CV Home Work-1 Summary Report – Stereo Reconstruction

Manideep Cherukuri
Cheru050@umn.edu

The assignment's goal is to determine how to recreate an image using stereo images. A stereo reconstructed image has a variety of advantages. Stereo reconstruction provides accurate depth information of the scene, which is useful in various applications such as robotics, autonomous driving, and 3D modeling. Stereo reconstruction systems are also low-cost compared to other depth estimation techniques such as LiDAR. The reconstructed image will be significantly bigger than the individual stereo images because stereo rebuilt images have a larger field of view.

Given the below 2 views of the scene, we are tasked to implement a stereo reconstruction algorithm.

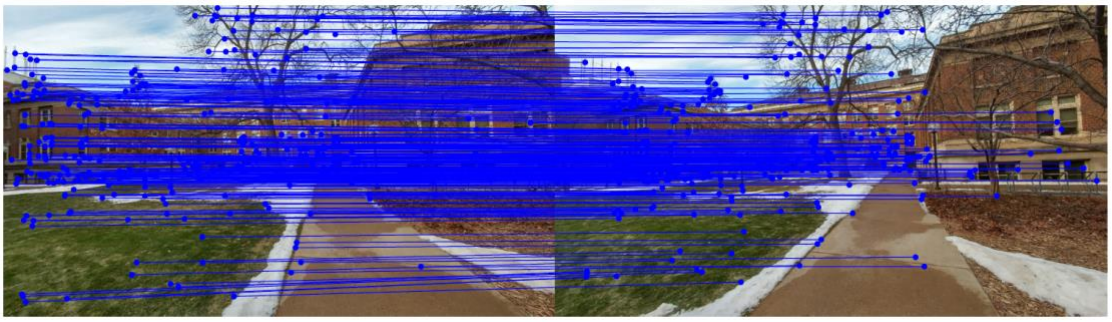
(a) Left image

(b) Right image

The corresponding points in both the stereo images is given in the correspondence.npz file. SIFT methods are typically used to find these points. It is a popular feature extraction approach that is frequently used in computer vision and image processing. It is used to find and extract recognizable details from photos that are independent of scale, rotation, and translation. A sample of the 316 correspondence points for both the stereo pictures can be viewed below.

```
pts1 shape :  (316, 2)        pts2 shape : (316, 2)
pts1 :                        pts2 :
[[ 11.49853  154.70865 ]      [[100.93673  174.15067 ]
 [ 14.190163 267.77185 ]       [104.90245  271.23688 ]
 [ 16.436968 163.63779 ]       [104.61186  181.42097 ]
 [ 19.189672 106.980644]       [109.80362  131.12547 ]
```

Data points



The correspondenses between the two images is shown by the blue line above.

Step 1: Given that the camera points are calibrated, our objective is to locate the 3D points of the reconstructed image using these inputs after we obtain the image points on the projected camera planes. We need to calculate the fundamental matrix to provide the relationship of the epipolar geometry existent between the two views after first determining the relation of rotation and translation between the views of the two cameras. It describes the mapping between a point in one image to its corresponding epipolar line in the other image.

The eight-point algorithm is used to find the fundamental matrix using an iterative algorithm such as RANSAC to remove the outliers and get the best F matrix. We first construct the matrix A using pts1 and pts2. Then we do the SVD of A to extract the smallest singular value from the right singular vector to solve this set of linear equation. Later, we enforce the rank 2 constraint. Upon doing the above steps, we get the following F matrix and the epipolar lines as shown below.

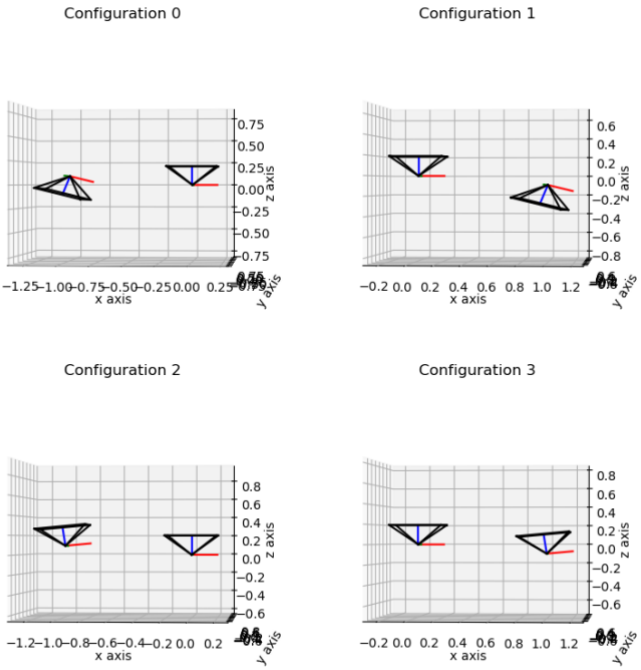
```
F matrix is:
[[ 2.41933079e-07  1.07446987e-05 -4.10709815e-03]
 [-4.84042353e-06 -1.23425336e-06 -1.52358218e-02]
 [ 2.04896740e-03  1.24802649e-02  9.99795502e-01]]
```

