Assignment 3

QINGTAN SHEN
1130945

This report will be divided into two parts. First part is to describe the steps to convert this algorithm into my code. Second part is to discuss about the performance of these pairs of images.

1. Algorithm steps

The first operation in my code is to find keypoints between these pairs of images, sift method is used in this step, then we find correspondences between these images, in this step, we use Flann based matcher in python. After getting keypoints and correspondences, we plot them as an output.

Then we come to the main part for this algorithm. In step a, we need to scale and shift the points. I found a paper named Revisiting Hartley's Normalized Eight Point Algorithm. My scale and shift operations totally follow this paper. If the point location in left image is $p1 = [m1, m2, 1]$, the point location in right image is $p2 = [m1', m2', 1]$. Then after operation, we have $p1 = \left[\frac{m1-\overline{m1}}{s}, \frac{m2-\overline{m2}}{s}, 1\right]$, $p2 = \left[\frac{m1'-\overline{m1'}}{s'}, \frac{m2'-\overline{m2'}}{s'}, 1\right]$, where $\overline{m1} = \frac{1}{n}\sum_{i=1}^{n} m1_i$, $\overline{m2} = \frac{1}{n}\sum_{i=1}^{n} m2_i$, $s = [(\sum_{i=1}^{n}(m1_i - \overline{m1})^2 + (m2_i - \overline{m2})^2)/2n]^{1/2}$. These formulas will also be used in step i, which will discuss later.

Step b is to compute a design matrix using eight points. A function named random_create_design_matrix was built for this step. We choose 8 pairs of correspondent keypoints randomly, for each pair we have two points [p1,q1] and [p2,q2], which contains the information about their x-position and y-position. Then we use them to create a row of matrix, which is $[p1 * q1, p2 * q1, q1, p1 * q2, p2 * q2, q2, p1, p2, 1]$. We use 8 pairs completing 8 rows for this matrix. We name this design matrix M.

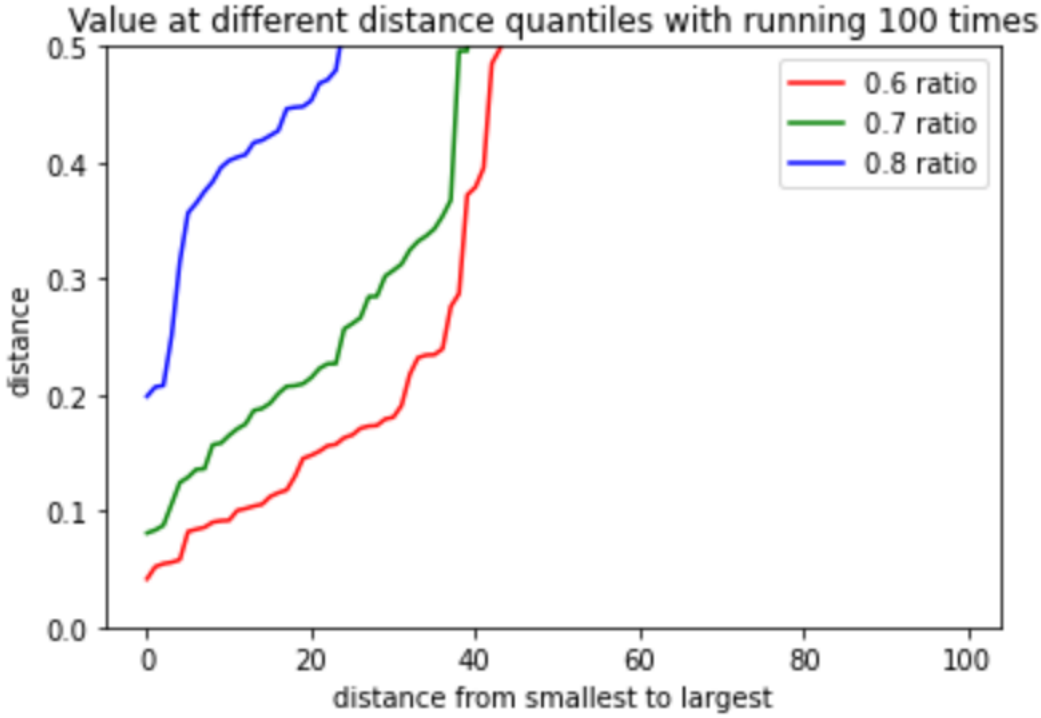
In step c, SVD method is used to find null space for the design matrix M. We have this formular $U * S * V^T = M$, because S is a diagonal matrix, and the smallest value for diagonal value is located at the right bottom of S matrix. So, we can use the last row of V^T as our estimate value of F.

