Project 3

PERCEPTION FOR AUTONOMOUS ROBOTS ENPM 673 SPRING 2021

UNIVERSITY OF MARYLAND, COLLEGE PARK

Siddharth Telang
stelang@umd.edu
116764520

STEREO VISION SYSTEM

The aim of this project is to compare two images from a single camera taken from two different positions, involving some rotation and translation and use the information from the second image to estimate the 3D information embedded in the images. There are four major steps involved in this process - Calibration, Rectification, Correspondence calculation and Depth estimation.

Calibration

In this step, we calculate the Fundamental and Essential matrix. The fundamental matrix encodes the intrinsic and extrinsic properties and is useful to find the relation between a set of points between images. On the other hand, the Essential matrix encodes only the extrinsic properties – translation and rotation. To compute this, we need matching points from one image onto another image and with the help of SIFT, we get the desired information.

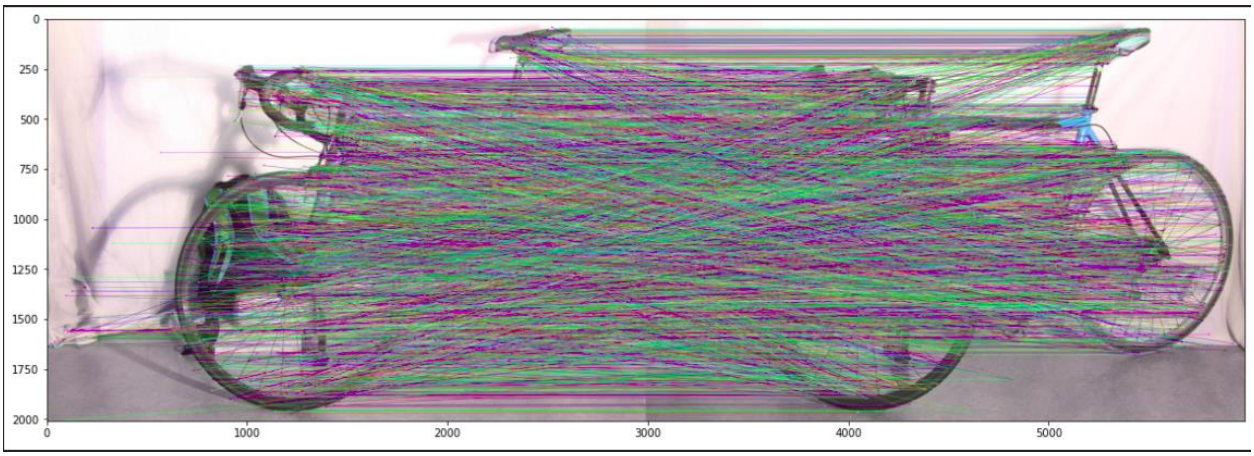


Figure showing matching of points on two images using SIFT

As we can see in the above image, SIFT gives us numerous key points and on matching them the results are not proper – we can see that all the points are not matched correctly. To estimate the F matrix we need the exact matching points. Hence, we eliminate outliers using RANSAC.

