Image Mosaicing

Song Jiang and Mengyun Xia

*EECE 5639, Computer Vision*

*March 14, 2016*

**Abstract**

In this project, we will apply a Harris corner detector to find corners in two images, automatically find corresponding features, estimate a homography between the two images, and warp one image into the coordinate system of the second one to produce a mosaic containing the union of all pixels in the two images.

# Introduction

Image Stitching is a widely used image processing technique. According matched feature points, we can combine many small perspective images into an image with a large viewing angle. This technique is also widely used in the synthesis of a wide-angle photo, satellite photo processing, medical image processing and other fields. The purpose of this project is to find corner features in multiple images and to align the images in a mosaic by estimating a homography between corresponding features.

1. **Description of Algorithms**
2. Read in two images and make them grayscale.
3. Apply Harris corner detector to both images: compute Harris R function over the image, and then do non-maximum suppression to get a sparse set of corner features.
4. Given two set of corners from the two images, compute normalized cross correlation (NCC) of image patches centered at each corner. Choose potential corner matches by finding pair of corners (one from each image) such that they have the highest NCC value. Then set a threshold to keep only matches that have a large NCC score to find correspondences between the two images.
5. Repeatedly sample minimal number of points needed to estimate a homography (4 pts in this case). Compute a homography from these four points. Map all points using the homagraphy and comparing distances between predicted and observed locations to determine the number of inliers. At the end, compute a least-squares homgraphy from all the inliers in the largest set of inliers.
6. Determine how big to make the final output image so that it contains the union of all pixels in the two images. Copy the image that does not have to be warped into the appropriate location in the output. Warp the other image into the output image based on the estimated homography (or its inverse). Use blending schemes to blend pixels in the area of overlap between both images.
7. **Experiments and values of parameters used**
8. Read in two images and make them grayscale.

img1=imread('DSC\_0308.JPG'); %read in image1

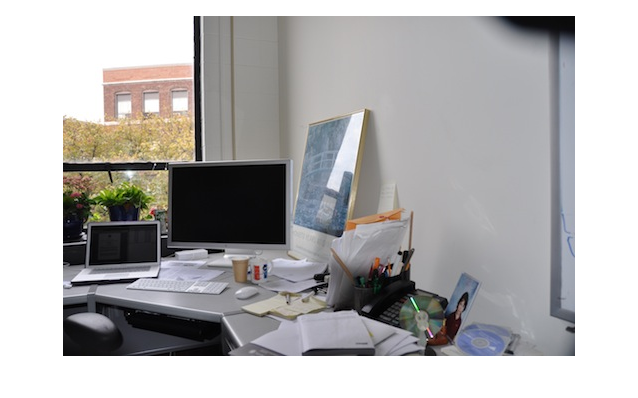
img2=imread('DSC\_0309.JPG'); %read in image2

pic1=rgb2gray(img1); %gray scale image1

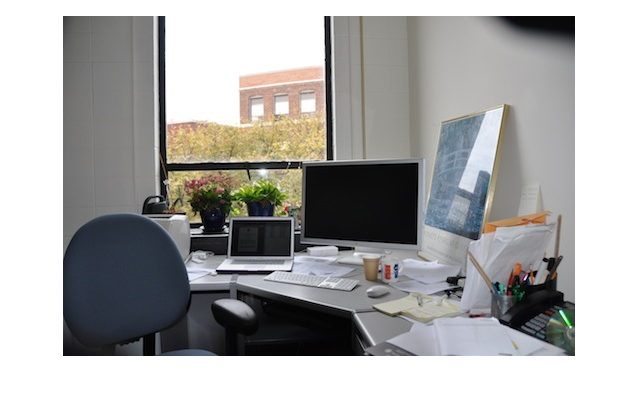
pic2=rgb2gray(img2); %gray scale image2

figure,imshow(img1);

figure,imshow(img2);



1. img1



1. img2
2. Apply Harris corner detector to both images: compute Harris R function over the image, and then do non-maximum suppression to get a sparse set of corner features.

