HOMEWORK 3 REPORT

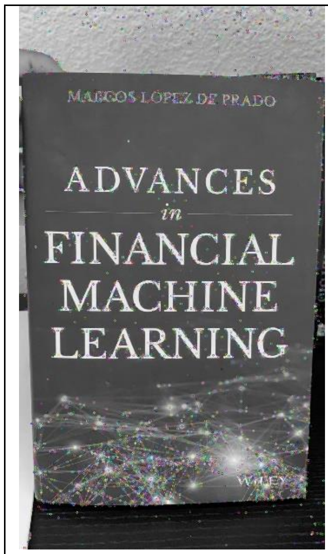
CSCI 677

SIFT, Homography, RANSAC

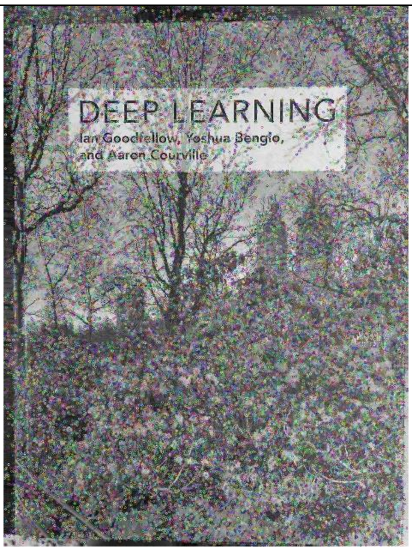
Hardik Prajapati
USC ID: 2678294168

a) SIFT features: No. of SIFT features detected and their plot on each source and destination images

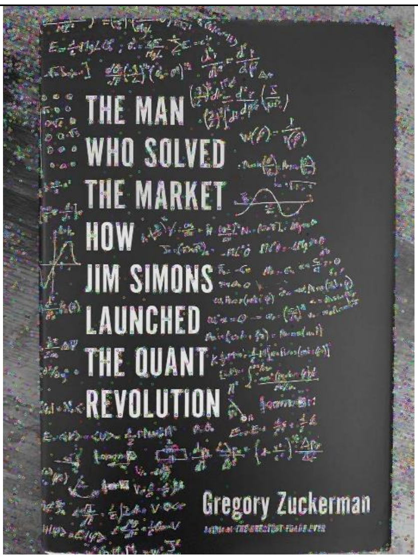
# Of SIFT features for different scales					
Scale (%)	Dst_0	Dst_1	Src_0	Src_1	Src_2
15	1805	362	384	1033	490
30	7510	1135	1013	3861	1641
60	26602	4059	2653	15497	4850
100	49979	10625	5259	42457	13004



