

function main

clear all;

close all;

clc;

% load data

set\_number=1;

[x1, x2] = readTextFiles(strcat('set',num2str(set\_number))); % default setting is to open set1 data

image1 = imread(strcat('set',num2str(set\_number),'/image1.jpg'));

image2 = imread(strcat('set',num2str(set\_number),'/image2.jpg'));

% Linear least squares version

F\_lin = cal\_F(x1,x2);

% Normalized version

T1=cal\_T(x1);

T2=cal\_T(x2);

xt1=T1\*x1;

xt2=T2\*x2;

F\_temp=cal\_F(xt1,xt2);

F\_normal=transpose(T1)\*F\_temp\*(T2);

% Epipolar line and error

[L\_linear\_1,L\_linear\_2,error\_lin\_1,error\_lin\_2] = epi\_line\_error(x1,x2,F\_lin);

[L\_norm\_1,L\_norm\_2,error\_norm\_1,error\_norm\_2] = epi\_line\_error(x1,x2,F\_normal);

%visualization

figure;

hold on;

draw\_pic\_linear(x1,x2,L\_linear\_1,L\_linear\_2,error\_lin\_1,error\_lin\_2,image1,image2,set\_number)

figure;

hold on;

draw\_pic\_normal(x1,x2,L\_norm\_1,L\_norm\_2,error\_norm\_1,error\_norm\_2,image1,image2,set\_number)

end

function draw\_pic\_normal(x1,x2,L1,L2,error1,error2,image1,image2,set\_number)

[~, n]=size(x1);

line\_len=15;

h\_title=suptitle({['Fundamental Matrix'],

['Result for Set',num2str(set\_number)]});

subplot(1,2,1)

hold on;

h\_title=title({['Image1 normalized version'];

['Average distance=',num2str(error1)]});

imshow(image1);

plot(x1(1,:),x1(2,:),'ro');

for i = 1:n

if L1(2,i)==0

p1 = [-L1(3,i)/L1(1,i),x1(2,i)-line\_len];

p2 = [-L1(3,i)/L1(1,i),x1(2,i)+line\_len];

else

p1 = [x1(1,i)-line\_len,x1(1,i)+line\_len];

p2 = [-(L1(1,i)\*p1(1,1)+L1(3,i))/L1(2,i), -(L1(1,i)\*p1(1,2)+L1(3,i))/L1(2,i)];

end

plot(p1,p2,'b');

end

% Plot image2

subplot(1,2,2)

hold on;

h\_title=title({['Image2 normalized version'];

['Average distance=',num2str(error2)]});

imshow(image2);

plot(x2(1,:),x2(2,:),'ro');

for i = 1:n

if L2(2,i)==0

p1 = [-L2(3,i)/L2(1,i),x2(2,i)-line\_len];

p2 = [-L2(3,i)/L2(1,i),x2(2,i)+line\_len];

else

p1 = [x2(1,i)-line\_len,x2(1,i)+line\_len];

p2 = [-(L2(1,i)\*p1(1,1)+L2(3,i))/L2(2,i), -(L2(1,i)\*p1(1,2)+L2(3,i))/L2(2,i)];

end

plot(p1,p2,'b');

end

print(gcf,'-djpeg' ,strcat('HW3\_2\_1\_normalized\_set',num2str(set\_number),'.jpeg'),'-r400')

end

function draw\_pic\_linear(x1,x2,L1,L2,error1,error2,image1,image2,set\_number)

[~, n]=size(x1);

line\_len=15;

% Plot image1

h\_title=suptitle({['Fundamental Matrix'],

['Result for Set',num2str(set\_number)]});

subplot(1,2,1)

hold on;

h\_title=title({['Image1 linear least square version'];

