Project 3

Gokul Hari
Directory ID: hgokul@umd.edu

I. Stereo Vision

The aim of the project is to build a pipeline for creating a Stereo Vision System to estimate the depth map of a given stereo image pair I and I' and their corresponding camera matrices K and K' . The following steps were followed to build the pipeline for depth map estimation.

Calibration

1) Feature Matching: For the given two images in a dataset, I computed the SIFT feature for both the images and performed a FLANN based feature matching using (K-nearest neighbors) KNN. The Lowe's distance for selecting high accuracy matches was chosen to be 0.7

2) Fundamental Matrix: Having obtained the feature correspondences, I need to compute the Fundamental matrix that describes the relationship between the feature correspondences of these stereo images. I estimated the Fundamental matrix using the normalized 8-point Hartley algorithm. I first normalised the 8 chosen point correspondences x, x' using the method suggested in [**?**], and retained the respective normalization matrices T, T'. I built a 8×9 linear solver matrix A, by computing the kroenker product between each point correspondences and stacking them vertically. This linear solver matrix A is

$$\begin{bmatrix} u_1'u_1 & u_1'v_1 & u_1' & v_1'u_1 & v_1'v_1 & v_1' & u_1 & v_1 & 1 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ u_n'u_n & u_n'v_n & u_n' & v_n'u_n & v_n'v_n & v_n' & u_n & v_n & 1 \end{bmatrix}$$

where $x = (u_1, v_1, 1)$ and $x' = (u_1', v_1', 1)$,the homogenous point correspondences. The solution for the fundamental matrix is computed by solving the homogenous sytem of equations given by

$$A.f = 0 \tag{1}$$

where f is the 9×1 vector which is later reshaped to a 3×3 fundamental matrix F'. The solution for F can be obtained using singular value decomposition and choosing the right singular vector of the last column The rank of the F matrix must be 2, but due to practical errors, generally due to mismatched point correspondences, the rank can be 3. Thus we enforce the rank as two by decomposing the F matrix using svd as USV^T. The last element of the diagonal matrix S is equated to zero and the F matrix is reconstructed. To undo the effect of normalizing the points, I computed F = $T'^T FT$.

This estimate of F is highly erroneous and thus I performed RANSAC using the epipolar constraint to to obtain a better estimate of F, by choosing a epipolar constraint error threshold at 0.002.

The fundamental matrix of all 3 datasets are given in image 1, 2, 3

3) Essential Matrix: Having obtained F, I computed the essential matrix E by equation

$$E = K'^T FK \tag{2}$$

4) Obtaining Camera Pose: The essential matrix E can be decomposed to 4 mathematically possible rotation and translation matrices, out of which only one of them is physically possible. We need to extract this physically possible camera pose.

Given the essential matrix E, I decomposed it to obtain R1, R2 and t, given as follows.

- decompose using SVD as $E = UDV^T$
- $R1 = UWV^T$ and $R2 = UW^TV^T$, where

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

- $t = U[:, 3]$ ie the third column of U

The 4 relative poses are (R1, t), (R1, -t), (R2, t), (R2, -t), with respect to the left camera which is registered to be at the reference pose $R = I_{3\times3}$, $t = [0, 0, 0]_{3\times1}$. Out of these 4 poses, only one of them is physically possible since, in the other 3 poses, the rays emanating from the points in image plane do not intersect, *in front of the camera*. Thus we perform the chierality check ie impose a depth postivity constraint, to get the only correct pose. This can be done by recovering the 3D point using linear triangulation and checking if the Z component of the 3D point is positive. The pose which has most positive values will be considered as the correct pose.

