ENPM673 Project 3: Stereo Vision

The link to the google drive containing the results for all the datasets can be viewed here. Some Images have been attached in the report as examples to explain the processes in the pipeline

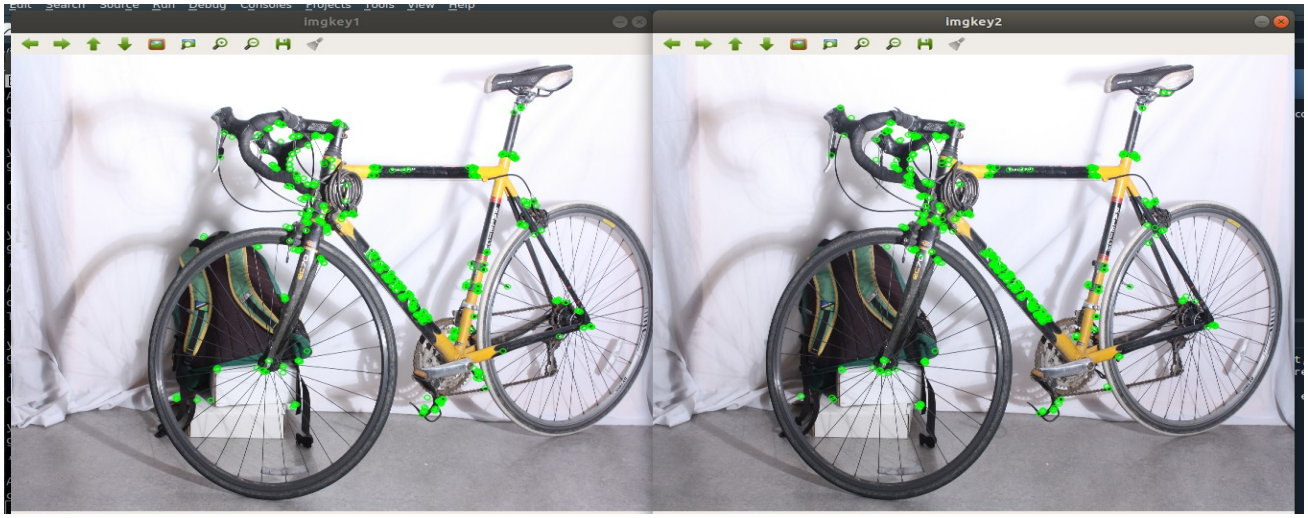
1. Calibration

(i). To compare 2 images in a dataset and select a set of matching features.

Techniques to identify matching features of two images taken from different perspectives are the SIFT and other corner detection techniques. I used a variant of SIFT called ORB which is also used to detect features from the images by identifying the keypoints and computing the descriptors.

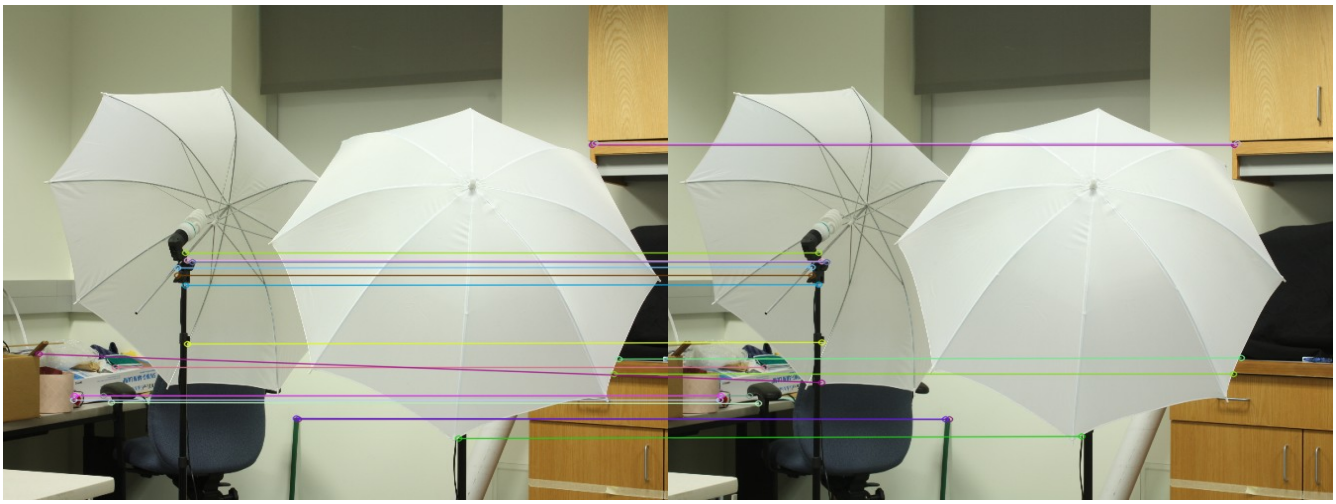
I re-scaled the image and obtained the keypoints by using the function orb.detectAndCompute().
The images showing the keypoints for the one of the datasets is as follows.