['Average distance=',num2str(error1)]});

imshow(image1);

plot(x1(1,:),x1(2,:),'ro');

for i = 1:n

if L1(2,i)==0

p1 = [-L1(3,i)/L1(1,i),x1(2,i)-line\_len];

p2 = [-L1(3,i)/L1(1,i),x1(2,i)+line\_len];

else

p1 = [x1(1,i)-line\_len,x1(1,i)+line\_len];

p2 = [-(L1(1,i)\*p1(1,1)+L1(3,i))/L1(2,i), -(L1(1,i)\*p1(1,2)+L1(3,i))/L1(2,i)];

end

plot(p1,p2,'b');

end

% Plot image2

subplot(1,2,2)

hold on;

h\_title=title({['Image2 linear least square version'];

['Average distance=',num2str(error2)]});

imshow(image2);

plot(x2(1,:),x2(2,:),'ro');

for i = 1:n

if L2(2,i)==0

p1 = [-L2(3,i)/L2(1,i),x2(2,i)-line\_len];

p2 = [-L2(3,i)/L2(1,i),x2(2,i)+line\_len];

else

p1 = [x2(1,i)-line\_len,x2(1,i)+line\_len];

p2 = [-(L2(1,i)\*p1(1,1)+L2(3,i))/L2(2,i), -(L2(1,i)\*p1(1,2)+L2(3,i))/L2(2,i)];

end

plot(p1,p2,'b');

end

print(gcf,'-djpeg' ,strcat('HW3\_2\_1\_LinearLS\_set',num2str(set\_number),'.jpeg'),'-r400')

end

function [L1,L2,error\_1,error\_2]=epi\_line\_error(x1,x2,F)

[~, n]=size(x1);

L1 = F\*x2;

L2 = transpose(F)\*x1;

% distance=|ax+by+c|/sqrt(a^2+b^2)

err1=sum(L1.\*x1); % calculate ax+by+c

den1=sqrt((L1(1,:).^2)+L1(2,:).^2); % calculate denominator

dist1=err1./den1; % calculate each distance

err2=sum(L2.\*x2);

den2=sqrt((L2(1,:).^2)+L2(2,:).^2);

dist2=err2./den2;

error\_1=sum(abs(dist1))/n;

error\_2=sum(abs(dist2))/n;

end

%% Calculate Transformation Matrix

function T=cal\_T(x)

[~, n]=size(x);

x\_bar=sum(x(1,:))/n;

y\_bar=sum(x(2,:))/n;

i=1;

num=sqrt((x(1,i)-x\_bar)^2+(x(2,i)-y\_bar)^2);

den=n\*sqrt(2);

d=num/den;

if n>=2

for i=2:n

num=sqrt((x(1,i)-x\_bar)^2+(x(2,i)-y\_bar)^2);

den=n\*sqrt(2);

d=d+num/den;

end

else

end

T=[1/d,0,-x\_bar/d;

0,1/d,-y\_bar/d;

0,0,1];

end

%% Calculate Fundamental Matrix

function F=cal\_F(x1,x2)

[~, n1]=size(x1);

[~, n2]=size(x2);

if n1~=n2

error=char('x1 and x2 does not match!')

return

else

n=n1;

end

%Build the matrix A

for i = 1:n

xx1 = x1(:,i);

xx2 = x2(:,i);

xx=xx2\*transpose(xx1);

for j=1:9

A(i,j)=xx(j);

end

end

%SVD

[u,s,v] = svd(A,0);

vv=v(:,9);

for i=1:3

F(1,i)=vv(i);

end

for i=1:3

F(2,i)=vv(i+3);

end

for i=1:3

F(3,i)=vv(i+6);

