ECE 661 Fall 2018
Homework #9

Ye Shi
shi349@purdue.edu

November 21, 2018

Contents

1 Methodology

In this section, I demonstrate the methods to reconstruction of 3D world frame and rectification by un-calibrated stereo images.

1.1 Estimate Fundamental Matrix F

For any given pair of corresponding points $(\vec{x_i}, \vec{x_i'})$, the epipoplar geometry satisfies

$$\vec{x_i}^T F \vec{x_i'} = 0,$$

where $\vec{x_i} = [x_i,\, y_i,\, 1]^H$, $\vec{x_i'} = [x_i',\, y_i',\, 1]^H$ and F is the 3×3 fundamental matrix

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Then we conduct it into

$$A_i \vec{f} = \begin{bmatrix} x_i'x_i & x_i'y_i & x_i' & y_i'x_i & y_i'y_i & y_i' & x_i & y_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ \vdots \\ f_{33} \end{bmatrix} = 0.$$

In this case, we need at least 8 pairs of points to estimate F. Please notice, the 8 pairs of points cannot be linear dependent, which will cause A to be a singular matrix.

We can solve $\vec{f}$ by singular value decomposition (SVD) and reshape it into 3×3 matrix F.

Please note F is a 7-DoF matrix, which means its rank is 2. So we need to keep this property by SVD F again as following steps.

1. $[U,\, D,\, V] \leftarrow SVD(F)$
2. $D_{33} \leftarrow 0$, set the least eigenvalue to 0.
3. $F = UDV^H$

For large group points, $(\vec{x}, \vec{x'})$ needs to be normalized before calculating F as $\hat{\vec{x}} = T\vec{x}$ and $\hat{\vec{x'}} = T\vec{x'}$. The normalization method is given as follows,

1. Find the center of the points group $\vec{c} = (c_x,\, c_y)$.
2. Calculate the average euclidean distance d from $\vec{c}$ to each point $\vec{x_i}$.
3. $$T = \begin{bmatrix} \frac{1}{d} & 0 & \frac{c_x}{d} \\ 0 & \frac{1}{d} & \frac{c_y}{d} \\ 0 & 0 & 1 \end{bmatrix}$$

If we use $(\hat{\vec{x}}, \hat{\vec{x'}})$ to find $\hat{F}$, where $\hat{\vec{x}}\hat{F}\hat{\vec{x'}} = 0$, we need to restore

$$F = T'^T \hat{F} T.$$

1.2 Estimate Epipoles $\vec{e}$ and $\vec{e'}$

$$\begin{cases} \vec{e}F = 0 \\ F\vec{e'} = 0 \rightarrow \vec{e'}F^T = 0 \end{cases}$$

It is obvious that $\vec{e}$ and $\vec{e'}$ are the left null space of F and F^T.

1.3 Estimates Camera Matrix P and P'

$$P = [I_{3\times3}|0] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P' = [\vec{e'}_X F|\vec{e'}]$$

$\vec{e'}_X$ is the skew-symmetric matrix $\begin{bmatrix} 0 & -e'_3 & e'_2 \\ e'_3 & 0 & -e'_1 \\ -e'_2 & e'_1 & 0 \end{bmatrix}$.

1.4 Reconstruct 3D Points $\vec{X}$ via Triangulation Method

We notate $P = [\vec{p_1}, \vec{p_2}, \vec{p_3}]^T$ and $P' = [\vec{p'_1}, \vec{p'_2}, \vec{p'_3}]^T$

$$\begin{cases} x_i(\vec{p_3}\vec{X_i}) - \vec{p_1}\vec{X_i} = 0 \\ y_i(\vec{p_3}\vec{X_i}) - \vec{p_2}\vec{X_i} = 0 \\ x'_i(\vec{p'_3}\vec{X_i}) - \vec{p'_1}\vec{X_i} = 0 \\ y'_i(\vec{p'_3}\vec{X_i}) - \vec{p'_2}\vec{X_i} = 0 \end{cases}$$

We then can apply SVD to solve $\vec{X}$.

1.5 Refine Fundamental Matrix F via Levenberg–Marquardt (LM)

Actually, we refine P' and X here instead of F and then reconstruct F by P'.
The cost fucntion is

$$\sum_i d(\vec{x_i}, PX_i) + d(\vec{x'_i}, P'X_i).$$

$d(x, y)$ is the euclidean distance.

1.6 Rectification

Rectification projects images onto a common image plane in such a way that the corresponding points have the same row coordinates. This image projection makes the image appear as though the two cameras are parallel. We will have

$$\hat{\vec{x_i}} = H\vec{x_i}$$

and

$$\hat{\vec{x'_i}} = H'\vec{x'_i}$$

The basic idea is to project the baseline to l_∞.

