ECE661: Computer Vision (Fall 2014)

Shaobo Fang: s-fang@purdue

October 16, 2014

Contents

1 Overview of Image Mosaicking Steps

Although image mosaicking could be done by simply establishing the points correspondences or using the two steps/single step method to establish homography across successive images, it will yield better results if method could be implemented to eliminate outliers in our interest points.

A good result of mosaicked image from 7 images taken at MSEE atrium would be presented as result. Another mosaicked image from 5 images taken outside MSEE building will also be tested.

1. **Interest Points Correspondence Establishment**

   First of all, interest points from successive images should be established. In order to complete the image mosaicking, we only need to find the Homography based on the correct sets of interest points found.

2. **Interest Points Classification: Outliers/Inliers?**

   Although SIFT is already a very good interest point detector, likely it will still return some false correspondences based on the descriptor vector. It has been tested that even just with a small set of outliers in our interest points correspondences, our homography would be significantly less accurate. Thus, RANSAC method was implemented to eliminate the outliers by random trial.

3. **Homography Estimation and Refinement**

   After a set of inliers were obtained, first we will use linear least squared method to roughly establish the 3×3 homography matrix H. Then, a non-linear method `DogLeg` will be implemented to refine the homography so that our results would look better. Please note that before we perform the non-linear `DogLeg` method we should make sure that our estimated homography H is already close enough to the real homography H_{real}, while H_{real} is absolutely correct and is not practical to obtain in most of the real world cases.

4. **Image Mosaicking**

   Finally, after we have obtained a rather accurate homography for each pair of the successive images, we will perform image mosaicking (stitching the successive images together to form a panoramic/wide angle image).

2 Interest Points Correspondences Establishment

Among the several popular feature matching algorithms SURF, SIFT and Harris Corner Detector, we will pick SIFT for this assignment.

As SIFT will return a 128 descriptor vector at each interest position, we will match the interest points based on euclidean distance. Assume we have two vectors, $\vec{V}$ and $\vec{V}'$, the euclidean distance is defined as:

$$||\vec{V} - \vec{V}'|| = \sqrt{(v_1 - v_1')^2 + (v_2 - v_2')^2 + (v_3 - v_3')^2 + ... + (v_{128} - v_{128}')^2}$$

The correspondence will be established if euclidean distance of two interested points' feature descriptor is:
1. Under a certain threshold value, in our case $T_{euclidean} = 30$
2. The Euclidean distance would be the minima between point A and point B for all possible combination of point A and a random point.

The results for the established interest points correspondences would be shown in the results section.
After we have obtained a set of corresponding interest points, we then need to look into the algorithm of eliminating the outliers.

3 RANSAC Algorithm

As the outliers will decrease the homography accuracy significantly, before we establish the final homography we need to remove several outliers using RANSAC algorithm (Random Sampling And Consensus).

There are 3 important parameters for RANSAC algorithm:

1. δ: This is a decision threshold value to construct inlier set. For example, if a certain random sample will return an overall error below the threshold σ, we can assume all the sample points in the sample set are inliers. Similarly, if for example, we have 5 different sample sets and all of them return high error, none of those sample data points could be safely moved into the inliers set. We need to continue the random sample experiment till the satisfaction criteria is met.

2. **N**: N is the number of the trials for random sample to be conducted. Method on obtaining the exact digit N will be explained later.

3. **M**: M is the size of the minimum number of inliers that would be required so the the final set of inliers is acceptable.

3.1 10 % Rule in RANSAC

In our RANSAC algorithm, we will use 10% rule to establish parameter M and N. `10% Rule` means that roughly we know that about 10% of the correspondences are false:

$$\epsilon = 0.1$$

Assume that the data points we pick in each random sample set is

$$n = 6$$

Then the probability of all correspondences chosen within single random set is:

$$(1-\epsilon)^6$$

This means that the probability of at least one sample points that n correspondences used for calculating H is an outlier is

$$1-(1-\epsilon)^6$$

Therefore, the probability of everyone of the N random sets will involved at least one outlier is:

$$[1-(1-\epsilon)^6]^N$$

Therefore, the probability that at least one of the N trials will be free of outliers is:

$$p = 1-[1-(1-\epsilon)^6]^N$$

As we are trying to set N such that $p = 0.99$, N could be calculated accordingly:

$$N = \frac{\log(1-p)}{\log[1-(1-\epsilon)^6]},\ while\ p = 0.99$$

The process of calculating for M is simpler:

$$M \approx (1-\epsilon) * (Total\ Number\ of\ Sample\ Points)$$

3.2 Procedure of Establishing Outliers

To establish a single homography matrix, we will follow the method that was utilized in Fall 2012 of picking 6 different correspondences randomly.

Assume we wan to project the image X to the reference image X', simply:

$$HX = X'$$

Now, in a single random set, we have 6 different random points, that means we can establish the equation:

$$H_{3\times3}X_{3\times6} = X'_{3\times6}$$

More specifically,

$$H_{3\times3}[\vec{X}_1,\ \vec{X}_2,\ \vec{X}_3,\ \vec{X}_4,\ \vec{X}_5,\ \vec{X}_6] = [\vec{X}'_1,\ \vec{X}'_2,\ \vec{X}'_3,\ \vec{X}'_4,\ \vec{X}'_5,\ \vec{X}'_6],\ while\ X_i = \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

In order to establish the correspondence, we need to form the matrix of the form as below:

$$A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \end{pmatrix},\ while\ A_i = \begin{bmatrix} 0 & 0 & 0 & -x_i & -y_i & -1 & y'_i x_i & y'_i y_i & y'_i \\ x_i & y_i & 1 & 0 & 0 & 0 & x'_i x_i & x'_i y_i & x'_i \end{bmatrix}$$

Now, set

$$[U, S, V] = svd(A)$$

Then, the estimated homography is:

$$\vec{h} =\ column\ in\ V\ that\ corresponde\ to\ the\ smallest\ singular\ value$$

$$\hat{H} = \begin{bmatrix} h(1) & h(2) & h(3) \\ h(4) & h(5) & h(6) \\ h(7) & h(8) & h(9) \end{bmatrix}$$

We then define the error as

$$Error = ||HX - X'||^2$$

Now, the procedures to establish inliers are as followed:

1. First, assume all sample points will be outliers
2. Pick N different random sets, each random sets consist 6 different random data points (correspondences).
3. Estimate the homography for each of the random set. We then will have N homography
4. Calculate the error for each random set, based on $Error = ||HX - X'||^2$.
5. If $Error < Threshold$ ($set\ for\ around\ 10^5$) and the error returned by the random set i is not the maximum error across N random sets, classify all 6 points into inliers
6. Check the total number of the inliers. If the inliers is less than M, go to step 2, otherwise, stop.

Following the above preocedure, we should be able to obtain a set of inliers. Then we will proceed to homographt estimation based on linear least squared method and non-linear least square method.

Note that although we assume 10% of the total data points are false correspondences, there might be less/more false correspondences (very likely less). Hence we will dump some correct correspondences by mistake.

4 Linear Least Square Homography Estimation and DogLeg Refinement

4.1 Linear Least Square Homography Estimation

Similar as in the previous step, the homography is estimated using the singular value decomposition method. From all the inliers we obtained using RANSAC, form the matrix as below:

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_M \end{pmatrix}, \ while \ A_i = \begin{bmatrix} 0 & 0 & 0 & -x_i & -y_i & -1 & y'_i x_i & y'_i y_i & y'_i \\ x_i & y_i & 1 & 0 & 0 & 0 & x'_i x_i & x'_i y_i & x'_i \end{bmatrix}$$

Now, set

$$[U, S, V] = svd(A)$$

Then, the estimated homography is:

$$\vec{h} = \ column \ in \ V \ that \ corresponde \ to \ the \ smallest \ singular \ value$$

$$\hat{H} = \begin{bmatrix} h(1) & h(2) & h(3) \\ h(4) & h(5) & h(6) \\ h(7) & h(8) & h(9) \end{bmatrix}$$

4.2 DogLeg Refinement

DogLeg algorithm is a combination of Gradient Descent Method (GD) and Gauss-Newton Method (GN).

First of all we need to establish the initial trust region, denoted as r_k. The increment for each iteration regarding p_k is defined as:

$$\vec{p}_{k+1} = \vec{p}_k + \begin{cases} \vec{\sigma}_{p,GN} & \text{if } ||\vec{\sigma}_{p,GN}|| < r_k \\ \vec{\sigma}_{p,GD} + \beta(\vec{\sigma}_{p,GN} - \vec{\sigma}_{p,GD}) & \text{if } ||\vec{\sigma}_{p,GD}|| < r_k < ||\vec{\sigma}_{p,GN}|| \\ \frac{r_k}{||\vec{\sigma}_{p,GD}||}\vec{\sigma}_{p,GD} & \text{otherwise} \end{cases}$$

With $\vec{\sigma}_{p,GD}$ and $\vec{\sigma}_{p,GN}$ defined as below:

$$\vec{\sigma}_{p,GD} = \frac{||J_f^T \vec{\epsilon}(\vec{p_k})||}{||J_f J_f^T \vec{\epsilon}(\vec{p_k})||} J_f^T \vec{\epsilon}(\vec{p_k})$$

$$\vec{\sigma}_{p,GN} = \frac{1}{J_f J_f^T + \mu_k I} J_f^T \vec{\epsilon}(\vec{p_k})$$

For initial condition, we set

$$\mu_0 = \tau * \max\{diag(J_f J_f^T)\}$$

μ_k value would be updated for each iteration:

$$\mu_{k+1} = \frac{1}{3}\mu_k$$

For each iteration, β value is set so:

$$||\vec{\sigma}_{p,GD} + \beta(\vec{\sigma}_{p,GN} - \vec{\sigma}_{p,GD})||^2 = r_k^2$$

As r_k value also need to be updated for each iteration, set

$$\rho_{DogLeg} = \frac{C(\vec{p_k}) - C(\vec{p_{k+1}})}{2\delta_p J_f^T \vec{\epsilon}(\vec{p_k}) - \delta_p^T J_f^T J_f \delta_p}$$

$$r_{k+1} = \begin{cases} \frac{r_k}{4} & \text{if } \rho^{DL} < \frac{1}{4} \\ r_k & \text{if } \frac{1}{4} < \rho^{DL} \leq \frac{3}{4} \\ 2r_k & \text{otherwise} \end{cases}$$

After we perform DogLeg method on the homography matrix we obtained from linear least squared method, we will move to the final step: Image Mosaicking.

