Yiyue Zhang

Requirements:

According to the homework description, the goal is to use SIFT features to locate desired objects in images that may contain other objects, occlusion and general clutter. It is assumed that the SIFT points on objects will arise primarily from a planar surface and that they can be matched in the other view by a homography transformation.

According to homework requirements, in this assignment, we need to display some intermediate results to understand and show the internal workings of the program. They are:

1. SIFT features: show the detected features overlaid on the images (just the location and directions and not the 128 dimensional vectors). Also give the number of features that are found in each image.

2. Show, graphically, the top 20 scoring matches found by the matcher before RANSAC operation is applied. Provide statistics of how many matches are found (per image pair)

3. Show the top 10 (or more) matches that are found after homography has been computed; also provide the total numbers consistent with the computed homography.

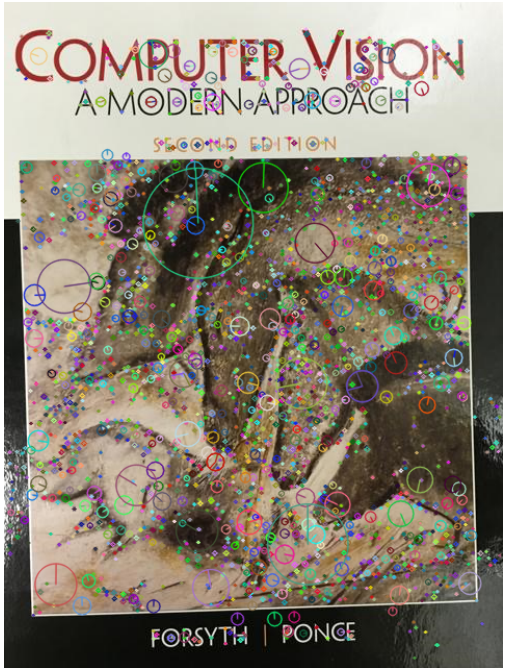
4. Output the computed homography matrix.

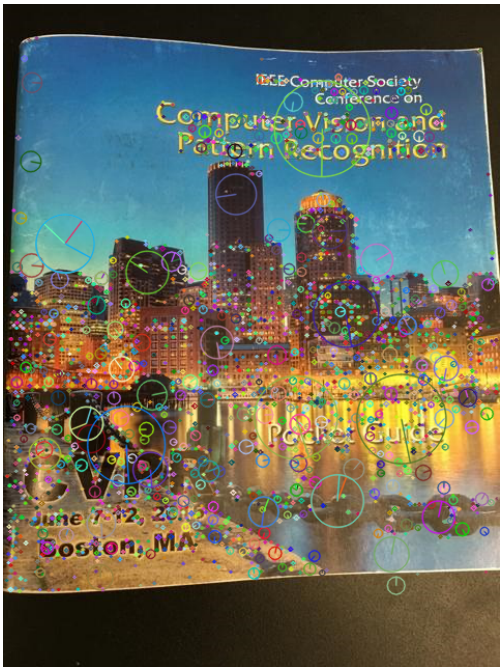
5. The built in homography finder also applies a non-linear optimization step at the end; you can ignore the details of this step or disable it if you wish.

Brief description of my program:

```
'''read the images'''
'''Convert images to gray scale '''
''' threshold for the matches '''
'''Create the SIFT Class '''
'''Get key points and descriptors for each image'''
'''Number of Key points found'''
'''Visualize Key Points in Images and Save them for Report'''
'''Create a BF matcher'''
'''          Experiments          '''
'''Match and Sort image 1 and image 3'''
'''Match and Sort image 1 and image 4'''
'''Match and Sort image 1 and image 5'''
'''Match and Sort image 2 and image 3'''
'''Match and Sort image 2 and image 4'''
'''Match and Sort image 2 and image 5'''
'''Save top  20  scoring  matches'''
'''Preprocessing and Find the Matrix for RANSAC homography'''
'''Output the computed homography matrix'''
'''Display them in images'''
```

Results:





```
# keypoints in image1: 3253

# keypoints in image2: 2815

# keypoints in image3: 1722

# keypoints in image4: 1623

# keypoints in image5: 1655
```