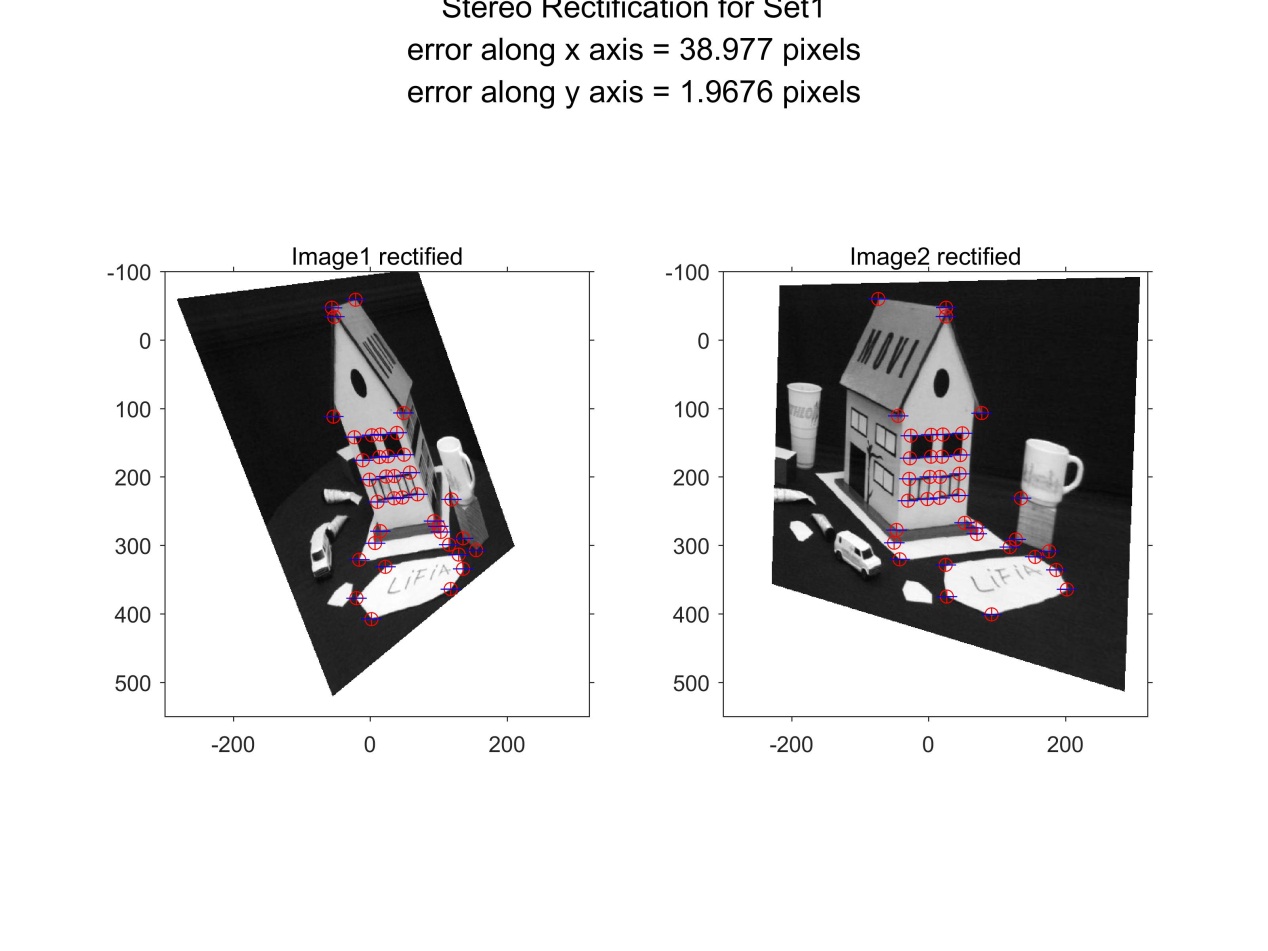
end

% let rank(F)=2

[u,s,v] = svd(F);