5 Image Mosaicking

Our goal is to project all 7 images to the center image, denoted as `imgC`. Similarly, based on the view angle, from left to right, each image is tagged as:
`imgL3,imgL2,imgL1,imgC,imgR1,imgR2,imgR3` As only the homography from successive images will be estimated, denoted as:

$$H_{L3L2} * imgL3 = imgL2$$

$$H_{L2L1} * imgL2 = imgL1$$

$$H_{L1C} * imgL1 = imgC$$

$$H_{R3R2} * imgR3 = imgR2$$

$$H_{R2R1} * imgR2 = imgR1$$

$$H_{R1C} * imgR1 = imgC$$

Then, the image mosaicking will be performed as below:

$$H_{L3L2} * H_{L2L1} * H_{L1C} * imgL3 = imgC$$

$$H_{L2L1} * H_{L1C} * imgL2 = imgC$$

$$H_{L1C} * imgL1 = imgC$$

$$H_{R3R2} * H_{R2R1} * H_{R1C} * imgR3 = imgC$$

$$H_{R2R1} * H_{R1C} * imgR2 = imgC$$

$$H_{R1C} * imgR1 = imgC$$

In order to optimize the result, pixel interpolation will also be performed!

6 Appendix: Important Parameters

There are many important parameters in the system. Some of them are calculated based on specific case, for example the M and the N in RANSAC algorithm, and those parameters in `DogLeg` algorithm. We can not list every single of those variable parameters as a huge number of iterations was performed.

However, there are several hard-coded fixed-value parameters in the system while most of the fixed-value parameters serves as the cut-off threshold.

Image Pair	`L3&L2`	`L2&L1`	`L1&C`	`R1&C`	`R2&R1`	`R3&R2`
$T_{euclidean\ distance}$	30	30	30	30	30	30
$N_{total\ correspondences}$	161	181	139	74	190	179
n_{RANSAC}	6	6	6	6	6	6
N_{RANSAC}	6	6	6	6	6	6
M_{RANSAC}	144	162	125	66	161	171
δ_{RANSAC}	5×10^5	5×10^4	10^6	4×10^4	1.6×10^5	10^6

Table 1. Important Fixed-Value Parameters. n_{RANSAC} denote number of data points picked in each random trial.

