Fundamental Matrix
Result for Set1

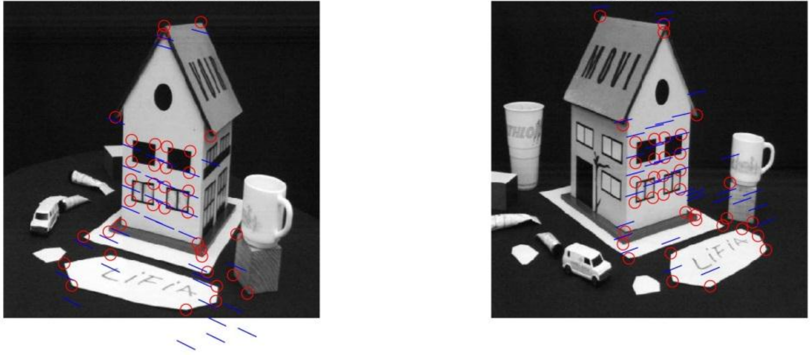


Image1 linear least square version
Average distance=28.0257

Image2 linear least square version
Average distance=25.1629

Fundamental Matrix
Result for Set1

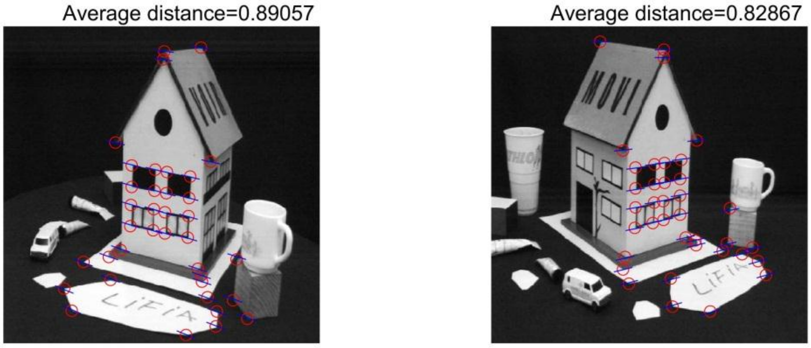


Image1 normalized version
Average distance=0.89057

Image2 normalized version
Average distance=0.82867

Fundamental Matrix
Result for Set2

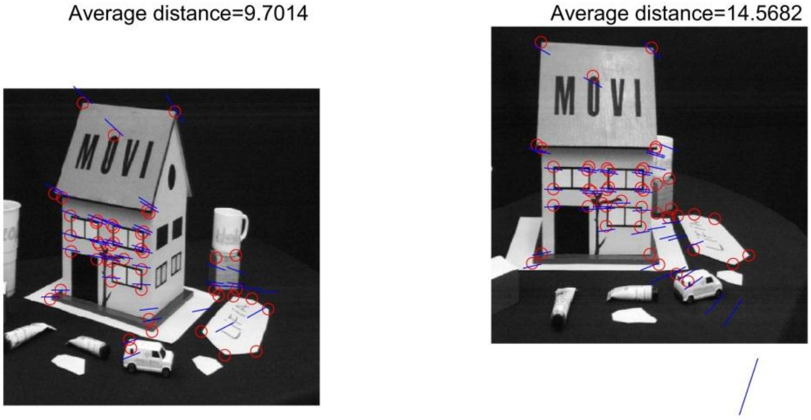


Image1 linear least square version
Average distance=9.7014

Image2 linear least square version
Average distance=14.5682

Fundamental Matrix
Result for Set2

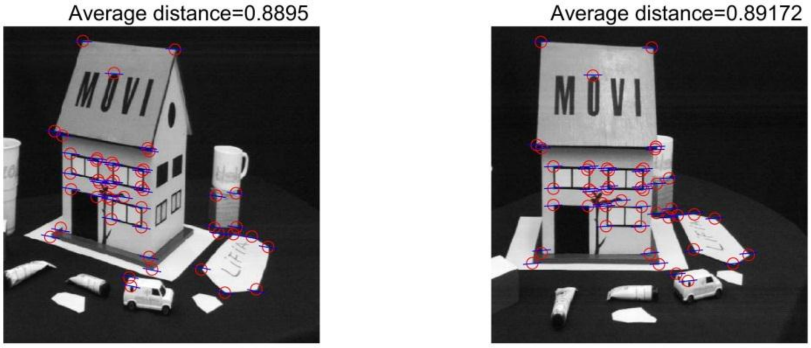
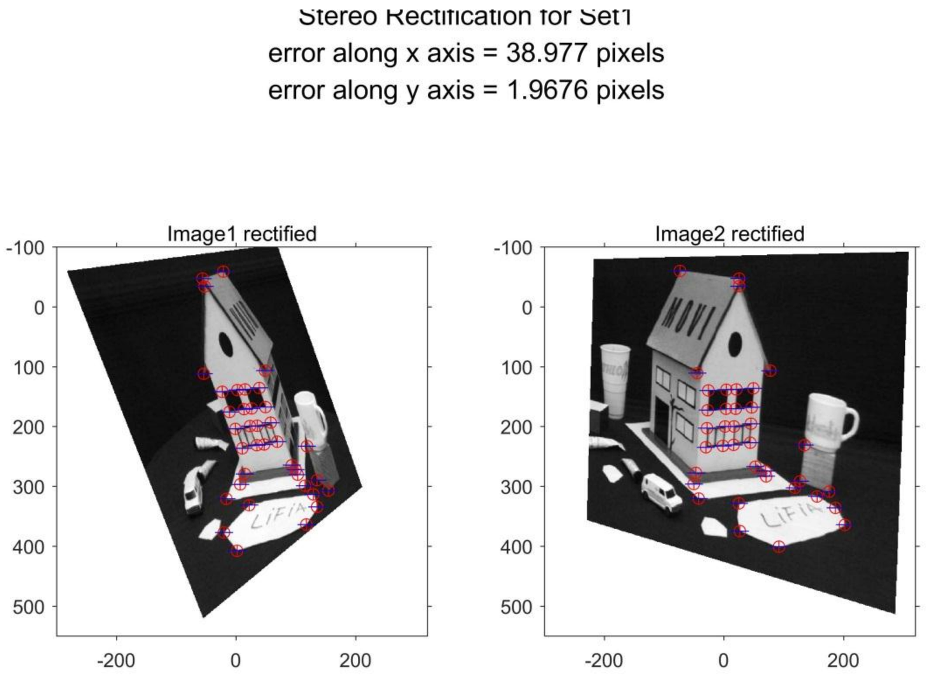


Image1 normalized version
Average distance=0.8895

Image2 normalized version
Average distance=0.89172

```matlab
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
```


Stereo Rectification for Set1
