EECS 442 F14
HW2
WU Tongshuang 40782356

1.

a) Let

$$H_1, H_2 be\ in\ form \begin{bmatrix} W_{3\times3} & X_{3\times1} \\ Y_{1\times3} & Z_{1\times1} \end{bmatrix}$$

where $H = H_1 H_2$

Since camera matrix has rank 3, we can get a matrix H_1 with form:

$$H_1 = \begin{bmatrix} A^{-1} & -A^{-1}b \\ 0 & 1 \end{bmatrix}$$

Such that

$$\widehat{M} = MH_1 = [I \quad 0]$$

Apply H_1 to M',

$$M'H_1 = [A' \quad b'] \begin{bmatrix} A^{-1} & -A^{-1}b \\ 0 & 1 \end{bmatrix} = [A'A^{-1} \quad -A'A^{-1}b + b']$$

Since $\widehat{M} = MH$ and $\widehat{M'} = M'H$,

$$M'H = M'H_1H_2 = [A'A^{-1} \quad -A'A^{-1}b + b']H_2 = \begin{bmatrix} A'' & b'' \\ 0 & 1 \end{bmatrix}$$

$$MH = M'H_1H_2 = [I \quad 0]H_2 = [I \quad 0]$$

To assure $[I \quad 0]H_2 = [I \quad 0]$, H_2 has to be in form of $\begin{bmatrix} I & 0 \\ Y_{1\times3} & Z_{1\times1} \end{bmatrix}$, meaning:

$$[A'A^{-1} \quad -A'A^{-1}b + b'] \begin{bmatrix} I & 0 \\ Y_{1\times3} & Z_{1\times1} \end{bmatrix} = \begin{bmatrix} A'' & b'' \\ 0 & 1 \end{bmatrix}$$

To assure $\widehat{M'}$ last row to be $[0 \quad 1]$, denote $[A'A^{-1} \quad -A'A^{-1}b + b']$ with n_{ij}:

$$[n_{31} \quad n_{32} \quad n_{33} \quad n_{34}] \begin{bmatrix} I & 0 \\ Y_{1\times3} & Z_{1\times1} \end{bmatrix} = [0 \quad 1]$$

$\because \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} [A'A^{-1} \quad -A'A^{-1}b + b'] \neq 0$, meaning $n_{34} \neq 0$,

$$H_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{n_{31}}{n_{34}} & -\frac{n_{32}}{n_{34}} & -\frac{n_{33}}{n_{34}} & \frac{1}{n_{34}} \end{bmatrix}$$

Therefore

$$H = \begin{bmatrix} A^{-1} & -A^{-1}b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{n_{31}}{n_{34}} & -\frac{n_{32}}{n_{34}} & -\frac{n_{33}}{n_{34}} & \frac{1}{n_{34}} \end{bmatrix}$$

(b)

$\because MX = MHH^{-1}X$

For fundamental matrix $x'^{-1}Fx = 0$,

If x and x' are corresponding points with respect to (M, M') for 3D point X,

There must also exist 3D point $H^{-1}X$, that matches x and x' using $(MH, M'H)$.

(c)

As shown in (a),

$$\widehat{M} = [I \quad 0], \widehat{M'} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$\therefore$ Their fundamental matrix is $[b]_\times A$.

$\because$ From the conclusion in (b), the fundamental matrices corresponding to the pairs of camera matrices (M, M') and $(MH = \widehat{M}, M'H = \widehat{M'})$ are the same, which is $[b]_\times A$.

$$\therefore F = \begin{bmatrix} 0 & -1 & b_2 \\ 1 & 0 & -b_1 \\ -b_2 & b_1 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 \end{bmatrix}$$

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

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

$\therefore$ Its seven parameter expression are $-\frac{a_{21}}{a_{12}}, \frac{a_{22}}{a_{12}}, -\frac{a_{23}}{a_{12}}, \frac{a_{11}}{a_{12}}, \frac{a_{13}}{a_{12}}, b_1, b_2$

2.

