4) Computing Planar Homographies

4.1. Feature Detection, Description, and Matching

- Converted the images to rgb2gray necessarily.
- Detected the features using the detectSURFFeatures function.
- Extracted & matched the features points from both the images using the extractFeatures and matchFeatures function.
- Took the index pairs and showed the matched locations as shown below:-

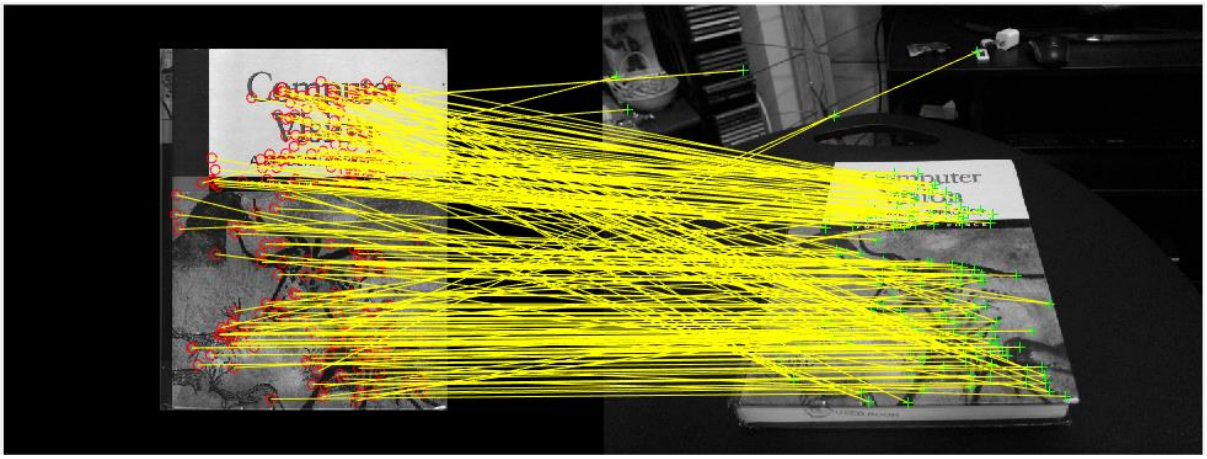


Figure 1:- Showing the matched locations for 2 images.

4.2) Brief And Rotations:-

- Read the image using the imread function
- Rotated the image using imrotate function
- Computed the detect and match Features necessarily and received the count of the image locations matched.
- Updated the histogram for each of the orientations and and displayed the histogram.
- Sample Rotations were received in the sample file. Some of the sample rotations of the images are saved in the results folder. Shown 2 of the sample rotations below:-