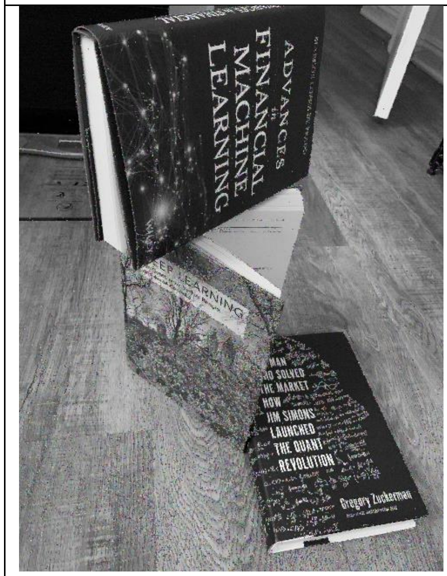
Src0



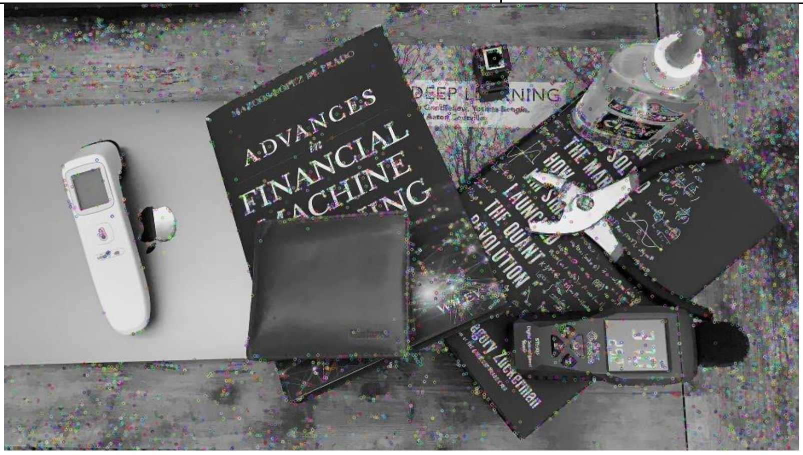
Src1


THE MAN WHO SOLVED THE MARKET HOW JIM SIMONS LAUNCHED THE QUANT REVOLUTION
Gregory Zuckerman

Src2



Dst0

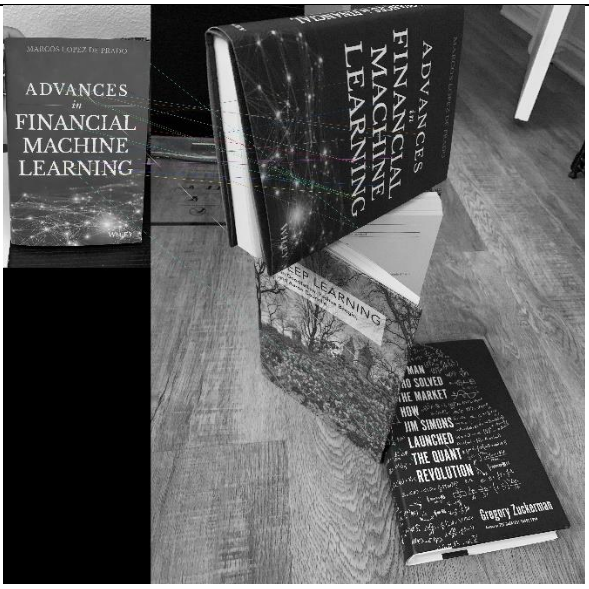
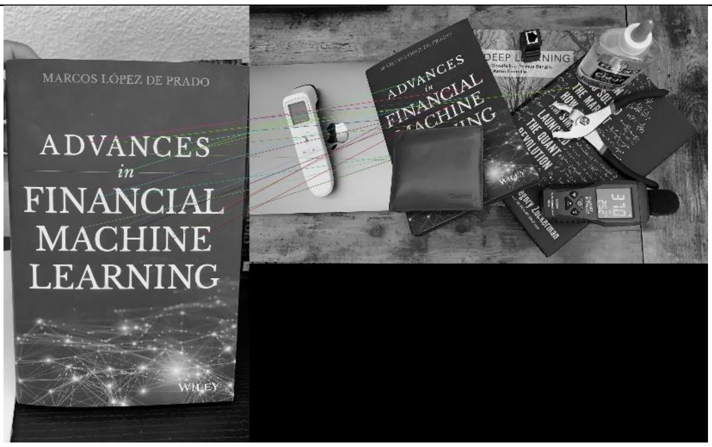
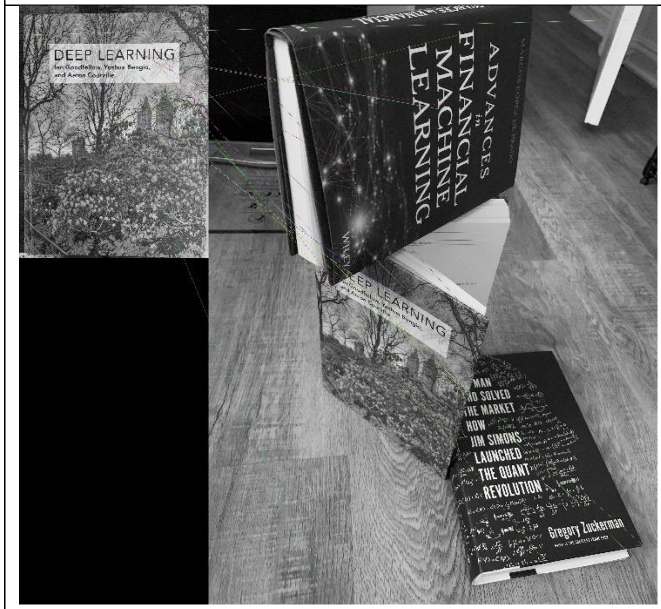
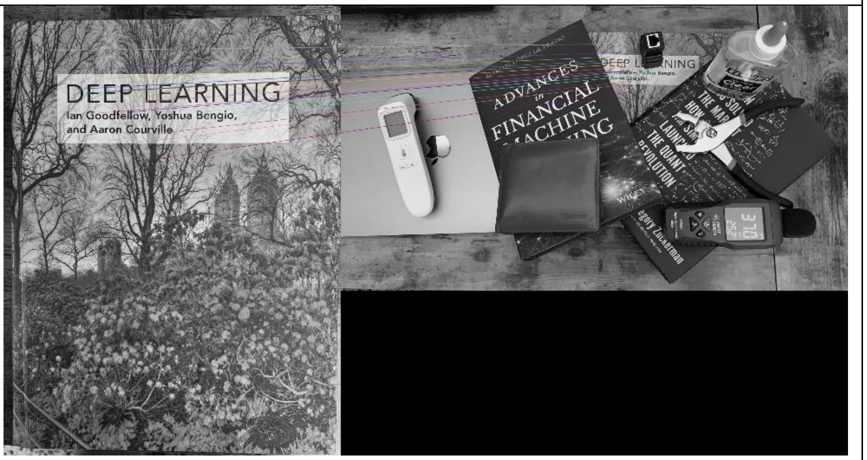


