CMPT 412
Assignment 2 Report

Maoshun Shang

301280479

Introduction

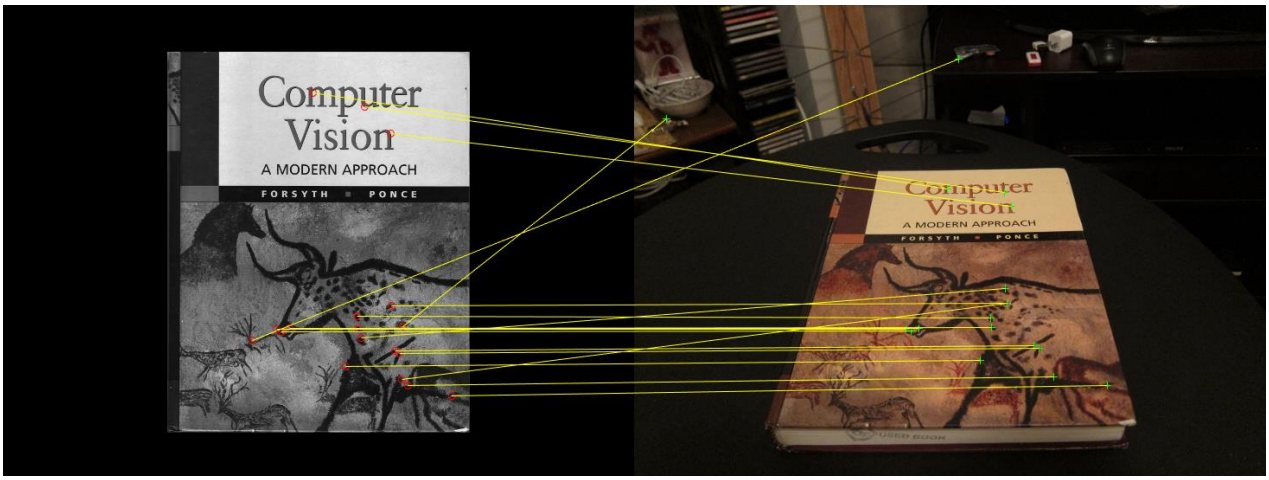
This assignment mainly focuses on the technique studied in lectures, including basics about feature detector, feature descriptor, planar homographies, and direct linear transform. The programming language used is MATLAB, which helps with reducing the complexity of heavy computation on matrix and vector calculations.

How to Run the Code

The main entrance for running the whole project is the file named “q2_1_4.m”, “briefRotTest.m”, “HarryPotterize_auto.m”, and “ar.m”, please prepare input images and videos in data folder as “.jpg” and “.mov” format. After running ar.m, the final output will be stored in “result” folder. Please note according to computer’s CPU speed, the execution time may vary.

Feature Detection, Description, and Matching

For this task, FAST detector and BRIEF descriptor are chosen as the desired feature detector and descriptor respectively. The function matches the features selected by the detector and descriptors on image 1 and image 2, then return the location of these features. The following is a sample output for book cover.

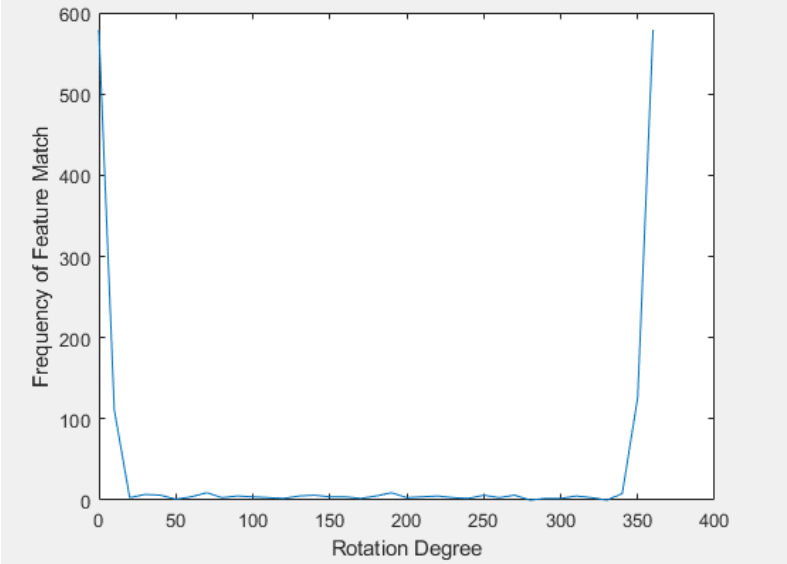


While there are few outliers, for example, there are two features mapped to somewhere outside the book, it is obvious that most of the features from one picture matches to the same feature location on the other one.