Yiyue Zhang

Top 20 Matches

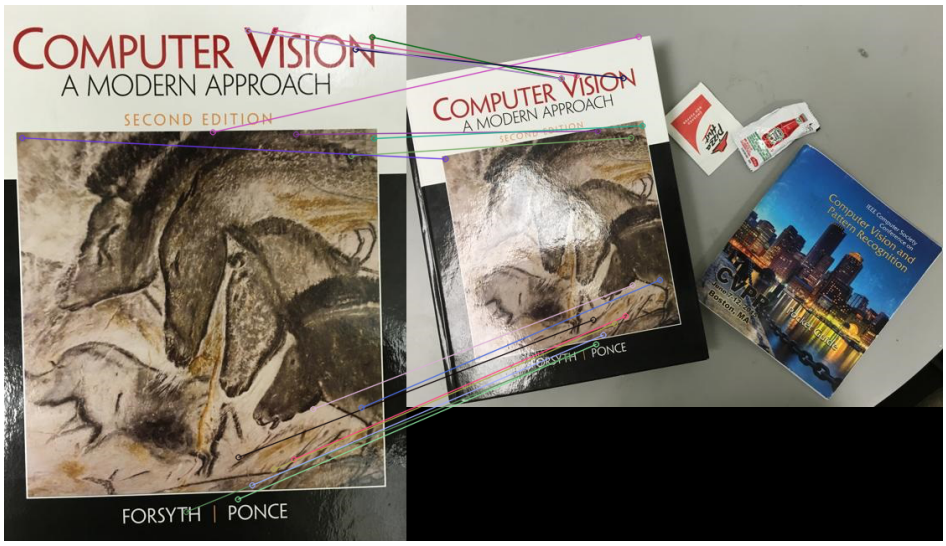


Image1 and Image3

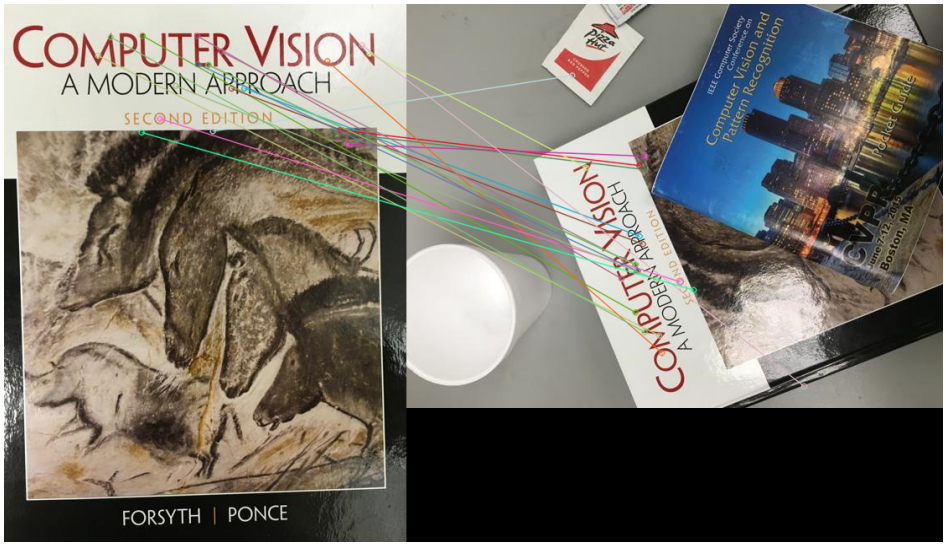
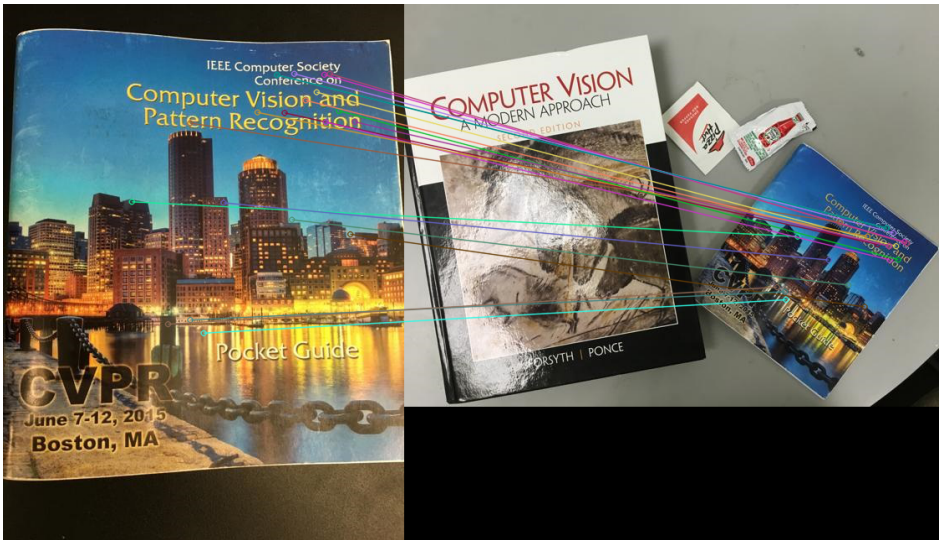