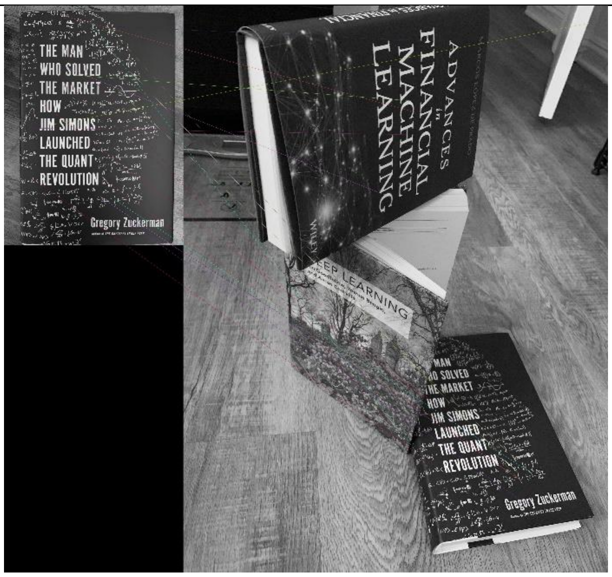
Dst1

Fig1: SIFT features

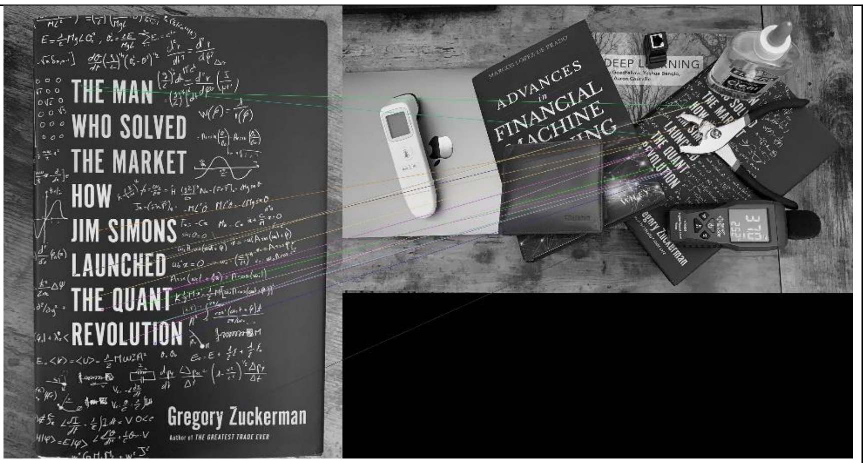
b) Top-20 matches before RANSAC AND Statistics of # of matches found