In this section, we have computed the R, T matrices which describes the relative pose of the right camera with respect to the left camera registered at reference pose. The estimated matrices are given in figures 1, 2, 3

Rectification

The estimated fundamental matrix from the previous step can use used to compute the epipolar lines in the image. The epipolar lines plotted in the image are shown in figure 6, 7, 8

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

```
F matrix  1

[[-1.64968287e-10 -9.73135072e-08  4.14839256e-06]
 [ 1.17415164e-07  1.30515429e-08  4.09598417e-03]
 [-1.18388101e-05 -4.11905763e-03  6.13443570e-03]]


E matrix  1

[[-5.61311953e-05 -1.20359267e-01 -2.02784182e-02]
 [ 1.44766623e-01  8.40836537e-03  9.89202130e-01]
 [ 2.51169766e-02 -9.92405540e-01  7.34038864e-03]]


Rotation matrix  1

[[ 0.99968486  0.00371665 -0.02482673]
 [-0.00391454  0.99996092 -0.00792694]
 [ 0.0247963   0.00802162  0.99966034]]

Translation matrix  1

[ 0.99252326  0.0212384  -0.1201936 ]
```

Fig. 1. Fundamental, Essential and Extrinsics of Dataset 1

```
F matrix  2

[[ 2.55310303e-12 -4.30377302e-09 -1.06302722e-06]
 [ 1.02870021e-08  4.59732097e-09 -3.48830633e-03]
 [ 4.68113617e-07  3.48100325e-03  7.71839812e-04]]


E matrix  2

[[ 3.46192643e-06 -5.44347406e-03 -1.54796209e-03]
 [ 1.30256248e-02  3.03279493e-03 -9.99909391e-01]
 [ 3.02585367e-03  9.99975886e-01  3.06406985e-03]]


Rotation matrix  2

[[ 0.99997017 -0.00151762  0.00757316]
 [ 0.00149454  0.99999422  0.00305252]
 [-0.00757775 -0.00304111  0.99996666]]

Translation matrix  2

[ 0.99998399 -0.00153138  0.00544816]
```

Fig. 2. Fundamental, Essential and Extrinsics of Dataset 2

```
F matrix  3

[[-4.00126105e-11 -1.67313688e-07  4.60022409e-05]
 [ 1.70364260e-07 -3.25076772e-08 -2.67628043e-03]
 [-4.62372867e-05  2.69289513e-03 -1.50921842e-03]]


E matrix  3

[[-0.0052991  -0.37271632 -0.03327967]
 [ 0.37325332 -0.03693938 -0.92585469]
 [ 0.05939986  0.92603528 -0.03142886]]


Rotation matrix  3

[[ 0.99989252 -0.01444171  0.00252794]
 [ 0.01451938  0.99931579 -0.03401689]
 [-0.00203495  0.03404994  0.99941806]]

Translation matrix  3

[ 0.92733323 -0.04594049  0.37140617]
```

Fig. 3. Fundamental, Essential and Extrinsics of Dataset 3

```
Homography matrix left image  1

[[ 3.51271192e-03  8.58694699e-05 -1.58860761e-01]
 [-1.40369274e-05  3.44664070e-03  7.15015762e-03]
 [ 6.46763850e-08 -7.90030911e-09  3.42176203e-03]]

Homography matrix right image  1

[[ 1.00768282e+00  9.70519530e-03 -6.36316563e+00]
 [-4.43770207e-03  1.00000364e+00  1.98699533e+00]
 [ 1.72526638e-05  1.66163865e-07  9.92220791e-01]]


Homography matrix left image  2

[[ 2.96671669e-03 -3.04047918e-04 -9.70816157e-02]
 [-3.77098264e-05  2.89204345e-03  1.66634082e-02]
 [-8.91562817e-08  9.99970064e-09  2.92677076e-03]]

Homography matrix right image  2

[[ 9.88113446e-01  3.71246032e-03  4.03239073e+00]
 [-1.19242468e-02  9.99962257e-01  5.16248423e+00]
 [-2.74988342e-05 -1.03316407e-07  1.01191018e+00]]


Homography matrix left image  3

[[-2.69515462e-03 -7.16375650e-05 -2.42354595e-03]
 [ 1.70313689e-04 -2.99162779e-03 -8.42710368e-02]
 [ 5.32322966e-07  2.54810876e-08 -3.26221264e-03]]

Homography matrix right image  3

[[ 9.21342245e-01  2.36146051e-03  3.42108823e+01]
 [-5.60622692e-02  9.99859593e-01  2.49340503e+01]
 [-1.77149709e-04 -4.54046306e-07  1.07879159e+00]]
```