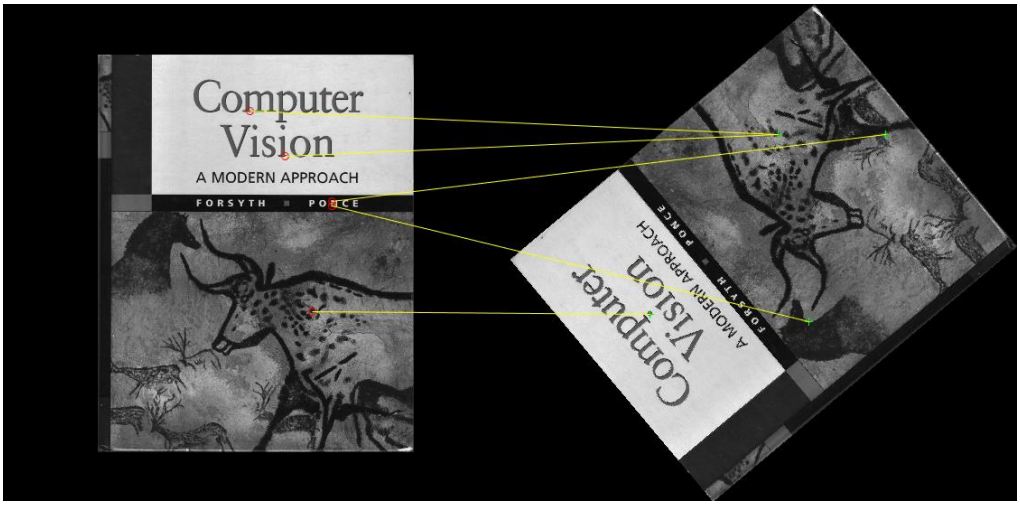
BRIEF and Rotations

In this section, we examine how rotation change the number of matched features. For this task, first, we use FAST feature detector and BRIEF descriptor, then compare the result with SURF feature detector and its corresponding feature descriptor. For each type of detector, we choose 3 orientations to visualize the matched points. Note the direction of rotation is counterclockwise.

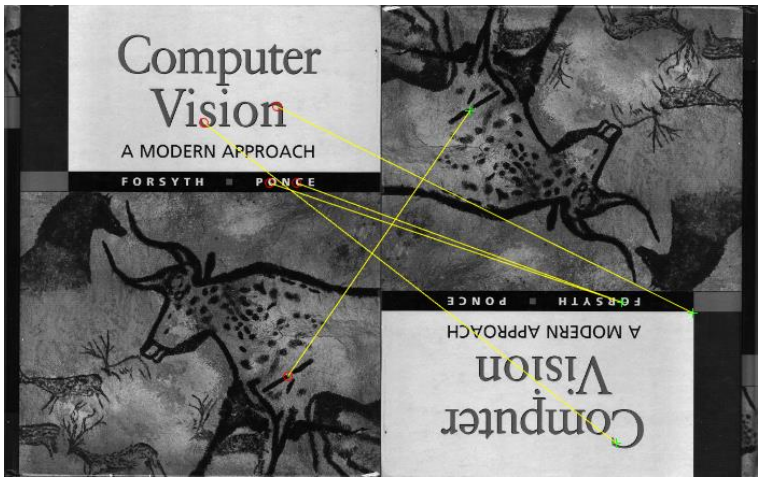
For FAST Detector with BRIEF Descriptor:



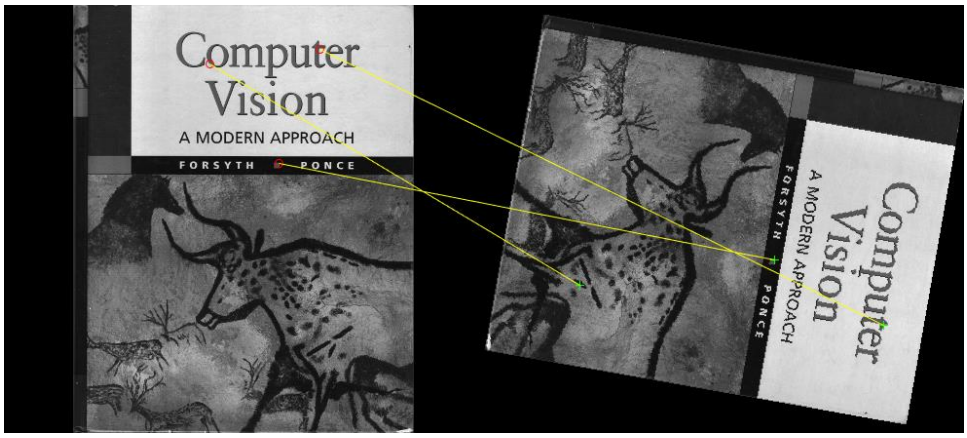
Rotation Degree vs. Count of Matched Features (BRIEF)