7 Results For 7 Images in MSEE Atrium

7.1 Interest Points Correspondences For Successive Images

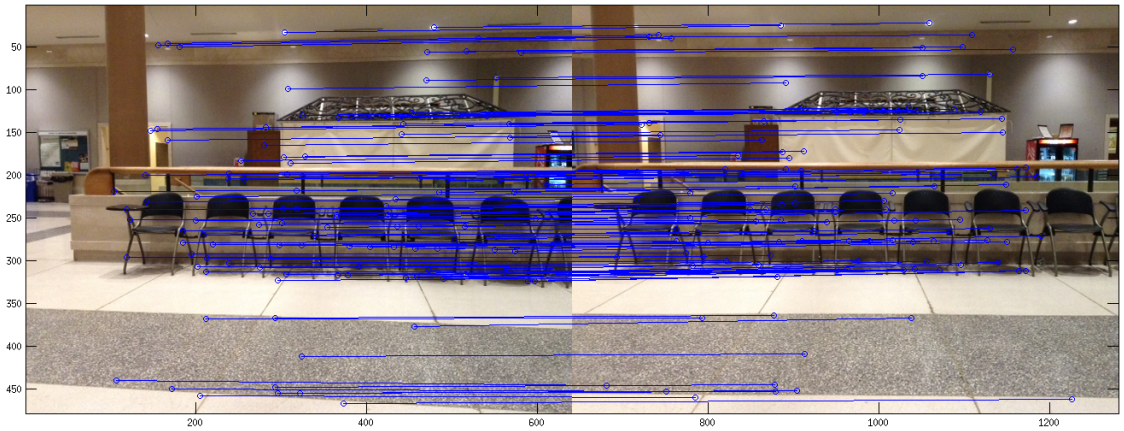

Figure 1. Correspondence Between `imgL1` and `imgC`

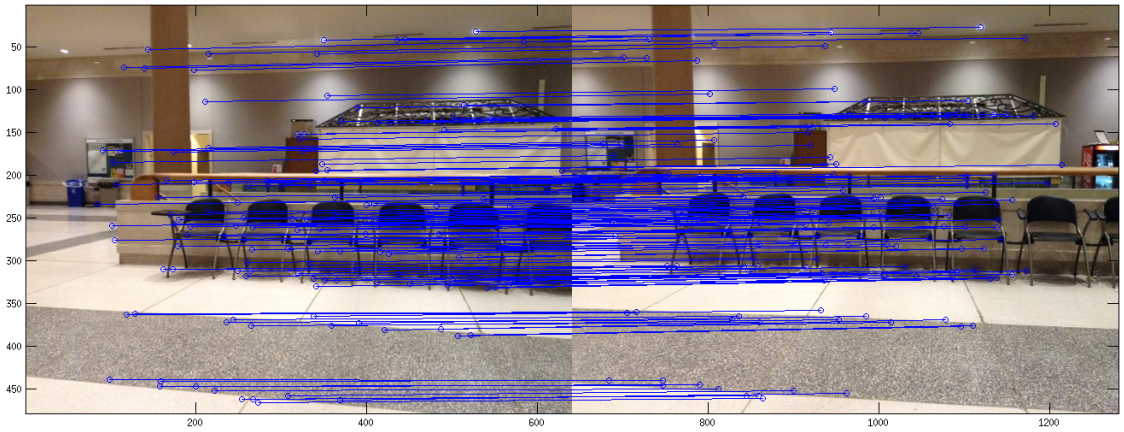

Figure 2. Correspondence Between `imgL2` and `imgL1`

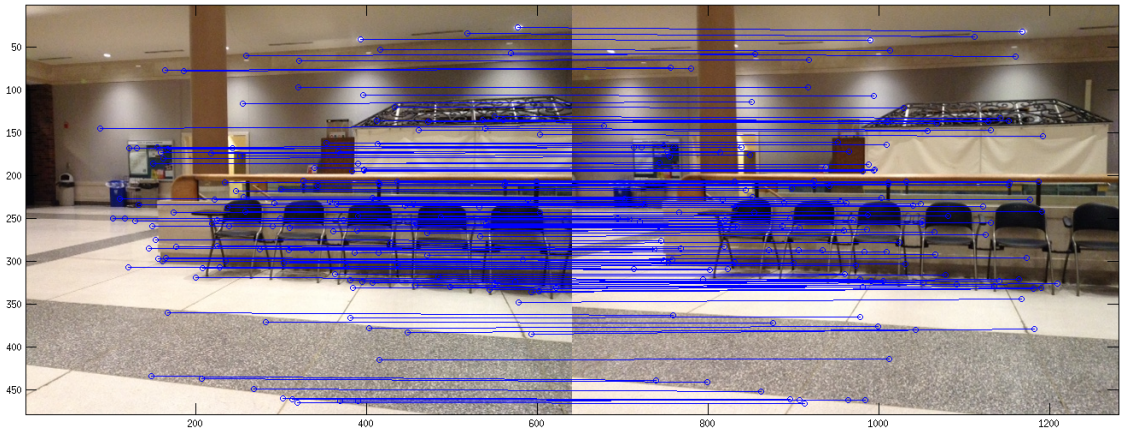

Figure 3. Correspondence Between `imgL3` and `imgL2`

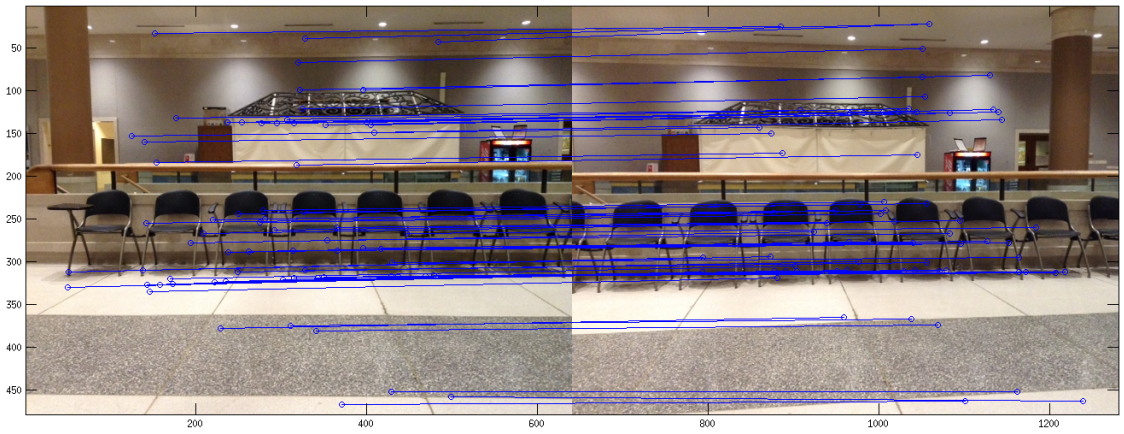

Figure 4. Correspondence Between `imgC` and `imgR1`

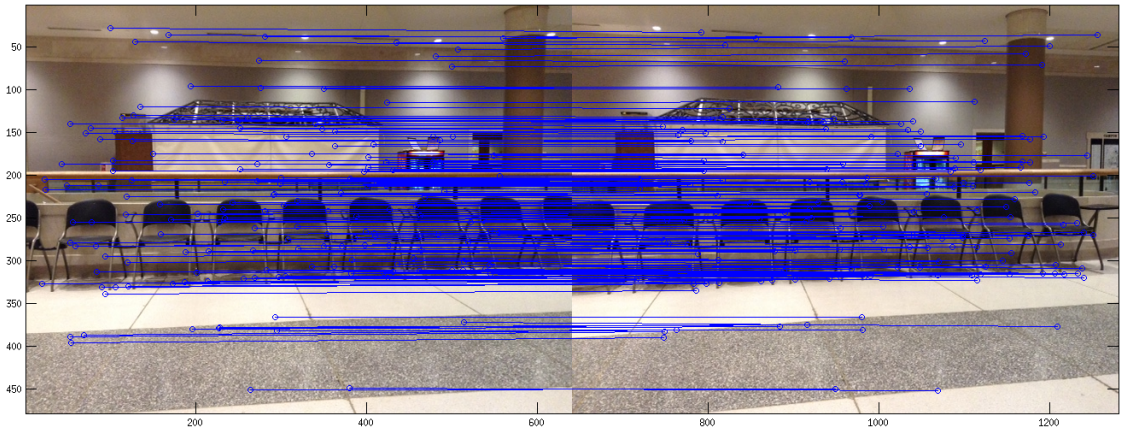

Figure 5. Correspondence Between `imgR1` and `imgR2`

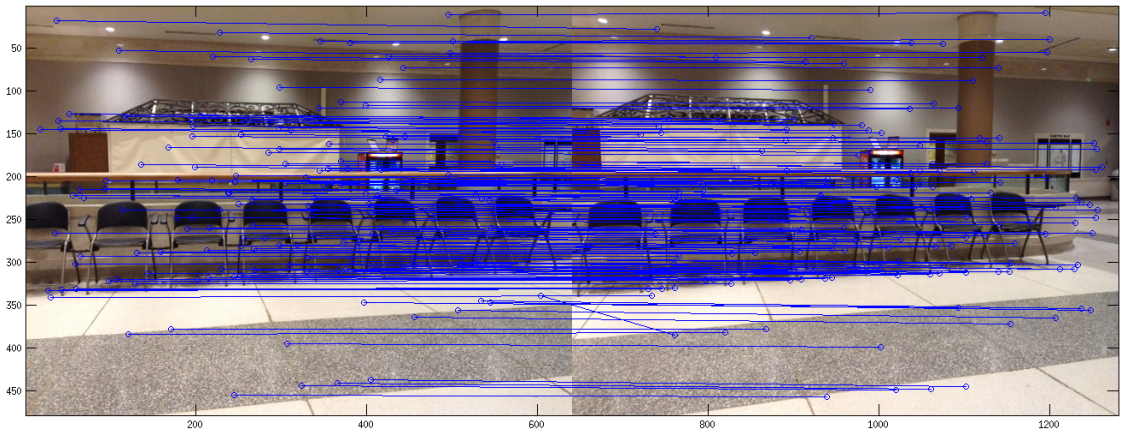

Figure 6. Correspondence Between `imgR2` and `imgR3`

7.2 Inliers (Green) and Outliers (Red)

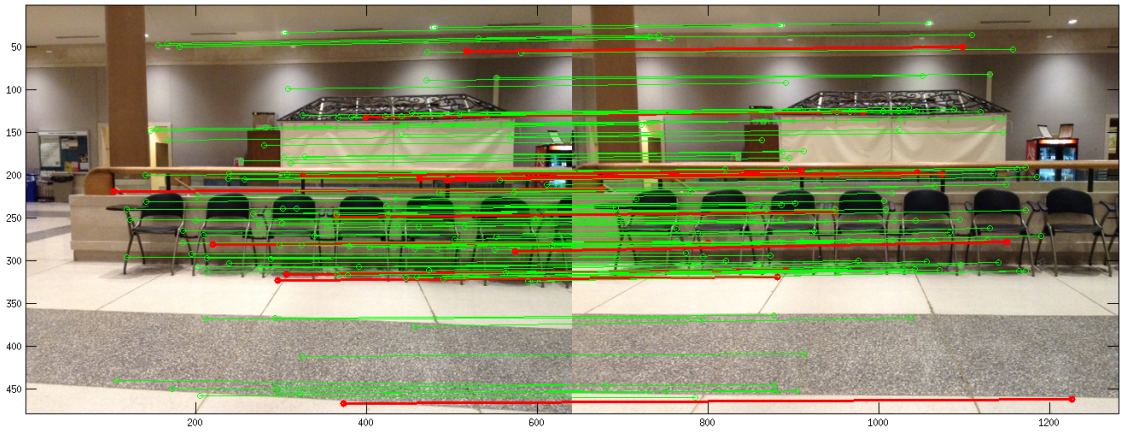

Figure 7. Inliers(green) vs. Outliers(red) for `imgL1` and `imgC`

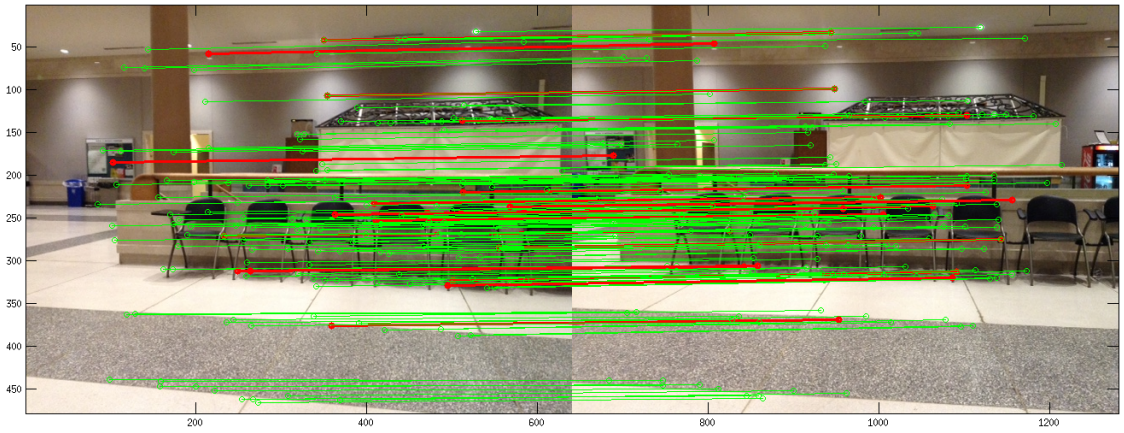

Figure 8. Inliers(green) vs. Outliers(red) for `imgL2` and `imgL1`

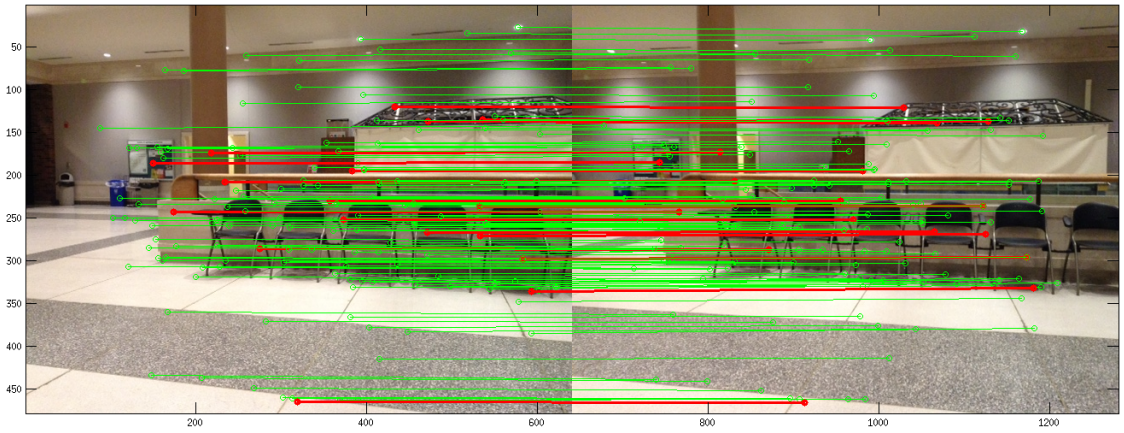

Figure 9. Inliers(green) vs. Outliers(red) for `imgL3` and `imgL2`

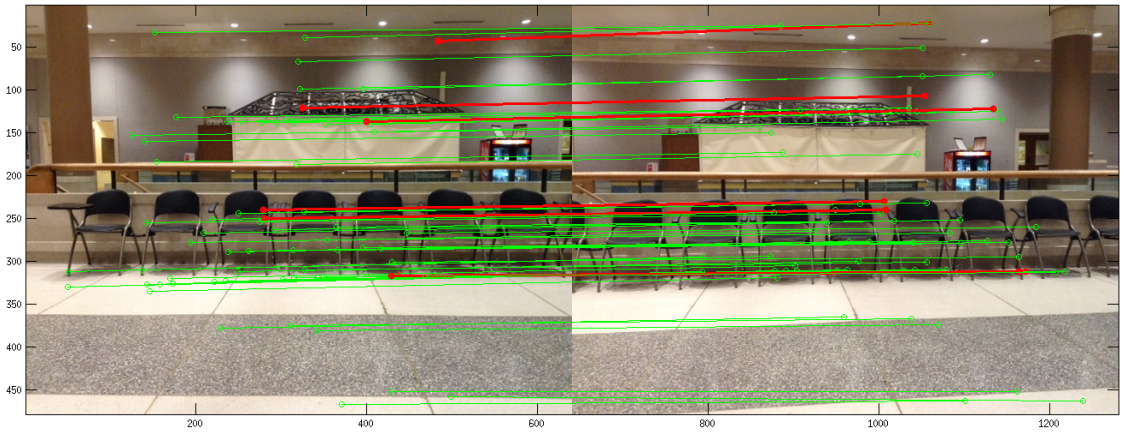

Figure 10. Inliers(green) vs. Outliers(red) for `imgC` and `imgR1`

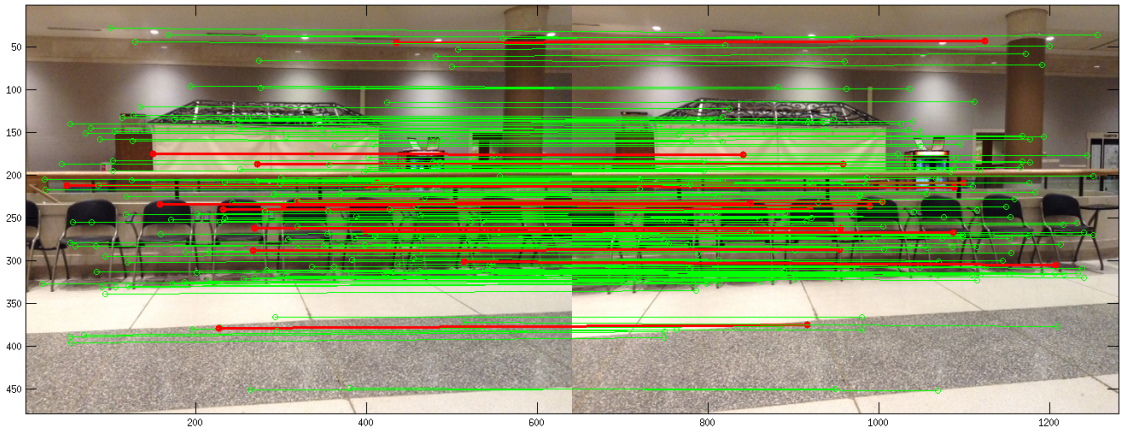

Figure 11. Inliers(green) vs. Outliers(red) for `imgR1` and `imgR2`

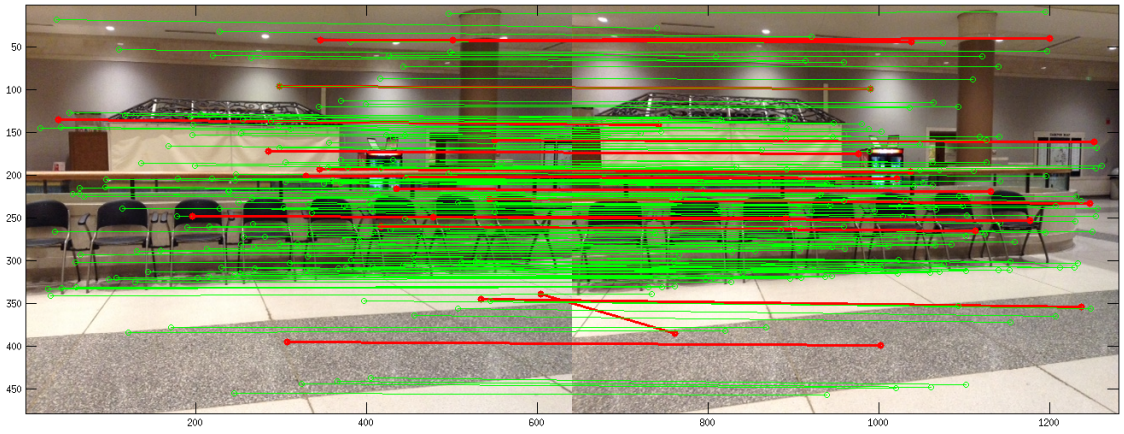

Figure 12. Inliers(green) vs. Outliers(red) for `imgR2` and `imgR3`

7.3 Image Mosaicking Result

Figure 13. Image Mosaicking Result Without Interpolation

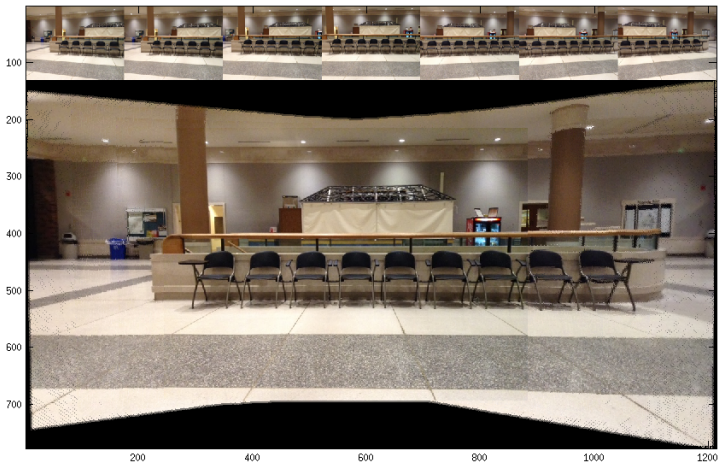

Figure 14. Image Mosaicking Result With Interpolation

8 Results For 5 Images Outside MSEE

8.1 Interest Points Correspondences For Successive Images

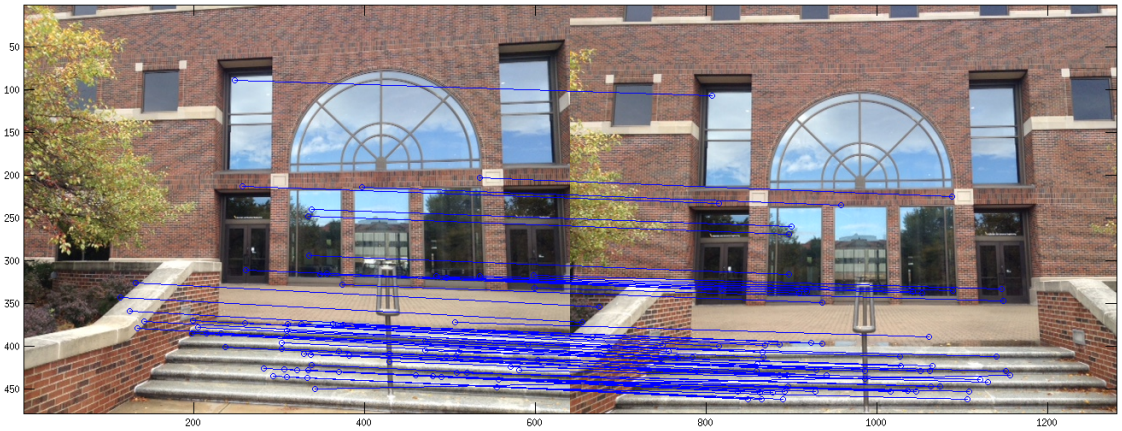

Figure 15. Correspondence Between `imgL1` and `imgC`

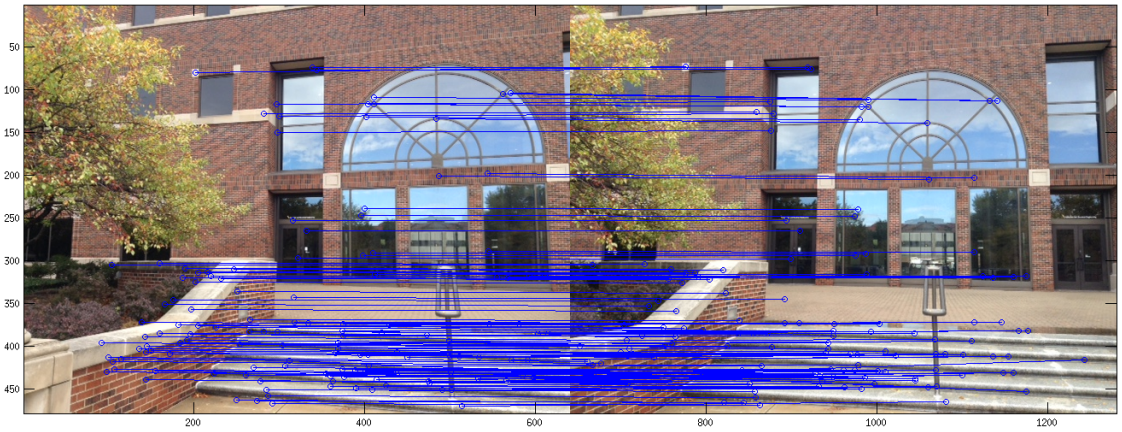

Figure 16. Correspondence Between `imgL2` and `imgL1`

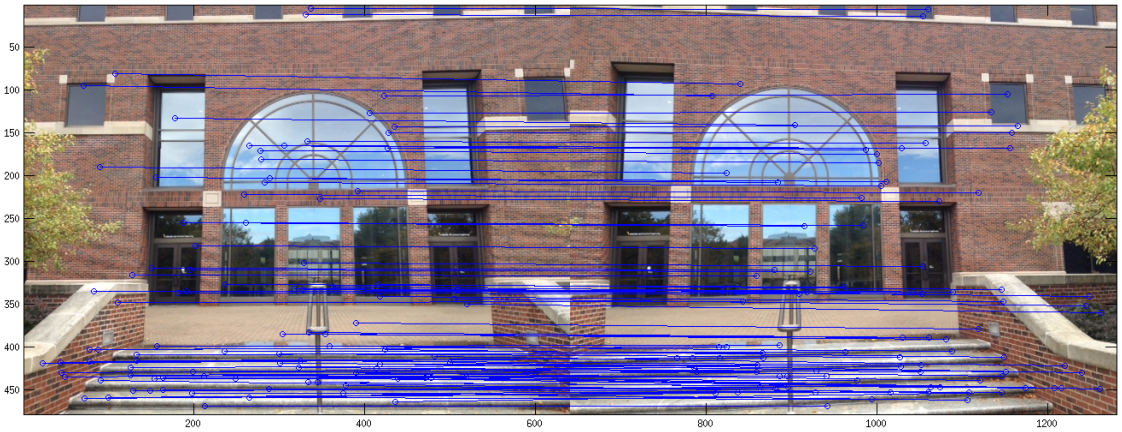

Figure 17. Correspondence Between `imgC` and `imgR1`

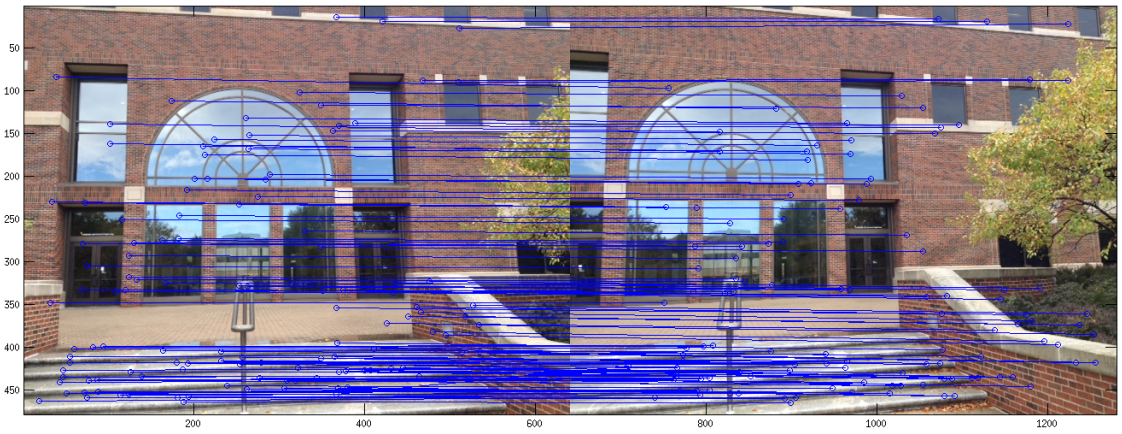

Figure 18. Correspondence Between `imgR1` and `imgR2`

8.2 Inliers (Green) and Outliers (Red)

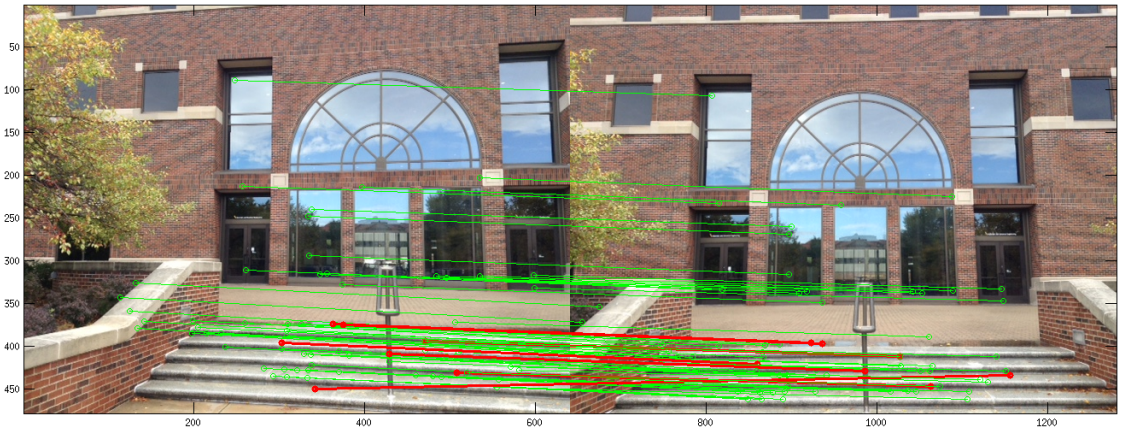

Figure 19. Inliers(green) vs. Outliers(red) for `imgL1` and `imgC`

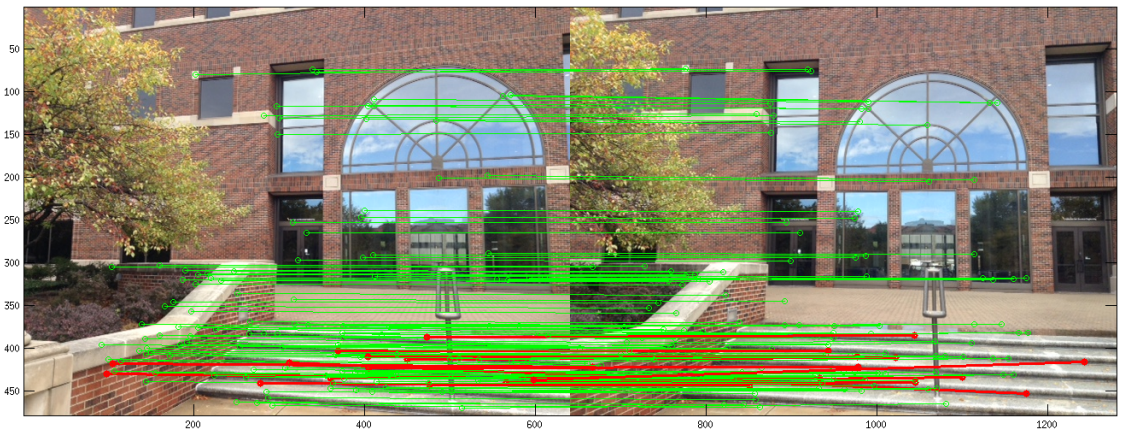

Figure 20. Inliers(green) vs. Outliers(red) for `imgL2` and `imgL1`

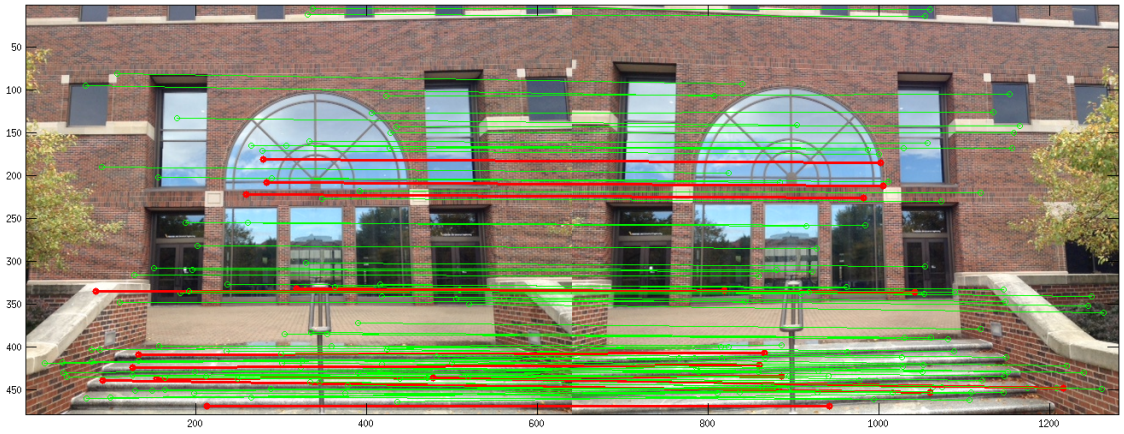

Figure 21. Inliers(green) vs. Outliers(red) for `imgC` and `imgR1`

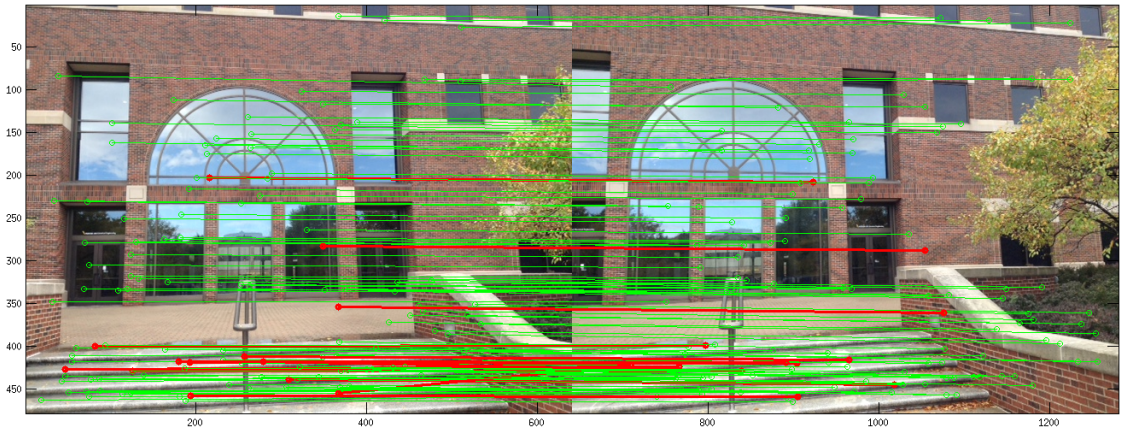

Figure 22. Inliers(green) vs. Outliers(red) for `imgR1` and `imgR2`

8.3 Image Mosaicking Result

Figure 23. The mosaicking result without pixel interpolation

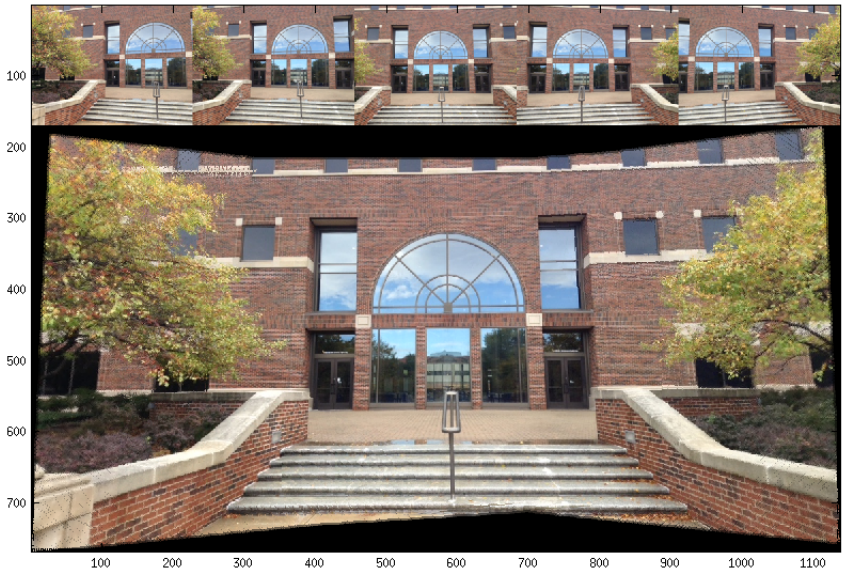

Figure 24. The mosaicking result with pixel interpolation

9 Appendix: Matlab Code

9.1 Main Function: `main.m`

```
close all; clear all; clc

Teuc = 30;                    % Threshold for euclidean distance
num_homo = 6;                 % number of data points in each random trial
e = 0.1;                      % 10% rule for error data
sigma = 1e6;                  % pre-set threshold for RANSAC

%% Initiate the algorithms by loading all the images and convert color
%  image to gray scale image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
imgl1 = imread('imgl1.jpg');
imgl2 = imread('imgl2.jpg');
imgl3 = imread('imgl3.jpg');
imgr1 = imread('imgr1.jpg');
imgr2 = imread('imgr2.jpg');
imgr3 = imread('imgr3.jpg');
imgc = imread('imgc.jpg');

imgl1mono = rgb2gray(imgl1);
