EECS 442
Fan Bu, 68836435
unique name: fanbu

Prob 1. (a)

$\because$ Camera Matrix has rank of 3.

So, $\exists H$, s.t. $MH = [I \ 0]$

$\because M = [A, b]$

Without loss of generality, let $H_1 = \begin{bmatrix} A^{-1} & -A^{-1}b \\ 0 & 1 \end{bmatrix}$

We have $MH_1 = [I \ 0]_{3\times4}$

In this case, $M'H_1 = [A', b'] \begin{bmatrix} A^{-1} & -A^{-1}b \\ 0 & 1 \end{bmatrix}$

$$= [A' \cdot A^{-1} + 0, \ A'(-A^{-1}b) + b']$$

$$\Longrightarrow M'H_1 = [A'A^{-1}, \ -A'A^{-1}b + b']$$

Since we know that $e_3^T(-A'A^{-1}b + b') \neq 0$

$\therefore [M'H_1]_{3,4} \neq 0$

let's explicit $M'H_1$ as $\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix}$

we can construct a matrix $H_2 = \begin{bmatrix} I_{3\times3} & , 0_{3\times1} \\ -\frac{x_{31}}{x_{34}}, \ -\frac{x_{32}}{x_{34}}, \ -\frac{x_{33}}{x_{34}}, & \frac{1}{x_{34}} \end{bmatrix}$

then we have

$$[M'H_1] \cdot H_2 = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix} \begin{bmatrix} I_{3\times3} & , 0_{3\times1} \\ -\frac{x_{31}}{x_{34}}, \ -\frac{x_{32}}{x_{34}}, \ -\frac{x_{33}}{x_{34}}, & \frac{1}{x_{34}} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11} - \frac{x_{14}x_{31}}{x_{34}}, & x_{12} - \frac{x_{14}x_{32}}{x_{34}}, & x_{13} - \frac{x_{14}x_{33}}{x_{34}}, & \frac{x_{14}}{x_{34}} \\ x_{21} - \frac{x_{24}x_{31}}{x_{34}}, & x_{22} - \frac{x_{24}x_{32}}{x_{34}}, & x_{23} - \frac{x_{24}x_{33}}{x_{34}}, & \frac{x_{24}}{x_{34}} \\ 0, & 0, & 0, & 1 \end{bmatrix}$$

In this case, $M'H_1 \cdot H_2$ is in a form a $\hat{M}'$ and $MH_1 \cdot H_2$ is in a form as $\hat{M}$.

Thus $H = H_1 \cdot H_2 = \begin{bmatrix} A^{-1}_{3\times3} & -A^{-1}b_{3\times1} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} & I_{3\times3} & & , O_{3\times1} \\ -\frac{x_{31}}{x_{34}}, & -\frac{x_{32}}{x_{34}}, & -\frac{x_{33}}{x_{34}} & , \frac{1}{x_{34}} \end{bmatrix}$

is the H that we are looking for.

(b) For the fundametal matrix F, we have

$$x'^T \cdot F \cdot x = 0$$

where $F = M'^{-T} E \cdot M^{-1}$

Thus $x'^T M'^{-T} E M^{-1} x = 0$

$\Rightarrow (x'^T H^T)(M'H)^{-T} E (MH)^{-1} (Hx) = 0$

we can see that the fundamental matrix corresponding to the pairs of matrices (M, M') and $(MH, M'H)$ are the same, where (M, M') corresponding to (x, x') and $(MH, M'H)$ corresponding to (Hx, Hx').

(c) According to Hint, the fundemental matrix corresponding to a pair of camera matrices $M = [I \mid 0]$ and $M' = [A \mid b]$ is equal to $[b]_\times A$

Thus, applying resalt from question (a), we have

$$F = [b]_\times A,$$

where $b = \begin{bmatrix} b_1 \\ b_2 \\ 1 \end{bmatrix}$ $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 \end{bmatrix}$.

$$\therefore F = [b]_\times \cdot A = \begin{bmatrix} 0 & -1 & b_2 \\ 1 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -a_{21} & -a_{22} & -a_{23} \\ a_{11} & a_{12} & a_{13} \\ -a_{11}b_2 + a_{21}b_1 & -a_{12}b_2 + a_{22}b_1 & -a_{13}b_2 + a_{23}b_1 \end{bmatrix}$$

Thus, we can multiply any scale factor to make one element as 1.