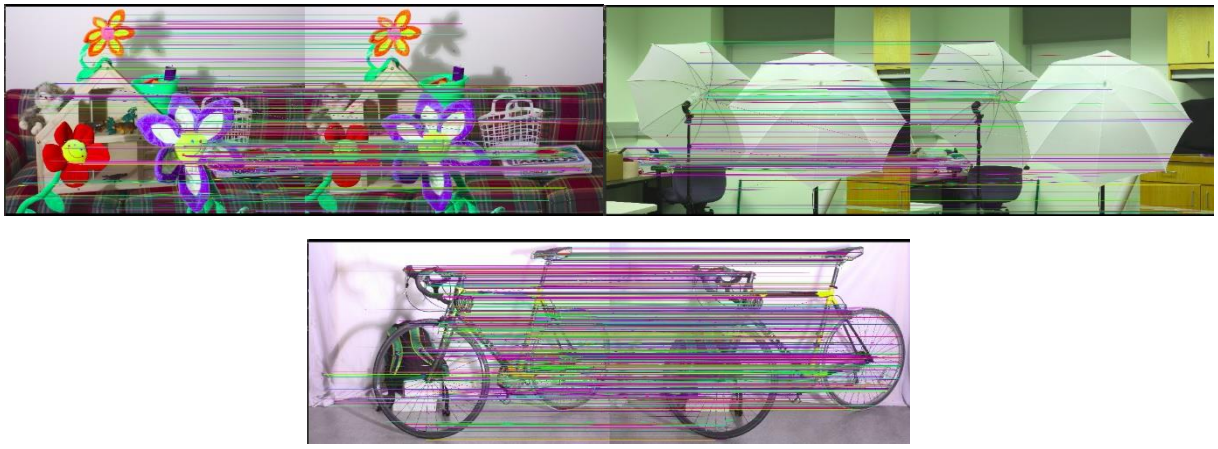
We know that $x'^{T}.F.x = 0$ for a matching point and this can be used to reject the outliers in the matching to get the exact matching points on the two images. Reference: https://cmsc733.github.io/2019/proj/p3/ and https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj3/html/sdai30/index.html)

For better results, we first normalize the points before applying the Eight Point algorithm to compute the F matrix.
- calculate the mean of the points and recenter them by subtracting from mean
- calculate a scale which is the average distance of the centered points form origin
- construct Transformation matrix (T_a, T_b) using mean and scale value
- get new points $x_i = T_a x$; $x'_i = T_b x'$

Using eight random points, we get F matrix using SVD – minimizing $x'^T.F.x$ by creating the decomposition matrix using the normalized points. F matrix can hence be obtained from the last row of V^T in the SVD decomposition.

However, the calculate F matrix can be of rank 3 and this would mean that there is no null space and hence, we won't be able to find the epipoles. Therefore, we decompose the F matrix again using SVD, set the last singular value to be 0 and then compose the matrix back using the updated singular matrix. In additiom, we un-normalize the matrix using Ta and Tb calculated above.

We apply RANSAC and calculate the error in the equation $x'^T.F.x = 0$ for all the matched points. The maximum permissible error is set as 0.001 and if the error goes past the threshold, we discard the points. This step is repeated several times until we get the best F matrix having the highest count of inlier matching points.



Figures showing the matching points after RANSAC

To estimate the Essential Matrix (E), we use the calculate best F and use that in the equation – $E = K_2^T.F.K_1$, enforce rank 2 by decomposing the E matrix using SVD, setting the last singular value to be zero and recombine to get the final E matrix.

Pose estimation:

To get the actual 3D points, we need to have the projection matrix – which is comprised of rotation and translation. The Essential matrix encodes the extrinsic parameters which are nothing but the rotation and translation.

However, there can be four possible solutions to the decomposition of E matrix and the one which gives us positive value of depth (z) will be the correct one.

SVD (E) = U, S, V^T

T1=U(:,3) and R1=UWV^T T2=−U(:,3) and R2=UWV^T

T3=U(:,3) and R3=UW^TV^T T4=−U(:,3) and R4=UW^TV^T where W is the skew symmetric matrix

It is also important to note that the determinant of matrix R is always positive, so if the value is negative, we correct the sign of R and T.

Estimation of 3D points:

$P = KR(I_3 \mid -T)$

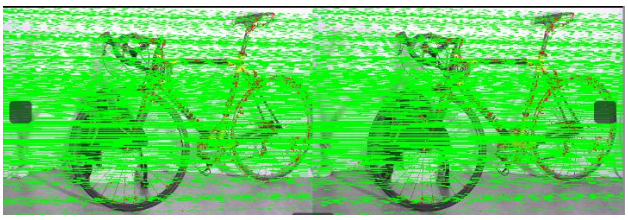
We have two projection matrices – P1 and P2 (four possible values) (P1 – taken as reference – where the translation and rotation is zero).
OpenCV has an inbuilt function to calculate the 3D points given the projection matrices and the source and destination points (matching points on left and right image). With the usage of *cv2.triangulatePoints*, the 3D points are estimated.