Dataset 1



Once the keypoints for each image is obtained, the descriptor of a feature in the first image needs to matched with all the features in the second set. The closest feature in the second image would be obtained by using a threshold distance. To perform this process I have used the brute force matcher object called cv2.BFMatcher(). I then obtain the best matches using the command BFMatcher.knnMatch() specifying a value of k. The set of matches within the particular threshold distance is obtained. A ratio test is applied to get a list of the matching features in both the images. The next image depict the matching features for one of the datasets

Dataset 3



(ii). To estimate the Fundamental matrix by using the set of features obtained in the previous step.

Now that the features are obtained. The fundamental matrix needs to be computed. The steps to obtain this matrix are as follows:

(a) Choose a set of 8 random points from both sets of features.
(b) Create an A matrix using these 8 points. This matrix would look similar to the homography matrix used in the previous projects.
(c) The SVD of the A matrix needs to be computed to obtain the U, sigma and V matrices.
(d) Reshaping the V matrix would give the corresponding Fundamental matrix.
(e) The F matrix obtained due to noise would be of rank 3. To enforce a rank of 2 in the matrix the last singular value needs to be set as zero before obtaining the final value of the fundamental matrix.
(f) To select the inliers the following epipolar constraint needs to be satisfied. The inliers selected would be a set of features which have a value very close to zero.

$x1^{T}.F.x2=0.$

Then the equation can be written as:
x1x2f11 + y1 x2 f21 + x2 f31 + x1y2f12 + y1y2f22 + y2f32 + x1f13 + y1f23 + f33 = 0

The RANSAC operation was repeated 10000 times till the maximum number of inliers was obtained based on which the desired fundamental matrix was computed.

(iii) To estimate the essential matrix and to recover the rotational/translational parameters

The essential matrix can be determined from the Fundamental matrix and the camera calibration matrices K1 and K2.

$$E= K2^{T}.F.K1$$

Once the Essential matrix was calculated from the above equation the following steps are taken to estimate the camera position with respect to the object.

Similarly as in the case of the fundamental matrix, noise can be removed from the calibration by setting the last singular value to 0 and computing E again.

Once the essential matrix is obtained the camera pose needs to be estimated. The camera pose represents the position of the cameras with respect to the world frame. The camera pose would consist of 6 DOFs. Solving the E matrix can give give 4 solutions for the camera configuration with respect to the world frame (R1,C1),(R2,C2),(R3,C3) and (R4,C4). These configurations can be obtained by considering a W matrix W=[[0 -1 0]; [1 0 0]; [0 0 1]].

If the determinant of the obtained rotation matrices are equal to 1. The camera pose needs to be corrected by negating the R and C matrices.

(iv) Determination of a unique camera pose by Triangulation

Now that 4 camera configurations are obtained, a unique pose needs to be obtained by the concept of traingulation. The cheirality condition needs to be checked if the estimated 3d point is in front of the camera point or not. The 3d points can be traingulated using least squares to obtain the depth Z. A 3d point is in front of the camera if r3 the third row of a rotation matrix multiplied by the difference between the 3d point and camera centre is greater than zero. The configuration with the maximum number of depth points is chosen to be th unique camera pose.