%gaussian smoothing

der\_sig=1;

x=-3:3;

Filter\_der1D=exp(-(x).^2/(2\*der\_sig^2))/(der\_sig\*sqrt(2\*pi));

pic1\_smoothed=conv2(Filter\_der1D,Filter\_der1D',pic1,'same');

pic2\_smoothed=conv2(Filter\_der1D,Filter\_der1D',pic2,'same');

%derivative filter

Filter\_derx=[1 0 -1];

Filter\_dery=Filter\_derx';

%computer image 1 derivative by convolution in both x and y directions

%compute the derivative with respect to x

pic1\_derx=conv2(pic1\_smoothed,Filter\_derx,'same');

%compute the derivative with respect to x at the border pixels

pic1\_derx(:,1)=2\*(pic1\_smoothed(:,2) -pic1\_smoothed(:,1));

pic1\_derx(:,end)=2\*(pic1\_smoothed(:,end) -pic1\_smoothed(:,end-1));

%compute the derivative with respect to y

pic1\_dery=conv2(pic1\_smoothed,Filter\_dery,'same');

%compute the derivative with respect to y at the border pixels

pic1\_dery(1,:)=2\*(pic1\_smoothed(2,:) -pic1\_smoothed(1,:));

pic1\_dery(end,:)=2\*(pic1\_smoothed(end,:) -pic1\_smoothed(end-1,:));

%compute image 2 derivative by convolution in both x and y directions

%compute the derivative with respect to x

pic2\_derx=conv2(pic2\_smoothed,Filter\_derx,'same');

%compute the derivative with respect to x at the border pixels

pic2\_derx(:,1)=2\*(pic2\_smoothed(:,2) -pic2\_smoothed(:,1));

pic2\_derx(:,end)=2\*(pic2\_smoothed(:,end) -pic2\_smoothed(:,end-1));

%compute the derivative with respect to y

pic2\_dery=conv2(pic2\_smoothed,Filter\_dery,'same');

%compute the derivative with respect to y at the border pixels

pic2\_dery(1,:)=2\*(pic2\_smoothed(2,:) -pic2\_smoothed(1,:));

pic2\_dery(end,:)=2\*(pic2\_smoothed(end,:) -pic2\_smoothed(end-1,:));

%compute quadratic terms of the derivatives

pic1\_der\_x2=pic1\_derx.^2;

pic1\_der\_y2=pic1\_dery.^2;

pic1\_der\_xy=pic1\_derx.\*pic1\_dery;

pic2\_der\_x2=pic2\_derx.^2;

pic2\_der\_y2=pic2\_dery.^2;

pic2\_der\_xy=pic2\_derx.\*pic2\_dery;

%apply Gaussian filter to suppress noise and perform the smoothing in a separable way

int\_sig=2;

x=-6:6;

Filter\_int1D=exp(-(x).^2/(2\*int\_sig^2))/(int\_sig\*sqrt(2\*pi));

pic1\_der\_x2=conv2(Filter\_int1D,Filter\_int1D',pic1\_der\_x2,'same');

pic1\_der\_y2=conv2(Filter\_int1D,Filter\_int1D',pic1\_der\_y2,'same');

pic1\_der\_xy=conv2(Filter\_int1D,Filter\_int1D',pic1\_der\_xy,'same');

pic2\_der\_x2=conv2(Filter\_int1D,Filter\_int1D',pic2\_der\_x2,'same');

pic2\_der\_y2=conv2(Filter\_int1D,Filter\_int1D',pic2\_der\_y2,'same');

pic2\_der\_xy=conv2(Filter\_int1D,Filter\_int1D',pic2\_der\_xy,'same');

kappa=0.04;

%compute Harris R function over the images

r1=(pic1\_der\_x2.\*pic1\_der\_y2-pic1\_der\_xy.^2)-kappa\*(pic1\_der\_x2+pic1\_der\_y2).^2;

r2=(pic2\_der\_x2.\*pic2\_der\_y2-pic2\_der\_xy.^2)-kappa\*(pic2\_der\_x2+pic2\_der\_y2).^2;

%calculate the maximum of Harris R function

r1\_max=max(r1(:));

r2\_max=max(r2(:));

%set the threshold

threshold1=0.001\*r1\_max;

threshold2=0.001\*r2\_max;

height1=size(img1,1); %calculate the height of the image1

width1=size(img1,2); %calculate the width of the image1

height2=size(img2,1); %calculate the height of the image2

width2=size(img2,2); %calculate the width of the image2

result1 = zeros(height1, width1); %record the position of the corner of image 1

result2 = zeros(height2, width2); %record the position of the corner of image 2

%do non-maximum suppression of image 1

count1 = 0; %record the quantity of corners in image 1

for i = 2 : height1-1

for j = 2 : width1-1

% the size of window is 3\*3

if (r1(i, j) > threshold1 && r1(i, j) > r1(i-1, j-1) && r1(i, j) > r1(i-1, j) && r1(i, j) > r1(i-1, j+1) && r1(i, j) > r1(i, j-1) && r1(i, j) > r1(i, j+1) && r1(i, j) > r1(i+1, j-1) && r1(i, j) > r1(i+1, j) && r1(i, j) > r1(i+1, j+1))

result1(i, j) = 1; % a sparse matrix

count1 = count1 + 1;

end;

end;

end;