1. Compute translate matrix T_1 sending image center into origin.
2. Compute rotation matrix R that rotate epipole $\vec{e}$ to $[f, 0, 1]^T$, f is any real number.
3. Compute protective matrix G that send $[f, 0, 1]^T$ to an idea point $[f, 0, 0]^T$
4. Compute translate matrix T_2 sending origin back to image center.
5. $$H = T_2GRT_1$$

The H' is the same method.

After this, we need to match the vertical axis between the stereo images. We simply apply one more homography $H_A = \begin{bmatrix} 1 & 0 & 0 \\ a & b & c \\ 0 & 0 & 1 \end{bmatrix}$.

Please notice I used Matlab which regard the horizonal axis as x-axis, so I used the second row of H_A to match vertical axis. So we simply use least square estimation to minimize

$$\sum_i (a\hat{x_i} + b\hat{y_i} + c - \hat{y_i'})^2.$$

Then $H \to H_A H$.

Since the baseline goes to l_∞, now the fundamental matrix is

$$F \to H'^{T^{-1}} F H^{-1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix}.$$

However, this is the ideal situation.

2 Results

The photos in Fig.1 are taken by a Iphone 7 and downscale to 800×600 for computation.

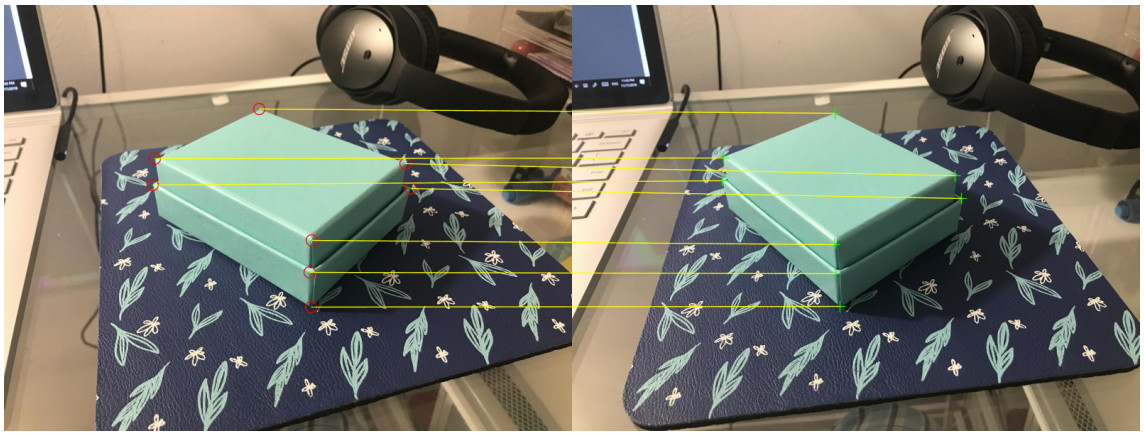

Figure 1: Images with Corresponding Points. The 8 points are manually marked.

- The fundamental matrix before refinement and before rectification

$$F_0 \begin{pmatrix} 0 & 0 & -0.003764 \\ 0 & 0 & -0.0281 \\ 0.003033 & 0.0218 & 1.0 \end{pmatrix}$$

The epipoles are

$$e_0 = \begin{pmatrix} 2511.0 \\ -395.3 \\ 1.0 \end{pmatrix},$$

$$e_0' = \begin{pmatrix} 2900.0 \\ -352.9 \\ 1.0 \end{pmatrix}.$$

The camera matrices are

$$P_0 = \begin{pmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \end{pmatrix},$$

$$P_0' = \begin{pmatrix} -1.07 & -7.692 & -352.9 & 2900.0 \\ -8.796 & -63.21 & -2900.0 & -352.9 \\ 0.03428 & 0.008251 & -82.8 & 1.0 \end{pmatrix}.$$

- The fundamental matrix after refinement and before rectification is

$$F = \begin{pmatrix} 0 & 0 & -0.003764 \\ 0 & 0 & -0.0281 \\ 0.003033 & 0.0218 & 1.0 \end{pmatrix}.$$