Image1 and Image4

Image1 and Image5

Yiyue Zhang



Top 20 Matches

Image2 and Image3

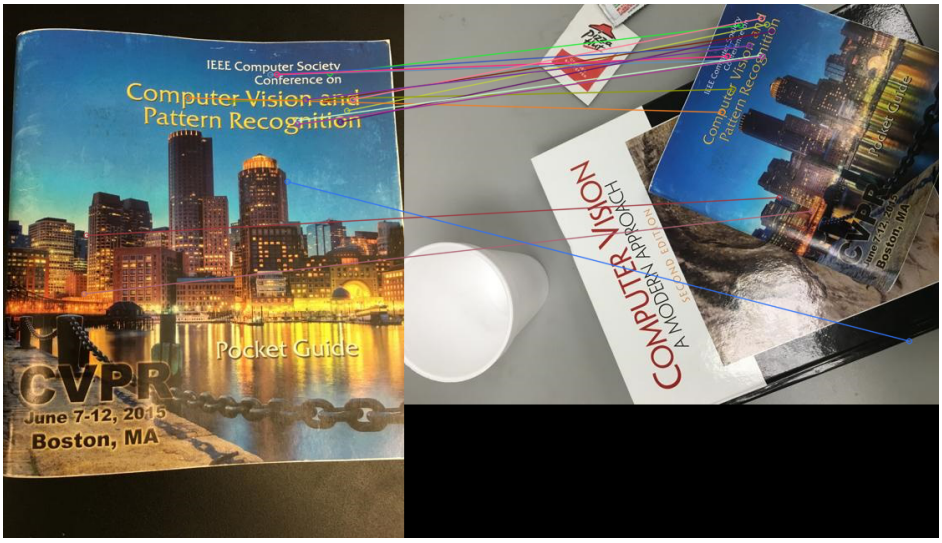


Image2 and Image4

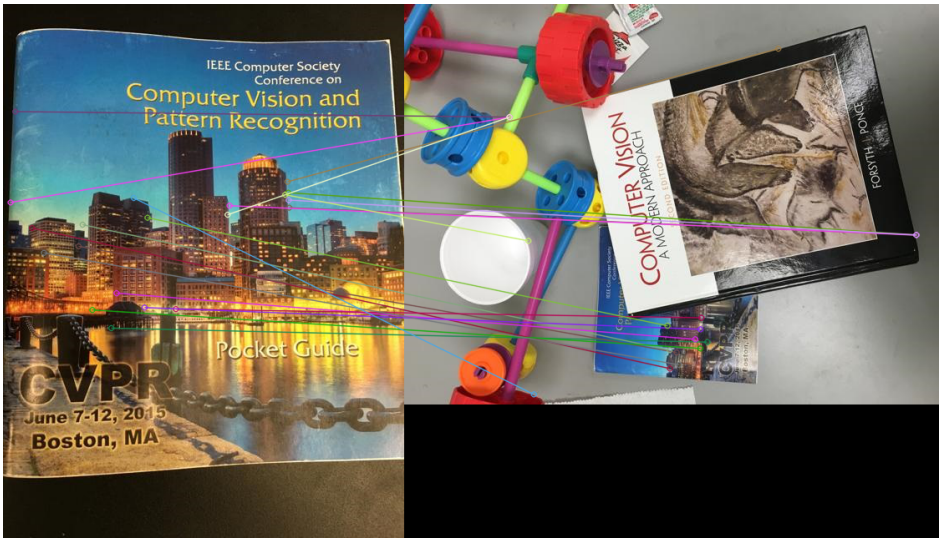


Image2 and Image5

```
# Matches in image1 and image3: 3253

# Matches in image1 and image4: 3253

# Matches in image1 and image5: 3253

# Matches in image2 and image3: 2815

# Matches in image2 and image4: 2815

# Matches in image2 and image5: 2815
```

# of Matches in each image pair

```
# consistent matches with  the computed homography in image1 and image3: 88

# consistent matches with  the computed homography in image1 and image4: 81

# consistent matches with  the computed homography in image1 and image5: 100

# consistent matches with  the computed homography in image2 and image3: 87

# consistent matches with  the computed homography in image2 and image4: 82

# consistent matches with  the computed homography in image2 and image5: 30
```