$\because$ l and l' are epipolar lines corresponding to x and x', According to the definition, $l' = Fx$.

Suppose x is the intersection of l and k, $x = k \times l = [k]_\times l$. $\therefore$

$$l' = Fx = F[k]_\times l$$

3.

3.1

Algorithm 1: for 8-point logarithm without normalization

1. Construct matrix W using all the corresponding points
   - Each row in form $[u'u \quad u'v \quad u' \quad v'u \quad v'v \quad v' \quad u \quad v \quad 1]$
2. Solve F from $WF = 0$ using SVD
3. Replace F by $\hat{F}$, such that $\det(\hat{F}) = 0$ using SVD to ensure rank 2.
4. Compute epiporlar line using $l = F \cdot x'$, $l' = F' \cdot x$
5. Compute error

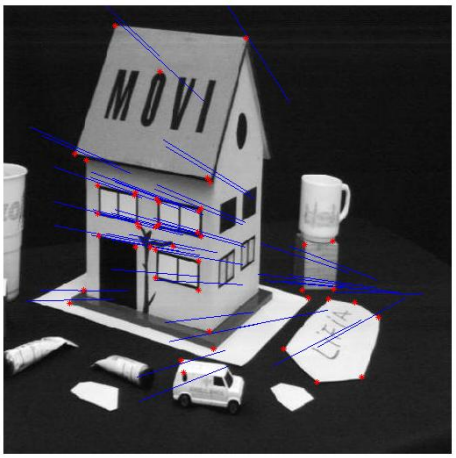
For the code copy, please see the following helper function: *FMatrix.m*

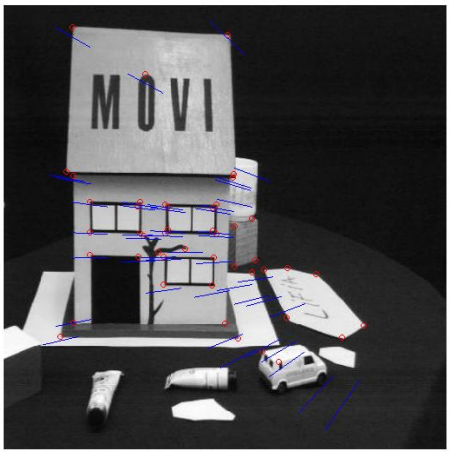
For data set 1:

$$dis_{img1} = 28.026$$
$$dis_{img2} = 25.163$$

For data set 2:

$$dis_{img1} = 9.7014$$
$$dis_{img2} = 14.568$$





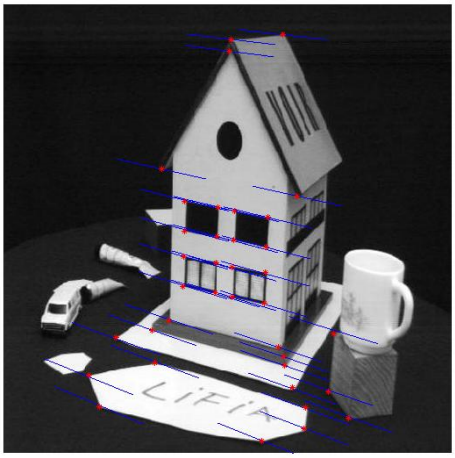
Algorithm 2: for 8-point logarithm with normalization

1. Normalization: transform the image coordinates according to
$$\hat{x}_i = Tx, \hat{x}'_i = T'x'$$
   - T involves Isotropic scaling and centroid
2. Perform step 1-3 in algorithm 1
3. De-normalize $x, x', F: F = T'^T FT$
4. Perform step 4 and 5 in algorithm 1