imgl2mono = rgb2gray(imgl2);
imgl3mono = rgb2gray(imgl3);
imgr1mono = rgb2gray(imgr1);
imgr2mono = rgb2gray(imgr2);
imgr3mono = rgb2gray(imgr3);
imgcmono = rgb2gray(imgc);

[M,N] = size(imgcmono); %% Calculate the size of the image

imgl1mono = single(imgl1mono);
imgl2mono = single(imgl2mono);
imgl3mono = single(imgl3mono);
imgr1mono = single(imgr1mono);
imgr2mono = single(imgr2mono);
imgr3mono = single(imgr3mono);
imgcmono = single(imgcmono);

%% Finding the SIFT features for all images using 'vl_sift' %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
disp('Finding the SIFT descriptors')
[fc,dc] = vl_sift(imgcmono);
[fl1,dl1] = vl_sift(imgl1mono);
[fl2,dl2] = vl_sift(imgl2mono);
[fl3,dl3] = vl_sift(imgl3mono);
[fr1,dr1] = vl_sift(imgr1mono);
[fr2,dr2] = vl_sift(imgr2mono);
[fr3,dr3] = vl_sift(imgr3mono);
toc
```

```
49
50 %% Form the euclidean distance of SIFT feature vectors for each successive
       image pair
51 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
52 eucl1c = dist(dl1',dc);
53 eucl2l1 = dist(dl2',dl1);
54 eucl3l2 = dist(dl3',dl2);
55 eucr1c = dist(dr1',dc);
56 eucr2r1 = dist(dr2',dr1);
57 eucr3r2 = dist(dr3',dr2);
58
59 %% Stitch the successive images together for correspondence demonstration
60 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61 imgl3l2 = [imgl3,imgl2];
62 imgl2l1 = [imgl2,imgl1];
63 imgl1c = [imgl1,imgc];
64 imgcr1 = [imgc,imgr1];
65 imgr1r2 = [imgr1,imgr2];
66 imgr2r3 = [imgr2,imgr3];
67
68 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
69 %% Now for pair imgL1/imgC
70 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
71 cnt_l1c = 0;  %% Counter for correspondences in pair imgL1/imgC
72
73 size_sift_l1c = size(eucl1c);
74 %% Establish correspondences in pair imgL1/imgC
75 for i = 1:1:size_sift_l1c(1)
76     for j = 1:1:size_sift_l1c(2)
77             if(eucl1c(i,j) == min(eucl1c(i,:)))
78                if(eucl1c(i,j) < Teuc)
79                    cnt_l1c = cnt_l1c + 1;
80                    corr_l1c_l1(1:2,cnt_l1c) = [round(fl1(2,i));round(fl1(1,i))
                          ];
81                    corr_l1c_c(1:2,cnt_l1c) = [round(fc(2,j));round(fc(1,j))];
82                    j = size_sift_l1c(2);
83                else
84                end
85             else
86             end
87     end
88 end
89 %% Calculate RANSAC parameters for pair imgL1/imgC
90 Ransac_l1c_M = fix((1-0.1)*cnt_l1c);
91 Ransac_l1c_N = fix(log(1-0.99)/log(1-(1-e)^num_homo));
92 %% Show all the correspondences established
93 figure
94 image(imgl1c)
95 truesize;
96 hold on;
97 for k = 1:1:cnt_l1c
98      plot([corr_l1c_l1(2,k),corr_l1c_c(2,k)+N],[corr_l1c_l1(1,k),corr_l1c_c(1,
           k)],'-ob','linewidth',1);
99 end
```

```
%% Perform RANSAC algorithm for pair imgL1/imgC
H  = ransac_linear_mini(N, cnt_l1c, num_homo, Ransac_l1c_M, Ransac_l1c_N,
    imgl1c, corr_l1c_l1, corr_l1c_c,sigma);
%% Save the final homography for pair imgL1/imgC
H_l1c = H;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Now for pair imgL2/imgL1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cnt_l2l1 = 0; %% Counter for correspondences in pair imgL2/imgL1

size_sift_l2l1 = size(eucl2l1);
%% Establish correspondences in pair imgL2/imgL1
for i = 1:1:size_sift_l2l1(1)
    for j = 1:1:size_sift_l2l1(2)
            if(eucl2l1(i,j) == min(eucl2l1(i,:)))
               if(eucl2l1(i,j) < Teuc)
                   cnt_l2l1 = cnt_l2l1 + 1;