F = F - u(:,3)\*s(3,3)\*transpose(v(:,3));

end



For dataset 1,

0.6023 0.3521 -227.3008

-0.1462 0.9681 -48.3259

0.0007 0.0001 0.8016

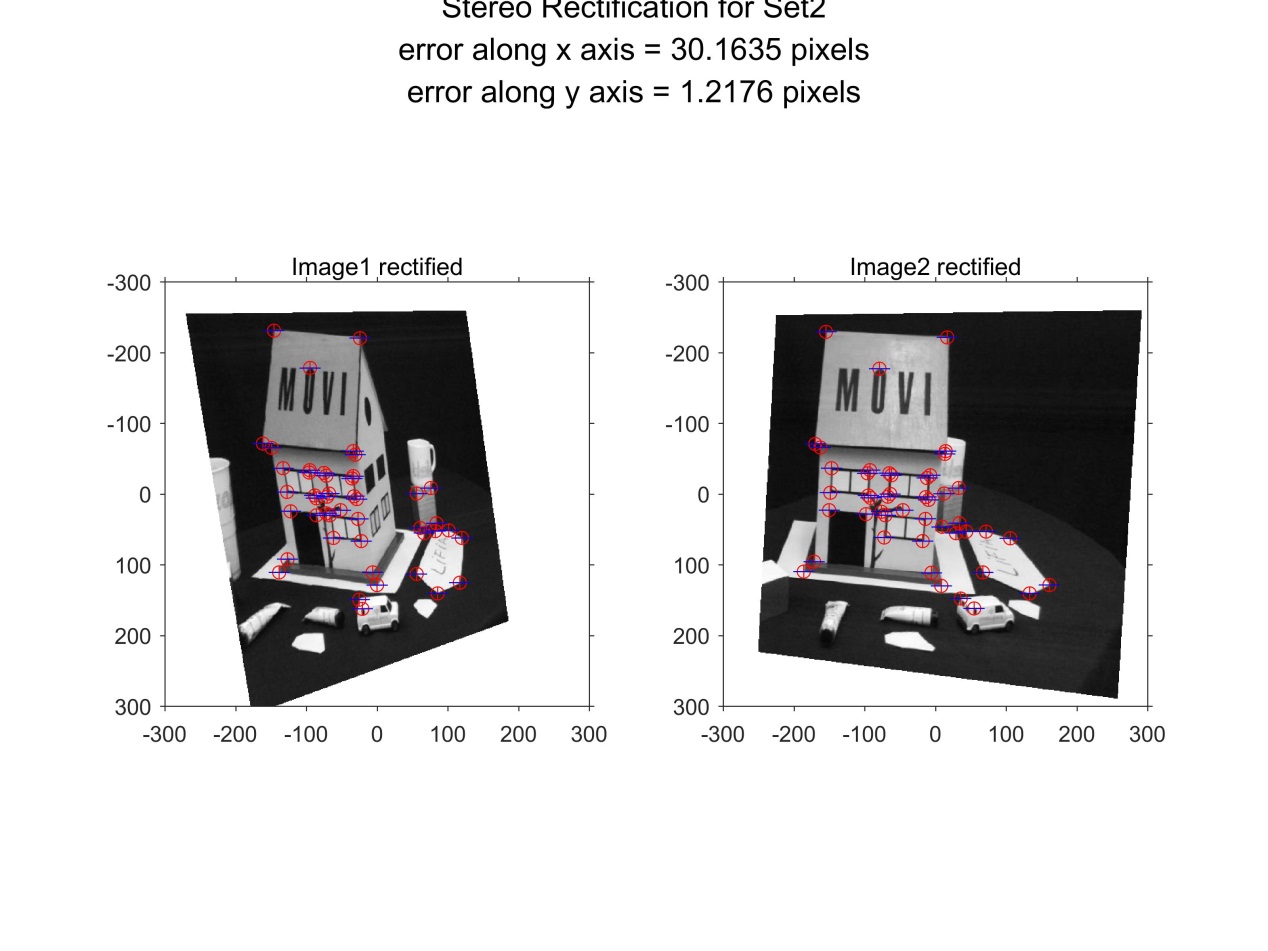
H1=

0.9995 -0.0312 -253.2156

0.0312 0.9995 -93.2719

-0.0006 0.0000 1.1626

H2=



For dataset 2,

0.7250 0.1438 -231.0188

-0.1391 0.9678 -217.5532

0.0005 0.0001 0.8521

H1=

0.9983 -0.0578 -240.7667

0.0578 0.9983 -270.3764

-0.0003 0.0000 1.0678

H2=

function main

clear all;

close all;

clc;