Matches	Src0_dst0	Src0_dst1	Src1_dst0	Src1_dst1	Src2_dst0	Src2_dst1
Good matches	219	529	155	606	1241	1047
Total matches	5259	5259	42457	42457	13004	13004
% Good Matches	**4.16**	**10.05**	**0.36**	**1.42**	**9.54**	**8.05**

 **Src0-Dst0**	 **Src0-Dst1**
 **Src1-Dst0**	 **Src1-Dst1**



Src2-Dst0

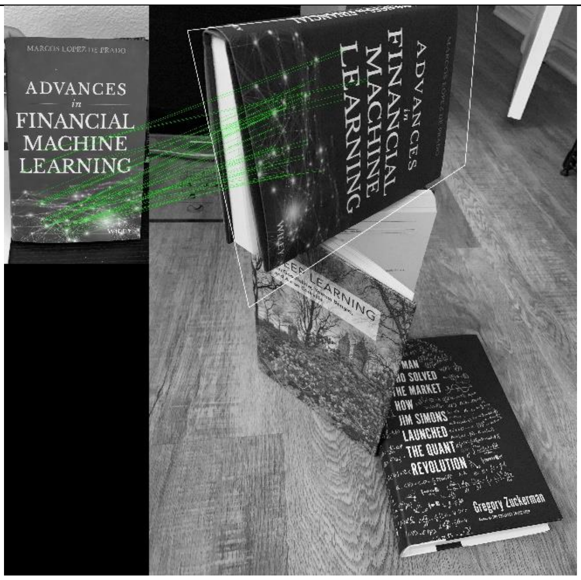


Src2-Dst1

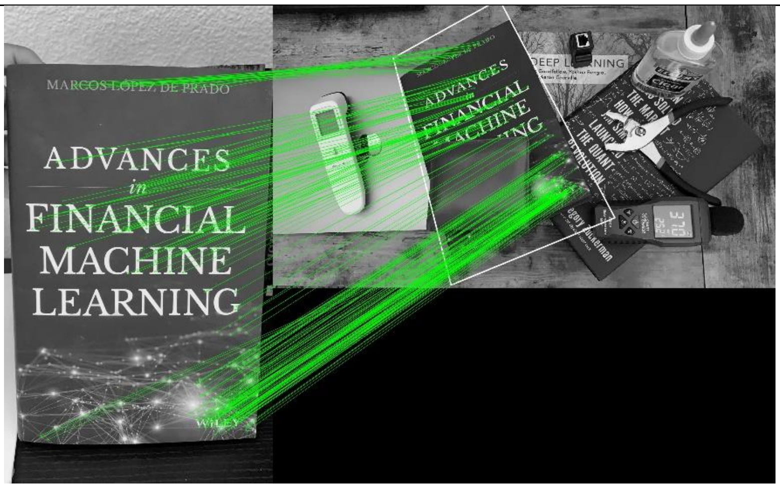
Fig2: Top20 matches before RANSAC

c) Total # of inliner matches after Homography estimation. Top-10 matches

	Src0_dst0	Src0_dst1	Src1_dst0	Src1_dst1	Src2_dst0	Src2_dst1
# inliners	**47**	**242**	**29**	**426**	**634**	**401**
% inliners of total good matches	**21.46%**	**45.74%**	**18.71%**	**70.29%**	**51.08%**	**38.299**