For the code copy, please see the following helper function: *FMatrix.m*

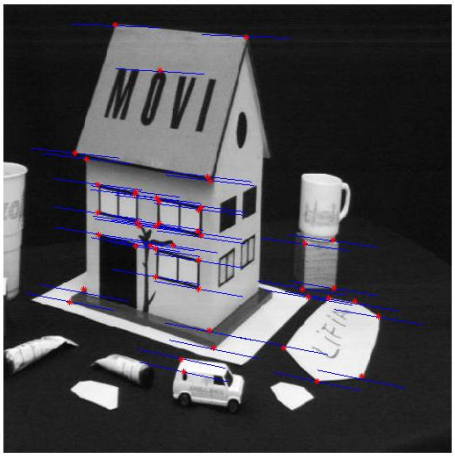
For data set 1:

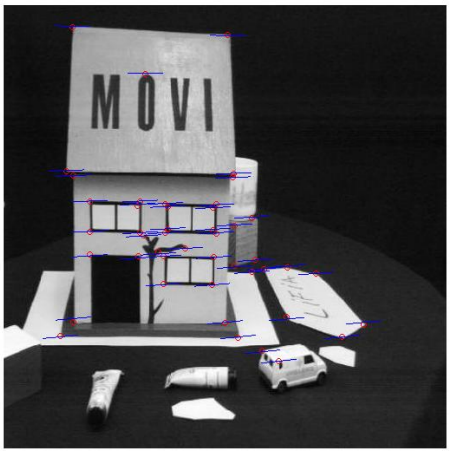
$$dis_{img1} = 0.8844$$
$$dis_{img2} = 0.82422$$



For data set 2:

$$dis_{img1} = 0.8914$$
$$dis_{img2} = 0.89353$$





3.2

Algorithm for Stereo Rectification

1. Compute fundamental matrix F and find epipoles e and e' in both images
   - As discussed in session 3.1
2. Select a projective transformation H' that maps e' to the point at infinity, $\begin{bmatrix}1\\0\\0\end{bmatrix}$
   - Transform e' to e'' $\begin{bmatrix}len\\0\\1\end{bmatrix}$ with H_T and H_R where len is its length
   - Compute $H_1 = H_R \cdot H_T$
   - Send it to infinity which scaling it with len^{-1}, so to normalize it using H_2
   - Compute $H' = H_1 \cdot H_2$
3. Find the matching H minimizing the least-squares distance
   - Since H' transforms e' to infinity, $H = H_A H' M$, where $F = [e]'_\times M$
   - Compute M from $F = [e]'_\times M$
   - Compute H_0 from $H_0 = H'M$
   - Compute transformed matrices: $\hat{x}_i = H_0 x$ and $\widehat{x'} = H'x'$
   - Compute $H_A = \begin{bmatrix}a & b & c\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$ by minimizing $\sum_i (a\hat{x}_i + b\hat{y}_i + c - \widehat{x'}_i)^2$ using least square equation: $Ax = b$ leads to $x = A^+ b$
   - $H = H_A \cdot H_0$
4. Resample the first image according to the projective transformation H and the second image according to the projective transformation H′

For the code copy, please see the following helper function: *imageRect.m*

The computed results are summarized as the following:

For set 1:

$$H = \begin{bmatrix}2.1065 \cdot 10^{-6} & -1.647 \cdot 10^{-6} & -3.354 \cdot 10^{-3}\\1.2335\text{e} \cdot 10^{-6} & -5.9127 \cdot 10^{-6} & -1.717 \cdot 10^{-3}\\-6.2663 \cdot 10^{-9} & 2.013 \cdot 10^{-1} & 1.0212 \cdot 10^{-5}\end{bmatrix}$$

$$H' = \begin{bmatrix}-9.9489 \cdot 10^{-1} & -1.01 \cdot 10^{-1} & -2.8055 \cdot 10^{2}\\1.01 \cdot 10^{-1} & -9.9489 \cdot 10^{-1} & -2.2884 \cdot 10^{2}\\9.3144 \cdot 10^{-4} & 9.4559 \cdot 10^{-5} & 1.2627\end{bmatrix}$$

$$err = 74.279$$

For set 2:

$$H = \begin{bmatrix}-7.3225 \cdot 10^{-7} & 2.7156 \cdot 10^{-6} & 2.2629 \cdot 10^{-3}\\-5.4272\text{e} \cdot 10^{-7} & 5.2278 \cdot 10^{-6} & 1.4334 \cdot 10^{-3}\\2.2287 \cdot 10^{-9} & -9.5275 \cdot 10^{-11} & -7.5068 \cdot 10^{-6}\end{bmatrix}$$

$$H' = \begin{bmatrix}-9.9279 \cdot 10^{-1} & -1.1988 \cdot 10^{-1} & -2.8484 \cdot 10^{2}\\1.1988 \cdot 10^{-1} & -9.9279 \cdot 10^{-1} & -2.2347 \cdot 10^{2}\\5.8878 \cdot 10^{-4} & 7.1092 \cdot 10^{-5} & 1.1689\end{bmatrix}$$

$$err = 58.188$$

Matlab code for Q3

%- Run code: q3solution -%

```
%% EECS 442 - HW 02 - Q3 Fundamental matrix and Image rectification

%  Declaration
%  ------------
%  Date: 2014 / 10 / 03
%  Author: WU Tongshuang, 40782306

%  used helper function
%  --------------------
%  FMatrix(pathdata1,pathdata2,pathimg1,pathimg2)
%  FMatrix_normalization(pathdata1,pathdata2,pathimg1,pathimg2)
%  imageRect(pathdata1,pathdata2,pathimg1,pathimg2)

%% Initialization
clear; close all; clc
%% ================= Part 1: fundamental matrix ============================
fprintf('Running HW2 Q3.1 exercise ... \n');
fprintf('Computing FM and distances for set 1...');
% without normalization
[F1,dist1_1,dist1_2]= FMatrix(...
    'set1/pt_2D_1.txt','set1/pt_2D_2.txt',...
    'set1/image1.jpg','set1/image2.jpg')
[F1_norm,dist1_1_norm,dist1_2_norm]=FMatrix_normalization(...
    'set1/pt_2D_1.txt','set1/pt_2D_2.txt',...
    'set1/image1.jpg','set1/image2.jpg')
fprintf('Computing FM and distances for set 2...');
% without normalization
[F2,dist2_1,dist2_2]= FMatrix(...
    'set2/pt_2D_1.txt','set2/pt_2D_2.txt',...
    'set2/image1.jpg','set2/image2.jpg')
[F2_norm,dist2_1_norm,dist2_2_norm]=FMatrix_normalization(...
    'set2/pt_2D_1.txt','set2/pt_2D_2.txt',...
    'set2/image1.jpg','set2/image2.jpg')
%% ================= Part 2: stereo fectification ==========================
fprintf('Running HW2 Q3.2 exercise ... \n');
fprintf('Performing rectification for set 1...');
[HL_1 HR_1 err_1] = imageRect('set1/pt_2D_1.txt','set1/pt_2D_2.txt',...
              'set1/image1.jpg','set1/image2.jpg')
fprintf('Performing rectification for set 2...');
[HL_2 HR_2 err_2] = imageRect('set2/pt_2D_1.txt','set2/pt_2D_2.txt',...
              'set2/image1.jpg','set2/image2.jpg')
```