Now the algrebric error is minimized the reprojection error can be minmized using non linear traingulation. I tried implementing this but I was getting an error when using the scipy least squares method.

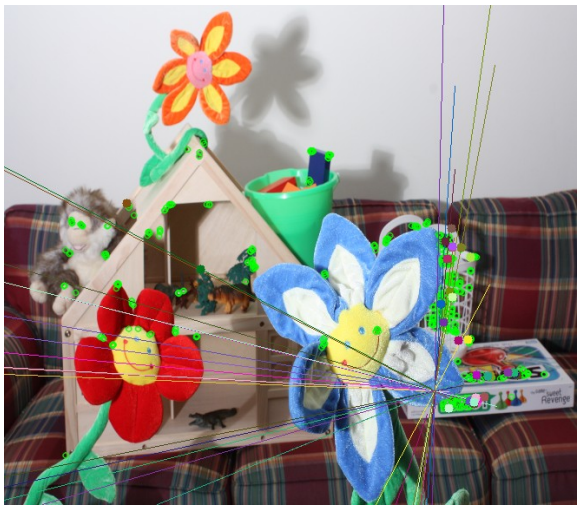
Therefore in this way the E matrix can be decomposed into a rotational and translational matrix to obtain the unique camera pose.

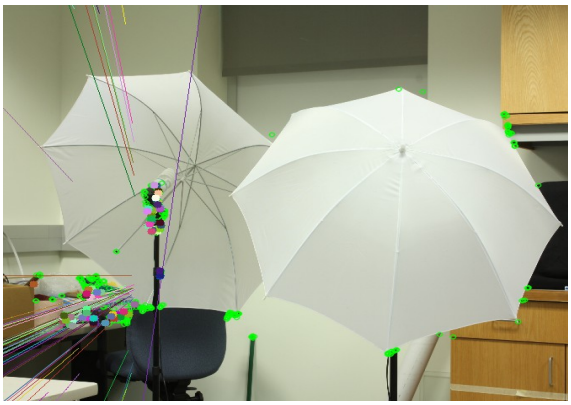
2. Rectification

Before performing rectification it is important to draw and visualize the epipolar lines between the two images. The comand cv2.computeCorrespondEpilines() can be used to plot the epipolar lines using the fundamental matrix and the inliers of one image with respect to the other

The epipolar lines before rectification of the two images for the datasets are as follows:








As we can see in the images the epipolar lines are not always horizontal and are at different angles. Therefore matching each pixel of the two images is essential if we need to calculate the disparity and finally estimate the depth.

I used the cv2.stereoRectifyUncalibrated() function using the features of the two images and the fundamental matrix to obtain the 2 homography matrix and used the cv2.warpPerspective() to warp the image and this caused the epipolar lines in both the images to be horizontal

The following image shows the results after rectification for one of the datasets

Dataset 2

From the results we can observe that the epipolar lines are now horizontal and this can make the computation of the disparity easier.

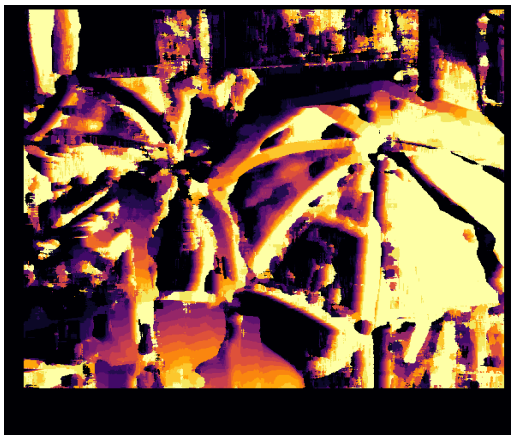
3. Correspondence

In this part of the pipeline for each epipolar line the matching windows concept is applied usng SSD to calculate the disparity of the 2 images.