In step d, we use these nine values we get to form a 3*3 matrix F (change the matrix dimension), to make sure it can be used in step e.

In step e, we use SVD method on matrix F, the formula is $U * S * V^T = F$, then we set the right bottom value in matrix S to zero (get S1), then we calculate F by $F = U * S1 * V^T$, to make sure $|F| = 0$.

In step f, the most important part is to define our threshold. My idea is to use the first image, which has middle size of correspondences. I decide to run the 8-point algorithm 100 times, each time we output a distance list, which contains all the distance from the points to the line we drawn, and I sort these 100 distance lists from the smallest to the largest. Then, we use three ratios 0.6, 0.7 and 0.8. We use these three values as three quantiles in our 100 sorted

distance lists. For example, if I choose 0.6 ratio, and the length of our sorted distance list is 200, then we choose the value in the 0.6*200=120 position in this condition. So, for each ratio, we get a list with 100 values, then we sort them and plot them. Here is the plot.



The y scale of this plot is only 0 to 0.5, because other larger values is unimportant in setting threshold. Then we find lines of 0.6 and 0.7 ratios are close to each other, while line of 0.8 ratio is far away from them. So, I choose 0.7 as the ratio threshold to pursue a higher accuracy. Then we can see the lowest 0.7 quantile in these 100 loops are nearly 0.07, I set this threshold to be 0.06, little smaller than 0.07 to pursue a higher accuracy. In conclusion, in this step, two thresholds are defined, the ratio threshold is 0.7, and the value threshold is 0.06. When the distance is greater than 0.06, the point is outlier point. If the distance is smaller than 0.06, it is an inlier point.

In step g, we run the 8-point algorithm 500 times for each pair of images. We use 0.06 as the distance threshold to choose inliers and outliers. After each loop, we count the inlier number, which can be represented as NI, the total point number is NT. Then we use $\frac{NI}{NT}$ to get our ratio. We always trace the largest ratio. If we find that in one loop this ratio is greater than 0.7, then we break this loop and keep it as the result. If we cannot find any loop in these 500 loops which ratio is greater than 0.7, then we use the largest ratio value during these 500 loops as our result. Our result contains many values, such as the F matrix we use for the best loop, and the test points list for this loop. In this step, we find that only the ratio of booksh, box, corr and valbonne plots are greater than 0.7.

Here is the inlier proportion for these 16 images.

```
booksh : 0.7183098591549296
box : 0.7146814404432132
castle : 0.5552763819095478
corr : 0.8951048951048951
graff : 0.09473684210526316
head : 0.6878172588832487
kampa : 0.5780141843971631
Kyoto : 0.41498047568335106
leafs : 0.12299465240641712
plant : 0.30945558739255014
rotunda : 0.3108108108108108
shout : 0.5641025641025641
valbonne : 0.7
wall : 0.41885964912280704
wash : 0.24
zoom : 0.4699248120300752
```

We can see that the graff and leafs images have smallest ratios, more details will be discussed in the result part.

In step h, firstly we use test points list and the best F matrix we get from step g to choose the inlier points. Because it is inconvenience and inefficient to store inlier points in step g directly, so we choose the inlier points again in this step. The result must be the same in the best loop of step g. Then we still use the design matrix creation method we use before, there are more rows in this time. More samples mean more accurate. Here is the row number (sample size) for each plot:

```
booksh : 153
box : 258
castle : 223
corr : 384
graff : 20
head : 274
kampa : 165
Kyoto : 1173
leafs : 49
plant : 116
rotunda : 24
shout : 134
valbonne : 140
wall : 197
wash : 41
zoom : 129
```