Consistent Matches between each image pair after RANSAC

Yiyue Zhang

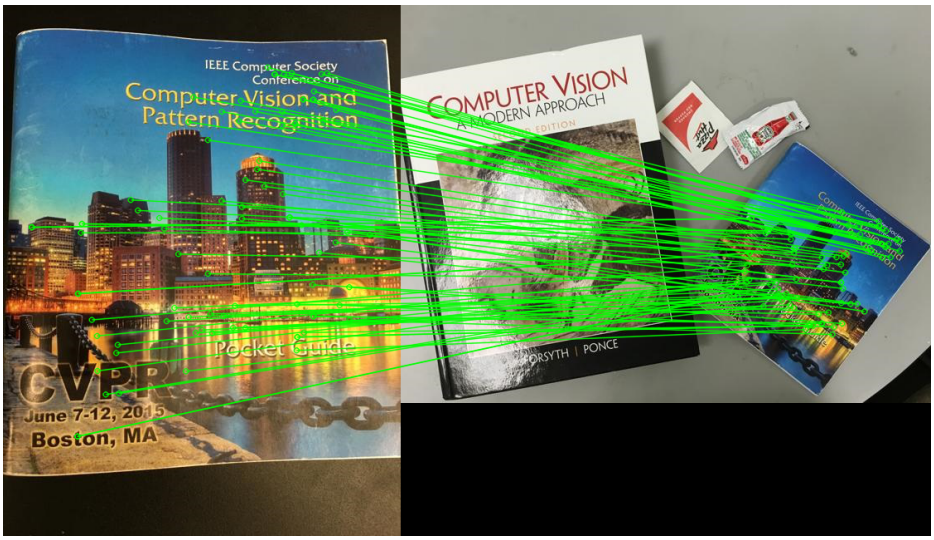
After RANSAC

Image1 and Image3

Image1 and Image4

Image1 and Image5

Yiyue Zhang



After RANSAC

Image2 and Image3

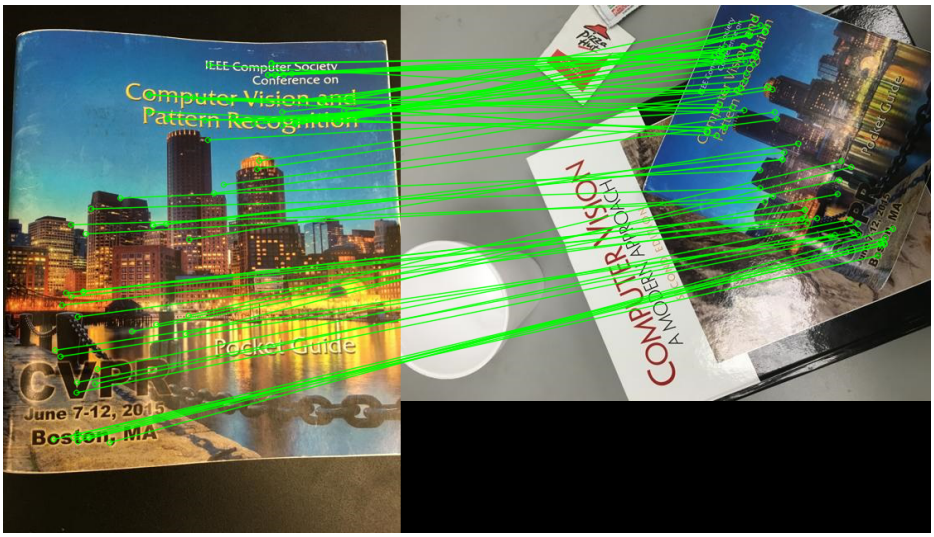


Image2 and Image4

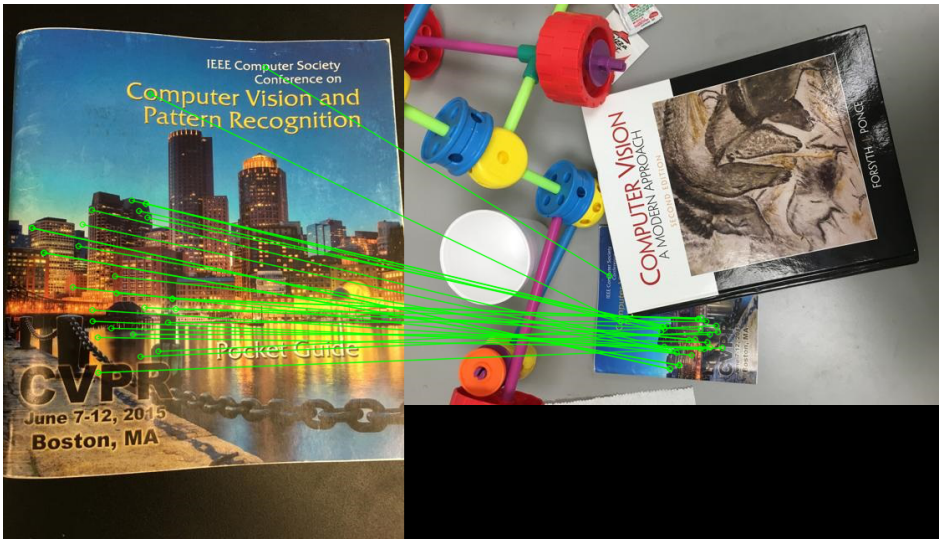


Image2 and Image5

Homography Matrices

```
The homography matrix for image1 and image3:

[[  5.38523484e-01   7.23306319e-02   2.25143349e+01]
 [ -1.03468522e-01   5.31339912e-01   1.00010851e+02]
 [ -5.74274784e-05  -5.73968543e-05   1.00000000e+00]]


----------------------------------------


The homography matrix for image1 and image4:

[[ -2.57923182e-01   5.35559084e-01   2.93149914e+02]
 [ -6.08309789e-01  -3.02844390e-01   4.76121752e+02]
 [  9.98655836e-06  -1.92685188e-04   1.00000000e+00]]


----------------------------------------


The homography matrix for image1 and image5:

[[ -1.19249834e-01   4.05292022e-01   2.74684328e+02]
 [ -4.77061548e-01  -1.33516163e-01   3.45180773e+02]
 [ -5.59272043e-05  -1.63752600e-04   1.00000000e+00]]

The homography matrix for image2 and image3:

[[  2.52211826e-01  -1.67330905e-01   4.82667963e+02]
 [  2.09517631e-01   4.26645242e-01   1.50578735e+02]
 [ -1.94730378e-04   2.34262823e-04   1.00000000e+00]]


----------------------------------------


The homography matrix for image2 and image4:

[[  1.39183804e-01   6.03966210e-01   2.70114293e+02]
 [ -5.60773519e-01   2.39070258e-01   2.22525461e+02]
 [ -1.84446245e-04   8.98188636e-05   1.00000000e+00]]


----------------------------------------


The homography matrix for image2 and image5:

[[ -4.43722580e-02   4.26960083e-01   2.16131722e+02]
 [ -4.48102804e-01   8.61984281e-02   4.44442400e+02]
 [ -1.78183806e-04   1.57502893e-04   1.00000000e+00]]
```

Yiyue Zhang

Analysis:

By applying SIFT features, I plotted the key points on top of each image. I saw most of the key points contain feature information of the image. There are also a few points drawn on the background (grey or black desk). And the number of outliers is proportional to the complexity of the image background. For example, image 3, 4, 5 contains more objects, so their backgrounds are more complex than image 1 and 2. Thus, there are more outliers in image 3, 4, 5 than that of in image 1 and 2. This shows, although, most of SIFT are meaningful, there are sill some outliers. We need to pay attention to these outliers before using SIFT features. However, among around 1.5k to 3.5k key points, there are only a few outliers. Thus, I think the SIFT features are meaningful and can represent the image.

Similarly, after I apply RANSAC, I saw most of the matches are correct, but there are a few mismatches too, such as the mismatch between the book and the “pizzahut" white bag between image 1 and image 4. Again, by comparing each experiment, it show images with a complex background tend to have more mismatches. And, if part of the object is blocked (image 2 and image5,) there are more mismatches. I think this is because the BFMatcher always find a match pair for all key points between images. If it cannot find the “correct” pair, it will assign it to the most “closest” pair. This explains why the matches between image 1 and 3/4/5 are always 3254, and the matches between image 2 and 3/4/5 are always 2815.