With ComputeBrief:-

Sample 1

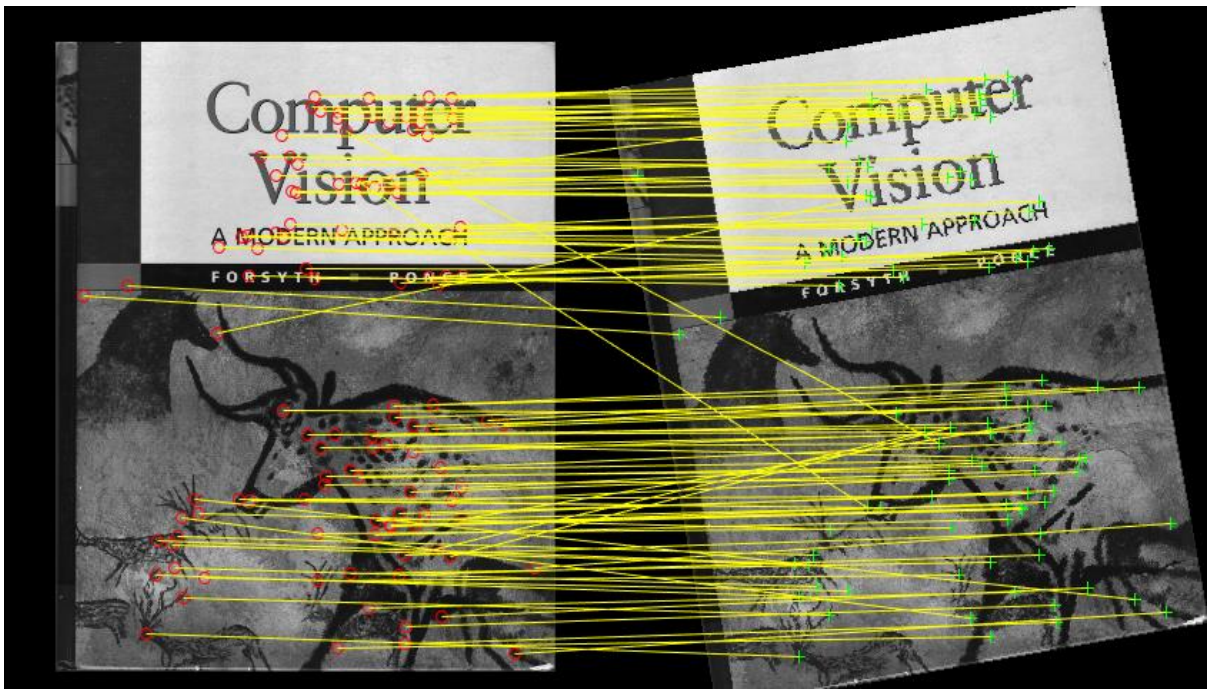


Figure 2:- Showing the original image with the rotated image(of some degree)

Sample 2:-

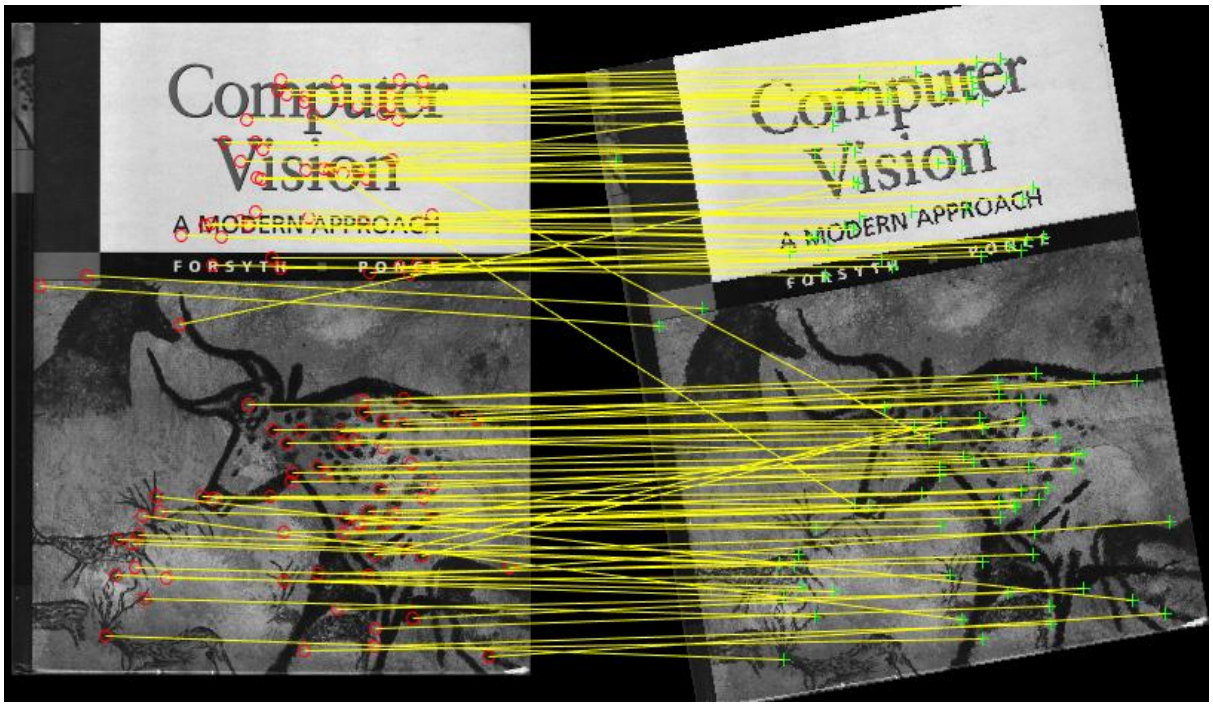


Figure 3:- Showing the original image with the rotated image(of some degree)