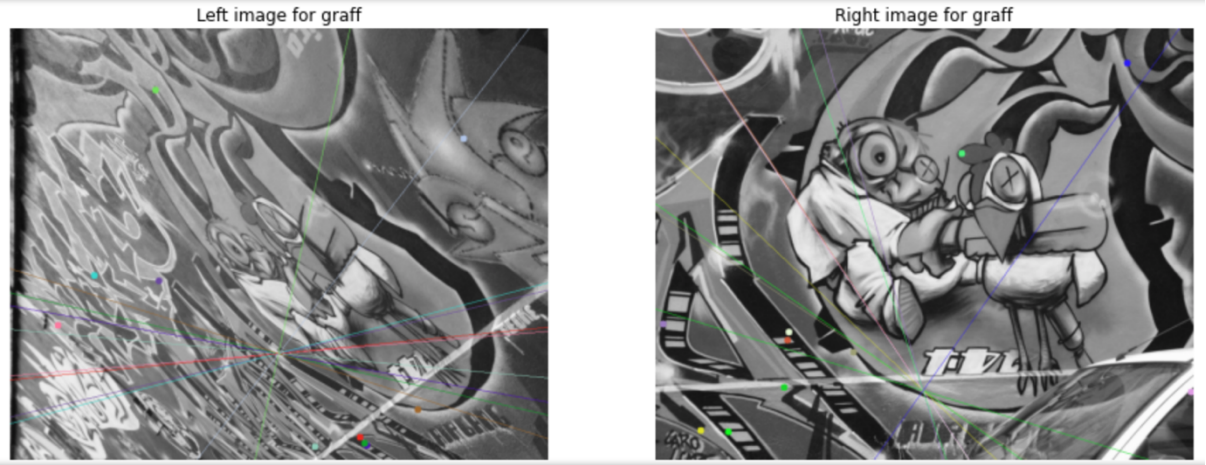
We will discuss this table later in the result part. Then SVD matrix calculation method also provided to get a new F matrix (same steps from b to e).

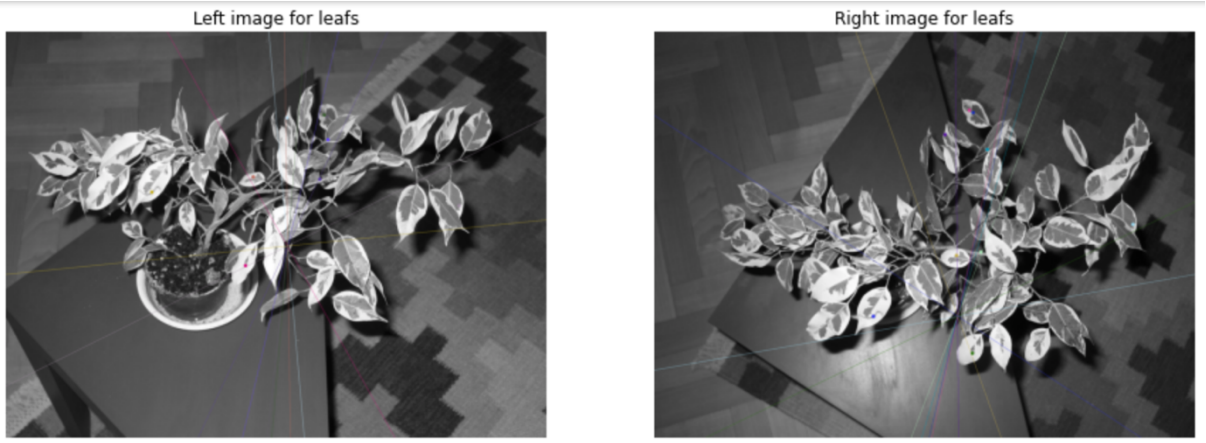
In the last step i, we should undo the effect of step a, it is also a preparation step for visualization. We use matrix calculation method in this step. There are two matrixes $T = \begin{bmatrix} 1/s & 0 & -\overline{m1}/s \\ 0 & 1/s & -\overline{m2}/s \\ 0 & 0 & 1 \end{bmatrix}$, $T' = \begin{bmatrix} 1/s' & 0 & -\overline{m1'}/s' \\ 0 & 1/s' & -\overline{m2'}/s' \\ 0 & 0 & 1 \end{bmatrix}$, where $s, \overline{m1}, \overline{m2}$ are the same meaning in step1 (using left points position value to calculate), $s', \overline{m1'}, \overline{m2'}$ only means

using right points position value to calculate, others are the same. Then we can get our new F matrix $newF = T * F * T'$. We finish our second part. The final F value we get is $newF$.