Rotation = 130 degrees

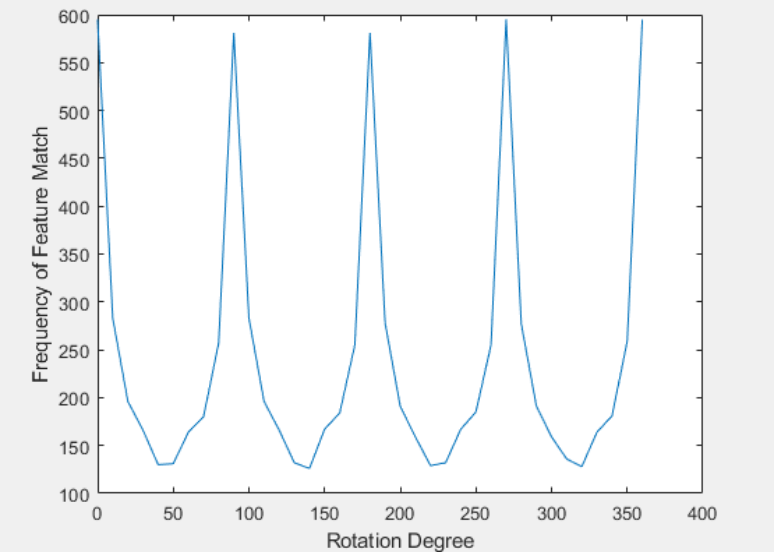


Rotation = 180 degrees

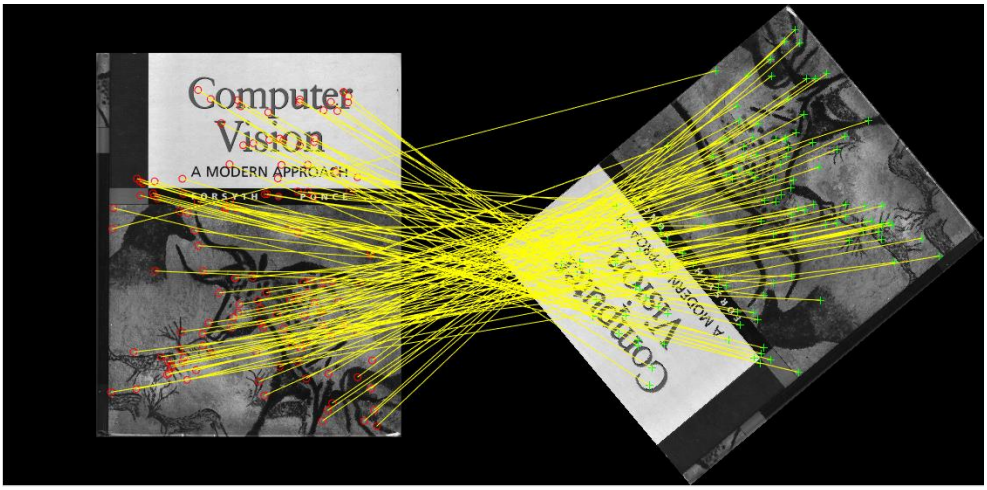


Rotation = 260 degrees

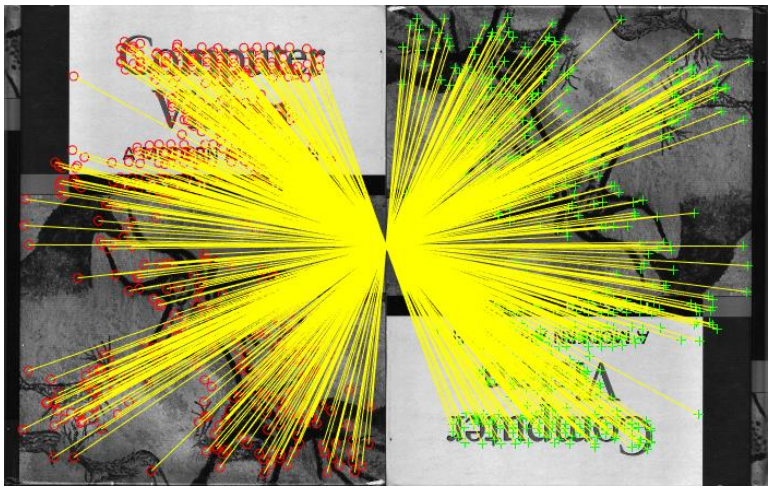
For SURF Detector with SURF Descriptor:



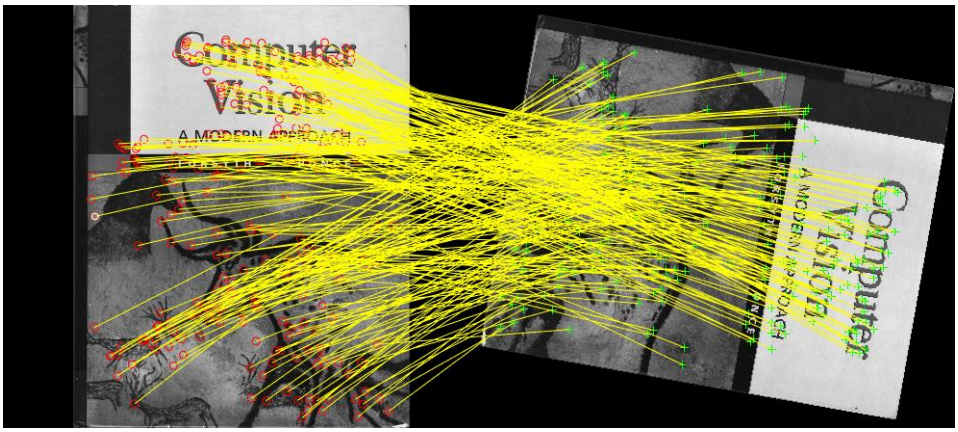
Rotation Degree vs. Count of Matched Features (SURF)



Rotation = 130 degrees



Rotation = 180 degrees



Rotation = 260 degrees

The BRIEF descriptor matches only few points unless the rotated image is very close to its upright position. That is, only at around 0 degree or at around 360 degrees, the number of matched features is normal. Otherwise, the number of matched features stays at a very low count.

Because BRIEF descriptor, which is provided in “computeBrief.m”, approximates the feature location instead of using interpolation, such behavior leads to a less accurate result. After rotation, the original image pixel might be settled in some location that is not exactly inside a pixel. In another word, the rotated pixel location might not be an integer. Thus, BREIF feature descriptor cannot extract those features due to non-interpolating result. Therefore, this phenomenon suggests that BRIEF detector is not good at detecting the features under such a case.