In the last part, we are required to apply homography RANSAC. I used a subset of all matches that have top scores. After applying homography to RANSAC, I saw a even better matching results. In general, there are less mismatches than that of before applying homography. In its plot, I plotted the top 100 matches. The result shows this method solves the mismatch problem that caused by “blocking” part of the object (image 2 and image5).

```
import cv2
import numpy as np

'''read the images'''
image_1 = cv2.imread('/Users/YiyueZhang/Downloads/hw3Data/image_1.jpg')
image_2 = cv2.imread('/Users/YiyueZhang/Downloads/hw3Data/image_2.jpg')
image_3 = cv2.imread('/Users/YiyueZhang/Downloads/hw3Data/image_3.jpg')
image_4 = cv2.imread('/Users/YiyueZhang/Downloads/hw3Data/image_4.jpg')
image_5 = cv2.imread('/Users/YiyueZhang/Downloads/hw3Data/image_5.jpg')

'''Convert images to gray scale '''
gray_1 = cv2.cvtColor(image_1, cv2.COLOR_BGR2GRAY)
gray_2 = cv2.cvtColor(image_2, cv2.COLOR_BGR2GRAY)
gray_3 = cv2.cvtColor(image_3, cv2.COLOR_BGR2GRAY)
gray_4 = cv2.cvtColor(image_4, cv2.COLOR_BGR2GRAY)
gray_5 = cv2.cvtColor(image_5, cv2.COLOR_BGR2GRAY)


''' threshold for the matches '''
thresh = 100


'''Create the SIFT Class '''
sift = cv2.xfeatures2d.SIFT_create()

'''Get key points and descriptors for each image'''
keypoint_1, descriptor_1 = sift.detectAndCompute(gray_1, None)
keypoint_2, descriptor_2 = sift.detectAndCompute(gray_2, None)
keypoint_3, descriptor_3 = sift.detectAndCompute(gray_3, None)
keypoint_4, descriptor_4 = sift.detectAndCompute(gray_4, None)
keypoint_5, descriptor_5 = sift.detectAndCompute(gray_5, None)

'''Number of Key points found'''
print ('---------------------------------------')
print ('\n')
print ('# keypoints in image1:', len(keypoint_1))
print ('\n')
print ('# keypoints in image2:', len(keypoint_2))
print ('\n')
print ('# keypoints in image3:', len(keypoint_3))
print ('\n')
print ('# keypoints in image4:', len(keypoint_4))
print ('\n')
print ('# keypoints in image5:', len(keypoint_5))
print ('\n')
print ('---------------------------------------')
print ('\n')

'''Visualize Key Points in Images and Save them for Report'''
img1 = cv2.drawKeypoints(image_1, keypoint_1, None, flags=cv2.
    DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/keypoints_image_01.png', img1)

img2 = cv2.drawKeypoints(image_2, keypoint_2, None, flags=cv2.
    DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/keypoints_image_02.png', img2)
```

```
img3 = cv2.drawKeypoints(image_3, keypoint_3, None, flags=cv2.
    DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/keypoints_image_03.png', img3)

img4 = cv2.drawKeypoints(image_4, keypoint_4, None, flags=cv2.
    DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/keypoints_image_04.png', img4)

img5 = cv2.drawKeypoints(image_5, keypoint_5, None, flags=cv2.
    DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/keypoints_image_05.png', img5)

'''Create a BF matcher'''
bf = cv2.BFMatcher()


'''           Experiments            '''
'''Match and Sort image 1 and image 3'''
'''Match and Sort image 1 and image 4'''
'''Match and Sort image 1 and image 5'''
'''Match and Sort image 2 and image 3'''
'''Match and Sort image 2 and image 4'''
'''Match and Sort image 2 and image 5'''
bf13 = bf.match(descriptor_1, descriptor_3)
bf13_sorted = sorted(bf13, key = lambda x:x.distance)

bf14 = bf.match(descriptor_1, descriptor_4)
bf14_sorted = sorted(bf14, key = lambda x:x.distance)

bf15 = bf.match(descriptor_1, descriptor_5)
bf15_sorted = sorted(bf15, key = lambda x:x.distance)

bf23 = bf.match(descriptor_2, descriptor_3)
bf23_sorted = sorted(bf23, key = lambda x:x.distance)

bf24 = bf.match(descriptor_2, descriptor_4)
bf24_sorted = sorted(bf24, key = lambda x:x.distance)

bf25 = bf.match(descriptor_2, descriptor_5)
bf25_sorted = sorted(bf25, key = lambda x:x.distance)

print ('----------------------------------------')
print ('\n')
print ('# Matches in image1 and image3:', len(bf13))
print ('\n')
print ('# Matches in image1 and image4:', len(bf14))
print ('\n')
print ('# Matches in image1 and image5:', len(bf15))
print ('\n')
print ('# Matches in image2 and image3:', len(bf23))
print ('\n')
print ('# Matches in image2 and image4:', len(bf24))
print ('\n')