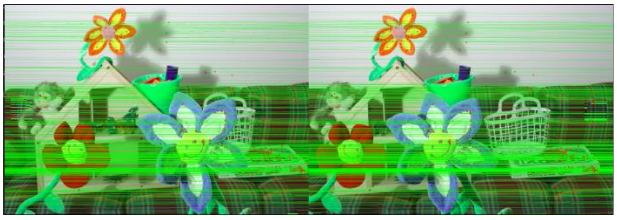
Correct camera pose estimation:

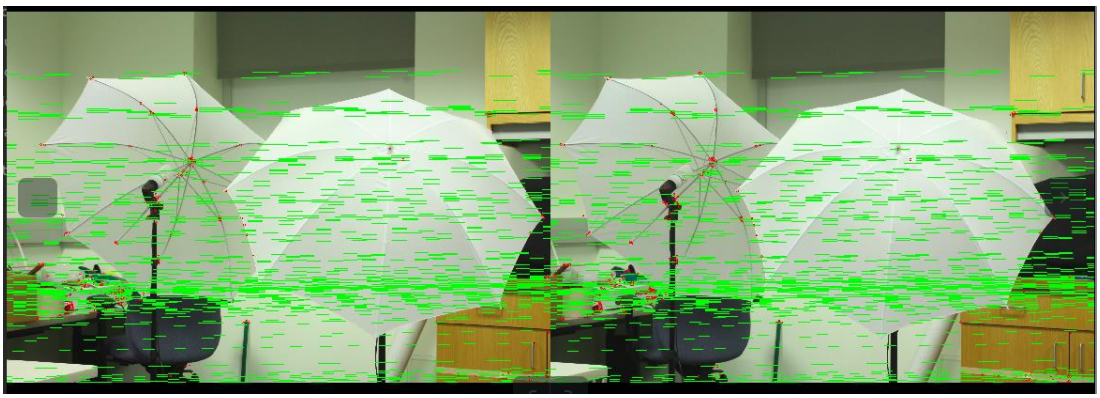
We know that the correct pose would be the one which gives us positive depth. The 3D points from above are multiplied with projection matrices P1 and P2 (four values), the points are normalized, and z is calculated. If both the projection matrices give us positive depth, we increment the count of z and repeat for other values of P2. The best P2 will be the one having the max positive z count which will be the correct camera pose.

Epipolar Lines:

The epipolar lines in the right and left image are calculated as below: $e2 = F.x1$ and $e1 = F^T.x2$ - this gives us the coefficients a, b, c of line in the format of $ax + by + c = 0$. Using this, we can easily plot the epipolar lines in the image.