For example, we may multiply $\frac{1}{a_{11}}$

Then, $F = \begin{bmatrix} -\frac{a_{21}}{a_{11}} & -\frac{a_{22}}{a_{11}} & \frac{-a_{23}}{a_{11}} \\ 1 & \frac{a_{12}}{a_{11}} & \frac{a_1'}{a_{11}} \\ -b_2 + \frac{a_{21}b_1}{a_{11}} & \frac{-a_{12}b_2 + a_{22}b_1}{a_{11}} & \frac{-a_{13}b_2 + a_{23}b_1}{a_{11}} \end{bmatrix}$

which is expressed by seven parameters.

Prob 2. Let k pass through x but not epipole e.

Then x can be expressed as the cross-mutiply of k and l, i.e.

$$x = [k]_\times l \quad \text{------} \ (1)$$

Since we know that fundamental matrix F has the property $F \cdot x = l'$ ------ (2)

put (1) into (2), we have.

$$l' = F \cdot x = F \cdot [k]_\times l$$

i.e. $l' = F \cdot [k]_\times \cdot l$ ■

Prob 3. 3.1 Fundamental Matrix.

① Linear Least Square

For each pair of point, $p' \cdot p^T$ will generate a 3×3 matrix

$$\begin{bmatrix} x_1'x_1 & y_1'x_1 & x_1 \\ x_1'y_1 & y_1'y_1 & y_1 \\ x_1' & y_1' & 1 \end{bmatrix}$$

Same reasoning, for all the N pairs of points, we can write a matrix A, such that

$$A = \begin{bmatrix} x_1'x_1 & x_1'y_1 & x_1' & y_1'x_1 & y_1'y_1 & y_1' & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n'x_n & x_n'y_n & x_n & y_n'x_n & y_n'y_n & y_n' & x_n & y_n & 1 \end{bmatrix}$$

Theoretically. $rank(A) = 8$.

So we do SVD for matrix A.

$[u\ s\ v] = svd(A)$

Then pick the column of V that corresponds to the minimum singular value $V(:,9) = [v_1\ v_2\ \cdots\ v_9]^T$

$$\text{Let } F = \begin{bmatrix} v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 \\ v_7 & v_8 & v_9 \end{bmatrix}$$

Since we know that $rank(F) = 2$, so this F is not the final Fundamental Matrix. Instead, we shall set its rank into 2.

Thus. $[u\ s\ v] = svd(F)$

Then $F \leftarrow F - u(:,3) * S(3,3) * [V(:,3)]^T$

② Normalized Version.

Before actually compute the Fundamental Matrix, in order to reduce the error by the uncentered origin, we shall center our data into a circle by multiplying a matrix T, where

$$T = \begin{bmatrix} 1/d & 0 & -\bar{x}/d \\ 0 & 1/d & -\bar{y}/d \\ 0 & 0 & 1 \end{bmatrix}$$

in which, $\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}$

$$\bar{y} = \sum_{i=1}^{n} \frac{y_i}{n}$$

$$d = \sum_{i=1}^{n} \frac{\sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}}{n\sqrt{2}}$$

By this way, for data x1 and X2, we can produce two matrix T1 and T2.

Then we put $xx1 = T1 \cdot X1$; $xx2 = T2 \cdot X2$ into the Linear least square step, it will result in a Scaled Fundamental Matrix F_s.

To get the final Fundamental Matrix

$$F = T_1^T \cdot F_s \cdot T_2$$

☆ Matlab Code is attached here and uploaded in CANVAS.

☆ Image result is also attached here or can be generated by my code. Errors are shown in images.

Fundamental Matrix
Result for Set1

Image1 linear least square version
Average distance=28.0257

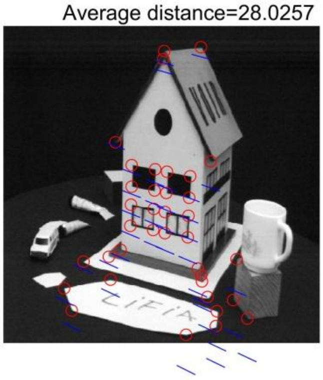


Image2 linear least square version
Average distance=25.1629

Fundamental Matrix
Result for Set1

Image1 normalized version
Average distance=0.89057

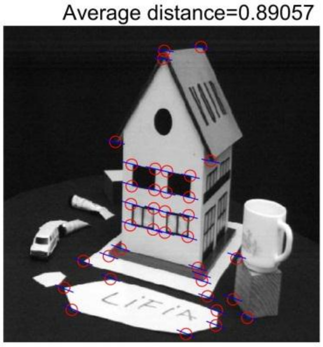
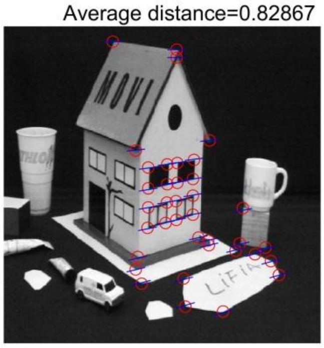