With ComputeBrief histogram

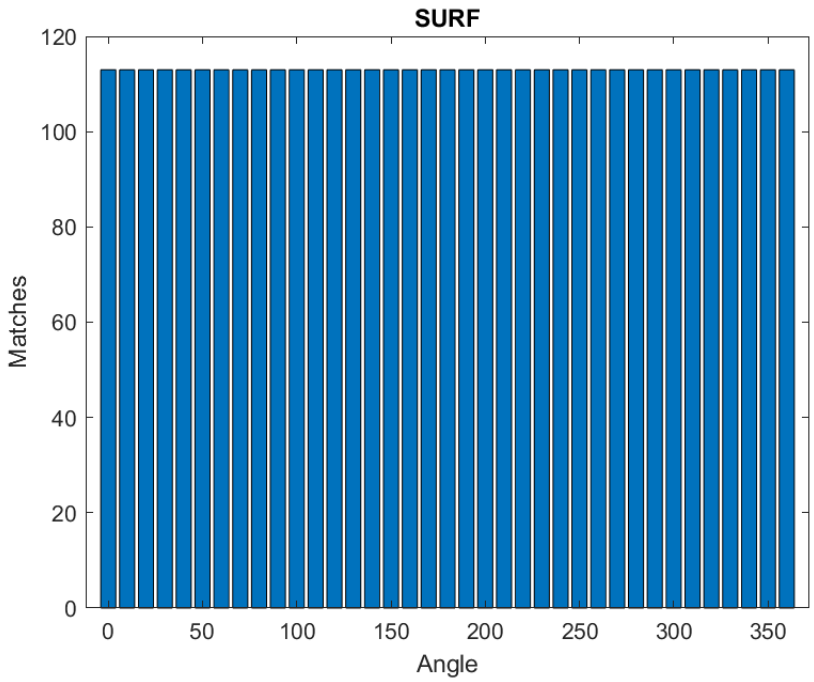
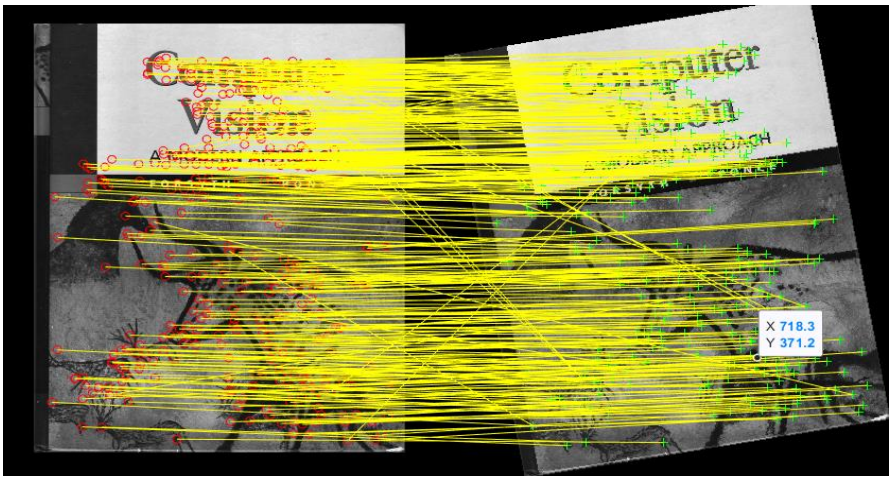