%do non-maximum suppression of image 2

count2 = 0; %record the quantity of corners in image 2

for i = 2 : height2-1

for j = 2 : width2-1

% the size of window is 3\*3

if (r2(i, j) > threshold2 && r2(i, j) > r2(i-1, j-1) && r2(i, j) > r2(i-1, j) && r2(i, j) > r2(i-1, j+1) && r2(i, j) > r2(i, j-1) && ...

r2(i, j) > r2(i, j+1) && r2(i, j) > r2(i+1, j-1) && r2(i, j) > r2(i+1, j) && r2(i, j) > r2(i+1, j+1));

result2(i, j) = 1; % a sparse matrix

count2 = count2 + 1;

end;

end;

end;

i = 1;

for j = 1 : height1

for k = 1 : width1

if result1(j, k) == 1;

% cor1 is a Nx2 matrix, stores the position of corners in image1

cor1(i, 1) = j;

cor1(i, 2) = k;

i = i + 1;

end;

end;

end;

i = 1;

for j = 1 : height2

for k = 1 : width2

if result2(j, k) == 1;

% cor2 is a Nx2 matrix, stores the position of corners in image2

cor2(i, 1) = j;

cor2(i, 2) = k;

i = i + 1;

end;

end;

end;

%%display the Harris corners in image 1

[poscol1, posrow1] = find(result1 == 1);

figure,imshow(img1);

hold on;

plot(posrow1, poscol1, '\*');

%%display the Harris corners in image 2

[poscol2, posrow2] = find(result2 == 1);

figure,imshow(img2);

hold on;

plot(posrow2, poscol2, '\*');



1. Harris corners in img1



1. Harris corners in img2
2. Given two set of corners from the two images, compute normalized cross correlation (NCC) of image patches centered at each corner. Choose potential corner matches by finding pair of corners (one from each image) such that they have the highest NCC value. Then set a threshold to keep only matches that have a large NCC score to find correspondences between the two images.

f=cell(1,count1);

g=cell(1,count2);

pic1\_db=double(pic1);

pic2\_db=double(pic2);

for i=1:count1

patch1=pic1\_db(cor1(i,1)-1:cor1(i,1)+1,cor1(i,2)-1:cor1(i,2)+1); %compute the gray level of each corner and its 3x3 neighbours in image 1

f{i}=patch1./(sqrt(sum(sum(patch1.^2))));

end

for i=1:count2

patch2=pic2\_db(cor2(i,1)-1:cor2(i,1)+1,cor2(i,2)-1:cor2(i,2)+1); %compute the gray level of each corner and its 3x3 neighbours in image 2

g{i}=patch2./(sqrt(sum(sum(patch2.^2))));

end

ncc1=zeros(count2,count1);

for i=1:count1

for j=1:count2

ncc1(j,i)=sum(sum(f{i}.\*g{j})); % compute the NCC values

end

end

%index1 records the exact corner in image2 which corresponds to image1

%max1\_num records the highest NCC values of every corner in image1 corresponding to the corners in image2

[max1\_num,index1]=max(ncc1);

tres=0.95; %set treshold be 0.95

% only the hightest NCC values bigger than treshold will be recorded

[max1pos\_rol,max1pos\_col]=find(max1\_num>tres);

match1=zeros(length(max1pos\_col),2); %records the corners of two images which have the correspondence

for i=1:length(max1pos\_col)

match1(i,1)=max1pos\_col(i);%records corners in image1 which correspond to corners in image2

match1(i,2)=index1(max1pos\_col(i));%records corners in image2 which correspond to corners in image1

end

ncc2=zeros(count1,count2);

for i=1:count2

for j=1:count1

ncc2(j,i)=sum(sum(f{j}.\*g{i})); % compute the NCC values

end

end

%index2 stores the exact corner in image1 which corresponds to image2

%max2\_num stores the highest NCC values of every corner in image2 corresponding to the corners in image1

[max2\_num,index2]=max(ncc2);