To compute the disparity the following steps were followed
(a) Specifying a block size to compare pixels from one image with the pixels from another image along the window. I have given a block size of 7.
(b) Specify the number of disparities to consider ie the window size in the second image. This would represent the window. I selected a window size of 16.
(c) The sum of squared differences is found by obtaining the difference between the block from the left image and each block along the window in the right image.
(d) The pixels with the minimum difference is chosen as the candidate for the disaparity image.
(e) The disparity map is rescaled from 0-255 and heatmap conversion is applied. The results below show the disparity image obtained and the corresponding heatmap conversion.

The following image shows the results after SSD for one of the datasets3

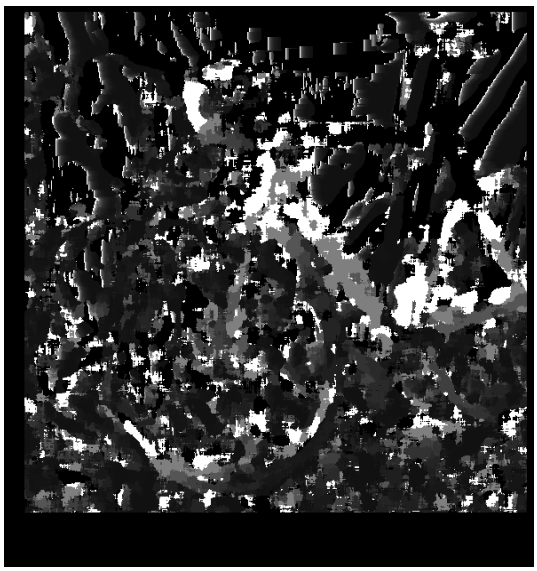
Dataset 3

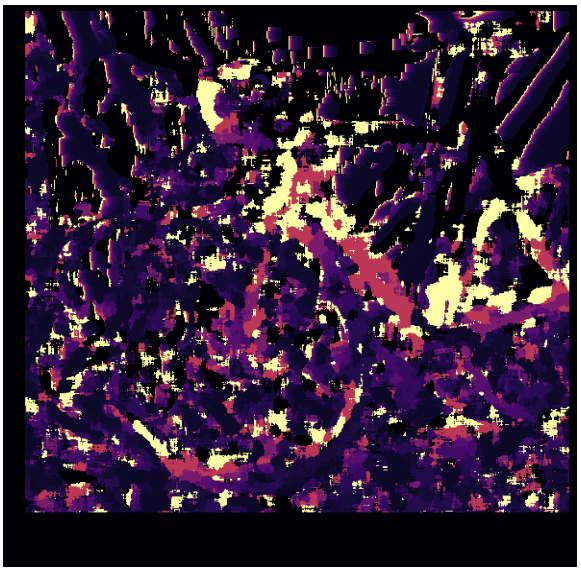


4. Computing Depth Image

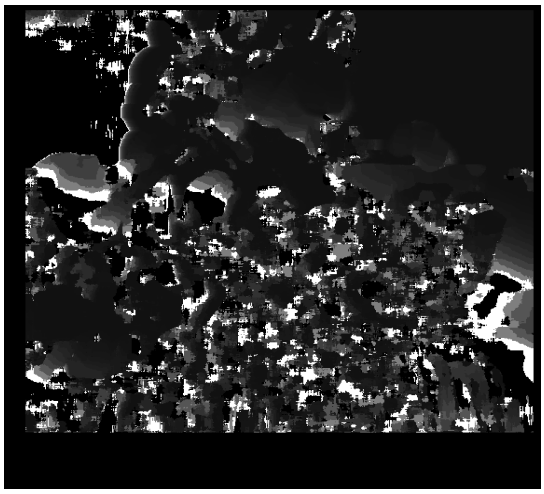
To obtain the depth image from the disparity map we need to understand the concept of depth in stereo images which is the distance of a point/pixel from the camera. Depth (Z) is inversely proportional to disaparity. Depth can be computed by multiplying the lense focal length and the baseline distance between the two cameras divided by the disparity. The image needs to be rescaled from 0-255 and then heatmap conversion needs to be applied to obtain the final image.