Src0-Dst0



Src0-Dst1

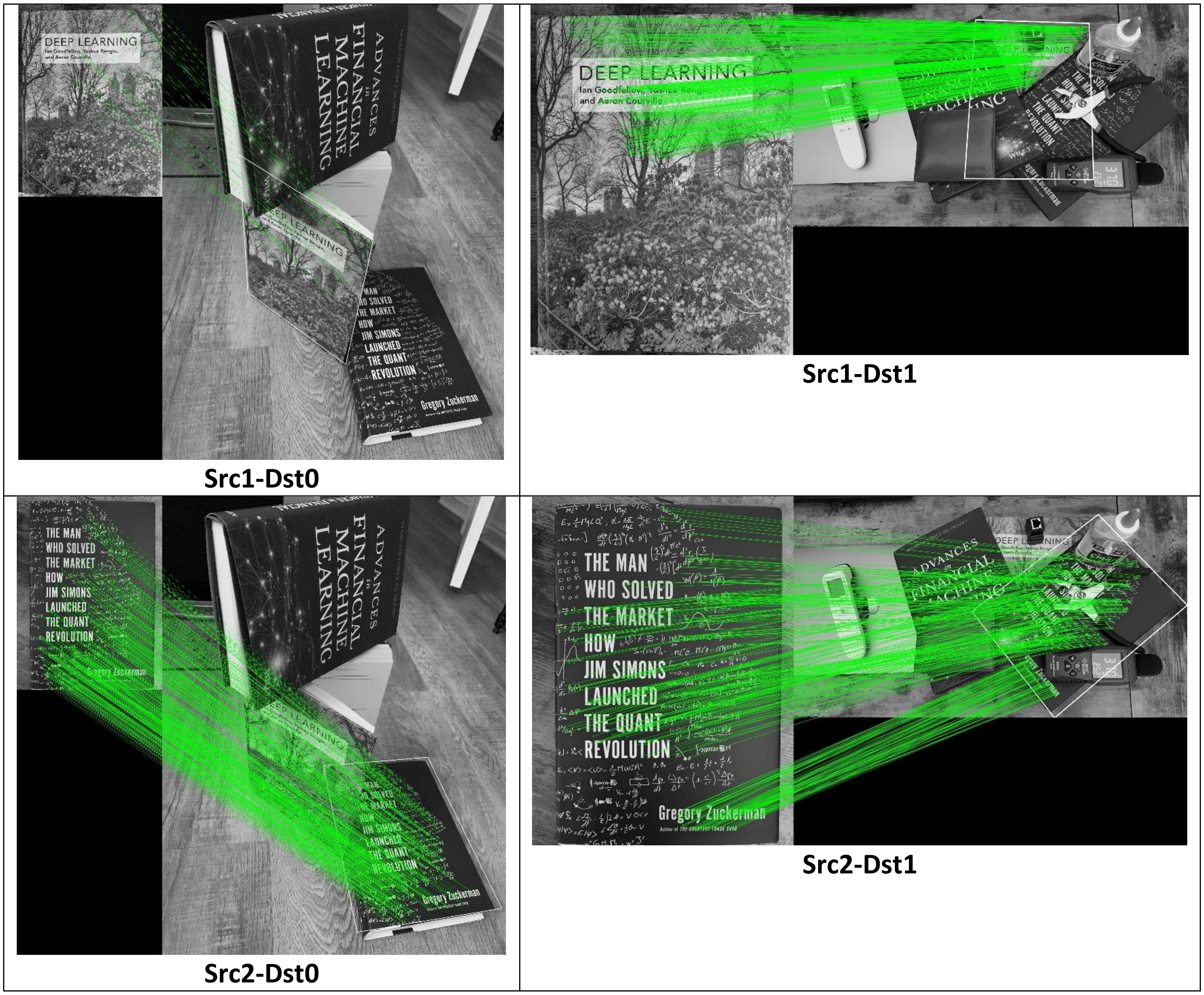


Fig3: Inliners after Homography Estimation

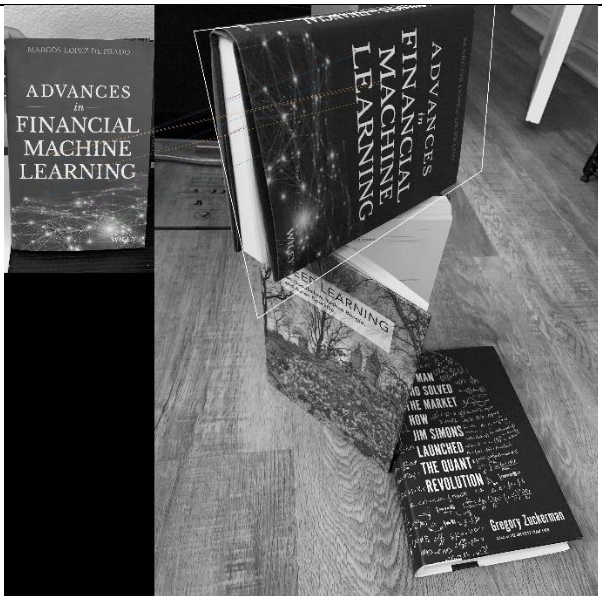
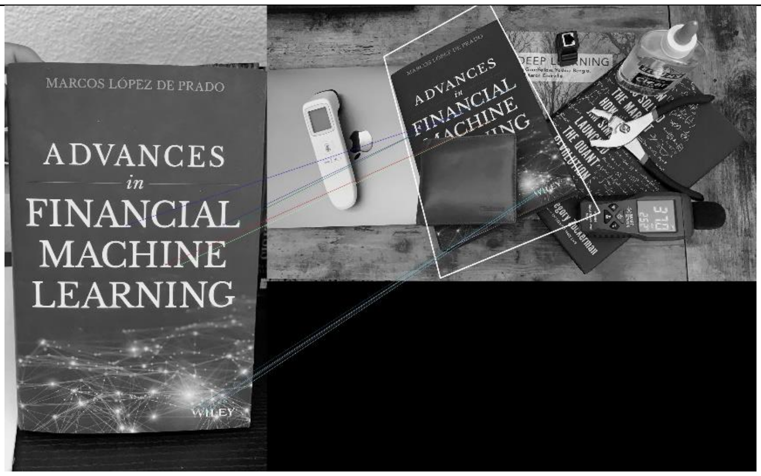