                   corr_l2l1_l2(1:2,cnt_l2l1) = [round(fl2(2,i));round(fl2(1,i
                       ))];
                   corr_l2l1_l1(1:2,cnt_l2l1) = [round(fl1(2,j));round(fl1(1,j
                       ))];
                   j = size_sift_l2l1(2);
               else
               end
            else
            end
    end
end
%% Calculate RANSAC parameters for pair imgL2/imgL1
Ransac_l2l1_M = fix((1-0.1)*cnt_l2l1);
Ransac_l2l1_N = fix(log(1-0.99)/log(1-(1-e)^num_homo));
%% Show all the correspondences established
figure
image(imgl2l1)
truesize;
hold on;
for k = 1:1:cnt_l2l1
     plot([corr_l2l1_l2(2,k),corr_l2l1_l1(2,k)+N],[corr_l2l1_l2(1,k),
         corr_l2l1_l1(1,k)],'-ob','linewidth',1);
end

sigma = 5e4;  %% Adjust threshold for RANSAC
%% Perform RANSAC algorithm for pair imgL2/imgL1
H  = ransac_linear_mini(N, cnt_l2l1, num_homo, Ransac_l2l1_M, Ransac_l2l1_N,
    imgl2l1, corr_l2l1_l2, corr_l2l1_l1,sigma);
%% Save the final homography for pair imgL2/img1
H_l2l1 = H;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Now for pair imgR1/imgC
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
cnt_r1c = 0; %% Counter for correspondences in pair imgR1/imgC
size_sift_r1c = size(eucr1c);
%% Establish correspondences in pair imgR1/imgC
for i = 1:1:size_sift_r1c(1)
    for j = 1:1:size_sift_r1c(2)
            if(eucr1c(i,j) == min(eucr1c(i,:)))
               if(eucr1c(i,j) < Teuc)
                   cnt_r1c = cnt_r1c + 1;
                   corr_r1c_r1(1:2,cnt_r1c) = [round(fr1(2,i));round(fr1(1,i))
                       ];
                   corr_r1c_c(1:2,cnt_r1c) = [round(fc(2,j));round(fc(1,j))];
                   j = size_sift_r1c(2);
               else
               end
            else
            end
    end
end
%% Calculate RANSAC parameters for pair imgR1/imgC
Ransac_r1c_M = fix((1-0.1)*cnt_r1c);
Ransac_r1c_N = fix(log(1-0.99)/log(1-(1-e)^num_homo));
%% Show all the correspondences established
figure
image(imgcr1)
truesize;
hold on;
for k = 1:1:cnt_r1c
     plot([corr_r1c_r1(2,k),corr_r1c_c(2,k)+N],[corr_r1c_r1(1,k),corr_r1c_c(1,
        k)],'-ob','linewidth',1);
end
sigma = 4e4;%% Adjust threshold for RANSAC
%% Perform RANSAC algorithm for this pair
H  = ransac_linear_mini(N, cnt_r1c, num_homo, Ransac_r1c_M, Ransac_r1c_N,
    imgcr1, corr_r1c_r1, corr_r1c_c,sigma);
%% Save the final homography for this pair
H_r1c = H;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Now for pair imgR2/imgR1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cnt_r2r1 = 0;%% Counter for correspondences in this pair
size_sift_r2r1 = size(eucr2r1);
%% Establish correspondences in this pair
for i = 1:1:size_sift_r2r1(1)
    for j = 1:1:size_sift_r2r1(2)
            if(eucr2r1(i,j) == min(eucr2r1(i,:)))
               if(eucr2r1(i,j) < Teuc)
                   cnt_r2r1 = cnt_r2r1 + 1;
                   corr_r2r1_r2(1:2,cnt_r2r1) = [round(fr2(2,i));round(fr2(1,i
                       ))];
                   corr_r2r1_r1(1:2,cnt_r2r1) = [round(fr1(2,j));round(fr1(1,j
                       ))];
                   j = size_sift_r2r1(2);
```

```
198                 else
199                 end
200             else
201             end
202     end
203 end
204 %% Calculate RANSAC parameters for this pair
205 Ransac_r2r1_M = fix((1-0.1)*cnt_r2r1);
206 Ransac_r2r1_N = fix(log(1-0.99)/log(1-(1-e)^num_homo));
207 %% Show all the correspondences established
208 figure
209 image(imgr1r2)
210 truesize;
211 hold on;