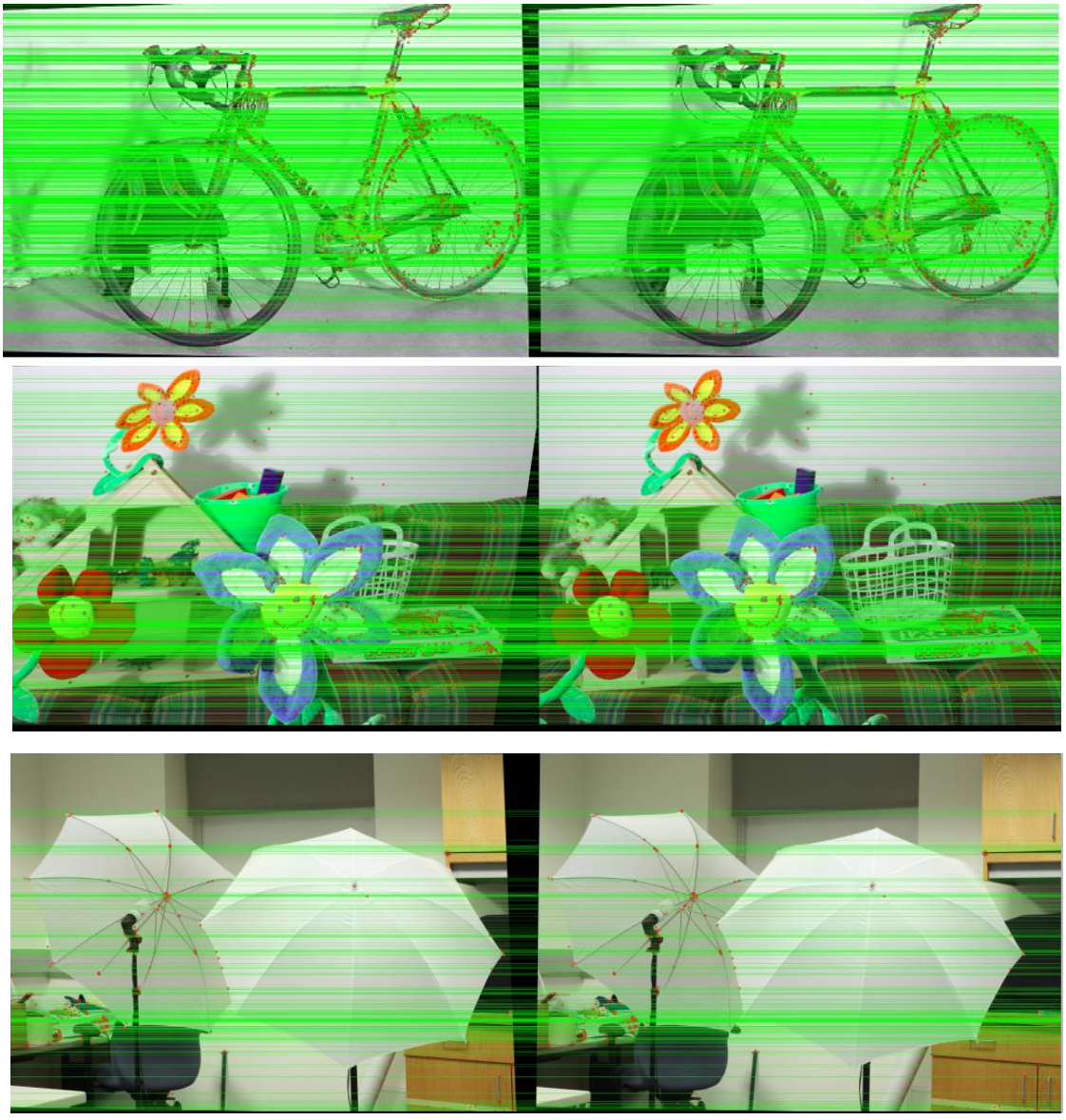






Rectification

The above epipolar lines are not horizontal and need to be rectified – made horizontal so that we can match pixels from one image to another on the same level. For this reason, we apply perspective transform to both the images using OpenCV's function - "*stereoRectifyUncalibrated*", which will give homography matrices for both the images to be warped. The combination of rectified images with epipolar lines and feature points drawn are shown below:



Rectified images with epilines and feature points

Correspondence and Disparity calculation

Now after we have got the epilines horizontal, we can proceed with the window matching concept to match the points on these lines. We take a specified size window and for every window block on the epiline of left image, we scan the corresponding epiline on the right image. During the scan process, we calculate the SSD (sum of squared differences) for every block in that line. For the block to be same at both the images, the SSD should be minimum, this concept is applied and disparity of the blocks having least SSD is found. The disparity matrix is then scaled to a maximum value of 255.