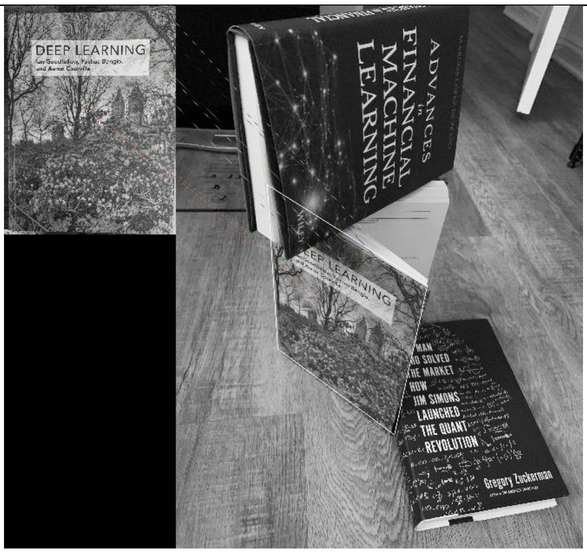
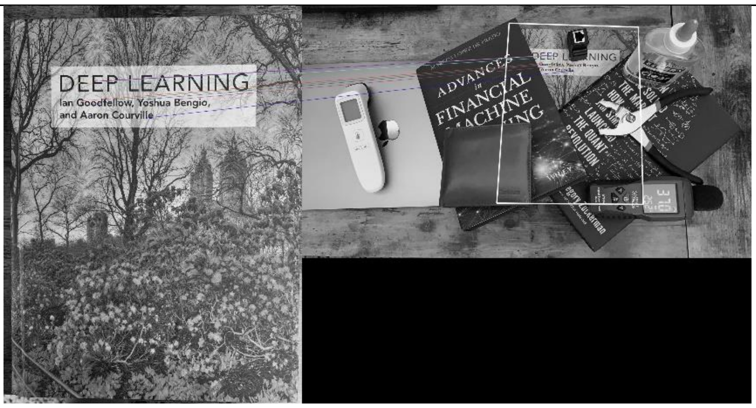
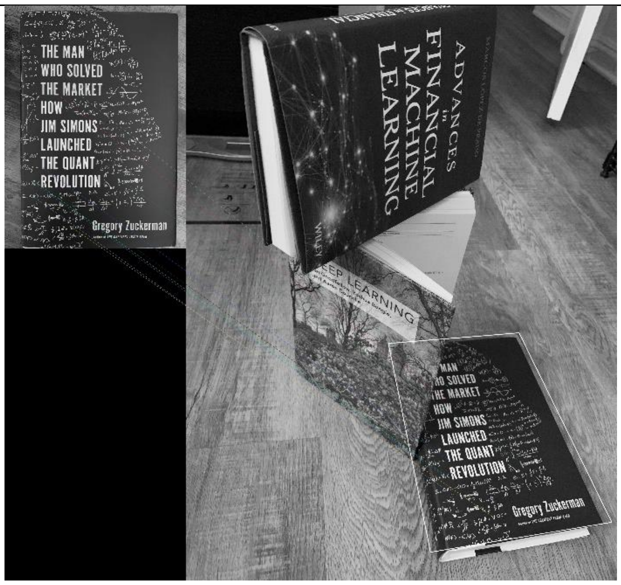
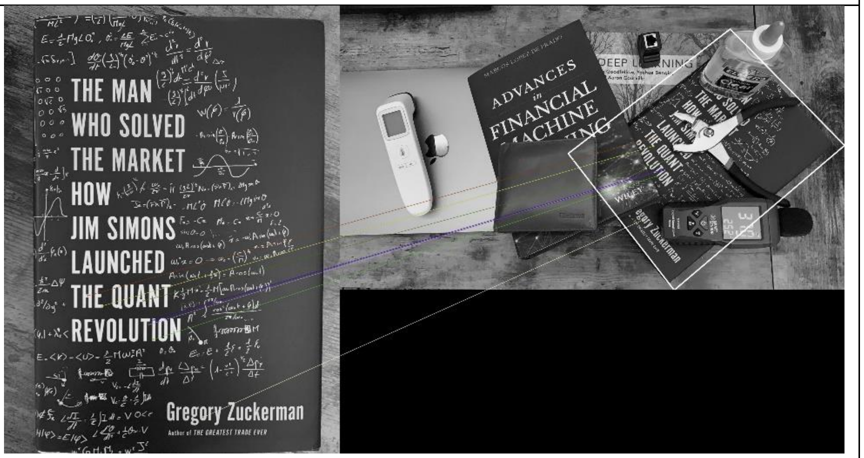
 Src0-Dst0	 Src0-Dst1
 Src1-Dst0	 Src1-Dst1
 Src2-Dst0	 Src2-Dst1