% only the hightest NCC values bigger than treshold will be recorded

[max2pos\_rol,max2pos\_col]=find(max2\_num>tres);

match2=zeros(length(max2pos\_col),2); %records the corners of two images which have the correspondence

for i=1:length(max2pos\_col)

match2(i,2)=max2pos\_col(i);%records corners in image2 which correspond to corners in image1

match2(i,1)=index2(max2pos\_col(i));%records corners in image1 which correspond to corners in image2

end

matchs=[]; %record the final feature points match relationship using two-way match

loc1=[]; %record the location of the final matching points in image1

loc2=[]; %record the location of the final matching points in image2

for i=1:length(match1)

for j=1:length(match2)

if match1(i,:)==match2(j,:)

matchs=[matchs;match1(i,:)];

loc1=[loc1;cor1(match1(i,1),:)];

loc2=[loc2;cor2(match1(i,2),:)];

end

end

end

%Draw lines between matching corners

X1=loc1(:,2);

Y1=loc1(:,1);

X2=loc2(:,2);

Y2=loc2(:,1);

dif=size(img1,2);

figure();

imshowpair(img1,img2,'montage');

hold on

for k=1:length(X1)

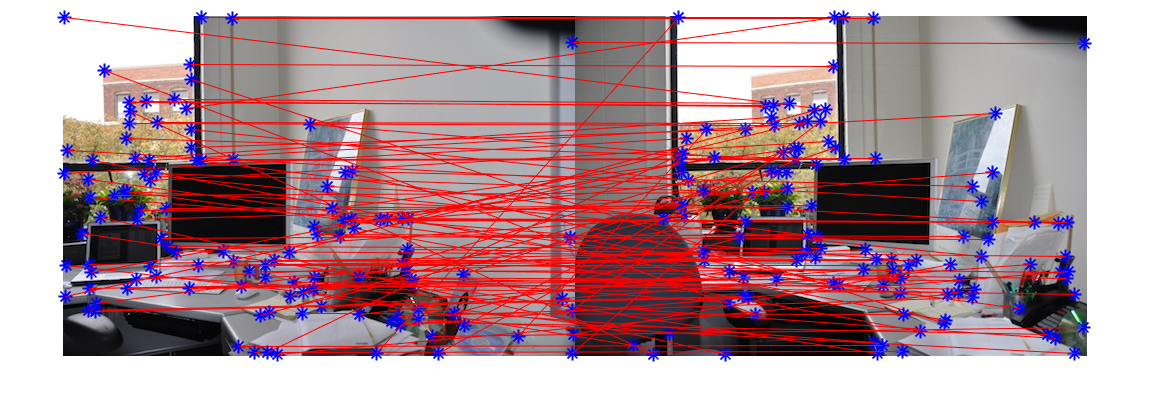
plot(X1(k),Y1(k),'b\*');

plot(X2(k)+dif,Y2(k),'b\*');

line([X1(k),X2(k)+dif],[Y1(k),Y2(k)],'Color','r');

end

set(gcf,'Color','w');



1. the correspondences between the two images after NCC
2. Use RANSAC to robustly estimate the homography from the noisy correspondences. Repeatedly sample minimal number of points needed to estimate a homography (4 pts in this case). Compute a homography from these four points. Map all points using the homagraphy and comparing distances between predicted and observed locations to determine the number of inliers. At the end, compute a least-squares homgraphy from all the inliers in the largest set of inliers.

inlier\_nummax=4;

inlier\_loc=[];

for k=1:20000

a=ceil(length(loc1)\*rand(1,4)); %randomly select corresponding four points

while a(1)==a(2)||a(1)==a(3)||a(1)==a(4)||a(2)==a(3)||a(2)==a(4)||a(3)==a(4)

a=ceil(length(loc1)\*rand(1,4));

end

x11=loc1(a(1),1);

y11=loc1(a(1),2);

x12=loc2(a(1),1);

y12=loc2(a(1),2);

x21=loc1(a(2),1);

y21=loc1(a(2),2);

x22=loc2(a(2),1);

y22=loc2(a(2),2);

x31=loc1(a(3),1);

y31=loc1(a(3),2);

x32=loc2(a(3),1);

y32=loc2(a(3),2);

x41=loc1(a(4),1);

y41=loc1(a(4),2);

x42=loc2(a(4),1);

y42=loc2(a(4),2);

A=[x11,y11,1,0,0,0,-x11\*x12,-y11\*x12;

0,0,0,x11,y11,1,-x11\*y12,-y11\*y12;

x21,y21,1,0,0,0,-x21\*x22,-y21\*x22;

0,0,0,x21,y21,1,-x21\*y22,-y21\*y22;

x31,y31,1,0,0,0,-x31\*x32,-y31\*x32;

0,0,0,x31,y31,1,-x31\*y32,-y31\*y32;

x41,y41,1,0,0,0,-x41\*x42,-y41\*x42;