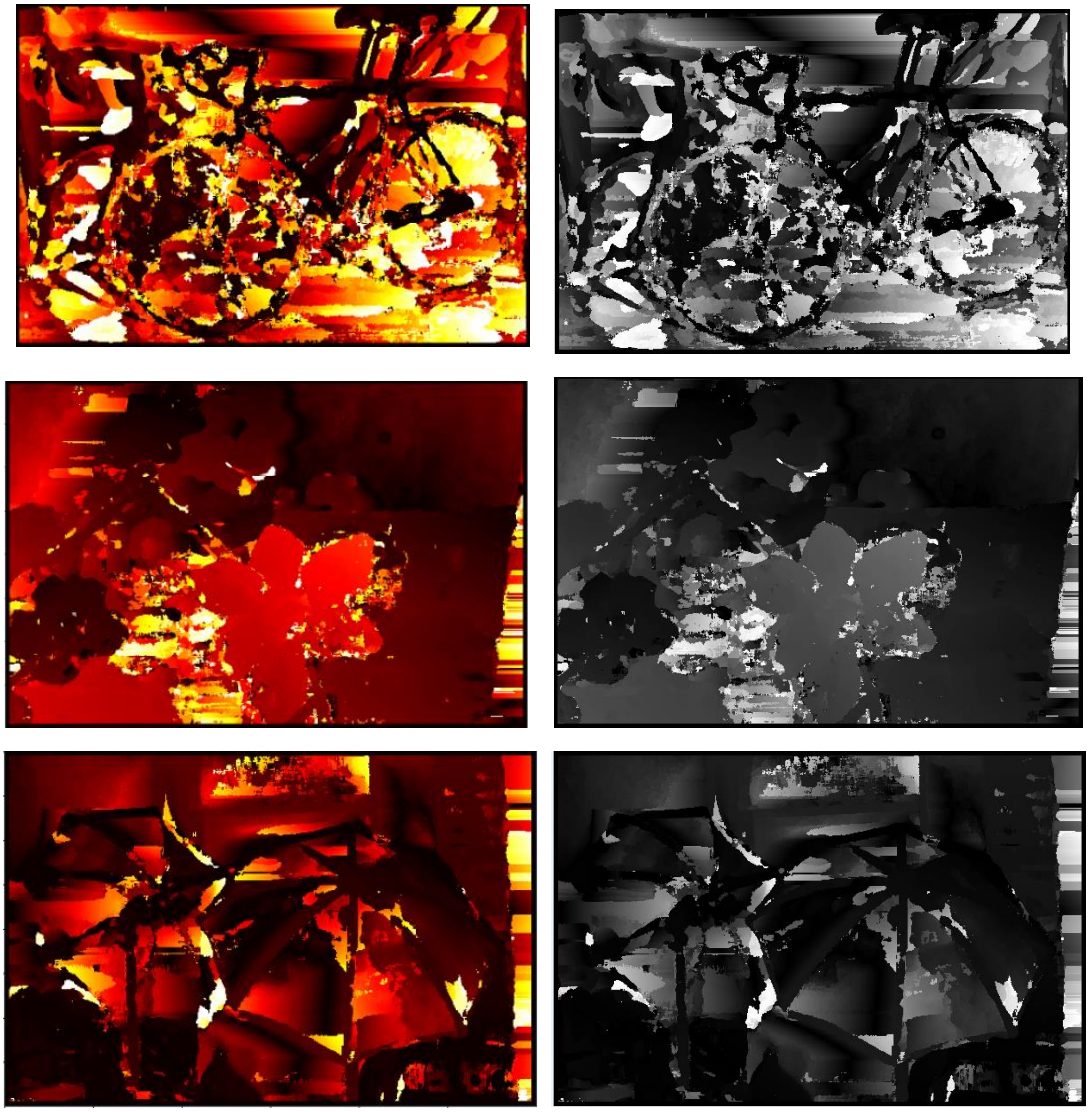


Figure showing the disparity map in heatmap and grayscale.

Depth estimation

Once we have got the disparity information, we can simply calculate the depth by using the relation: depth = (f*baseline)/disparity, where f = focal length, baseline = distance between the two camera centers. It has to be noted that at some places the depth can be zero, and dividing by zero will give ∞, hence we add a negligible number, tending towards zero to disparity to avoid this. In addition, we further scale the depth to a max of 255.