212 for k = 1:1:cnt_r2r1
213      plot([corr_r2r1_r2(2,k),corr_r2r1_r1(2,k)+N],[corr_r2r1_r2(1,k),
         corr_r2r1_r1(1,k)],'-ob','linewidth',1);
214 end
215 sigma = 1.6e5;%% Adjust threshold for RANSAC
216 %% Perform RANSAC algorithm for this pair
217 H  = ransac_linear_mini(N, cnt_r2r1, num_homo, Ransac_r2r1_M, Ransac_r2r1_N,
    imgr1r2, corr_r2r1_r2, corr_r2r1_r1,sigma);
218 %% Save the final homography for this pair
219 H_r2r1 = H;
220
221 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
222 %% Now for pair imgL3/imgL2
223 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
224
225 cnt_l3l2 = 0;%% Counter for correspondences in this pair
226 size_sift_l3l2 = size(eucl3l2);
227 %% Establish correspondences in this pair
228 for i = 1:1:size_sift_l3l2(1)
229     for j = 1:1:size_sift_l3l2(2)
230             if(eucl3l2(i,j) == min(eucl3l2(i,:)))
231                 if(eucl3l2(i,j) < Teuc)
232                     cnt_l3l2 = cnt_l3l2 + 1;
233                     corr_l3l2_l3(1:2,cnt_l3l2) = [round(fl3(2,i));round(fl3(1,i
                        ))];
234                     corr_l3l2_l2(1:2,cnt_l3l2) = [round(fl2(2,j));round(fl2(1,j
                        ))];
235                     j = size_sift_l3l2(2);
236                 else
237                 end
238             else
239             end
240     end
241 end
242 %% Calculate RANSAC parameters for this pair
243 Ransac_l3l2_M = fix((1-0.1)*cnt_l3l2);
244 Ransac_l3l2_N = fix(log(1-0.99)/log(1-(1-e)^num_homo));
245 %% Show all the correspondences established
246 figure
247 image(imgl3l2)
```

```
truesize;
hold on;
for k = 1:1:cnt_l3l2
     plot([corr_l3l2_l3(2,k),corr_l3l2_l2(2,k)+N],[corr_l3l2_l3(1,k),
        corr_l3l2_l2(1,k)],'-ob','linewidth',1);
end
sigma = 5e5;%% Adjust threshold for RANSAC
%% Perform RANSAC algorithm for this pair
H  = ransac_linear_mini(N, cnt_l3l2, num_homo, Ransac_l3l2_M, Ransac_l3l2_N,
    imgl3l2, corr_l3l2_l3, corr_l3l2_l2,sigma);
%% Save the final homography for this pair
H_l3l2 = H;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Now for pair imgR3/imgR2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cnt_r3r2 = 0;%% Counter for correspondences in this pair
size_sift_r3r2 = size(eucr3r2);
%% Establish correspondences in this pair
for i = 1:1:size_sift_r3r2(1)
    for j = 1:1:size_sift_r3r2(2)
            if(eucr3r2(i,j) == min(eucr3r2(i,:)))
               if(eucr3r2(i,j) < Teuc)
                   cnt_r3r2 = cnt_r3r2 + 1;
                   corr_r3r2_r3(1:2,cnt_r3r2) = [round(fr3(2,i));round(fr3(1,i
                       ))];
                   corr_r3r2_r2(1:2,cnt_r3r2) = [round(fr2(2,j));round(fr2(1,j
                       ))];
                   j = size_sift_l3l2(2);
               else
               end
            else
            end
    end
end
%% Calculate RANSAC parameters for this pair
Ransac_r3r2_M = fix((1-0.1)*cnt_r3r2);
Ransac_r3r2_N = fix(log(1-0.99)/log(1-(1-e)^num_homo));
%% Show all the correspondences established
figure
image(imgr2r3)
truesize;
hold on;
for k = 1:1:cnt_r3r2
     plot([corr_r3r2_r3(2,k),corr_r3r2_r2(2,k)+N],[corr_r3r2_r3(1,k),
        corr_r3r2_r2(1,k)],'-ob','linewidth',1);
end
sigma = 1e6;%% Adjust threshold for RANSAC
%% Perform RANSAC algorithm for this pair
H  = ransac_linear_mini(N, cnt_r3r2, num_homo, Ransac_r3r2_M, Ransac_r3r2_N,
    imgr2r3, corr_r3r2_r3, corr_r3r2_r2,sigma);
%% Save the final homography for this pair
H_r3r2 = H;