2. Result discussion

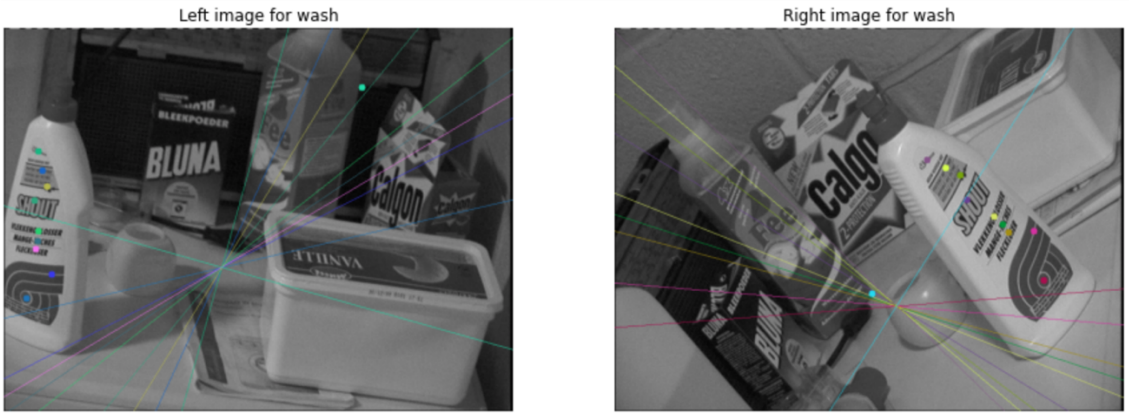
From the inlier proportion in each plot, graffs and leafs have lowest value, So we need to check these two images to find if the inliers are the real correspondent keypoints, here is the plot created in part 3.





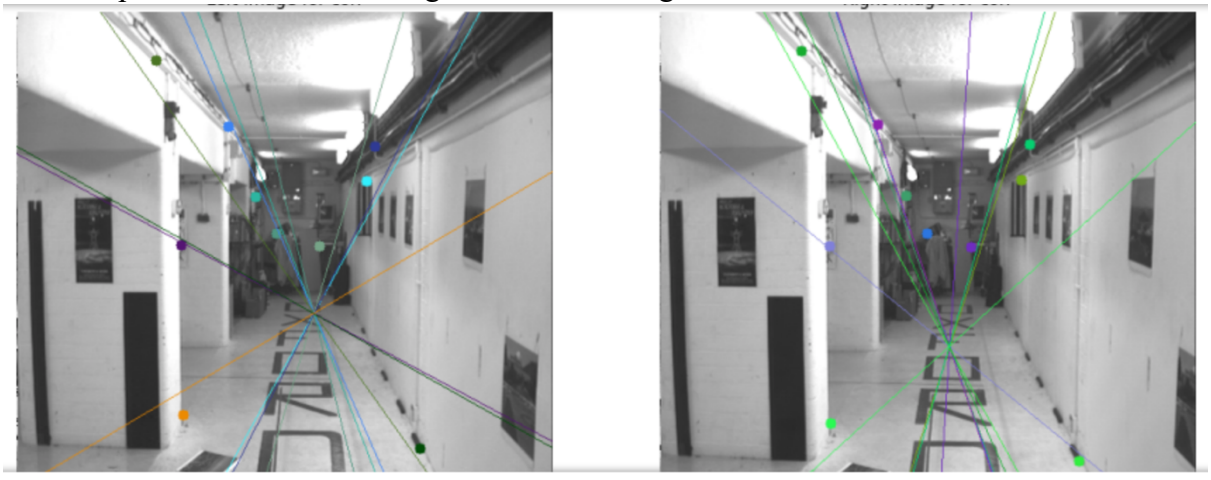
From graffs and leafs images (it is hard to find correspondence relationship because of their image character), we can see that although our threshold is high and only nearly only 10 percent correspondences are chosen as the top matches. But in our images, there are still some wrong correspondence relationships in the 10 samples we have chosen at part three. These wrong correspondence relationships have influence on F value, so the images shown in part three of graffs and leafs are poor.