Fig. 4. Homographies of all three datasets

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 using the function cv2.stereoRectifyUncalib() which intakes only the estimated fundamental matrix and point correspondences. These homographies are shown in figure 4. These homographies can be used for rectification. The epipolar lines after rectification are shown in figure 6, 7, 8

Correspondence

Using the rectified left and right images, for each corresponding epipolar line e_l, e_r I performed block matching between left and right images with the epipolar line as my search space. The block matching operations yields the disparity map which has an inversely proportional relationship with the depth map to be estimated

***Block matching**:* To perform block matching in a epipolar line as the search space, I performed the following steps. First, choose a 11×11 pixel window w_l along image coordinate index x_l in the epipolar line e_l in the left image. Next, for every non-overlapping 11×11 window w_r, scanned along the corresponding epipolar line e_r in the right image, I computed the sum of squared distance (SSD) between w_r and w_l. and chose the image coordinate index x_r in the right image, along e_r, which has the least SSD value. The matched coordinates x_r and x_l can be considered as the same real world region, projected in locations x_r and x_l in the right and left images. The disparity d is calculated as $|x_l - x_r|$ and recorded as an entry to the disparity map. I repeated this procedure of

computing disparity between all the windows scanned along epipolar lines e_l with their corresponding epipolar lines e_r in the right image to fill up the disparity map.

I noticed that there were black line regions where the epipolar lines are not available, since the sift features were not detected anywhere these regions. Thus, this method of scanning horizontally along only the available epipolar lines is bound to introduce irregular heatmaps. To account for this irregularity, I performed the block matching along every pixel along the left and right rectified images and constructed my disparity map. This results are the final disparity maps and are shown in results section

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 of window w_l. This search range was passed as a parameter to my block matching function. This drastically reduced the computational time, by avoiding redundant SSD computations. However, this approach seems to create a block of undefined disparity values in the right most edge of the image. I was unable to find a solution to this due to lack of time

Compute depth maps

The depth map is given by the formula

$$z = \frac{baseline * f}{d} \tag{4}$$

where d is the disparity. The parameters baseline and f are already given. The depth maps computed are shown in next section.

II. Results

The feature matchings, rectified images, disparity and depth images of all three datasets are given in this section