Since we don't have enough points, the improvement of refinement is limited, which can be implied from

$$F - F_0 = \begin{pmatrix} 3.483 \times 10^{-20} & 3.083 \times 10^{-19} & 3.383 \times 10^{-17} \\ -8.47 \times 10^{-21} & -3.812 \times 10^{-20} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The epipoles are

$$e = \begin{pmatrix} 2511.0 \\ -395.3 \\ 1.0 \end{pmatrix},$$

$$e' = \begin{pmatrix} 2900.0 \\ -352.9 \\ 1.0 \end{pmatrix}.$$

The camera matrices are

$$P = \begin{pmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \end{pmatrix},$$

$$P' = \begin{pmatrix} -1.07 & -7.692 & -352.9 & 2900.0 \\ -8.796 & -63.21 & -2900.0 & -352.9 \\ 0.03428 & 0.008251 & -82.8 & 1.0 \end{pmatrix}.$$

Fig.2 shows the corresponding points after rectification. The homographies of the two views are

$$H = \begin{pmatrix} 0.69 & -0.2273 & 146.6 \\ 0.1417 & 0.7941 & -29.12 \\ 0 & 0 & 1.0 \end{pmatrix},$$

$$H' = \begin{pmatrix} 0.7298 & -0.1906 & 122.2 \\ 0.1253 & 0.8897 & -49.27 \\ 0 & 0 & 1.0 \end{pmatrix}.$$

The fundamental matrix after rectification and before refinement is

$$F_0 = {H'}^{-1} F H^{-1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -0.03212 \\ 0 & 0.02762 & 1.0 \end{pmatrix}.$$

Please notice

$$F_0 \simeq \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 1 \end{pmatrix}.$$

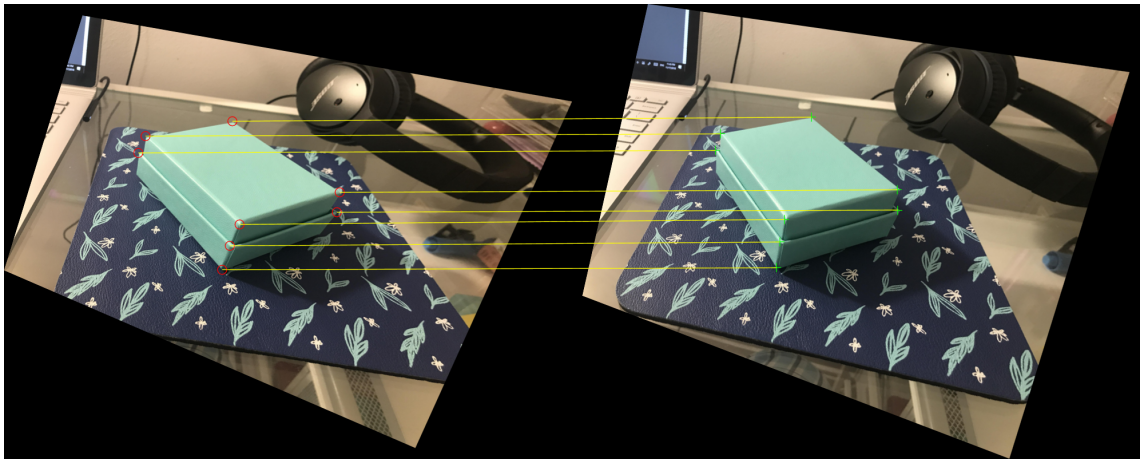

Figure 2: Corresponding Points after Rectification

- The fundamental matrix after rectification and after refinement is

$$F_{rect} = \begin{pmatrix} 0 & 0 & -0.003764 \\ 0 & 0 & -0.0281 \\ 0.003033 & 0.0218 & 1.0 \end{pmatrix}.$$

The epipoles are

$$e_{rect} = \begin{pmatrix} -0.9878 \\ 0.1555 \\ 0 \end{pmatrix},$$

$$e'_{rect} = \begin{pmatrix} 0.9927 \\ -0.1208 \\ 0 \end{pmatrix}.$$

Please notice here e_{rect} and e'_{rect} are ideal points, which means the baseline is a vanishing line. The camera matrices are

$$P_{rect} = \begin{pmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \end{pmatrix},$$

$$P'_{rect} = \begin{pmatrix} -1.07 & -7.692 & -352.9 & 2900.0 \\ -8.796 & -63.21 & -2900.0 & -352.9 \\ 0.03428 & 0.008251 & -82.8 & 1.0 \end{pmatrix}.$$

Fig.4 and Fig.5 show the 3D reconstruction and matching points.