Figure 3:- Showing the histogram using Brief

Using SURF sample 1



Using SURF Sample 2

Using SURF Histogram

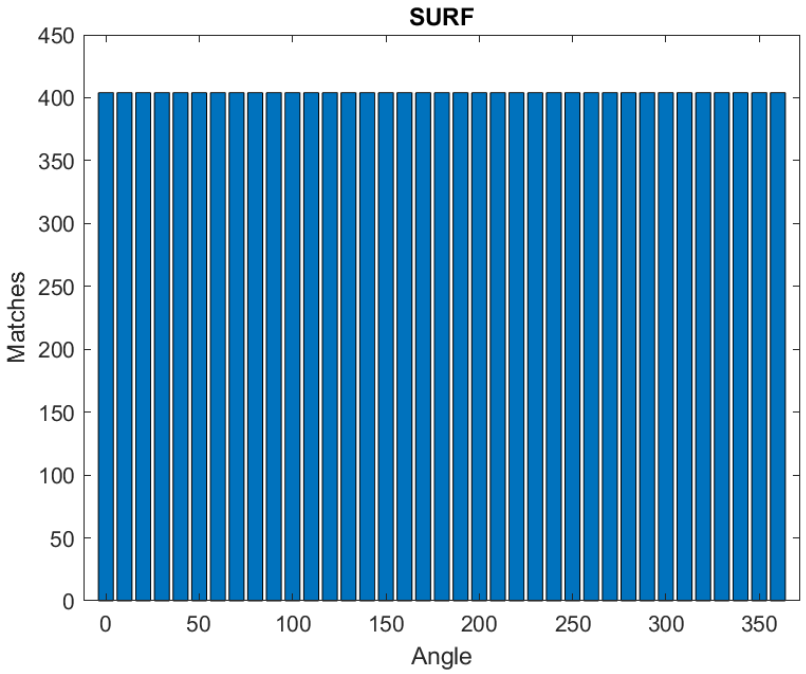


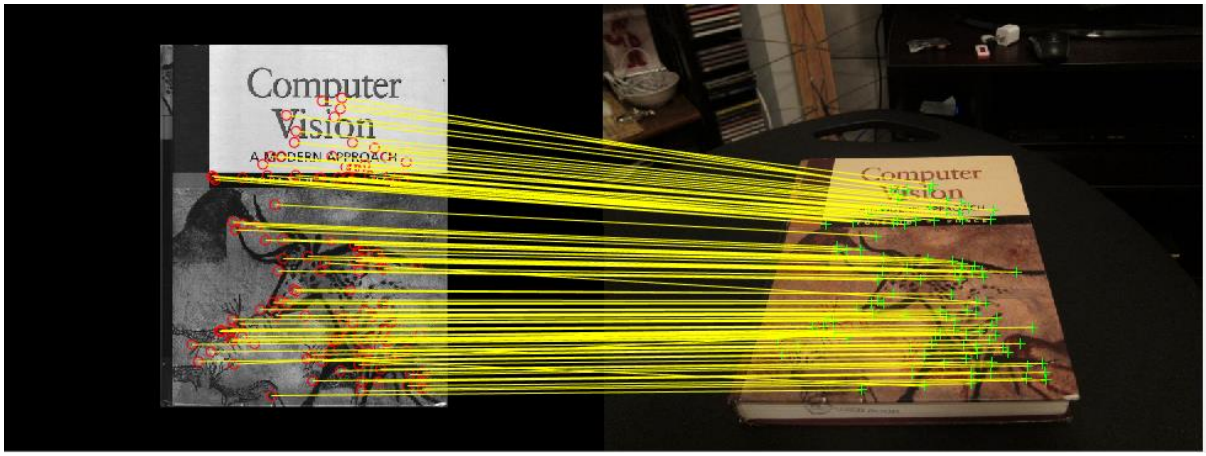
Figure 4,5,6:- Showing the rotated images using SURF and histogram using SURF(extract Features)

4.3) Homography Computation, Homography Normalization and RANSAC Implementation :-

- Homography for the provided images.
- As we can see in the first figure, that there are lots of outliers while matching the features.
- I have achieved through the concept of ransac and SVD to remove the outliers and provide maximum number of inliers that match correspondingly to the images that are given.
- I have minimised the number of errors that were received and computated the homography each and every time through the computeH_ransac function.
- Normalizing the homography is achieved via the compute_norm function. The mean of the points to the origin has been calculated and scaling is done for finding the largest distance and received the linear transform equation.
- The homogenous coordinates are normalized and received the H2to1 value.
- By calling the computeH function, we have reshaped the coordinates and performed the SVD logic.
- This is performed throughout until the noisy data's are removed and received a correct inlier images with the correct inliers(there were no outliers found after performing the RANSAC). Hence the RANSAC algorithm was learnt via this.

During the computeH implementation

After implementing the RANSAC



4.6 HarryPotterizing a Book

- In the file HarryPoterize_auto.m warping is achieved using the warpH function. Upon calling the warpH function, the bestH2to1 which was found using the ransac algorithm is used for finding the best feature match. Warping is achieved successfully.
- Using the compositeH function, the masking is performed to match the image with the warped image. Calling the compositeH function, I have created a mask for the warped image with the expected homography. The composite image has been identified and hence achieved the HarryPotter image on top of the book with the expected resolution. The HarryPotterized output is given below:-