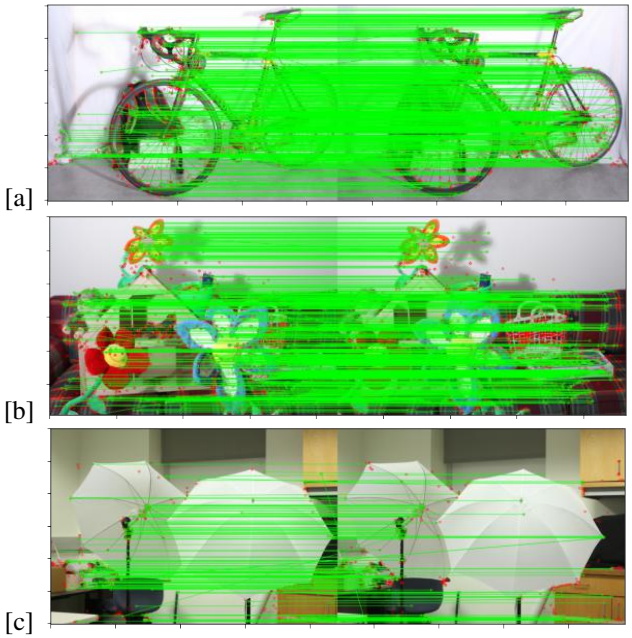

Fig. 5. all Sift matches of images 1 2 and 3,

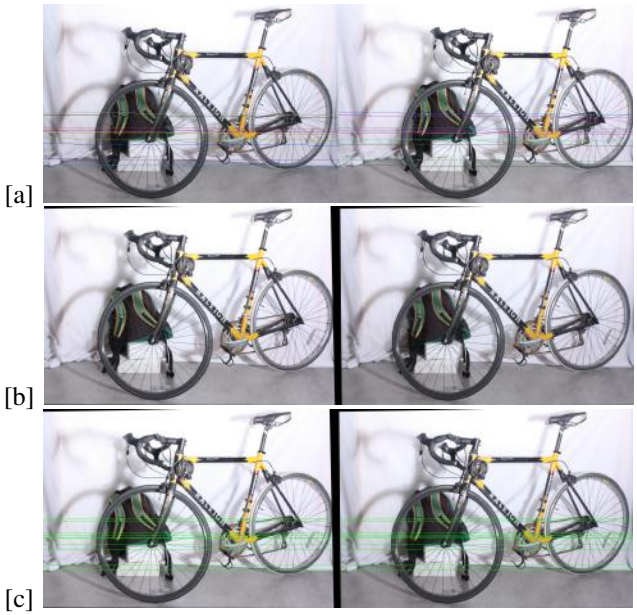

Fig. 6. dataset 1 - a) epipolar lines (only 10) b) recitified image c) epipolar lines after rectification

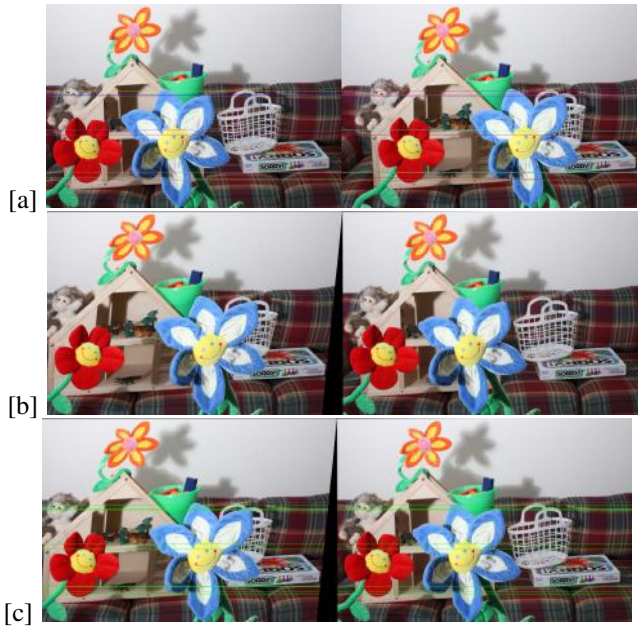

Fig. 7. dataset 2 - a) epipolar lines (only 10) b) recitified image c) epipolar lines after rectification