% load data

set\_number=1;

[x1, x2] = readTextFiles(strcat('set',num2str(set\_number))); % default setting is to open set1 data

image1 = imread(strcat('set',num2str(set\_number),'/image1.jpg'));

image2 = imread(strcat('set',num2str(set\_number),'/image2.jpg'));

% Calculate fundamental matrix by Normalized version

T1=cal\_T(x1);

T2=cal\_T(x2);

xt1=T1\*x1;

xt2=T2\*x2;

F\_temp=cal\_F(xt2,xt1);

F=transpose(T1)\*F\_temp\*(T2);

% Find epipole for each picture

e1=null(F);

e2=null(transpose(F));

H1=cal\_H2(e1,image1);

H2=cal\_H1(e2,image2);

[~, n]=size(x1);

A=zeros(2\*n,5);

xx1=H1\*x1;

xx2=H2\*x2;

% tarnsform to homogeneuos coordinates

for i=1:n

xx1(:,i)=xx1(:,i)/xx1(3,i);

xx2(:,i)=xx2(:,i)/xx2(3,i);

end

A(1:n,1)=transpose(xx1(1,:));

A(1:n,2)=transpose(xx1(2,:));

A(1:n,3)=ones(n,1);

A(1+n:2\*n,4)=transpose(xx1(2,:));

A(1+n:2\*n,5)=ones(n,1);

b=zeros(2\*n,1);

b(1:n)=transpose(xx2(1,:));

b(1+n:2\*n)=transpose(xx2(2,:));

sd=A\b;

s1=sd(1);

s3=sd(2);

d1=sd(3);

s2=sd(4);

d2=sd(5);

H0=eye(3);

H0(1,1)=s1;

H0(2,2)=s2;

H0(1,2)=s3;

H0(1,3)=d1;

H0(2,3)=d2;

H1=H0\*H1;

% calculate transformed errors.

new\_x1=H1\*x1;

new\_x2=H2\*x2;

% tarnsform new\_x to homogeneuos coordinates

for i=1:n

new\_x1(:,i)=new\_x1(:,i)/new\_x1(3,i);

new\_x2(:,i)=new\_x2(:,i)/new\_x2(3,i);

end

% then calculate errors

error=new\_x1-new\_x2;

error\_x=sqrt(sum(error(1,:).\*error(1,:))/n);

error\_y=sqrt(sum(error(2,:).\*error(2,:))/n);

% calculate epiline

new\_F\_temp=cal\_F(new\_x1,new\_x2);

new\_F=transpose(T1)\*new\_F\_temp\*(T2);

L1=new\_F\*new\_x2;

L2=new\_F\*new\_x1;

% draw original images

figure;

h\_title=suptitle({['Original Images for Set',num2str(set\_number)]});

subplot(1,2,1);hold on;

h\_title=title({['Image1 original']});

imshow(image1);

subplot(1,2,2);hold on;

h\_title=title({['Image2 original']});

imshow(image2);

% draw transform images

RA = imref2d([512, 512], [0, 512], [0, 512]);

[IMG1, RB1] = imwarp(image1, RA, projective2d(H1'), 'fillvalues', 255);

[IMG2, RB2] = imwarp(image2, RA, projective2d(H2'), 'fillvalues', 255);

figure

clf()

ax1 = subplot(1,2,1);

imshow(IMG1, RB1); hold on

plot(new\_x1(1,:), new\_x1(2,:), 'r+')

ax2 = subplot(1,2,2);

imshow(IMG2, RB2); hold on

plot(new\_x2(1,:), new\_x2(2,:), 'r+')

linkaxes([ax1, ax2], 'xy')

axis equal

axis([-300, 320, -100, 550])%ues for dataset1

% axis([-300, 300, -300, 300])%ues for dataset2

draw\_rect\_point(new\_x1,new\_x2,error\_x,error\_y,set\_number)

end

function H=cal\_H1(epipole,image)

epipole=epipole/epipole(3);

