Computer Vision Project 2

Author: Haolin Li, Honsong Li

Abstract: In this project, we are mainly working on automatic image mosaicking. In order to do image mosaicking, first we use Harries corner detection and Non-maximum suppression to find corners in two adjacent images, and then we use feature correlation to collect corners which appears in both images, and save the location of those points; because the previous step always have some "wrong pairs", so next we use RANSC to choose which pair is correct and which is not; after doing that, we can use those right point pairs from two adjacent images to generate the homograph matrix between these two images, which means now we can transform one of our images respect to the coordinate of another image; finally, we use the homograph matrix and image stitching to connect those different images and get a panorama from them.

Keyword: image mosaicking, corner detect, non-max suppression, RANSAC, homograph matrix, image stitching, panorama

I. Introduction[1]

IN this project, we are working on automatic image mosaicking, this is a very famous study concentration in computer vision field, because with image mosaicking, we can get a high resolution and wide range image just with some image taken by a low resolution and small range sensor or camera. In order to do image mosaicking, first we need to find some features appear in both image and belong to a same thing, this is a kind of correlation between these two images, we use Harris corner detection to find those features form two images separately and then we use Non-max suppression to collect some "sparse corner" in our image, this is because if we use dense corner to calculate homograph matrix between two images, the matrix will be very sensitive to the location of different pixels, and this will makes our result seems ugly; then we use these "sparse corner" or "sparse features" and normalized cross correlation (NCC) to find the feature point in image 1 is same like which feature point in image 2, and this always means these two points are coming from a same thing in the real world, so the pair of points shown some correlation between these two image, but some point in the image would really looks like each other even if they are coming from different real things, so, if we just use NCC to find the correlation points between two images, there will always have some "wrong point pairs", which means these points have a high NCC value, but they are coming from two different things in 3-D world, in order to solve this problem, we use RANSAC in our point pairs dataset and choose those points which have the most "neighbors" in the dataset as our result, so now we can get the correct correlation point pairs from two adjacent images. Secondly, after we get the point pairs from two adjacent images, we can use at least 4 points to generate homograph matrix of these two images, this means we can transform one of these two image respect to the coordinates of another image, during this way, we can use this homograph matrix to stitch these images.

II. Corner Detection

A. Harris Corner Detection

Harris corner detection is a way to detect corner, it has four steps:

1. Use edge detection mask to detect edge in x and y directions and save them as I_x and I_y.
2. Calculate ${I_x}^2$, ${I_y}^2$ and $I_x I_y$.
$$
\begin{aligned}
{I_{xij}}^2 &= I_{xij} I_{xij} \\
{I_{yij}}^2 &= I_{yij} I_{yij} \\
I_x I_y &= I_{xij} I_{yij}
\end{aligned}
$$
3. Find C matrix for each pixel
$$
\mathrm{C} = \begin{bmatrix} \sum w_i \, {I_x}^2 & \sum w_i I_x I_y \\ \sum w_i I_x I_y & \sum w_i \, {I_y}^2 \end{bmatrix}
$$
4. And then we can use R value to determine whether this pixel is a corner, edge or flat, where:
$$
\mathrm{R} = \det(\mathrm{C}) - k * (trace(C))^2 \quad (\mathrm{k} \in (0.04, 0.06))
$$
if $\lambda_1 \; and \; \lambda_2$ are eigen value for matrix C, then:
$$
\det(\mathrm{C}) = \lambda_1 * \lambda_2
$$
$$
\mathrm{trace(C)} = \lambda_1 + \lambda_2
$$
$$
\text{type} = \begin{cases} corner & R \gg 0 \\ edge & R < 0 \\ flat & R > 0 \end{cases}
$$

Then, with the Harris edge detection, we can find which pixel is a corner and which is not.

B. Non-max Suppression

Non-max suppression is a way to sparse the corner we find, because when we generate homograph matrix, using spare feature to generate the matrix will make the matrix non-sensitive to the position of pixels, so we will get a better result. There are six steps of Non-max suppression in our project:

1. Get the R matrix from the result of Harris corner detection and maximum value Rmax from R.
2. Set a high threshold H_T which is k*Rmax. ($k \in (0.3,0.99)$).
3. Find all the pixels whose R value is bigger than H_T, and we call them must_a_corner point.
4. Set a low threshold L_T = c*Rmax ($c \in (0,0.1)$), which means if a point's R value bigger than L_T, then we think this point is a corner.
5. Find all the point with R value bigger than L_T but smaller than H_T and calculate the distance for each of this point to it's nearest must_a_corner point, save this distance for each corner point.
6. Sort the distance we get from the previous step form large to small, and then choose those point which have large distance value as many as we want.
7. Then all points of must_a_corner point and the point you choose from step 6 will be our result.

This idea is a very good way to find spare corner, because after we calculate the distances between corner points and must_a_corner points, if the distance is large, it means the corner point is locate at a place where we don't have a lot of must_a_corner points, this means if we choose a corner point at this place, our point dataset will become sparse.

III. Points Correlation

A. Normalized Cross Correlation

Normalized cross correlation (NCC) is a way to measure the similarity between two matrix or two images, it is the same as cross correlation, but before we use cross correlation for two matrix, we normalize both matrix firstly and then use cross correlation, during this way, there are four steps in Normalized cross correlation.

1. Get the matrix you want to find, we call it model.
2. Get the background or image you want to use to find where the models locate.
3. Normalize the model and the background.

$$f^{\wedge} = \frac{f}{||f||} = \frac{f}{\sqrt{\sum_{[i,j]\epsilon R} f^2(i,j)}}$$
$$g^{\wedge} = \frac{g}{||g||} = \frac{g}{\sqrt{\sum_{[i,j]\epsilon R} g^2(i,j)}}$$

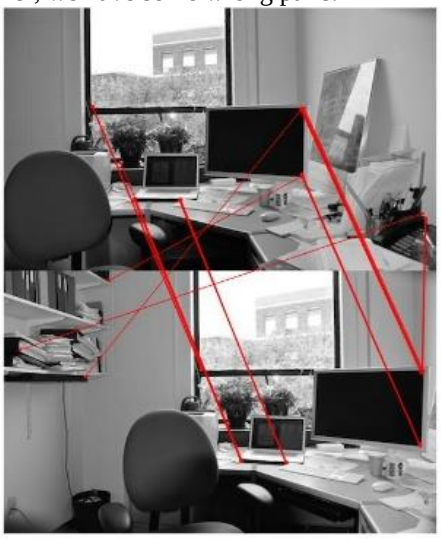
4. Calculate the normalized cross correlation with $f^{\wedge}$ and $g^{\wedge}$, with the equation shown below

$$N_{fg} = C_{f^{\wedge}g^{\wedge}} = \sum_{[i,j]\epsilon R} f^{\wedge}(i,j)g^{\wedge}(i,j)$$

After we get NCC for each corner, we use the point which have the highest NCC value as our result, so then we can find the correlation points in two adjacent images.