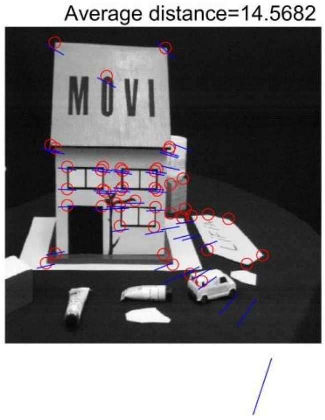
Image2 normalized version
Average distance=0.82867



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

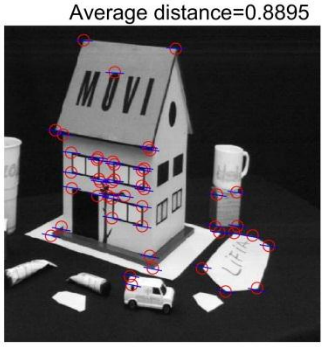
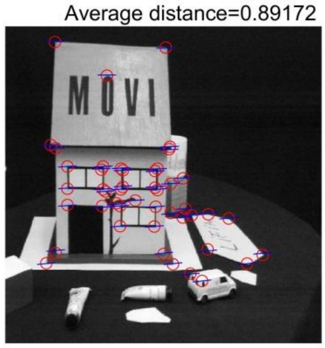


Image2 normalized version
Average distance=0.89172



```
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
```

3.2 Stereo Rectification

After we calculate the Fundamental Matrix F, we know that : $\text{rank}(F) = 2$. $p_2^T \cdot F \cdot p_1 = 0$

For epipole e_1 and e_2 in Image J_1 and J_2, we have:

$F \cdot e_1 = 0$ $F^T \cdot e_2 = 0$

$\Rightarrow e_1 \in \mathcal{N}(F)$ $e_2 \in \mathcal{N}(F^T)$

First, we shall find a matrix H_1 for J_1, H_2 for J_2.

In order to translate the epipole to infinty, and make sure we have less distortion. We shall first translate the origin of the picture to its center

$$\bar{e} = T \cdot e \quad \text{where} \quad T = \begin{bmatrix} 1 & 0 & -\frac{width}{2} \\ 0 & 1 & -\frac{length}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

The set the epipoler line horizontal, let $\phi = \angle \bar{e}$

$$R = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \hat{e} = R\bar{e} = \begin{bmatrix} \hat{e}_1 \\ 0 \\ 1 \end{bmatrix}$$

Then

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/\hat{e}_1 & 0 & 1 \end{bmatrix}$$

By this way, $H = G \cdot R \cdot T$, we can get H_1 and H_2

Now we transformed picture J_1 and J_2 into $\tilde{J}_1$ and $\tilde{J}_2$

$$J_1 \xrightarrow{H_1} \tilde{J}_1$$
$$J_2 \xrightarrow{H_2} \tilde{J}_2$$

However, this is NOT the final result. We should set $\tilde{J}_2$ as a standard and correct $\tilde{J}_1$ to the right position

So that

$$\begin{tikzcd} J_1 \arrow[d, "H_1"'] & J_2 \arrow[d, "H_2"] \\ \tilde{J}_1 \arrow[d, "H_0"'] & \tilde{J}_2 \\ \hat{J}_1 & \end{tikzcd}$$

Then $\hat{J}_1$ and $\tilde{J}_2$ is our final result. that minimizes $d(H_0 \tilde{p}_1, \tilde{p}_2)$.

To calculate H_0, let $H_0 = \begin{bmatrix} s_1 & s_3 & d_1 \\ 0 & s_2 & d_2 \\ 0 & 0 & 1 \end{bmatrix}$

we know that

$$\begin{bmatrix} \tilde{p}_{1,x_1} & \tilde{p}_{1,y_1} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \tilde{p}_{1,x_n} & \tilde{p}_{1,y_n} & 1 & 0 & 0 \\ 0 & 0 & 0 & \tilde{p}_{1,y_1} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \tilde{p}_{1,y_n} & 1 \end{bmatrix}_{2n\times 5} \cdot \begin{bmatrix} s_1 \\ s_3 \\ d_1 \\ s_2 \\ d_2 \end{bmatrix}_{5\times 1} = \begin{bmatrix} \tilde{p}_{2,x_1} \\ \vdots \\ \tilde{p}_{2,x_n} \\ \tilde{p}_{2,y_1} \\ \vdots \\ \tilde{p}_{2,y_n} \end{bmatrix}$$