```

```
296
297 %%%%%%%% END OF HOMOGRAPHY ESTIMATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
298 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
299 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
300
301 %% Find the proper offset we need shen mosaicking images
302 %  Although this part of cose is long, the task is simple, find the minimum
303 %  tx, ty values (for all the successive pairs) after projection and
304 %  apply tx, ty as offset when we are mosaicking all the imgaes together
305
306 ver_1 = H_r3r2*H_r2r1*H_r1c*[1;1;1];
307 ver_2 = H_r3r2*H_r2r1*H_r1c*[N;1;1];
308 ver_3 = H_r3r2*H_r2r1*H_r1c*[1;M;1];
309 ver_4 = H_r3r2*H_r2r1*H_r1c*[N;M;1];
310 ver_1 = round(ver_1/ver_1(3));
311 ver_2 = round(ver_2/ver_2(3));
312 ver_3 = round(ver_3/ver_3(3));
313 ver_4 = round(ver_4/ver_4(3));
314
315 txr3c = min([ver_1(1),ver_2(1),ver_3(1),ver_4(1)]);
316 tyr3c = min([ver_1(2),ver_2(2),ver_3(2),ver_4(2)]);
317
318 ver_1 = H_l3l2*H_l2l1*H_l1c*[1;1;1];
319 ver_2 = H_l3l2*H_l2l1*H_l1c*[N;1;1];
320 ver_3 = H_l3l2*H_l2l1*H_l1c*[1;M;1];
321 ver_4 = H_l3l2*H_l2l1*H_l1c*[N;M;1];
322 ver_1 = round(ver_1/ver_1(3));
323 ver_2 = round(ver_2/ver_2(3));
324 ver_3 = round(ver_3/ver_3(3));
325 ver_4 = round(ver_4/ver_4(3));
326
327 txl3c = min([ver_1(1),ver_2(1),ver_3(1),ver_4(1)]);
328 tyl3c = min([ver_1(2),ver_2(2),ver_3(2),ver_4(2)]);
329
330 ver_1 = H_l2l1*H_l1c*[1;1;1];
331 ver_2 = H_l2l1*H_l1c*[N;1;1];
332 ver_3 = H_l2l1*H_l1c*[1;M;1];
333 ver_4 = H_l2l1*H_l1c*[N;M;1];
334 ver_1 = round(ver_1/ver_1(3));
335 ver_2 = round(ver_2/ver_2(3));
336 ver_3 = round(ver_3/ver_3(3));
337 ver_4 = round(ver_4/ver_4(3));
338
339 txl2c = min([ver_1(1),ver_2(1),ver_3(1),ver_4(1)]);
340 tyl2c = min([ver_1(2),ver_2(2),ver_3(2),ver_4(2)]);
341
342 ver_1 = H_r2r1*H_r1c*[1;1;1];
343 ver_2 = H_r2r1*H_r1c*[N;1;1];
344 ver_3 = H_r2r1*H_r1c*[1;M;1];
345 ver_4 = H_r2r1*H_r1c*[N;M;1];
346 ver_1 = round(ver_1/ver_1(3));
347 ver_2 = round(ver_2/ver_2(3));
348 ver_3 = round(ver_3/ver_3(3));
349 ver_4 = round(ver_4/ver_4(3));
```

```
350
351 txr2c = min([ver_1(1),ver_2(1),ver_3(1),ver_4(1)]);
352 tyr2c = min([ver_1(2),ver_2(2),ver_3(2),ver_4(2)]);
353
354 ver_1 = H_l1c*[1;1;1];
355 ver_2 = H_l1c*[N;1;1];
356 ver_3 = H_l1c*[1;M;1];
357 ver_4 = H_l1c*[N;M;1];
358 ver_1 = round(ver_1/ver_1(3));
359 ver_2 = round(ver_2/ver_2(3));
360 ver_3 = round(ver_3/ver_3(3));
361 ver_4 = round(ver_4/ver_4(3));
362
363 txl1c = min([ver_1(1),ver_2(1),ver_3(1),ver_4(1)]);
364 tyl1c = min([ver_1(2),ver_2(2),ver_3(2),ver_4(2)]);
365
366 ver_1 = H_r1c*[1;1;1];
367 ver_2 = H_r1c*[N;1;1];
368 ver_3 = H_r1c*[1;M;1];
369 ver_4 = H_r1c*[N;M;1];
370 ver_1 = round(ver_1/ver_1(3));
371 ver_2 = round(ver_2/ver_2(3));
372 ver_3 = round(ver_3/ver_3(3));
373 ver_4 = round(ver_4/ver_4(3));
374
375 txr1c = min([ver_1(1),ver_2(1),ver_3(1),ver_4(1)]);
376 tyr1c = min([ver_1(2),ver_2(2),ver_3(2),ver_4(2)]);
377
378 tx = min([txl1c,txl2c,txl3c,txr1c,txr2c,txr3c]);
379 ty = min([tyl1c,tyl2c,tyl3c,tyr1c,tyr2c,tyr3c]);
380
381 %%%%%%%% END OF OFFSET ESTABLISHMENT    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
382 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
383
384 %% Final Step: Mosaicking all images %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
385 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
386 clear new_img;
387 clear interp_img;
388 for i = 1:1:M
389     for j = 1:1:N
390
391         new_coor = H_l3l2*H_l2l1*H_l1c*[j;i;1];
392         new_coor = round(new_coor/new_coor(3));
393         new_img(new_coor(2)-ty+2,new_coor(1)-tx+2,1:3) = imgl3(i,j,1:3);
394
395         new_coor = H_r3r2*H_r2r1*H_r1c*[j;i;1];
396         new_coor = round(new_coor/new_coor(3));
397         new_img(new_coor(2)-ty+2,new_coor(1)-tx+2,1:3) = imgr3(i,j,1:3);
398
399     end
400 end
401
402 for i = 1:1:M
403     for j = 1:1:N
```

```
404         new_coor = H_l2l1*H_l1c*[j;i;1];
405         new_coor = round(new_coor/new_coor(3));
406         new_img(new_coor(2)-ty+2,new_coor(1)-tx+2,1:3) = imgl2(i,j,1:3);
407 
408         new_coor = H_r2r1*H_r1c*[j;i;1];
409         new_coor = round(new_coor/new_coor(3));
410         new_img(new_coor(2)-ty+2,new_coor(1)-tx+2,1:3) = imgr2(i,j,1:3);
411 
412     end
413 end
414 
415 for i = 1:1:M
416     for j = 1:1:N
417         new_coor = H_l1c*[j;i;1];
418         new_coor = round(new_coor/new_coor(3));
419         new_img(new_coor(2)-ty+2,new_coor(1)-tx+2,1:3) = imgl1(i,j,1:3);
420 
421         new_coor = H_r1c*[j;i;1];
422         new_coor = round(new_coor/new_coor(3));
423         new_img(new_coor(2)-ty+2,new_coor(1)-tx+2,1:3) = imgr1(i,j,1:3);
424     end
425 end
426 
427 for i = 1:1:M
428     for j = 1:1:N
429     new_img(i-ty+2,j-tx+2,1:3) = imgc(i,j,1:3);
430     end
431 end
432 
433 figure
434 imagesc(new_img)
435 truesize
436 %%
437 %%%%%%%%%%%%%%%Pixel Interpolation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
438 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
439 disp('Interpolating the image...')
440 size_new_image = size(new_img);
441 interp_img = double(new_img);
442 new_img = double(new_img);
443 for i = 5:1:size_new_image(1)-5
444     for j = 5:1:size_new_image(2)-5
445         if (sumsqr(new_img(i,j,1:3)) == 0)
446             k = 1;
447             while(k < 4)
448                 if (sumsqr(new_img(i-k,j-k,:)) ~= 0)
449                 interp_img(i,j,1) = interp_img(i,j,1) + (1/4)*(new_img(i-k,j-k
                        ,1));
450                 interp_img(i,j,2) = interp_img(i,j,2) + (1/4)*(new_img(i-k,j-k
                        ,2));
451                 interp_img(i,j,3) = interp_img(i,j,3) + (1/4)*(new_img(i-k,j-k
                        ,3));
452                 k = 4;
453                 else
454                 k = k+1;
```

```
455                 end
456             end
457             k = 1;
458             while(k < 4)
459                 if (sumsqr(new_img(i-k,j+k,:)) ~= 0)
460                 interp_img(i,j,1) = interp_img(i,j,1) + (1/4)*(new_img(i-k,j+k
                        ,1));
461                 interp_img(i,j,2) = interp_img(i,j,2) + (1/4)*(new_img(i-k,j+k
                        ,2));
462                 interp_img(i,j,3) = interp_img(i,j,3) + (1/4)*(new_img(i-k,j+k
                        ,3));
463                 k = 4;
464                 else
465                 k = k+1;
466                 end
467             end
468             k = 1;
469             while(k < 4)
470                 if (sumsqr(new_img(i+k,j-k,:)) ~= 0)
471                 interp_img(i,j,1) = interp_img(i,j,1) + (1/4)*(new_img(i+k,j-k
                        ,1));
472                 interp_img(i,j,2) = interp_img(i,j,2) + (1/4)*(new_img(i+k,j-k
                        ,2));
473                 interp_img(i,j,3) = interp_img(i,j,3) + (1/4)*(new_img(i+k,j-k
                        ,3));
474                 k = 4;
475                 else
476                 k = k+1;
477                 end
478             end
479             k = 1;
480             while(k < 4)
481                 if (sumsqr(new_img(i+k,j+k,:)) ~= 0)
482                 interp_img(i,j,1) = interp_img(i,j,1) + (1/4)*(new_img(i+k,j+k
                        ,1));
483                 interp_img(i,j,2) = interp_img(i,j,2) + (1/4)*(new_img(i+k,j+k
                        ,2));
484                 interp_img(i,j,3) = interp_img(i,j,3) + (1/4)*(new_img(i+k,j+k
                        ,3));
485                 k = 4;
486                 else
487                 k = k+1;
488                 end
489             end
490         else
491         end
492     end
493 end
494 interp_img = uint8(interp_img);
495
496 whole_img = [img13,img12,img11,imgc,imgr1,imgr2,imgr3];
497 scale = size_new_image(2)/(7*N);
498 scale_img = imresize(whole_img,scale);
499 show_img = [scale_img;interp_img];
```

```
500 figure
501 imagesc(show_img)
502 truesize
503 disp('Image Mosaicking Done!')
```