B. Random Sample Consensus (RANSAC)

After we find the correlation points from two different image, due to some different area really looks the same as each other, we have some wrong pairs.



Img1. Have wrong pairs

In order to eliminant those wrong pairs, we use RANSAC.

RANSAC is a powerful method to eliminant the outlier points and keep the inlier point as much as possible, in order to use RANSAC, we need to find the "point" we will use in RANSAC and the measurement used to determine inlier and outlier points. In this project, we use a pair of points in two adjacent images as a "point" in RANSAC and use the mean square error between the real correlation point and point generated by homograph matrix as the measurement to determine inlier or outlier points. There are five steps to do RANSAC.

1. Randomly choose 4 pairs of points from dataset.
2. Calculate the homograph matrix based on points we get form step 1.
3. Apply the homograph matrix to all pairs in our dataset and use measurement we describe before to collect the number of inlier "points".
4. Because this question is in 4-dimensional space, so we do step 1,2,3 at least 75 times, and finally choose the homograph matrix which have the most inlier "points" as the right homograph matrix between these two images.
5. Apply the right homograph matrix with all "points" in our dataset, and only keep those pairs which have a small mean square error.

After doing RANSAC, our result looks much clearly and beautiful than before.

Img2. After RANSAC

As we can see, after RANSAC, the correlation points look perfect.

IV. Generate Homograph Matrix

Homograph matrix is a matrix to describe the perspective transformation between two images, although the H matrix have nine unknown variables, it's Dof is just eight, so we can use four pairs of point to generate H matrix. Meanwhile, we can also use more than four pairs of point to generate a more precision H matrix.

A. *Use Four Pairs of point*

Assume H matrix is:

$$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

Because we know:

$$\begin{bmatrix} x_{21} \\ y_{21} \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x_{11} \\ y_{11} \\ 1 \end{bmatrix}$$

So, we will get two equations from the previous matrix equation:

$$x_{21} = \frac{h_1 x_{11} + h_2 y_{11} + h_3}{h_7 x_{11} + h_8 y_{11} + h_9}$$

$$y_{21} = \frac{h_4 x_{11} + h_5 y_{11} + h_6}{h_7 x_{11} + h_8 y_{11} + h_9}$$

If we use four pairs of point, then we can assume $h_9 = 1$ and we will get the equation shown below:

$$\begin{bmatrix} x_{11} & y_{11} & 1 & 0 & 0 & 0 & (-x_{11}x_{21}) & (-y_{11}x_{21}) \\ 0 & 0 & 0 & x_{11} & y_{11} & 1 & (-x_{11}y_{21}) & (-y_{11}y_{21}) \\ & & & & \vdots & & & \\ x_{14} & y_{14} & 1 & 0 & 0 & 0 & (-x_{14}x_{24}) & (-y_{14}x_{24}) \\ 0 & 0 & 0 & x_{14} & y_{14} & 1 & (-x_{14}y_{24}) & (-y_{14}y_{24}) \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_7 \\ h_8 \end{bmatrix} = \begin{bmatrix} x_{21} \\ y_{21} \\ \vdots \\ x_{24} \\ y_{24} \end{bmatrix}$$

Then we can get H matrix.

B. *Use More than Four pairs*

After RANSAC, we want to use all pairs of point we have to generate a more precisely Homograph matrix for two images, we can use the same equation shown before, but add more points in it.

$$\begin{bmatrix} x_{11} & y_{11} & 1 & 0 & 0 & 0 & (-x_{11}x_{21}) & (-y_{11}x_{21}) \\ 0 & 0 & 0 & x_{11} & y_{11} & 1 & (-x_{11}y_{21}) & (-y_{11}y_{21}) \\ & & & & \vdots & & & \\ x_{14} & y_{14} & 1 & 0 & 0 & 0 & (-x_{14}x_{24}) & (-y_{14}x_{24}) \\ 0 & 0 & 0 & x_{14} & y_{14} & 1 & (-x_{14}y_{24}) & (-y_{14}y_{24}) \\ & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} x_{21} \\ y_{21} \\ \vdots \\ x_{24} \\ y_{24} \\ \vdots \end{bmatrix}$$