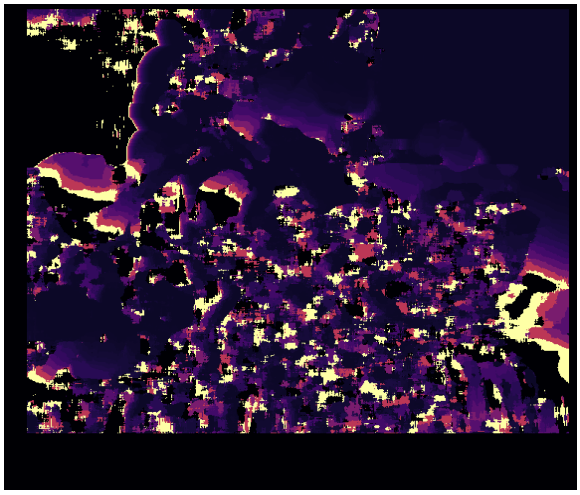
Dataset 1



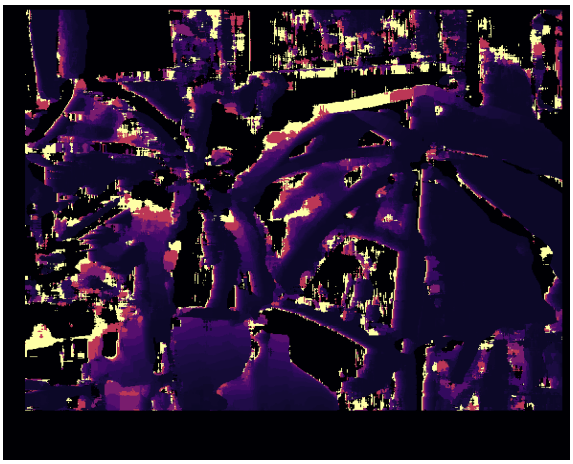


Dataset 2





Dataset 3



From the results we can observe that the pixels which appear brighter in the disparity image appears darker in the depth image. This tells us that those points were far away from the cameras and the images which appear brighter proves that they are closer to the camera.

Problems encountered

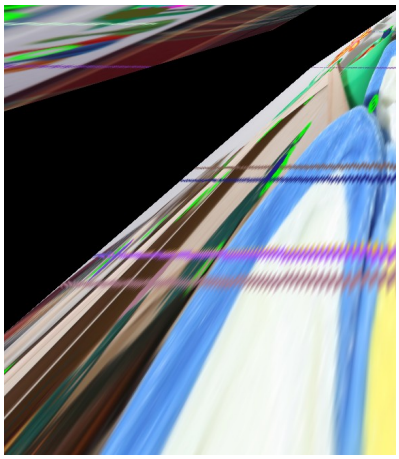
When finding matches for the two images the distance between the two points plays a key role in the further processes.

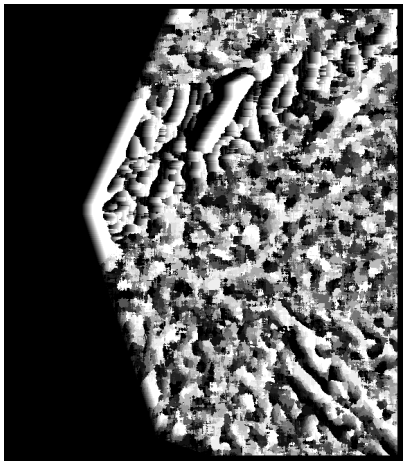
In the below code snippet:

*if m.distance < 0.8*n.distance:*

where m refers to the maching point in the first image and n represents the matching point in the second image and 0.8 refers to the scale factor by which the two points need to be close. Selecting this scale factor plays a significant role when finding the best set of matching features using RANSAC and subsequently performing the rectification.

The following are some of the results of stereo rectification which I obtained when I did not select a proper scale factor.





From the above images we can observe that the epipolar lines are not horizontal and it does not result in an appropriate disparity map.

The other issue was when I tried performing no linear triangulation, the scipy function resulted in an error. Performing a non linear triangulation would result in minimization of the geometrical error and thus give better results. If the camera pose was used in the subsequent steps of the pipeline.