9.2 RANSAC and Linear Least Square: `ransac_linear_mini.m`

```
function [ H ] = ransac_linear_mini(N, cnt_l1c, num_homo, Ransac_l1c_M,
    Ransac_l1c_N, imgl1c, corr_l1c_l1, corr_l1c_c,sigma)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1). This function perfrom RANSAC algorithms to eliminate outliers
% 2). Then the homography will be estimated based on linear least square
% 3). Finally, this function will call the DogLeg for homography refinement
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

outlier_flag = ones(cnt_l1c,1); % Intially Assume all points are outliers

while(sum(outlier_flag)>(cnt_l1c-Ransac_l1c_M)); %% Check if we have enough
    inliers
for rn = 1:1:Ransac_l1c_N
    samples(rn,1:num_homo) = randi(cnt_l1c,num_homo,1); % Generate 6 random
        sample points in one random set(trial)
    for k = 1:1:num_homo
        idx = samples(rn,k);
        %%%%%%%%% Contruct the 'A' matrix for homography estimation %%%%%%%
        A(2*k-1,1:9) = [0,0,0,-[corr_l1c_l1(2,idx),corr_l1c_l1(1,idx),1],
            corr_l1c_c(1,idx)*[corr_l1c_l1(2,idx),corr_l1c_l1(1,idx),1]];
        A(2*k,1:9) = [[corr_l1c_l1(2,idx),corr_l1c_l1(1,idx),1],0,0,0,-
            corr_l1c_c(2,idx)*[corr_l1c_l1(2,idx),corr_l1c_l1(1,idx),1]];
    end
    [U,D,V] = svd(A); %% Perform SVD to solve for homography
    h_vec(rn,1:9) = V(:,9); %% Obrtain homography vector
end

%% Calculate the error for each random trial(set), while each set(trial)
%  consist of 6 random data points
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    error = zeros(Ransac_l1c_N,1);
for rn = 1:1:Ransac_l1c_N
    H = [h_vec(rn,1),h_vec(rn,2),h_vec(rn,3);h_vec(rn,4),h_vec(rn,5),h_vec(rn
        ,6);h_vec(rn,7),h_vec(rn,8),h_vec(rn,9)];
    for k = 1:1:cnt_l1c
        idx = k;
        x_prime_calc = H*[corr_l1c_l1(2,idx);corr_l1c_l1(1,idx);1];
        x_prime_calc = (x_prime_calc/(x_prime_calc(3)));
        x_prime = [corr_l1c_c(2,idx);corr_l1c_c(1,idx);1];
        error(rn) = error(rn) + sumsqr(x_prime-x_prime_calc);
    end
end

%% RANSAC outlier elimination process, detalailed description see text part
%  in report
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for rn = 1:1:Ransac_l1c_N
    if(error(rn) ~= max(error) && error(rn) < sigma)
        for k = 1:1:num_homo
        idx = samples(rn,k);
```

```
45          outlier_flag(idx) = 0;
46          end
47      else
48      end
49  end
50     sum(outlier_flag); % check the sum of remaining outliers
51  end
52
53  %% Generate the plot of inliers vs. outliers %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
54  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
55  figure
56  image(img1c)
57  truesize;
58  hold on;
59  for k = 1:1:cnt_l1c
60      if (outlier_flag(k) == 1)
61       % Mark outliers as RED
62       plot([corr_l1c_l1(2,k),corr_l1c_c(2,k)+N],[corr_l1c_l1(1,k),corr_l1c_c(1,
             k)],'-or','linewidth',3);
63      else
64       % Mark inliers as green
65       plot([corr_l1c_l1(2,k),corr_l1c_c(2,k)+N],[corr_l1c_l1(1,k),corr_l1c_c(1,
             k)],'-og','linewidth',1);
66      end
67  end
68
69  %% For the inliers, perform linear least squared method to obatain a more
70  %  accurate homography
71  cnt = 0;
72  for k = 1:1:cnt_l1c
73      if(outlier_flag(k) ~= 1)
74      idx = k;
75      B(2*cnt+1,1:9) = [0,0,0,-[corr_l1c_l1(2,idx),corr_l1c_l1(1,idx),1],
           corr_l1c_c(1,idx)*[corr_l1c_l1(2,idx),corr_l1c_l1(1,idx),1]];
76      B(2*cnt+2,1:9) = [[corr_l1c_l1(2,idx),corr_l1c_l1(1,idx),1],0,0,0,-
           corr_l1c_c(2,idx)*[corr_l1c_l1(2,idx),corr_l1c_l1(1,idx),1]];
77      cnt = cnt + 1;
78      else
79      end
80  end
81  [U,D,V] = svd(B); %% Perform SVD to solve for homography
82  h = V(:,9); %% Obrtain homography vector
83  H = [h(1),h(2),h(3);h(4),h(5),h(6);h(7),h(8),h(9)];%% Form homography matrix
84
85  %% Additioanl step: check error after RANSAC algorithms %%%%%%%%%%%%%%%%%%%%
86  error_after_RANSAC = 0;
87  for k = 1:1:cnt_l1c
88      if(outlier_flag(k) ~= 1)
89          idx = k;
90          x_prime_calc = H*[corr_l1c_l1(2,idx);corr_l1c_l1(1,idx);1];
91          x_prime_calc = (x_prime_calc/(x_prime_calc(3)));
92          x_prime = [corr_l1c_c(2,idx);corr_l1c_c(1,idx);1];
93          error_after_RANSAC = error_after_RANSAC + sumsqr(x_prime-x_prime_calc)
               ;
```

```
    else
    end
end

%% Construct the initial condition of Ax = b for DogLeg Algorithms %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
row_cnt = 1;
for k = 1:1:cnt_l1c
    if(outlier_flag(k) ~= 1)
        idx = k;
        A_dogleg(row_cnt,1:3) = [corr_l1c_l1(2,idx),corr_l1c_l1(1,idx),1];
        b_dogleg(row_cnt,1:3) = [corr_l1c_c(2,idx);corr_l1c_c(1,idx);1];
        row_cnt = row_cnt + 1;
    else
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
H_dogleg = DogLeg(A_dogleg,H',b_dogleg); % Call DogLeg to refine homography
H = H_dogleg'; % Update and return the refined homography for DogLeg

end
```

9.3 DogLeg Algorithm: `DogLeg.m`

```
function [ H_dogleg ] = DogLeg(A,H,b)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
Jf = A;
AHresult = A*H;
size_AH = size(AHresult);
size_I = size(Jf'*Jf);

for k = 1:1:size_AH(1)
    AH(k,1:3) = AHresult(k,1:3)/AHresult(k,3);
end

Ep = b-AH;
rk = 0.05*norm(H);

sigma_GD = (norm(Jf'*Ep)/norm(Jf*Jf'*Ep))*(Jf'*Ep);
up = rk*max(diag(Jf'*Jf));

sigma_GN = ((Jf'*Jf + up*eye(size_I(1)))^(-1))*Jf'*Ep;

if(norm(sigma_GN)<rk)
    H = H + sigma_GN;
elseif (norm(sigma_GD)<rk) && (rk < sigma_GN)
    H = H + sigma_GD;
else
    H = H + rk/(norm(sigma_GD))*sigma_GD;
end

```

```
29 H_dogleg = H;
30 end
```