V. Shown Panorama Result

A. *The Flowchart of our Idea*

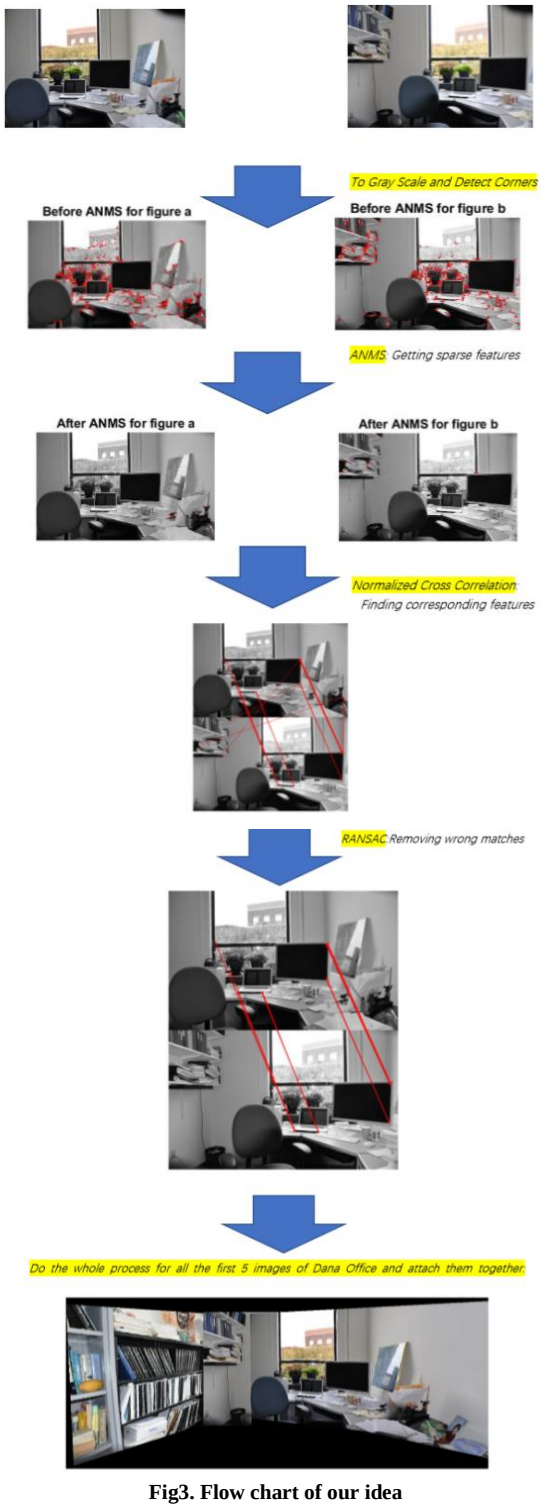


Fig3. Flow chart of our idea

B. The result of corner detection and corner detection after NMS

We get the result of corner detection with and without NMS, some results are shown below:

For DanaHallWay1 is:

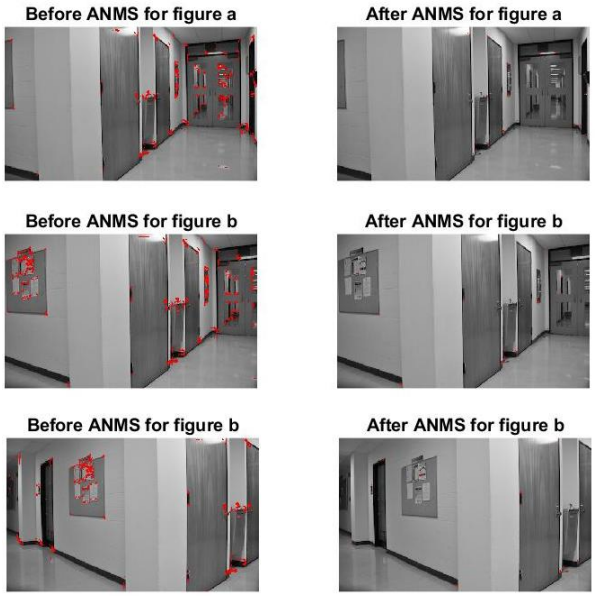


Fig4. Image form DanaHallWay1

For DanaHallWay2 is:

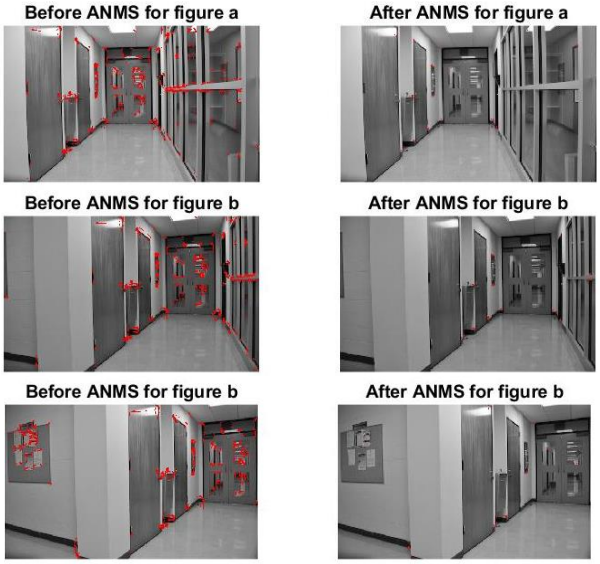


Fig5. Image form DanaHallWay2

And we also calculate the result of Dana office: (just some of it)

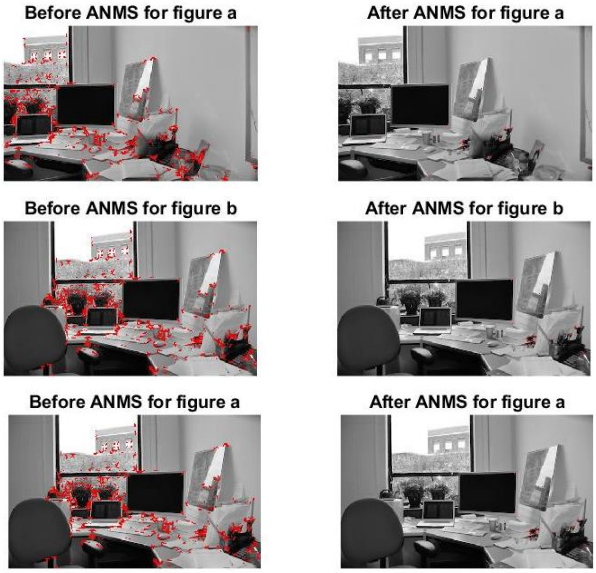


Fig6. Image form DanaOffice

As we can seen from the above result, our code works well for all these different images.

C. Using NCC and RANSAC to find the correlation points

In this section, we are shown some of our result about correlation points between two images with or without RANSAC, pair points are connected with red line.