%- Helper function : **FMatrix** -%

```
function [F,dist1,dist2] = FMatrix(pathdata1,pathdata2,pathimg1,pathimg2)
% EECS 442 HW2 Q3a
% fundenmantal matrix without normalization

% load feature points from two images. column wise data
% format: dimention(3) * N
[X1,X2] = readTextFiles(pathdata1,pathdata2);

% construction of W, every row is a set of corresponding points
W=ones(9,size(X1,2));
W(1,:) = X2(1,:).*X1(1,:);
W(2,:) = X2(1,:).*X1(2,:);
W(3,:) = X2(1,:);
W(4,:) = X2(2,:).*X1(1,:);
W(5,:) = X2(2,:).*X1(2,:);
W(6,:) = X2(2,:);
W(7,:) = X1(1,:);
W(8,:) = X1(2,:);
% calculate SVD of A
W=W';
[U, D, V] = svd (W);
f = V(:,end); % f is the last column of V
F = reshape(f,3,3);
F = F';
[U, D, V] = svd(F);
D(:,3:end) = 0; % only use the first two columns
F = U * D * V';
F = F';
% calculate distances
% epipolar lines for X1
l1 = F * X2; % 3-3 * 3-n
% epipolar lines for X2
l2 = F'* X1;

% distance between X1 and l1
temp = abs(l1(1,:).*X1(1,:)+l1(2,:).*X1(2,:)+l1(3,:));
dist1 = temp./(l1(1,:).^2+l1(2,:).^2).^.5;
dist1 = mean(dist1(:));
% distance between X2 and l2
temp = abs(l2(1,:).*X2(1,:)+l2(2,:).*X2(2,:)+l2(3,:));
dist2 = temp./(l2(1,:).^2+l2(2,:).^2).^.5;
dist2 = mean(dist2(:));
```

```
% draw feature points and epipolar lines
figure;
img1 = imread(pathimg1);
imshow(img1);
hold on;
for i=1:size(X1,2)
    x = X1(1,i)-50:X1(1,i)+50;
    y = -(l1(1,i).*x+l1(3,i))./l1(2,i);
    plot(x,y,'b');
end
plot(X1(1,:),X1(2,:),'r*');
hold off;

figure;
img2 = imread(pathimg2);
imshow(img2);
hold on;
plot(X2(1,:),X2(2,:),'ro');
for i=1:size(X2,2)
    x = X2(1,i)-20:X2(1,i)+20;
    y = -(l2(1,i).*x+l2(3,i))./l2(2,i);
    plot(x,y,'b');
end
hold off;
```

%- Helper function : **FMatrix_normalization** -%

```
function [F,dist1,dist2] = FMatrix_normalization(pathdata1,pathdata2,pathimg1,pathimg2)
% EECS 442 HW2 Q3a
% fundenmantal matrix with normalization

% load feature points from two images. column wise data
% format: dimention(3) * N
[X1,X2] = readTextFiles(pathdata1,pathdata2);
% normalization
% centroid
X1Trans = [-mean(X1(1,:)); -mean(X1(2,:)); 1];
X2Trans = [-mean(X2(1,:)); -mean(X2(2,:)); 1];
% Isotropic scaling
X1Scale = sqrt(2)./std(X1,1,2);
X2Scale = sqrt(2)./std(X2,1,2);
% transform matrix
transM1 = eye(3,3);
transM1(:,3) = X1Trans;
transM2 = eye(3,3);
transM2(:,3) = X2Trans;
scaleM1 = eye(3,3);
scaleM1(1,1) = X1Scale(1);
scaleM1(2,2) = X1Scale(2);
scaleM2 = eye(3,3);
scaleM2(1,1) = X2Scale(1);
scaleM2(2,2) = X2Scale(2);
normM1 = scaleM1 * transM1;
normM2 = scaleM2 * transM2;
normX1 = normM1 * X1;
normX2 = normM2 * X2;
% construction of W, every row is a set of corresponding points
W=ones(9,size(X1,2));
W(1,:) = normX2(1,:).*normX1(1,:);
W(2,:) = normX2(1,:).*normX1(2,:);
W(3,:) = normX2(1,:);
W(4,:) = normX2(2,:).*normX1(1,:);
W(5,:) = normX2(2,:).*normX1(2,:);
W(6,:) = normX2(2,:);
W(7,:) = normX1(1,:);
W(8,:) = normX1(2,:);
% calculate SVD of A
W=W';
[U, D, V] = svd (W);
f = V(:,end); % f is the last column of V
```

```
F = reshape(f,3,3);
F = F';
[U, D, V] = svd(F);
D(:,3:end) = 0; % only use the first two columns
F = U * D * V';
% denormalization
F  = normM2' * F * normM1;
F = F';
% calculate distances
% epipolar lines for X1
l1 = F * X2; % 3-3 * 3-n
% epipolar lines for X2
l2 = F'* X1;
% distance between X1 and l1
temp = abs(l1(1,:).*X1(1,:)+l1(2,:).*X1(2,:)+l1(3,:));
dist1 = temp./(l1(1,:).^2+l1(2,:).^2).^.5;
dist1 = mean(dist1(:));
% distance between X2 and l2
temp = abs(l2(1,:).*X2(1,:)+l2(2,:).*X2(2,:)+l2(3,:));
dist2 = temp./(l2(1,:).^2+l2(2,:).^2).^.5;
dist2 = mean(dist2(:));
% draw feature points and epipolar lines
figure;
img1 = imread(pathimg1);
imshow(img1);
hold on;
for i=1:size(X1,2)
    x = X1(1,i)-50:X1(1,i)+50;
    y = -(l1(1,i).*x+l1(3,i))./l1(2,i);
    plot(x,y,'b');
end
plot(X1(1,:),X1(2,:),'r*');
hold off;
figure;
img2 = imread(pathimg2);
imshow(img2);
hold on;
plot(X2(1,:),X2(2,:),'ro');
for i=1:size(X2,2)
    x = X2(1,i)-20:X2(1,i)+20;
    y = -(l2(1,i).*x+l2(3,i))./l2(2,i);
    plot(x,y,'b');
end
hold off;
```