print ('# Matches in image2 and image5:', len(bf25))
print ('\n')
```

```
print ('----------------------------------------')
print ('\n')



'''Save top  20  scoring  matches'''
bf13_img = cv2.drawMatches(image_1, keypoint_1, image_3, keypoint_3,
    bf13_sorted[:20], None, flags=2)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/top_match13.png', bf13_img)

bf14_img = cv2.drawMatches(image_1, keypoint_1, image_4, keypoint_4,
    bf14_sorted[:20], None, flags=2)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/top_match14.png', bf14_img)

bf15_img = cv2.drawMatches(image_1, keypoint_1, image_5, keypoint_5,
    bf15_sorted[:20], None, flags=2)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/top_match15.png', bf15_img)

bf23_img = cv2.drawMatches(image_2, keypoint_2, image_3, keypoint_3,
    bf23_sorted[:20], None, flags=2)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/top_match23.png', bf13_img)

bf24_img = cv2.drawMatches(image_2, keypoint_2, image_4, keypoint_4,
    bf24_sorted[:20], None, flags=2)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/top_match24.png', bf14_img)

bf25_img = cv2.drawMatches(image_2, keypoint_2, image_5, keypoint_5,
    bf25_sorted[:20], None, flags=2)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/top_match25.png', bf15_img)



'''Preprocessing and Find the Matrix for RANSAC homography'''
src13 = np.float32([ keypoint_1[m.queryIdx].pt for m in bf13_sorted[:thresh] ])
    .reshape(-1,1,2)
dst13 = np.float32([ keypoint_3[m.trainIdx].pt for m in bf13_sorted[:thresh] ])
    .reshape(-1,1,2)
acc13, mask_a13 = cv2.findHomography(src13, dst13, cv2.RANSAC, 0.0)
rough13, mask_r13 = cv2.findHomography(src13, dst13, cv2.RANSAC, 5.0)

src14 = np.float32([ keypoint_1[m.queryIdx].pt for m in bf14_sorted[:thresh] ])
    .reshape(-1,1,2)
dst14 = np.float32([ keypoint_4[m.trainIdx].pt for m in bf14_sorted[:thresh] ])
    .reshape(-1,1,2)
acc14, mask_a14 = cv2.findHomography(src14, dst14, cv2.RANSAC, 0.0)
rough14, mask_r14 = cv2.findHomography(src14, dst14, cv2.RANSAC, 5.0)

src15 = np.float32([ keypoint_1[m.queryIdx].pt for m in bf15_sorted[:thresh] ])
    .reshape(-1,1,2)
dst15 = np.float32([ keypoint_5[m.trainIdx].pt for m in bf15_sorted[:thresh] ])
    .reshape(-1,1,2)
acc15, mask_a15 = cv2.findHomography(src15, dst15, cv2.RANSAC, 0.0)
rough15, mask_r15 = cv2.findHomography(src15, dst15, cv2.RANSAC, 5.0)

src23 = np.float32([ keypoint_2[m.queryIdx].pt for m in bf23_sorted[:thresh] ])
    .reshape(-1,1,2)
```

```
dst23 = np.float32([ keypoint_3[m.trainIdx].pt for m in bf23_sorted[:thresh] ])
    .reshape(-1,1,2)
acc23, mask_a23 = cv2.findHomography(src23, dst23, cv2.RANSAC, 0.0)
rough23, mask_r23 = cv2.findHomography(src23, dst23, cv2.RANSAC, 5.0)

src24 = np.float32([ keypoint_2[m.queryIdx].pt for m in bf24_sorted[:thresh] ])
    .reshape(-1,1,2)
dst24 = np.float32([ keypoint_4[m.trainIdx].pt for m in bf24_sorted[:thresh] ])
    .reshape(-1,1,2)
acc24, mask_a24 = cv2.findHomography(src24, dst24, cv2.RANSAC, 0.0)
rough24, mask_r24 = cv2.findHomography(src24, dst24, cv2.RANSAC, 5.0)

src25 = np.float32([ keypoint_2[m.queryIdx].pt for m in bf25_sorted[:thresh] ])
    .reshape(-1,1,2)
dst25 = np.float32([ keypoint_5[m.trainIdx].pt for m in bf25_sorted[:thresh] ])
    .reshape(-1,1,2)
acc25, mask_a25 = cv2.findHomography(src25, dst25, cv2.RANSAC, 0.0)
rough25, mask_r25 = cv2.findHomography(src25, dst25, cv2.RANSAC, 5.0)


print ('-----------------------------------------------------------------------')
print ('\n')
print ('# matches per image pair in image1 and 3:', np.count_nonzero(mask_a13))
print ('\n')
print ('# matches per image pair in image1 and 4:', np.count_nonzero(mask_a14))
print ('\n')
print ('# matches per image pair in image1 and 5:', np.count_nonzero(mask_a15))
print ('\n')
print ('# matches per image pair in image2 and 3:', np.count_nonzero(mask_a23))
print ('\n')
print ('# matches per image pair in image2 and 4:', np.count_nonzero(mask_a24))
print ('\n')
print ('# matches per image pair in image2 and 5:', np.count_nonzero(mask_a25))
print ('\n')
print ('-----------------------------------------------------------------------')
print ('\n')

print ('-----------------------------------------------------------------------