For DanaHallWay1 is:

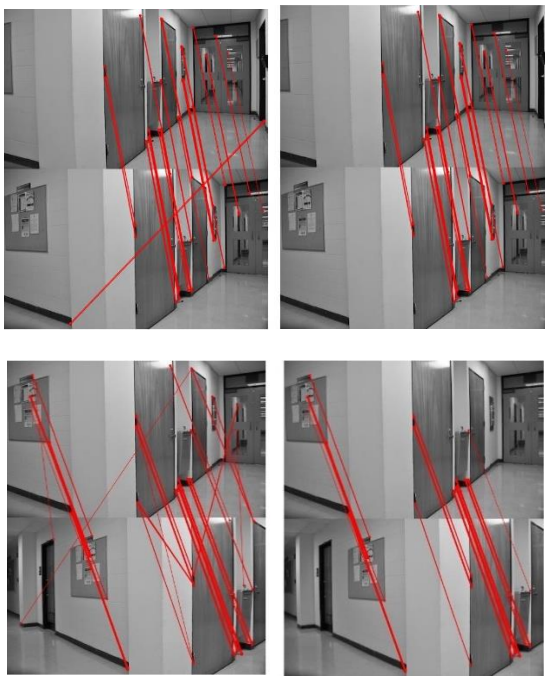


Fig7. Image form DanaHallWay1, left are result with NCC, right are result with NCC and RANSAC

For DanaHallWay2 is:

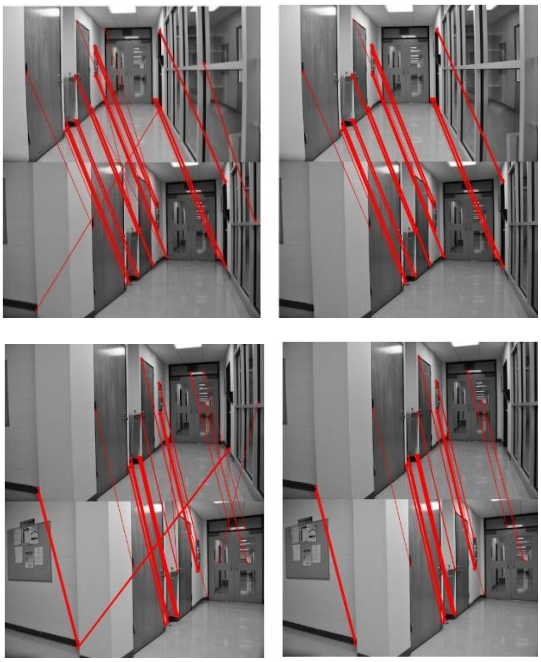


Fig8. Image form DanaHallWay2, left are result with NCC, right are result with NCC and RANSAC

Result of Dana office: (just some of it)

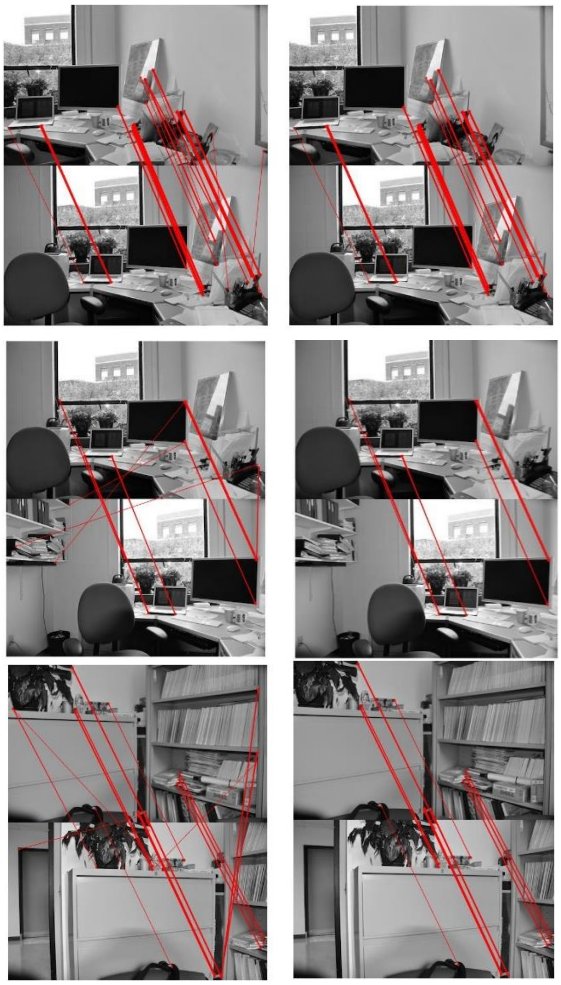


Fig9. Image form DanaOffice, left are result with NCC, right areresult with NCC and RANSAC

D. The Panorama for Each File

After we find the correlation points between every two adjacent images, we can calculate the homograph matrix of these two images and use this matrix to stitch image. The results of Panorama are shown here, we generate Panorama for each file.

For DanaHallWay1 is:

Fig10. Panorama of DanaHallWay1

For DanaHallWay2 is:

Fig11. Panorama of DanaHallWay2

Result of Dana office:

Because in Dana office file, image 312 and 313 have no correlation points, so we separate Dana office Panorama into two parts:

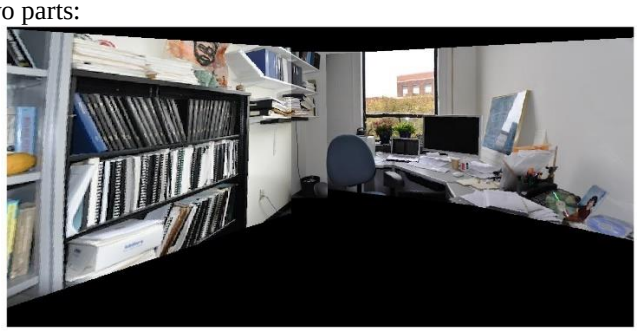


Fig12. Panorama of DanaOffice part 1

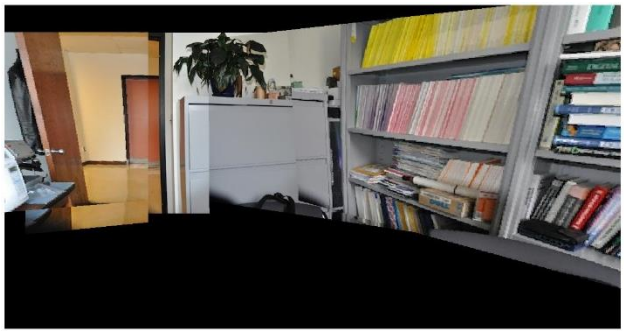


Fig13. Panorama of DanaOffice part 2

The results look not so good, the boundary between two images is obvious, but all results are acceptable.

VI. Conclusion

In this project, we are working on automatically image mosaicking, in order to do image mosaicking, we use corner as the feature to connect two adjacent images and use Harris corner detection to find corner, then we use non-max suppression (NMS) to sparse the corner we find, so we can get a better homograph matrix; after that, we use the NCC and RANSAC to find the correlation points from two images and use those points to calculate the homograph matrix for two adjacent images, finally, we use all image we have and their homograph matrix to generate panorama for each file. The code works well and gives the result.