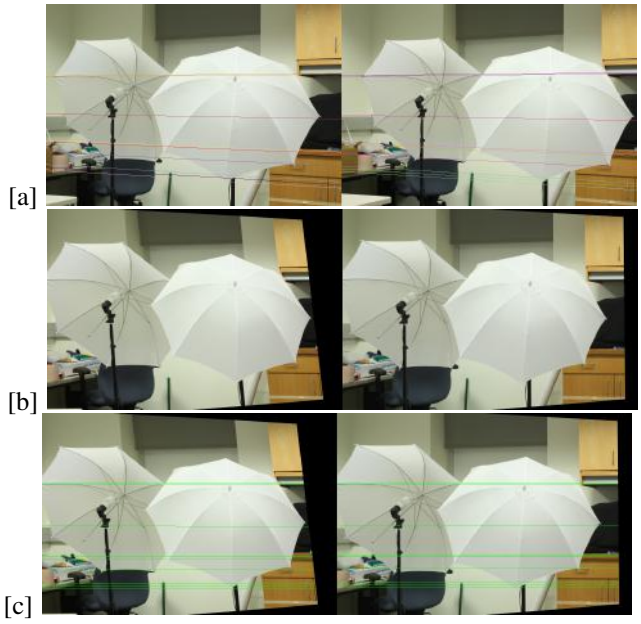

Fig. 8. dataset 1 - a) epipolar lines (only 10) b) recitified image c) epipolar lines after rectification

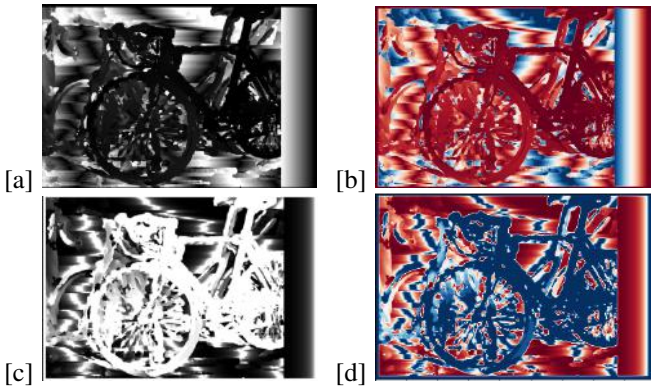

Fig. 9. dataset 1 - a) and b) depth maps in gray and red-blue heatmaps; c) and d) disparity map in gray and red-blue heatmaps

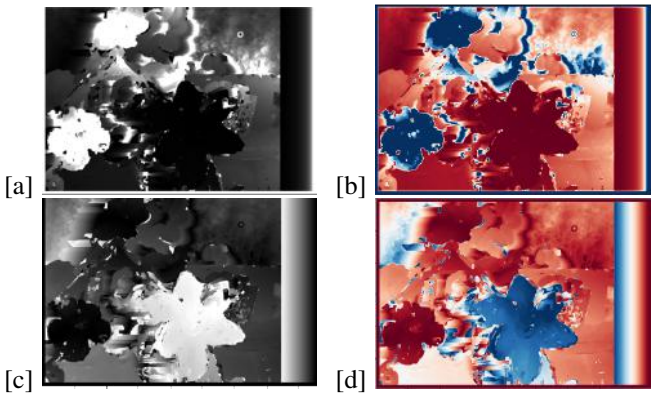

Fig. 10. dataset 2 - a) and b) depth maps in gray and red-blue heatmaps; c) and d) disparity map in gray and red-blue heatmaps

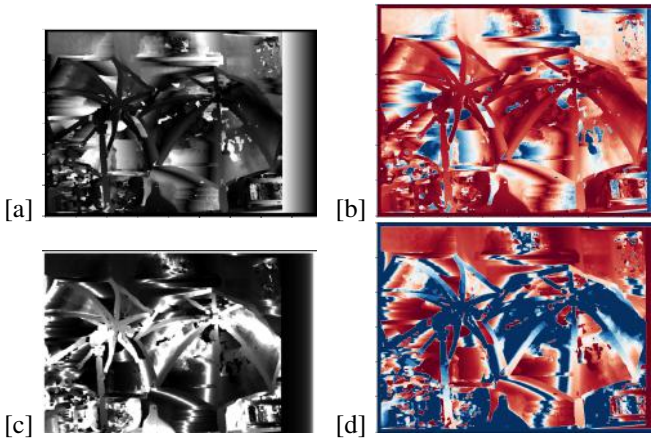

Fig. 11. dataset 3 - a) and b) depth maps in gray and red-blue heatmaps; c) and d) disparity map in gray and red-blue heatmaps