%- Helper function : **imageRect** -%

```
function [HL,HR,err] = imageRect(pathdata1,pathdata2,pathimg1,pathimg2)
% get fundamental matrix
F = FMatrix_normalization(pathdata1,pathdata2,pathimg1,pathimg2);
img1 = imread(pathimg1);
img2 = imread(pathimg2);

% calculate epipoles
[U, D, V] = svd (F);
epL = V(:,end); % last column of V
epL = epL/epL(3);
[U, D, V] = svd(F');
epR = V(:,end);
epR = epR/epR(3);

% map epi_prime to [1,0,1]
% x0 is center of the image
pc = [size(img2,2)/2 ; size(img2,1)/2];
TH=[
    1, 0, pc(1);
    0, 1, pc(2);
    0, 0, 1
    ];
epR_prime = TH * epR;
len = sqrt(epR_prime(1)^2+epR_prime(2)^2);
sine = epR_prime(2) / len;
cose = epR_prime(1) / len;
RH=[
    cose, -sine, 0;
    sine, cose, 0;
    0, 0, 1
    ];
% map e_prime to the x-axis at location [length,0,1]
H1=RH * TH;
% send epipole to infinity
H2 = [
    1 0 0;
    0 1 0;
    -1/len 0 1
    ];
% transformation for one image
HR = H2*H1;

% find H to align epipolar lines
```

```
% calculate M
% [e'_]xM = F
exM = pinv([0, -epR(3), epR(2); epR(3),0,-epR(1);-epR(2),epR(1),0]);
M = exM * F;
H0 = HR * M;
% find HA =  I + HR * epR * a'
[X1,X2] =  readTextFiles(pathdata1,pathdata2);
X2_hat = HR * X2;
X1_hat = H0 * X1;
X1_hat = X1_hat./repmat(X1_hat(3,:),3,1);
X2_hat = X2_hat./repmat(X2_hat(3,:),3,1);
A = X1_hat';
b = X2_hat(1,:)';
% use the method solving least square parameter
lsp = pinv(A) * b;
HA = [lsp(1),lsp(2),lsp(3);0 1 0;0 0 1];
% get H
HL = HA * H0;

% apply transform and find errors
X1H = HL*X1;
X1H = X1H./repmat(X1H(3,:),3,1);
X2H = HR*X2;
X2H = X2H./repmat(X2H(3,:),3,1);
err = sum((X1H(1:2,:)-X2H(1:2,:)).^2);
err = mean(sqrt(err));
```