Appendix

This is our code (In Matlab).

Main.m

```
clear all;
clc;
clear;clc;
image = [];
% % a =
imread('./DanaOffice/DSC_0308.JPG');
% % b =
imread('./DanaOffice/DSC_0309.JPG');
% % c =
imread('./DanaOffice/DSC_0310.JPG');
% % d =
imread('./DanaOffice/DSC_0311.JPG');
% % e =
imread('./DanaOffice/DSC_0312.JPG');
% f =
imread('./DanaOffice/DSC_0313.JPG');
% g =
imread('./DanaOffice/DSC_0314.JPG');
% h =
imread('./DanaOffice/DSC_0315.JPG');
% i =
imread('./DanaOffice/DSC_0316.JPG');
% j =
imread('./DanaOffice/DSC_0317.JPG');
% % image = cat(4,image,a);
% % image = cat(4,image,b);
% % image = cat(4,image,c);
% % image = cat(4,image,d);
% % image = cat(4,image,e);
% image = cat(4,image,f);
% image = cat(4,image,g);
% image = cat(4,image,h);
% image = cat(4,image,i);
% image = cat(4,image,j);
a =
imread('./DanaHallWay1/DSC_0283.JPG');
b =
imread('./DanaHallWay1/DSC_0282.JPG');
c =
imread('./DanaHallWay1/DSC_0281.JPG');
image = cat(4,image,a);
image = cat(4,image,b);
image = cat(4,image,c);
% a =
imread('./DanaHallWay2/DSC_0287.JPG');
% b =
imread('./DanaHallWay2/DSC_0286.JPG');
% c =
imread('./DanaHallWay2/DSC_0285.JPG');
% image = cat(4,image,a);
% image = cat(4,image,b);
% image = cat(4,image,c);
H_matrix =
[];%zeros([3,3,size(image,4)]);
middle_number = round(size(image,4)/2);
keep_correspond1 = {};
keep_correspond2 = {};
```

```
for i = 1:size(image,4)
    if i < middle_number
        [keep_correspond1_use,
keep_correspond2_use] =
connect_two_image(image(:,:,:,i),image(:
,:,:,i+1));

keep_correspond1=[keep_correspond1,keep_
correspond1_use];

keep_correspond2=[keep_correspond2,keep_
correspond2_use];

        H_matrix =
[H_matrix,cp2tform(keep_correspond1_use,
keep_correspond2_use,'projective')];
    elseif i > middle_number
        [keep_correspond1_use,
keep_correspond2_use] =
connect_two_image(image(:,:,:,i-
1),image(:,:,:,i));

keep_correspond1=[keep_correspond1,keep_
correspond1_use];

keep_correspond2=[keep_correspond2,keep_
correspond2_use];

        H_matrix =
[H_matrix,cp2tform(keep_correspond2_use,
keep_correspond1_use,'projective')];
    else
        keep_correspond1_use =
[1,1;1,size(image,1);size(image,2),1;siz
e(image,2),size(image,1)];
        keep_correspond2_use =
[1,1;1,size(image,1);size(image,2),1;siz
e(image,2),size(image,1)];
        H_matrix =
[H_matrix,cp2tform(keep_correspond2_use,
keep_correspond1_use,'projective')];
    end
end
%% connect image
figure()
after_project = {};
img_final = [];
for i = 1:size(image,4)
    %i = size(image,4)-t+1;
    if i <= middle_number
        kk = i;
        use_H = H_matrix(i);
        for j = kk+1:middle_number
            use_H.tdata.T =
(H_matrix(j).tdata.T'*use_H.tdata.T')';
            use_H.tdata.Tinv =
(use_H.tdata.Tinv'*H_matrix(j).tdata.Tin
v')';
        end
        after_project =
[after_project,imtransform(image(:,:,:,i
),use_H)];%,
    else
        kk = i;
        use_H = H_matrix(i);
        for j = 1:kk-middle_number
            use_H.tdata.T =
(H_matrix(kk-
j).tdata.T'*use_H.tdata.T')';
            use_H.tdata.Tinv =
(use_H.tdata.Tinv'*H_matrix(kk-
j).tdata.Tinv')';
        end
        after_project =
[after_project,imtransform(image(:,:,:,i
),use_H)]; %
    end

    mm = cell2mat(after_project(i));
    subplot(1,size(image,4),i);
    imshow(mm(:,:,:));
    %img_final = cat(4,img_final,mm);
end
%% the right part
stitch_two1 = image(:,:,:,1);
for i = 1:middle_number-1
    %if i <= middle_number-1
        stitch_two1 =
stitch_website_right_part( image(:,:,:,i
+1), stitch_two1,
H_matrix(i).tdata.Tinv');%homography)


end
figure()

imshow(stitch_two1);

%% the left part
stitch_two2 =
image(:,:,:,size(image,4));
stitch_two2 = stitch_two2(:,end:-1:1,:);
for i = 1:middle_number-1 %i =
middle_number+1:size(image,4)
    k = size(image,4)-i+1;
    use_image = image(:,:,:,k-1);
    use_image = use_image(:,end:-1:1,:);
    %stitch_two2 =
stitch_website_left_part( image(:,:,:,k-
1),stitch_two2,
H_matrix(k).tdata.Tinv');%homography)
    stitch_two2 =
stitch_website_right_part( use_image,sti
tch_two2, H_matrix(k).tdata.T');
%     use_H_use =
H_matrix(i).tdata.Tinv';
%     j = i-1;
%     while j > middle_number
%     use_H_use =
H_matrix(j).tdata.Tinv'*use_H_use;
%     j = j - 1;
```

```
%     end
%     stitch_two2 =
stitch_website_left_part( stitch_two2,im
age(:,:,:,i),
use_H_use);%H_matrix(i).tdata.Tinv');
end
%subplot(1,2,2)
%imshow(stitch_two2);
stitch_two2 = stitch_two2(:,end:-1:1,:);
figure()
imshow(stitch_two2);
%%
%stitch_two2 is left part, stitch_two1
is rigth part
for i= 1:size(stitch_two1,1)
    if stitch_two1(i,1) ~= 0
        notzero1 = i;
    end
end
for i= 1:size(stitch_two2,1)
    if
stitch_two2(i,size(stitch_two2,2)) ~= 0
        notzero2 = i;
    end
end
%if size(stitch_two1,1)>
size(stitch_two2,2)
    if notzero1 - notzero2 > 0
        stitch_two2 =
padarray(stitch_two2, [notzero1-notzero2
0], 0, 'pre');
    else%if notzero1 - notzero2 < 0
        stitch_two1 =
padarray(stitch_two1, [notzero2-notzero1
0], 0, 'pre');
    end
    if size(stitch_two1,1)-
size(stitch_two2,1) > 0
        stitch_two2 =
padarray(stitch_two2,
[size(stitch_two1,1)-size(stitch_two2,1)
0], 0, 'post');
    else
        stitch_two1 =
padarray(stitch_two1,
[size(stitch_two2,1)-size(stitch_two1,1)
0], 0, 'post');
    end

stitch_result =
[stitch_two2(:,1:(size(stitch_two2,2)-
size(image,2))+60,:),stitch_two1(:,60:en
d,:)];
%stitch_result =
stitch_website_left_part( stitch_two1,st
itch_two2, [1,0,0;0,1,0;0,0,1]);
figure()
imshow(stitch_result);
```