0,0,0,x41,y41,1,-x41\*y42,-y41\*y42];

b=[x12,y12,x22,y22,x32,y32,x42,y42]';

h=A\b; %Compute a homography from these corresponding four points

inlier\_num=0;

inlier\_loc=0;

for i=1:length(loc1)

%Map all points using the homagraphy

loc2\_predicted(i,1)=(h(1)\*loc1(i,1)+h(2)\*loc1(i,2)+h(3))/(h(7)\*loc1(i,1)+h(8)\*loc1(i,2)+1);

loc2\_predicted(i,2)=(h(4)\*loc1(i,1)+h(5)\*loc1(i,2)+h(6))/(h(7)\*loc1(i,1)+h(8)\*loc1(i,2)+1);

%Compare distances between predicted and observed locations

distance=sqrt((loc2\_predicted(i,1)-loc2(i,1))^2+(loc2\_predicted(i,2)-loc2(i,2))^2);

if distance<1 %the inliers will be recorded only when the distance less than 1

inlier\_num=inlier\_num+1;

inlier\_loc=[inlier\_loc;i];

end

end

if inlier\_num>inlier\_nummax

inlier\_nummax=inlier\_num;

hom=h; %compute a least-squares homgraphy from all the inliers in the largest set of inliers

inlier\_maxloc=inlier\_loc; %record the exact corners that are inliers

end

end

%record the position of inliers in image1 and image2

for i=1:length(inlier\_maxloc)-1

inlier1\_loc(i,1)=loc1(inlier\_maxloc(i+1),1);

inlier1\_loc(i,2)=loc1(inlier\_maxloc(i+1),2);

inlier2\_loc(i,1)=loc2(inlier\_maxloc(i+1),1);

inlier2\_loc(i,2)=loc2(inlier\_maxloc(i+1),2);

end

%Draw lines between matching inliers corners

X1=inlier1\_loc(:,2);

Y1=inlier1\_loc(:,1);

X2=inlier2\_loc(:,2);

Y2=inlier2\_loc(:,1);

dif=size(img1,2);

figure();

imshowpair(img1,img2,'montage');

hold on

for k=1:length(X1)

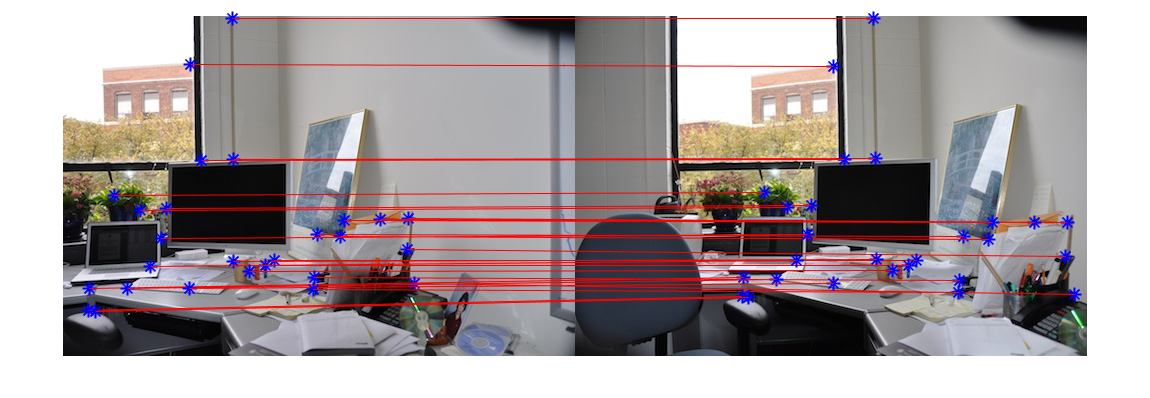
plot(X1(k),Y1(k),'b\*');

plot(X2(k)+dif,Y2(k),'b\*');

line([X1(k),X2(k)+dif],[Y1(k),Y2(k)],'Color','r');

end

set(gcf,'Color','w');



1. a least-squares homgraphy after using RANSAC
2. Determine how big to make the final output image so that it contains the union of all pixels in the two images. Copy the image that does not have to be warped into the appropriate location in the output. Warp the other image into the output image based on the estimated homography (or its inverse). Use blending schemes to blend pixels in the area of overlap between both images.
3. **Observations**
4. **Conclusion**

Using Harris corner detector can detect a large number of corners. After using NCC to find correspondence between the two images, the correspondences are likely to have many errors. Using RANSAC can robustly estimate the homography from the noisy correspondences. Warping the image into the output image based on the inverse of the estimated homography can have no gaps. It is a good way to produce an image mosaic.

1. **Appendix**