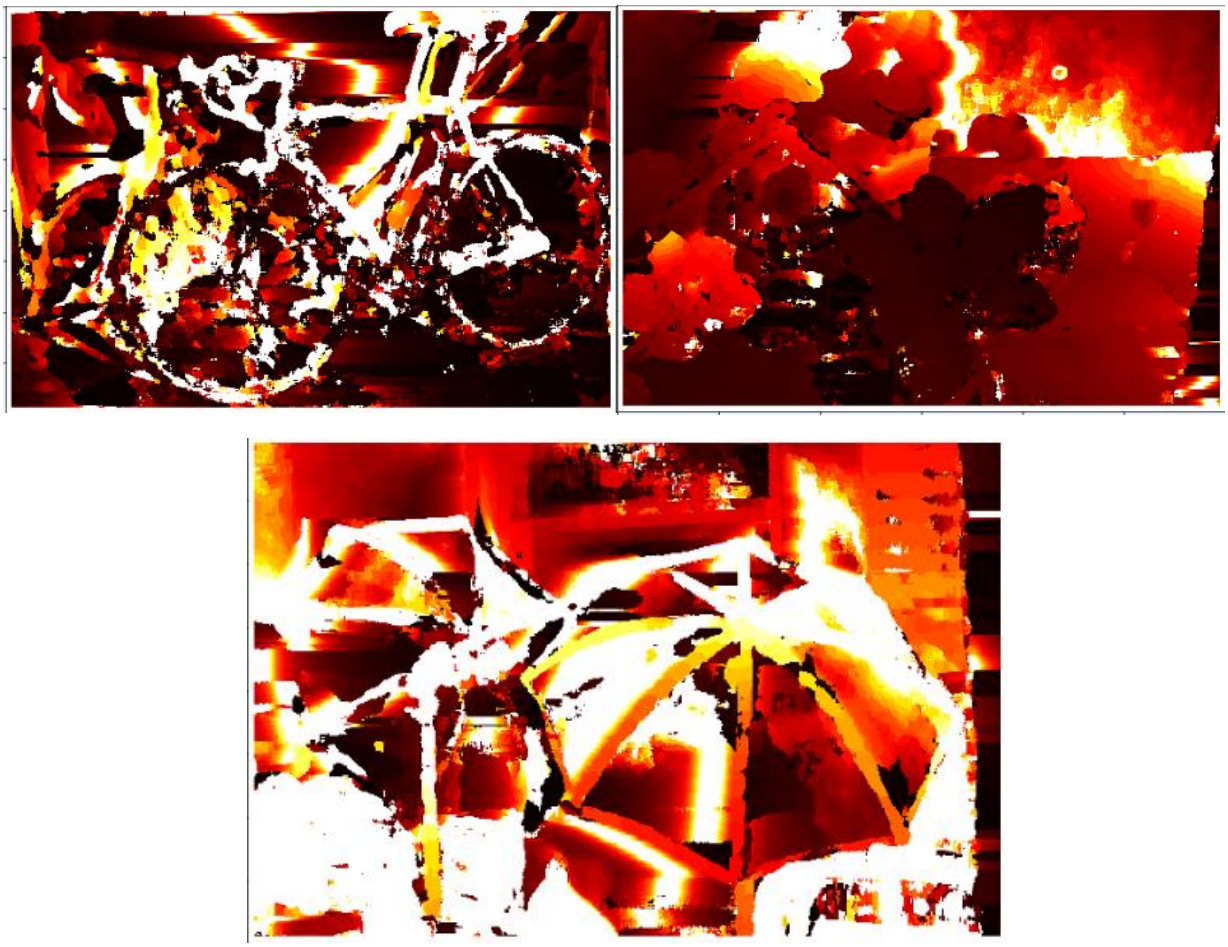


Figure shows the depth information.

Link to all the results:
https://drive.google.com/drive/folders/17bMJsVr_L3CIB7A0DQGo7w2xyyWFhgIO?usp=sharing