connect_two_image.m

```
function[ keep_correspond1,
keep_correspond2] =
connect_two_image(a,b)  %a,b are two
image you want to connect
% keep_correspond1 and  keep_correspond2
are the connected point from two
% different image
%% read in
    a_grey =
rgb2gray(a(30:size(a,1),:,:));
    b_grey =
rgb2gray(b(30:size(a,1),:,:));
    %% find corner
    [a_corner_ori, a_R,a_Rmax] =
find_corner(a_grey);
    [b_corner_ori, b_R,b_Rmax] =
find_corner(b_grey);
    %% apply ANMS
    n = 300;
    a_corner_ANMS = ANMS(a_R, a_Rmax,
n);
    b_corner_ANMS = ANMS(b_R, b_Rmax,
n);
    %% show the differences between the
result apply ANMS or not
    figure()
    subplot(1,2,1);
    show_corner(a_grey, a_corner_ori);
    title('Before ANMS for figure a')
    subplot(1,2,2);
    show_corner(a_grey, a_corner_ANMS);
    title('After ANMS for figure a');
    figure()
    subplot(1,2,1);
    show_corner(b_grey, b_corner_ori);
    title('Before ANMS for figure b');
    subplot(1,2,2);
    show_corner(b_grey, b_corner_ANMS);
    title('After ANMS for figure b');
    %% find the same corner in two image
    [correspond1,correspond2] =
correspondence(a_grey, b_grey,
a_corner_ANMS, b_corner_ANMS,n);
    %% show the reuslt
    connect = [a_grey;b_grey];
    %showMatchedFeatures(a_grey,b_grey,
matchedPoints1,matchedPoints2);
    %for i =
size(correspond,2)/2:size(correspond,2)
    %    use_correspond(2,i) =
use_correspond(2,i)+ size(a_grey,2);
    %end

    use_correspond1 =
[correspond1(:,2),correspond1(:,1)];
    use_correspond2 =
[correspond2(:,2),correspond2(:,1)];
    use_correspond2(:,2) =
use_correspond2(:,2)+ size(a_grey,1);
```

```
    figure
    imshow(connect);
    hold on
    for i = 1:size(use_correspond1,1)

plot([use_correspond1(i,1),use_correspon
d2(i,1)],
[use_correspond1(i,2),use_correspond2(i,
2)],'r-');
        hold on
    end
    plot(use_correspond1(:,1),
use_correspond1(:,2),'r.');
    plot(use_correspond2(:,1),
use_correspond2(:,2),'r.');
    %for i = 1:size(use_correspond,2)-1
    %
plot(use_correspond(1:2,i),use_correspon
d(1:2,i+1),'r-')
    %    hold on
    %end
    %% use RANSAC
    use_correspond2(:,2) =
use_correspond2(:,2)- size(a_grey,1);
    %% calculate H
    H_save = zeros([3,3,100]);
    save_online = zeros([1,100]);
    for
i=1:120%110%size(use_correspond2,1)
        %randomli choose 4 pair points,
and calculate the H transformation
        %matrix, then transfor every
point from one image two another,
        %calcualte the tranformation and
original error, choose the minimum
        %error 4 points, save H
        use_point1 = zeros([4,2]);
        use_point2 = zeros([4,2]);
        for j = 1:4
            choose_point_index1 =
ceil(rand(1,1)*size(use_correspond1,1));
            %choose_point_index2 =
ceil(rand(1,1)*size(use_correspond2,1));
%randomly choose point
            use_point1(j,:) =
use_correspond1(choose_point_index1,:);
            use_point2(j,:) =
use_correspond2(choose_point_index1,:);
        end
        %already get 4 pairs of
point,calculate matrix H
        H =
genreate_tranformation(use_point1,
use_point2);
        %now we get H, use H to
translate point from img2 to img1,
calculate
        %the error between tranlate
point and real pair point, choose the
pair
        %which have the minimum value;
and can also set a threshold, minimum
        %value should smaller than some
value
        if max(max(H)) == Inf ||
sum(sum(isnan(H))) > 0
            continue;
        end
        if rank(H) == 3
            H_save(:,:,i) =
H;%inv(H); %H is H matric from 1 to 2,
inv(H) is from 2 to 1
            online = 0;
            for j =
1:size(use_correspond1,1)
                fake1 =
H*[use_correspond1(j,:),1]';%H\[use_corr
espond2(j,:),1]';
                fake1 =
[round(fake1(1)/fake1(3)),round(fake1(2)
/fake1(3))];
                if
sqrt(sum((use_correspond2(j,:) -
fake1).^2)) < sqrt(6)%1.^2+1.^2)
                    online = online + 1;
                end
            end
            save_online(i) = online;
        end
    end
    %[online_result,id] =
sort(save_online,'descend');
    [online_max,id] = max(save_online);
    H_result = H_save(:,:,id); %H_result
should be the right tranformation H
matrix, from img2 to img1
    %% So the after RANSAC result
    keep_correspond1 = [];
    keep_correspond2 = [];
    for i = 1:size(use_correspond1,1)
        fake1 =
H_result*[use_correspond1(i,:),1]';
        fake1 =
[round(fake1(1)/fake1(3)),round(fake1(2)
/fake1(3))];
        if
sqrt(sum((use_correspond2(i,:) -
fake1).^2)) < sqrt(10)%2.^2+2.^2)  %the
generous
            keep_correspond1 =
[keep_correspond1;use_correspond1(i,:)];
            keep_correspond2 =
[keep_correspond2;use_correspond2(i,:)];
        end
    end
    keep_correspond2(:,2) =
keep_correspond2(:,2)+ size(a_grey,1);
    figure
    imshow(connect);
    hold on
    for i = 1:size(keep_correspond1,1)
```

```
plot([keep_correspond1(i,1),keep_corresp
ond2(i,1)],
[keep_correspond1(i,2),keep_correspond2(
i,2)],'r-');
        hold on
    end
     plot(keep_correspond1(:,1),
keep_correspond1(:,2),'r.');
    hold on
    plot(keep_correspond2(:,1),
keep_correspond2(:,2),'r.');
    keep_correspond2(:,2) =
keep_correspond2(:,2)- size(a_grey,1);
end
```