However, for SURF descriptor, which is a built-in function in MATLAB, the plot changes significantly. The number of matched features fluctuates with a pattern, and the minimum number of matched features is obviously exceeding that of BRIEF descriptor. Specifically, the number peaks at rotation = 0, 90, 180, 270, and 360 degrees. Even if the highest and lowest count differs dramatically, the SURF did a good job at detecting the features when the object is rotated. Comparing to BRIEF, SURF detector is good for detecting rotated objects.

Homography

Homography Computation

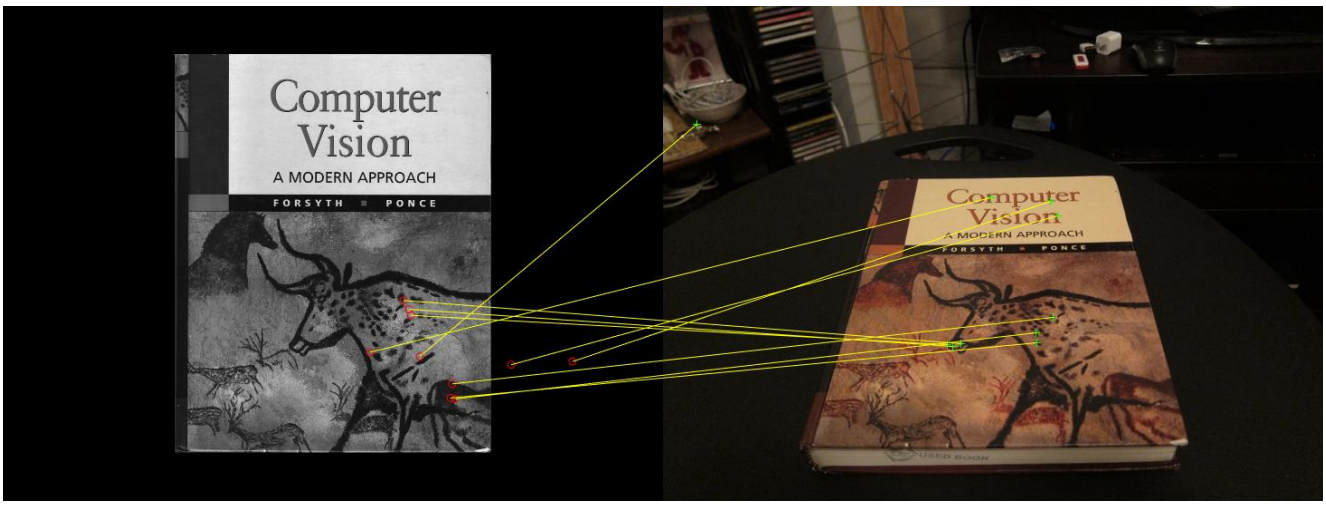
In this section, computeH(x1, x2) is a function that returns a 3 by 3 matrix, called H2to1, for transformation points from x2 to x1. By taking a pair of images, we visualize the corresponding location of 10 pairs of points.

Homography Normalization

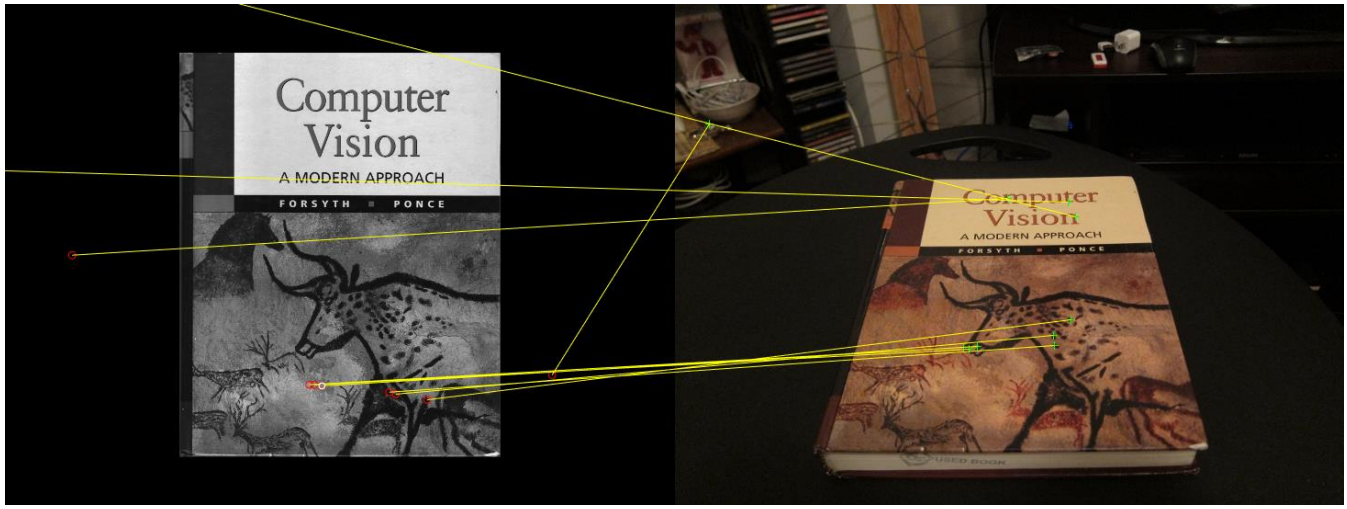
In addition to the previous procedure, we do additional two steps: translate and scale. Then plot on the same image.