Denote as $A \cdot x = b$

$\therefore\ x = A \backslash b$

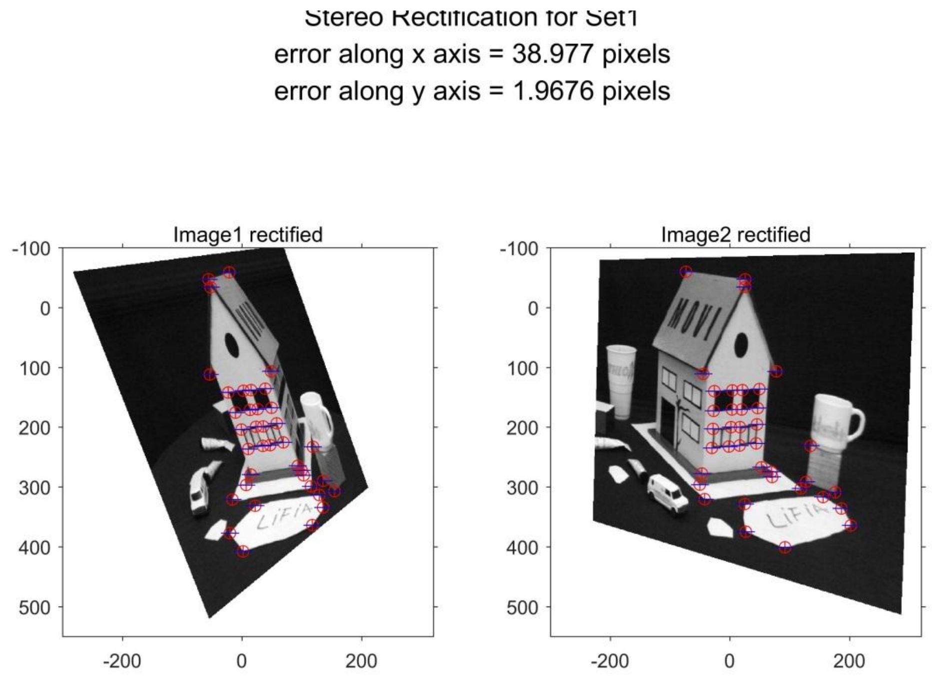
Thus we can calculate H_0,

Then $\begin{cases} H_1 \Leftarrow H_0 \cdot H_1 \\ H_2 \Leftarrow H_2 \end{cases} \Rightarrow \begin{cases} \hat{J}_1 = H_0 H_1 J_1 \\ \tilde{J}_2 = H_2 J_2 \end{cases}$

☆ Matlab code is attached below and uploaded.

☆ H transform Matrix H_1 and H_2 are attach below.

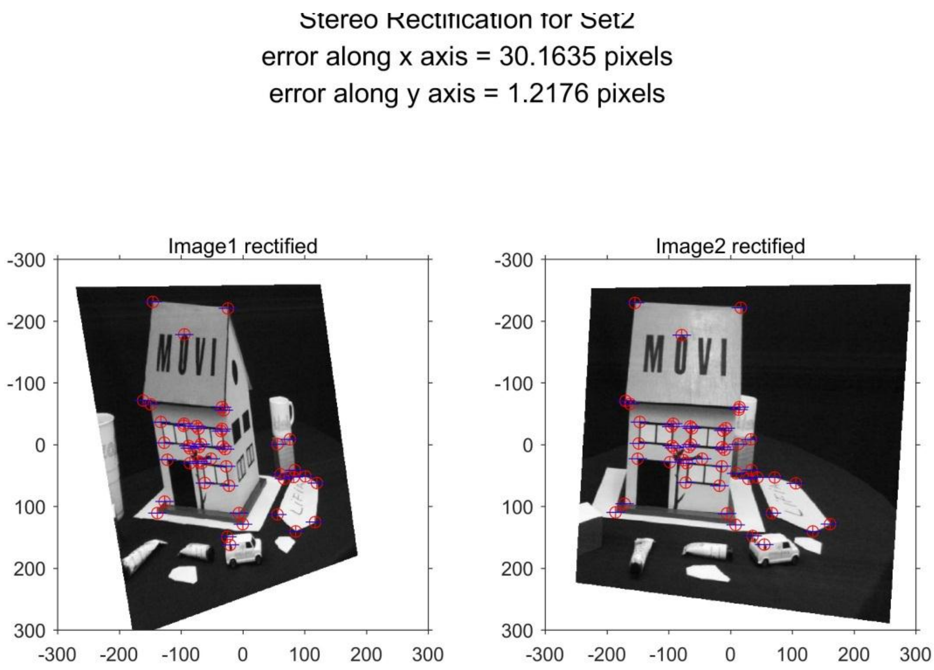
☆ Images and Errors are also attached and shown below.



For dataset 1,

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

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



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
```