F matrix

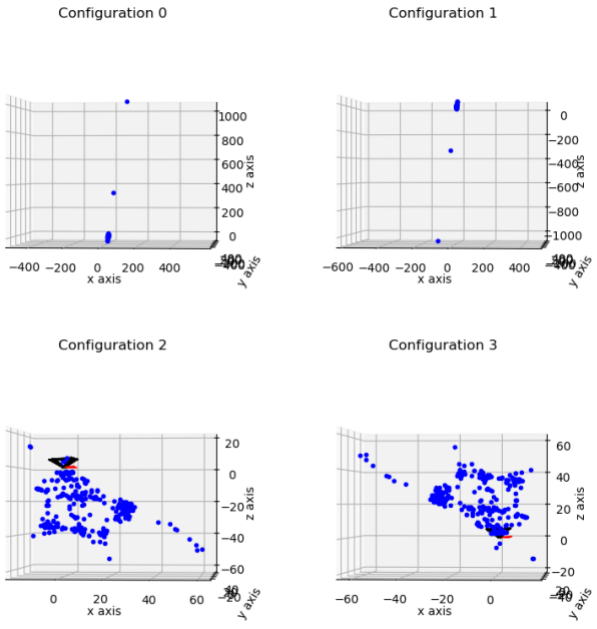
Epipolar lines observed between the corresponding points.

Step 2: Post this, the essential matrix is computed. The fundamental matrix is a mathematical representation that shows the relative orientation and position of two calibrated cameras seeing the same image. To determine the relative pose between the two cameras, the fundamental matrix which is a 3x3 matrix that links the coordinates of the given views in two cameras

perspective. We have four poses following the breakdown of the essential matrix. The 4 poses are (R_1, t) , (R_1, -t) , (R_2, t) , (R_2, -t) as shown below.



Step 3: Pose disambiguation is used to determine the ideal camera pose, and the one with the greatest number of triangulated points in front of both cameras is selected as the desired position. For each candidate pose, the function checks whether each reconstructed 3D point lies in front of the camera. This is done by transforming the 3D point into camera coordinates using the rotation and translation matrices, and then checking whether the z-coordinate of the transformed point is positive. If the point is in front of both cameras, its score is incremented by 1. As seen in the below snippet, the best pose is observed in configuration 3.



```
Rotation matrices :
[[ 0.99257128  0.01331856  0.12093334]
 [-0.01153484  0.99981429 -0.01543772]
 [-0.12111649  0.01392809  0.99254058]]
Camera centers :
[[ 0.99444607]
 [-0.04933522]
 [-0.09296796]]
3D reconstructed points :
(316, 3)
[[-28.73032326  -6.97628812  21.46958905]
 [-26.98997144  -0.0515493   20.28280261]
 [-27.98614192  -6.33902931  21.13549945]
 [-47.71539968 -16.74950668  36.25187692]
```

Values of the rotation matrix, camera center's and 3D reconstruction pts for the best pose observed.

Step 4: Next comes Image rectification. It is the technique of converting images to a common image plane to make stereo correspondence computations simpler and enhance the precision of depth or disparity maps in stereo vision. The following rectified images are obtained after performing a dense stereo matching between the two views.

Step 5: We do dense matching which is a technique used in computer vision to compute dense correspondences between two images or frames of a video. It involves finding correspondences between every pixel in one image and its corresponding pixel in the other image. The result of dense matching is a dense disparity map that encodes the depth information of the scene, where each pixel in the map corresponds to a 3D point in space. Dense matching is often used in stereo vision, where two images taken from different viewpoints are used to compute the depth information of the scene.

```
desp1 :
(270, 480, 128)
[[[0. 0. 0. ... 0. 2. 0.]
  [0. 0. 0. ... 0. 2. 0.]
  [0. 0. 0. ... 0. 2. 0.]
  ...
  [0. 0. 0. ... 0. 0. 0.]
  [0. 0. 0. ... 0. 0. 0.]
  [0. 0. 0. ... 0. 0. 0.]]

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

```
desp2 :
(270, 480, 128)
[[[0. 0. 0. ... 2. 3. 0.]
  [0. 0. 0. ... 2. 4. 0.]
  [0. 0. 0. ... 1. 4. 0.]
  ...
  [0. 0. 0. ... 0. 0. 0.]
  [0. 0. 0. ... 0. 0. 0.]
  [0. 0. 0. ... 0. 0. 0.]]

 [[0. 0. 0. ... 1. 2. 0.]
  [0. 0. 0. ... 1. 3. 0.]
  [0. 0. 0. ... 1. 3. 0.]
  ...
```

Dense descriptors

```
disparity :
(270, 480)
[[0. 0. 0. ... 1. 1. 1.]
 [0. 0. 0. ... 1. 1. 1.]
 [0. 0. 0. ... 1. 1. 1.]
 ...
 [0. 1. 1. ... 1. 1. 1.]
 [0. 1. 1. ... 1. 1. 1.]
 [0. 0. 1. ... 1. 1. 1.]]
```

Disparity matrix

The final visulualisation of the dense match function is as observed below.