Stereo Reconstruction

Dongha Kang

May 11, 2020

1 Find match

To estimate the epipolar line, first, the two corresponding images need to find whether or not they have matching points by applying SIFT feature. To make the matching more precise, bidirectional matching was applied. The ratio test was 0.7, which provided the number of matching points around 300-320 matching points.

2 Compute F (fundamental matrix)

To draw epipolar line, images need to find the corresponding fundamental matrix using 8-point algorithm and RANSAC. For each RANSAC iterations, it sets up homogeneous linear system with 8 unknowns (A) randomly from matching points, and solve $Ax = 0$ for x, where x is 9 unknown values, which becomes F (will be converted to `3x3` matrix). In this function, I used `numpy.linalg.svd` to calculate x. The RANSAC iterations occurred 10000 times with 0.05 as threshold. (Figure 1).

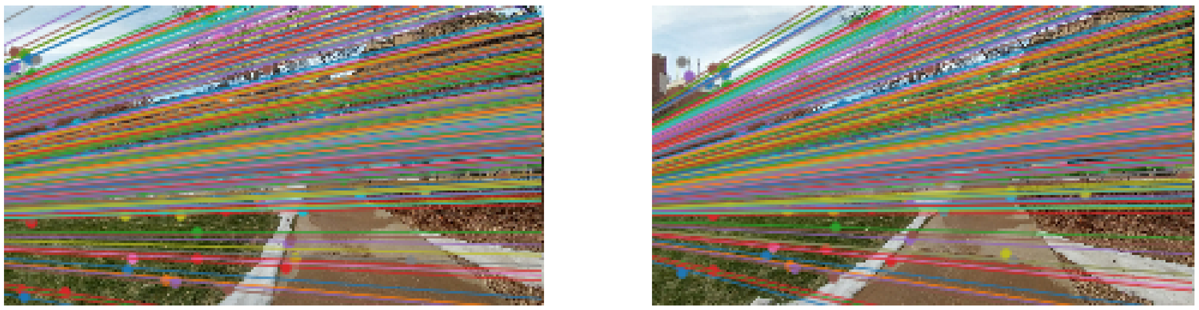


Figure 1: Fundamental matrix

3 Triangulation

Given camera pose and corresponding matching points, the 3D points can be reconstructed by the triangulation equation, which will return 4 different rotation matrices, camera centers and 3D reconstructed points.

4 Pose disambiguation

This function is to find the best configuration of relative camera pose and reconstructed points from 'Triangulation'. In other words, by verifying 3D point triangulation, the best camera pose will be chosen from four different camera poses. Not only applying cheirality condition, this function also checks whether the 3D reconstructed point(`pts3D`) is in front of the the world coordinate's 'default' camera. In most cases, number of valid points with cheirality condition (`nValid`) is around [4, 315, 0, 0]. (Figure **??**)

5 Compute rectification

With best disambiguate pose (configuration of relative camera pose, reconstructed points) and fundamental matrix, this function finds the best homography to make epipolar line horizontal. (Figure 2)

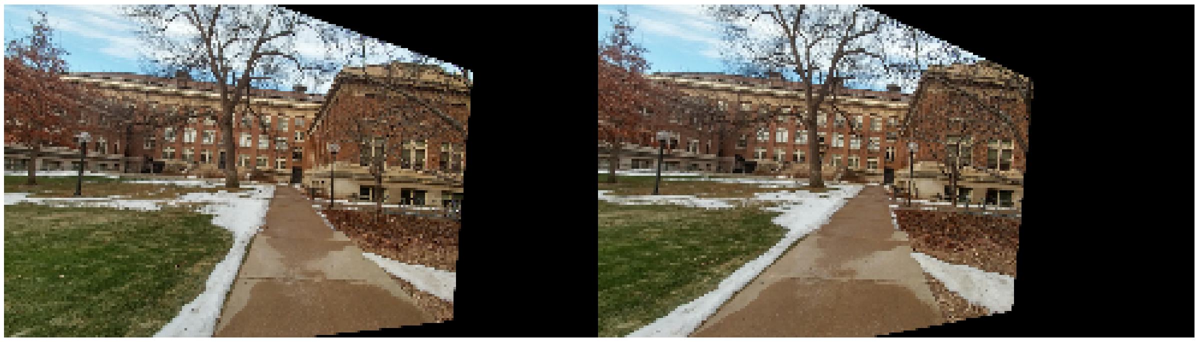
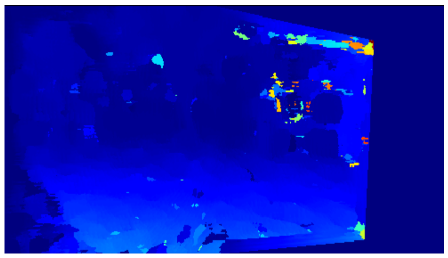


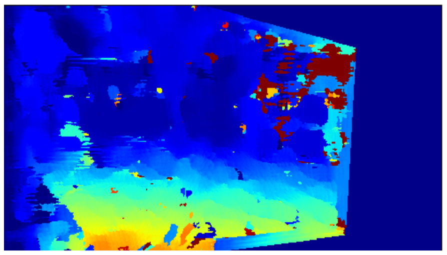
Figure 2: Warp images with rectification homography

6 Dense match

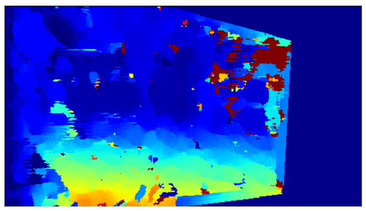
By using dense sift matches, this function finds disparity to (later on) visualize the disparity map between the images. To visualize vividly, I also normalized the image with alpha=0 ,and beta=150 (Figure **??**). Below (Figure **??** is the result of various dense sift sizes.



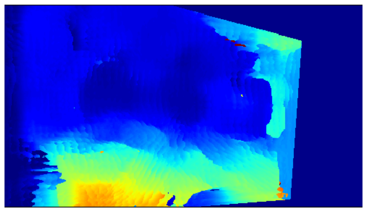
(a) Disparity without normalization



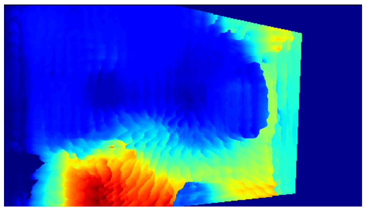
(b) Disparity with normalization



(a) Dense SIFT size: 3



(b) Dense SIFT size: 6



(c) Dense SIFT size: 10