find_corner.m

```
function[corner_img, result, Rmax] =
find_corner(img)  %corner_img
    %use prewitt filter to calculate
gradient
    %use 5*5 Gaussian filter to smooth
    prew_h = [1 2 1; 0 0 0; -1 -2 -1];
    prew_v = [-1 0 1; -2 0 2; -1 0 1];
    H=fspecial('gaussian',9, 2);%
generate Gaussian filter

img_gauss=imfilter(img,H,'replicate'); %
filt image
    %start calculate gradient
    img_gradient_x = conv2(img_gauss,
prew_h, 'valid');%'same');
    img_gradient_y = conv2(img_gauss,
prew_v, 'valid');%'same');
    x_square =
img_gradient_x.*img_gradient_x;
    y_square =
img_gradient_y.*img_gradient_y;
    xy_use =
img_gradient_x.*img_gradient_y;
    %process to generate C
    h = fspecial('gaussian',[5
5],1);%ones([5,5]); use gaussian weight
    xx = conv2(x_square,h,'same');
    yy = conv2(y_square,h,'same');
    xy = conv2(xy_use,h,'same');
    Rmax = 0;
    corner_img =
zeros([size(x_square,1),
size(x_square,2)]);
    result = zeros([size(x_square,1),
size(x_square,2)]);
    %R_matrix = zeros([size(x_square,1),
size(x_square,2)]);
    for i = 1:size(x_square,1)
        for j = 1:size(y_square,2)
            C = [xx(i,j) xy(i,j);
xy(i,j) yy(i,j)];
            R = det(C) -
0.04*(trace(C)^2);
            result(i,j) = R;
            if R > Rmax
                Rmax = R;
            end
        end
    end
    for i = 1:size(x_square,1)
        for j = 1:size(y_square,2)
            if result(i,j) > 0.0005 *
Rmax
                corner_img(i,j) = 1;
            end
        end
    end
end
```

ANMS.m

```
function[corner] = ANMS(R, Rmax,
number_choose)
    large_row1 = [];
    large_row2 = [];
    medium_row1 = [];
    medium_row2 = [];
    medium_distance = [];
    for i = 1:size(R,1)
        for j = 1:size(R,2)
            if R(i,j) >= 0.5*Rmax
                large_row1 = [large_row1
i];
                large_row2 = [large_row2
j];
            end
        end
    end
    large = [large_row1;large_row2];
    for i = 1:size(R,1)
        for j = 1:size(R,2)
            if R(i,j) < 0.5*Rmax &&
R(i,j) >= 0.05*Rmax
                use =
[i*ones([1,size(large,2)]);j*ones([1,siz
e(large,2)])];
                temp = large - use;
                medium_row1 =
[medium_row1 i];
                medium_row2 =
[medium_row2 j];
                medium_distance =
[medium_distance min(sqrt(temp(1,:).^2 +
temp(2,:).^2))];
            end
        end
    end
    medium = [medium_row1;medium_row2];
    %
    corner =
zeros([size(R,1),size(R,2)]);
    for i = 1:size(large,2)
```

```
        corner(large(1,i),large(2,i)) =
1;
        if number_choose > 0
            number_choose =
number_choose-1;
        else
            break;
        end
    end

    [medium_distance,id] =
sort(medium_distance,'descend');
    medium = medium(:,id);
    for i = 1:size(medium,2)
        if number_choose > 0

corner(medium(1,i),medium(2,i)) = 1;
            number_choose =
number_choose - 1;
        else
            break;
        end
    end
end
```

correspondence.m

```
function[correspond1,correspond2] =
correspondence(a_grey, b_grey, corner1,
corner2,n)
    [m1,n1] = size(corner1);
    %[m2,n2] = size(corner2); just
assume corner1 and corner2 are in same
    %size
    use1 = zeros([n,2]);
    use2 = zeros([n,2]);
    %use_use2 = zeros([n,2]);
    use_use1 = [];
    use_use2 = [];
    k1 = 1;
    k2 = 1;
    correspond1 = [];
    correspond2 = [];
    for i = 1:m1
        for j=1:n1
            if corner1(i,j) == 1
                use1(k1,:) = [i+1,j+1];
                k1 =k1 + 1;
            end
            if corner2(i,j) == 1
                use2(k2,:)= [i+1,j+1];
                k2 = k2 + 1;
            end
        end
    end

    for i = 1:n
        sr1 = 11;
        sr2 = 13;
        range1 = extract_range(a_grey,
use1(i,:),sr1);%7%5); %the last
parameter of size must be an odd number
        one_pixel = [];%zeros([1,n]);
        use_use2 = [];
        for j = 1:n
            range2 =
extract_range(b_grey,
use2(j,:),sr2);%9%11);
            use_range1 = range1 -
mean(mean(range1));
            use_range2 = range2 -
mean(mean(range2));
            if sum(sum(use_range1)) ~= 0
            use_range1 =
use_range1/sum(sum(use_range1));
            end
            if sum(sum(use_range2)) ~= 0
            use_range2 =
use_range2/sum(sum(use_range2));
            end
            C =
normxcorr2(use_range1,use_range2);%sum(s
um(range1.*range2))/sqrt(sum(sum(range1.
*range1)) +
sum(sum(range2.*range2)));%calculate NCC
            [ypeak, xpeak] =
find(C==max(C(:)));
            %Compute translation from
max location in correlation matrix
            %yoffSet = ypeak-
size(onion,1);
            %xoffSet = xpeak-
size(onion,2);
            if size(xpeak,1) > 1
                xpeak = xpeak(1);
                ypeak = ypeak(1);
            end
            if max(C(:)) > 0.94
                one_pixel =
[one_pixel,max(C(:))];
            %if one_pixel(j) > 0.7
                %use_use2(j,:) =
[use2(j,1)-(13-1)/2+xpeak-1,use2(j,2)-
(13-1)/2+ypeak-1];
                use_use2 =
[use_use2;[use2(j,1)-(sr2-1)/2+xpeak-1-
(sr1-1)/2,use2(j,2)-(sr2-1)/2+ypeak-1-
(sr1-1)/2]];
            end
            %find where is the maxmum
one_pixel value and make it as our
            %result
        end
        %choose the max as
correspondence
        if size(one_pixel) ~= 0
            [big,id] = max(one_pixel);
            correspond1 =
[correspond1;use1(i,:)];%[use1(i,1),use1
(i,2)]];
            correspond2 =
[correspond2;use_use2(id,:)];%[use2(id,1
),use2(id,2)]];
```

```
        end
    end
end
```