RANSAC

In addition to previous section, we do N iterations, randomly select 4 pairs of points to produce a H2to1 matrix in each iteration. Keeping the best H2to1 matrix on the fly. The image “computeH_RANSAC” below shows the inliers points produced by the best H2to1 matrix.



computeH

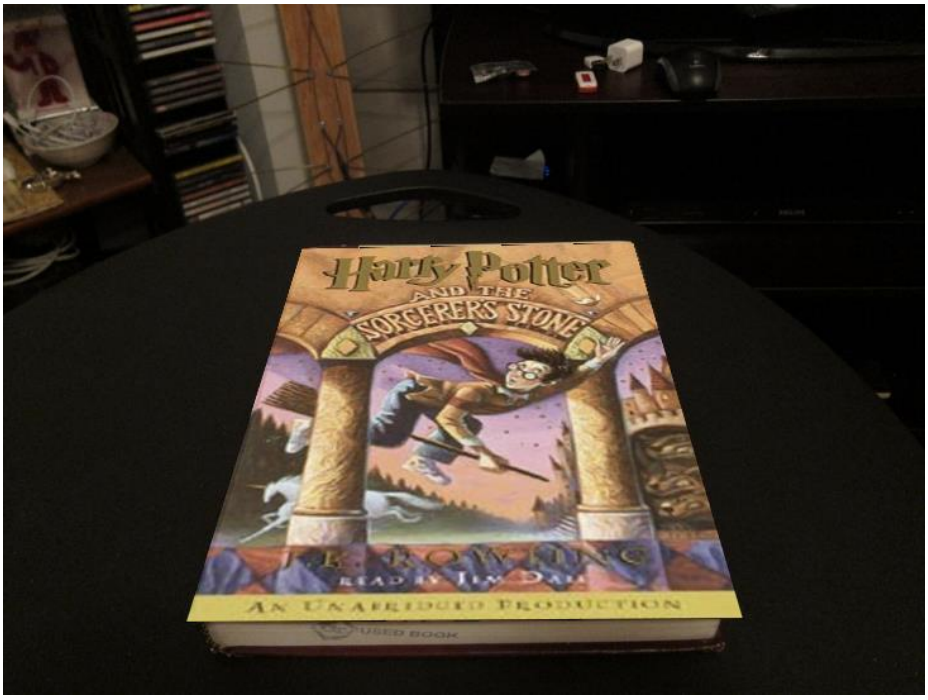


computeH_norm

computeH_RANSAC

HarryPotterizing a Book

The code computes a matrix that transfer cv_cover.jpg to desk.jpg, then using the same transformation and masking, we transpose Harry Potter cover to the book on the desk. The sample result is shown as below.



Augmented Reality Application

This section is similar to previous section. However, instead of doing computation on a single set of images, we do computation on multiple sets of images. That is, if we split the video frame by frame, each frame is identical to the job in the previous section. After running the script, the final video, named “output.avi”, is stored in the “results” folder. A sample frame from my output video is shown below for reference.