error along x axis = 38.977 pixels
error along y axis = 1.9676 pixels
Image1 rectified
Image2 rectified
-100
0
100
200
300
400
500
-200
0
200
MOVI
LiFIA

For dataset 1,

$$H1=\begin{bmatrix} 0.6023 & 0.3521 & -227.3008 \\ -0.1462 & 0.9681 & -48.3259 \\ 0.0007 & 0.0001 & 0.8016 \end{bmatrix}$$

$$H2=\begin{bmatrix} 0.9995 & -0.0312 & -253.2156 \\ 0.0312 & 0.9995 & -93.2719 \\ -0.0006 & 0.0000 & 1.1626 \end{bmatrix}$$

Stereo Rectification for Set2
error along x axis = 30.1635 pixels
error along y axis = 1.2176 pixels

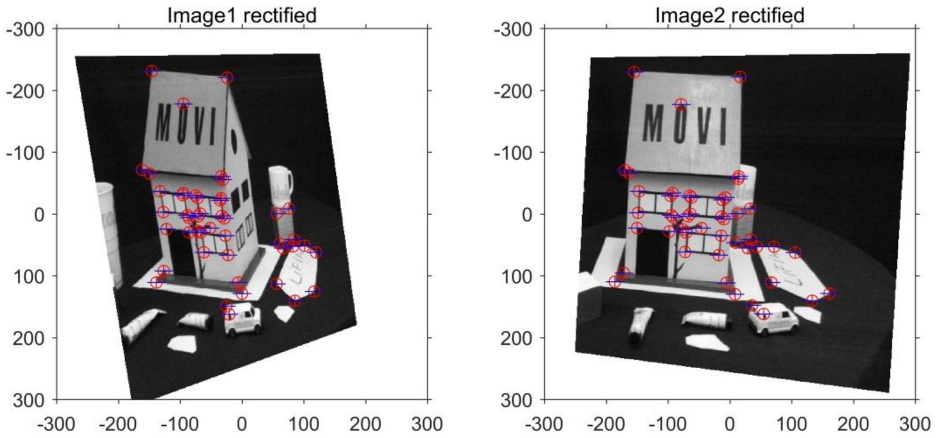


Image1 rectified

Image2 rectified

For dataset 2,

$$H1=\begin{bmatrix} 0.7250 & 0.1438 & -231.0188 \\ -0.1391 & 0.9678 & -217.5532 \\ 0.0005 & 0.0001 & 0.8521 \end{bmatrix}$$

$$H2=\begin{bmatrix} 0.9983 & -0.0578 & -240.7667 \\ 0.0578 & 0.9983 & -270.3764 \\ -0.0003 & 0.0000 & 1.0678 \end{bmatrix}$$

```
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
```