Fig4: Top10 matches after Homography estimation

d) Homography matrix

Src0_dst0	$\begin{pmatrix} 0.350942 & -1.20062 & 2327.03 \\ 1.37674 & 0.077976 & -54.2872 \\ 0.000198418 & -0.00027425 & 1 \end{pmatrix}$
Src0_dst1	$\begin{pmatrix} 0.3454 & 0.00416 & 459.783 \\ -0.2176 & 0.2921 & 200.66 \\ -0.0000564 & -0.000171 & 1 \end{pmatrix}$
Src1_dst0	$\begin{pmatrix} 0.78429 & 0.2001 & 679.394 \\ 0.4208 & 1.233 & 1331.6 \\ -0.0000794 & 0.000203 & 1 \end{pmatrix}$
Src1_dst1	$\begin{pmatrix} 0.4523 & -0.0893 & 884.559 \\ 0.0146 & 0.3939 & 75.437 \\ 0.0000134 & -0.0000697 & 1 \end{pmatrix}$
Src2_dst0	$\begin{pmatrix} 0.8199 & -0.1362 & 1419.08 \\ 0.041 & 0.1679 & 2369.21 \\ 0.0000429 & -0.00018 & 1 \end{pmatrix}$
Src2_dst1	$\begin{pmatrix} 0.21195 & -0.3924 & 1485.97 \\ 0.31 & 0.2583 & 89.992 \\ -0.0000696 & -0.0000347 & 1 \end{pmatrix}$

