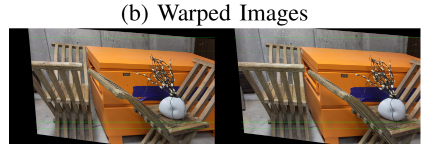
Depth is calculated using following equation

$$depth = \frac{baseline * f}{disparity} \tag{4}$$

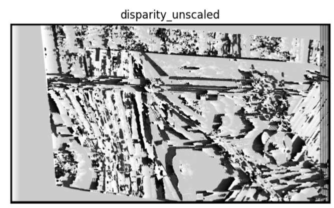
The parameters baseline and f are already given. The depth maps computed are shown in next section. For dataset 1

(a) Unrectified Epilines

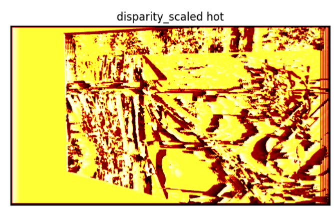
(b) Warped Images



(c) Rectified Epilines

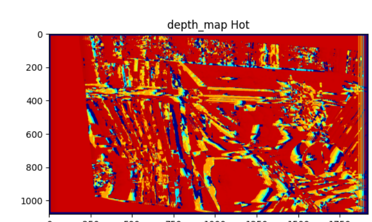


(d) Disparity Grayscale



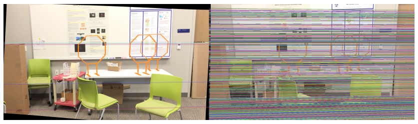
(e) Disparity HeatMap

(f) Depth Map Grayscale

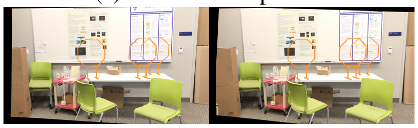


(g) Depth Map HeatMap

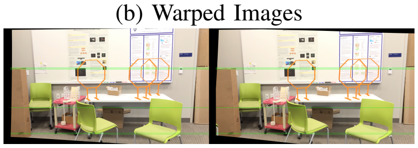
Fig. 4: Output of dataset 1



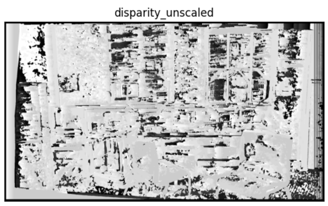
(a) Unrectified Epilines



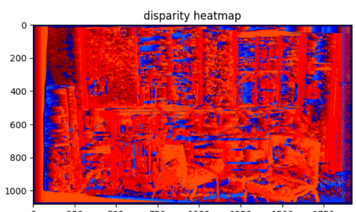
(b) Warped Images



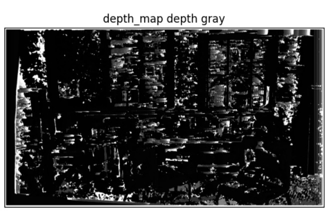
(c) Rectified Epilines



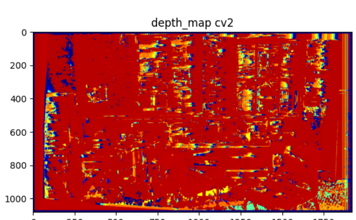
(d) Disparity Grayscale



(e) Disparity HeatMap



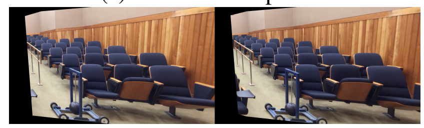
(f) Depth Map Grayscale



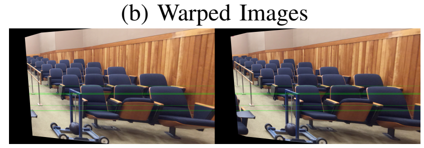
(g) Depth Map HeatMap

Fig. 5: Output of dataset 2

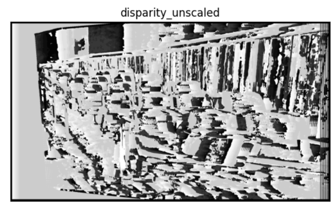
(a) Unrectified Epilines



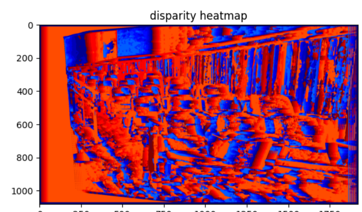
(b) Warped Images



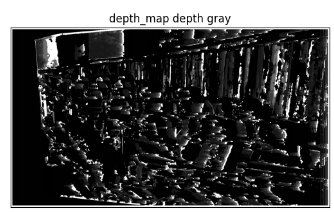
(c) Rectified Epilines



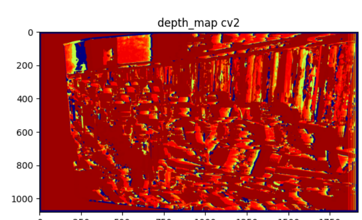
(d) Disparity Grayscale



(e) Disparity HeatMap



(f) Depth Map Grayscale



(g) Depth Map HeatMap

Fig. 6: Output of dataset 3