T=eye(3);

[width, length]=size(image);

T(1,3)=-width/2;

T(2,3)=-length/6;

e\_bar=T\*epipole;

phi=atan2(e\_bar(2),e\_bar(1));

R=[cos(phi),sin(phi),0;

-sin(phi),cos(phi),0;

0,0,1];

e\_hat=R\*e\_bar;

G=eye(3);

G(3,1)=-1/e\_hat(1);

H=G\*R\*T;

end

function H=cal\_H2(epipole,image)

epipole=epipole/epipole(3);

T=eye(3);

[width, length]=size(image);

T(1,3)=-width/2;

T(2,3)=-length/6;

e\_bar=T\*epipole;

phi=atan2(e\_bar(2),e\_bar(1));

phi=phi+pi();

R=[cos(phi),sin(phi),0;

-sin(phi),cos(phi),0;

0,0,1];

e\_hat=R\*e\_bar;

G=eye(3);

G(3,1)=-1/e\_hat(1);

H=G\*R\*T;

end

function draw\_rect\_point(x1,x2,error1,error2,set\_number)

[~, n]=size(x1);

line\_len=15;

subplot(1,2,1)

hold on;

h\_title=title({['Image1 rectified']});

plot(x1(1,:),x1(2,:),'ro');

for i = 1:n

p1=[x1(1,i)-line\_len,x1(1,i)+line\_len];

p2=[x1(2,i),x1(2,i)];

plot(p1,p2,'b');

end

% Plot image2

subplot(1,2,2)

hold on;

h\_title=title({['Image2 rectified']});

plot(x2(1,:),x2(2,:),'ro');

for i = 1:n

p1=[x2(1,i)-line\_len,x2(1,i)+line\_len];

p2=[x2(2,i),x2(2,i)];

plot(p1,p2,'b');

end

h\_title=suptitle({['Stereo Rectification for Set',num2str(set\_number)];

['error along x axis = ',num2str(error1),' pixels'];

['error along y axis = ',num2str(error2),' pixels']});

print(gcf,'-djpeg' ,strcat('HW3\_2\_2\_rectification\_set',num2str(set\_number),'.jpeg'),'-r400')

end

%% Calculate Transformation Matrix

function T=cal\_T(x)

[~, n]=size(x);

x\_bar=sum(x(1,:))/n;

y\_bar=sum(x(2,:))/n;

i=1;

num=sqrt((x(1,i)-x\_bar)^2+(x(2,i)-y\_bar)^2);

den=n\*sqrt(2);

d=num/den;

if n>=2

for i=2:n

num=sqrt((x(1,i)-x\_bar)^2+(x(2,i)-y\_bar)^2);

den=n\*sqrt(2);

d=d+num/den;

end

else

end

T=[1/d,0,-x\_bar/d;

0,1/d,-y\_bar/d;

0,0,1];

end

%% Calculate Fundamental Matrix

function F=cal\_F(x1,x2)

[~, n1]=size(x1);

[~, n2]=size(x2);

if n1~=n2

error=char('x1 and x2 does not match!')

return

else

n=n1;

end

%Build the matrix A

for i = 1:n

xx1 = x1(:,i);

xx2 = x2(:,i);

xx=xx2\*transpose(xx1);

for j=1:9

A(i,j)=xx(j);

end

end

%SVD

% [u s v] = svd(A,0);

[u s v] = svd(A);

vv=v(:,9);

for i=1:3

F(1,i)=vv(i);

end

for i=1:3

F(2,i)=vv(i+3);

end

for i=1:3

F(3,i)=vv(i+6);

end

% let rank(F)=2

[u s v] = svd(F);

F = F - u(:,3)\*s(3,3)\*transpose(v(:,3));

end