generate_tranformation.m

```
function[H] =
genreate_tranformation(use_point1,
use_point2) %from
    H = zeros([3,3]);
    A =
[use_point1(1,1),use_point1(1,2),1,0,0,0
,-use_point2(1,1)*use_point1(1,1),-
use_point2(1,1)*use_point1(1,2);

0,0,0,use_point1(1,1),use_point1(1,2),1,
-use_point2(1,2)*use_point1(1,1),-
use_point2(1,2)*use_point1(1,2);

use_point1(2,1),use_point1(2,2),1,0,0,0,
-use_point2(2,1)*use_point1(2,1),-
use_point2(2,1)*use_point1(2,2);

0,0,0,use_point1(2,1),use_point1(2,2),1,
-use_point2(2,2)*use_point1(2,1),-
use_point2(2,2)*use_point1(2,2);

use_point1(3,1),use_point1(3,2),1,0,0,0,
-use_point2(3,1)*use_point1(3,1),-
use_point2(3,1)*use_point1(3,2);

0,0,0,use_point1(3,1),use_point1(3,2),1,
-use_point2(3,2)*use_point1(3,1),-
use_point2(3,2)*use_point1(3,2);

use_point1(4,1),use_point1(4,2),1,0,0,0,
-use_point2(4,1)*use_point1(4,1),-
use_point2(4,1)*use_point1(4,2);

0,0,0,use_point1(4,1),use_point1(4,2),1,
-use_point2(4,2)*use_point1(4,1),-
use_point2(4,2)*use_point1(4,2)];

    result =
[use_point2(1,1),use_point2(1,2),use_poi
nt2(2,1),use_point2(2,2),use_point2(3,1)
,use_point2(3,2),use_point2(4,1),use_poi
nt2(4,2)];
    %A*h = result;
    H_temp = A\result';
    H_temp = [H_temp' 1]; %assume h9 = 1
    H(1,:) = H_temp(1:3);
    H(2,:) = H_temp(4:6);
    H(3,:) = H_temp(7:9);
end
```

stitch_website_right_part.m

```
function stitchedImage =
stitch_website_right_part( im1, im2,
homography)
stitchedImage = im1;
stitchedImage = padarray(stitchedImage,
[0 size(im2, 2)], 0, 'post');
stitchedImage = padarray(stitchedImage,
[size(im2, 1) 0], 0, 'both');
for i = 1:size(stitchedImage, 2)
    for j = 1:size(stitchedImage, 1)
        if  j > size(im1,1) %&&i >
size(im1,2)
        p2 = homography * [i; j-
floor(size(im2, 1)); 1];
        p2 = p2 ./ p2(3);
        x2 = floor(p2(1));
        y2 = floor(p2(2));
        if x2 > 0 && x2 <= size(im2, 2)
&& y2 > 0 && y2 <= size(im2, 1)
            stitchedImage(j, i,:) =
im2(y2, x2,:);
        end
        end
    end
end
%crop
[row,col] = find(stitchedImage);
c = max(col(:));
d = max(row(:));
st=imcrop(stitchedImage, [1 1 c d]);
[row,col] = find(stitchedImage ~= 0);
a = min(col(:));
b = min(row(:));
st=imcrop(st, [a b size(st,1)
size(st,2)]);
stitchedImage = st;
```

extract_range.m

```
function[range] = extract_range(img,
point, s)
    r = (s-1)/2;
    range = zeros([s,s]);

    range(r+1,r+1) =
img(point(1),point(2));
    for i = 1:r
        for j = 1:r
            if point(1)+i <= size(img,1)
                if point(2)+j <=
size(img,2)
                    range(r+1+i,r+1+j) =
img(point(1)+i,point(2)+j);
                end
                if point(2)-j > 0
                    range(r+1+i,r+1-j) =
img(point(1)+i,point(2)-j);
                end
            end
            if point(1)-i > 0
                if point(2)+j <=
size(img,2)
                    range(r+1-i,r+1+j) =
img(point(1)-i,point(2)+j);
                end
                if point(2)-j > 0
```

```
                    range(r+1-i,r+1-j) =
img(point(1)-i,point(2)-j);
                end
            end
        end
    end
end
```

show_corner.m

```
function[] = show_corner(img, corner)
    result =
cat(3,img,img,img);%zeros([size(img,1),s
ize(img,2)]);
    for i = 1:size(corner,1)
        for j = 1:size(corner,2)
            if corner(i,j) == 1
                result(i+1,j+1,1) = 255;
                result(i+1,j+1,2) = 0;
                result(i+1,j+1,3) = 0;
            else
                result(i,j,1) =
img(i,j);
                result(i,j,2) =
img(i,j);
                result(i,j,3) =
img(i,j);
            end
        end
    end
    %imshow(uint8(result))
    imshow(result);
end
```