Code implementation:

1) We read our input Image data which consists of 3 source image and 2 destination images. Objective is to locate the source image in each destination image.
2) First, we convert our image files to grayscale image using cv2.cvtcolor function.
3) Then, we detect keypoints using SIFT on both the source and destination images.
4) We draw the keypoints on the images using cv2.drawKeypoints
5) Now, the keypoints are matched using BFMatcher.knnMatch WITH k=2. This matches every keypoint in source image to a keypoint in destination image.
6) Hence, we compute the good matches by applying ratio test. The resulting array of matches is sorted using distance as the metric. Of these we draw the top 20 matches for each pair.
7) Now, we apply RANSAC to find global matches and calculate the HOMOGRAPHY matrix. Homography matrix is a 3x3 matrix and we set the last element to be 1. Hence, we would be requiring 8 equations i.e 4 point matches at least to compute the Homography matrix. We do this using cv2.findHomography function with one of the parameters as cv2.RANSAC
8) We apply the perspective transformation to the corner coordinates of source image using this Homography matrix and then draw the contour on our destination image using the computed coordinates. This basically draws a contour around (rectangle shape) around our target object in destination image.
9) Now, for drawing the top10 matches that have the minimum error between projected source keypoint and destination keypoint, we do the following:

a. We loop over all the good matches computed and check if the output masked matrix (computed when finding Homography matrix) at this particular index is equal to 1 or not. IF yes, then we store this matched Keypoints in an array, 'matches_masked'.

b. We sort this array 'matches_masked' on basis of distance.

c. Then we draw the the top10 matches of this sorted 'matches_masked' array using cv2.drawMatches function.

10) Now we draw all the inliners for each pair using the computed 'good_matches'. We make sure that we draw only the inliners by specifying the parameter 'matchesMask'.

11) All the required images are displayed and saved using cv2.imshow and cv2.imwrite functions respectively.

Analysis:

1) **The total number of SIFT features changes drastically if we resize the image. On downscaling the image, we lose many keypoints and hence some essential information.**
2) **Destination 0 has the highest number of keypoints.**
3) **After applying the ratio test, %of good matches of the total matches is highest in Src0-Dst1 pair (10.05%) and lowest in Src1-Dst0 pair (0.36%). This can be due to similar patterns in Src0 image as the branches of trees are very similar and hence there might have been a lot of local matches.**
4) **After RANSAC and applying Homography transformation, Number of good matches consistent with Homography estimation is highest in Src1-Dst1 pair (70.29%) and least in Src0-Dst0 pair (21.46%)**
5) **Quality Analysis table:**

Source image	Destination image	Total matches	Good matches	Matches after Homography estimation
0	0	**5259**	**219 (4.16%)**	**47 (21.46%)**
0	1	**5259**	**529 (10.05%)**	**242 (45.74%)**
1	0	**42457**	**155 (0.36%)**	**29 (18.71%)**
1	1	**42457**	**606 (1.42%)**	**426 (70.29%)**
2	0	**13004**	**1241 (9.54)**	**634 (51.08%)**
2	1	**13004**	**1047 (8.05)**	**401 (38.29%)**

6) **It can be inferred that finding source 0 and source 1 image is easier in destination 1 image. Whereas finding source 2 is slightly better in destination 0.**
7) **It can also be commented that, our program is reasonably robust in finding the target among several objects present in the destination image.**

8) **Also, our program was able to locate the target objects even if the orientation and size were changed in destination image.**
9) **Although, it's visible that our program faced difficulty against areas in destination image which had occlusions with our target object, particularly when Src2-dst0 and src2-dst1 pair are compared.**