    ----------------------------')
print ('\n')
print ('# consistent matches with  the computed homography in image1 and
    image3:', np.count_nonzero(mask_r13))
print ('\n')
print ('# consistent matches with  the computed homography in image1 and
    image4:', np.count_nonzero(mask_r14))
print ('\n')
print ('# consistent matches with  the computed homography in image1 and
    image5:', np.count_nonzero(mask_r15))
print ('\n')
print ('# consistent matches with  the computed homography in image2 and
    image3:', np.count_nonzero(mask_r23))
print ('\n')
print ('# consistent matches with  the computed homography in image2 and
    image4:', np.count_nonzero(mask_r24))
print ('\n')
print ('# consistent matches with  the computed homography in image2 and
```

```
    image5:', np.count_nonzero(mask_r25))
print ('\n')
print ('-------------------------------------------------------------------------
    ----------------------------')
print ('\n')


'''Output the computed homography matrix'''
print ('-----------------------------------------')
print ('\n')
print ('The homography matrix for image1 and image3:\n')
print (acc13)
print ('\n')
print ('-----------------------------------------')
print ('\n')
print ('The homography matrix for image1 and image4:\n')
print (acc14)
print ('\n')
print ('-----------------------------------------')
print ('\n')
print ('The homography matrix for image1 and image5:\n')
print (acc15)
print ('\n')
print ('-----------------------------------------')
print ('\n')
print ('The homography matrix for image2 and image3:\n')
print (acc23)
print ('\n')
print ('-----------------------------------------')
print ('\n')
print ('The homography matrix for image2 and image4:\n')
print (acc24)
print ('\n')
print ('-----------------------------------------')
print ('\n')
print ('The homography matrix for image2 and image5:\n')
print (acc25)
print ('\n')
print ('-----------------------------------------')
print ('\n')


'''Display them in images'''
matchem13 = mask_a13.ravel().tolist()
draw13 = dict(matchColor = (0,255,0),
                   singlePointColor = None,
                   matchesMask = matchem13,
                   flags = 2)
draw_m13 = cv2.drawMatches(image_1,keypoint_1,image_3,keypoint_3,bf13_sorted[:
    thresh],None,**draw13)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/final13.png', draw_m13)

matchem14 = mask_a14.ravel().tolist()
draw14 = dict(matchColor = (0,255,0),
                   singlePointColor = None,
                   matchesMask = matchem14,
```

```
                    flags = 2)
draw_m14 = cv2.drawMatches(image_1,keypoint_1,image_4,keypoint_4,bf14_sorted[:
    thresh],None,**draw14)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/final14.png', draw_m14)

matchem15 = mask_a15.ravel().tolist()
draw15 = dict(matchColor = (0,255,0),
                    singlePointColor = None,
                    matchesMask = matchem15,
                    flags = 2)
draw_m15 = cv2.drawMatches(image_1,keypoint_1,image_5,keypoint_5,bf15_sorted[:
    thresh],None,**draw15)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/final15.png', draw_m15)

matchem23 = mask_a23.ravel().tolist()
draw23 = dict(matchColor = (0,255,0),
                    singlePointColor = None,
                    matchesMask = matchem23,
                    flags = 2)
draw_m23 = cv2.drawMatches(image_2,keypoint_2,image_3,keypoint_3,bf23_sorted[:
    thresh],None,**draw23)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/final23.png', draw_m23)

matchem24 = mask_a24.ravel().tolist()
draw24 = dict(matchColor = (0,255,0),
                    singlePointColor = None,
                    matchesMask = matchem24,
                    flags = 2)
draw_m24 = cv2.drawMatches(image_2,keypoint_2,image_4,keypoint_4,bf24_sorted[:
    thresh],None,**draw24)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/final24.png', draw_m24)

matchem25 = mask_a25.ravel().tolist()
draw25 = dict(matchColor = (0,255,0),
                    singlePointColor = None,
                    matchesMask = matchem25,
                    flags = 2)
draw_m25 = cv2.drawMatches(image_2,keypoint_2,image_5,keypoint_5,bf25_sorted[:
    thresh],None,**draw25)
cv2.imwrite('/Users/YiyueZhang/Downloads/hw3Data/final25.png', draw_m25)
```