After that, we talk about sample numbers, except graffs and leafs we discussed, rotunda and wash have smallest sample sizes. Let's see their images.



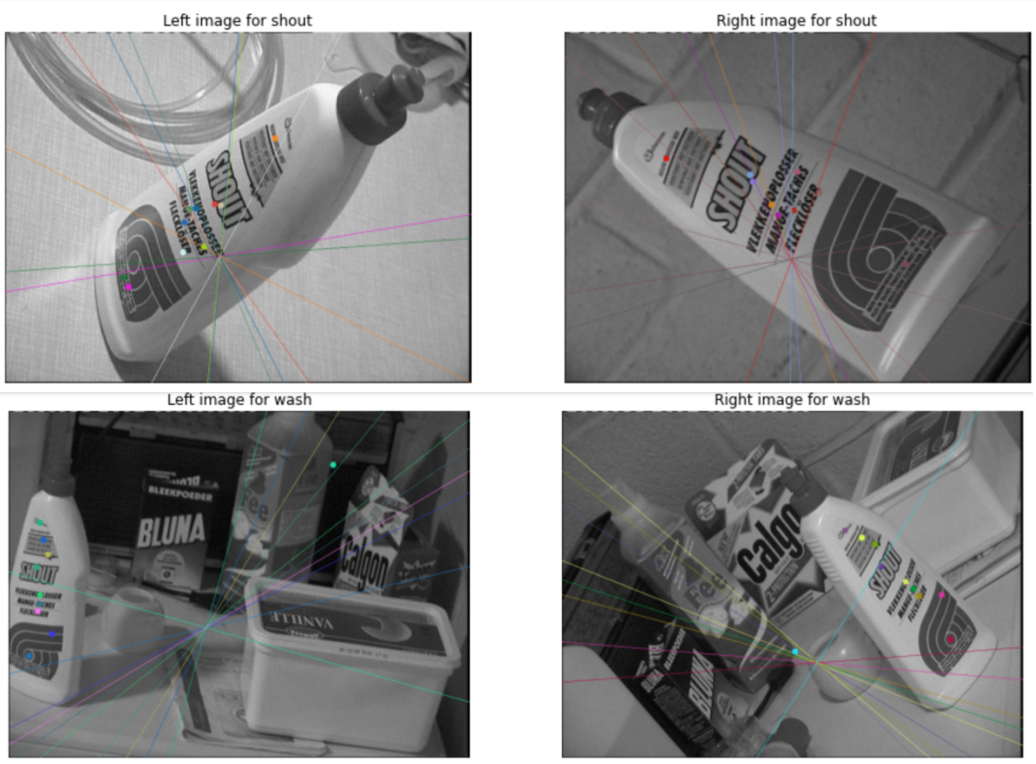
From the images, we can see that some of the keypoint relationships are incorrect. In the rotunda image, the result is not so good (because of the brightness change). And in wash image, the result is worse because we had almost all the points in only one object, these points are in a small scale, which will affect the accuracy of our results.

The best performance is corr image, here is the images:



These pair of images are easy to find correspondences, so there are more than 1000 samples used to calculate the final F, also the correspondence relationships are accurate, also there are only little rotation from left to right, so the result is the best. And the brightness for all the parts is same. The position change is little from left to right, so its result is the best.

It will also be reasonable to compare these two pairs:



We can see the shout image has a better performance, because it only has one object in the image, although the rotation angle is large. While in wash images, there are more than one objects, which is harder to operate.

Then we can have a conclusion in this result part. Firstly, if the image pair is complex itself, which means hard to find correct correspondences between keypoints, then the F matrix will be not accurate. Secondly, if one image has a large rotation and this image has many objects, and some objects may change their shape or brightness in the rotation, then the F matrix may not accurate. Thirdly, if the correspondence points are in a small scale of image, the F matrix may not be the best. Fourthly, if the image zoomed in our out too much, it would cause more mistakes.