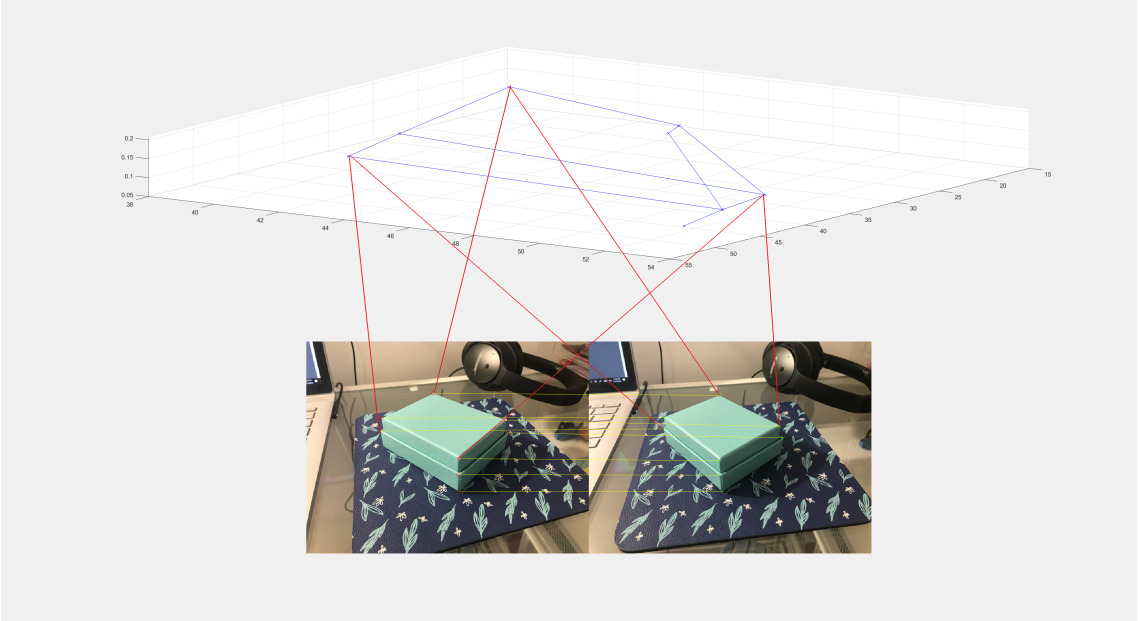

Figure 3: 3D Reconstruction with Corresponding Points before refinement and before Rectification

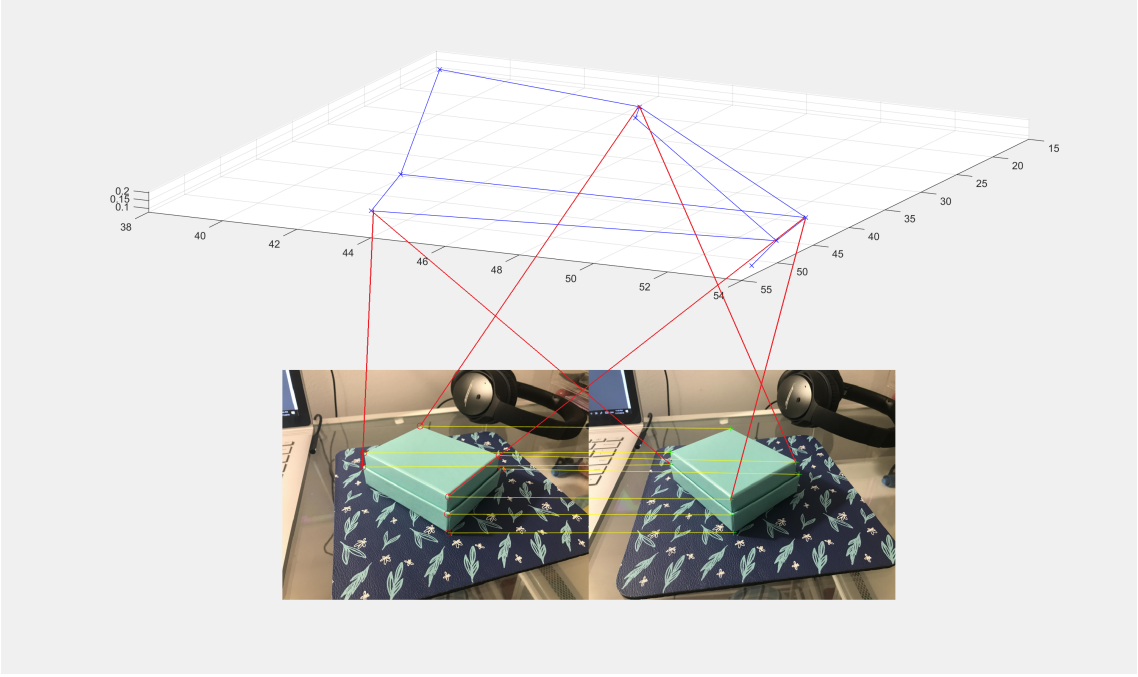

Figure 4: 3D Reconstruction with Corresponding Points after refinement and before Rectification

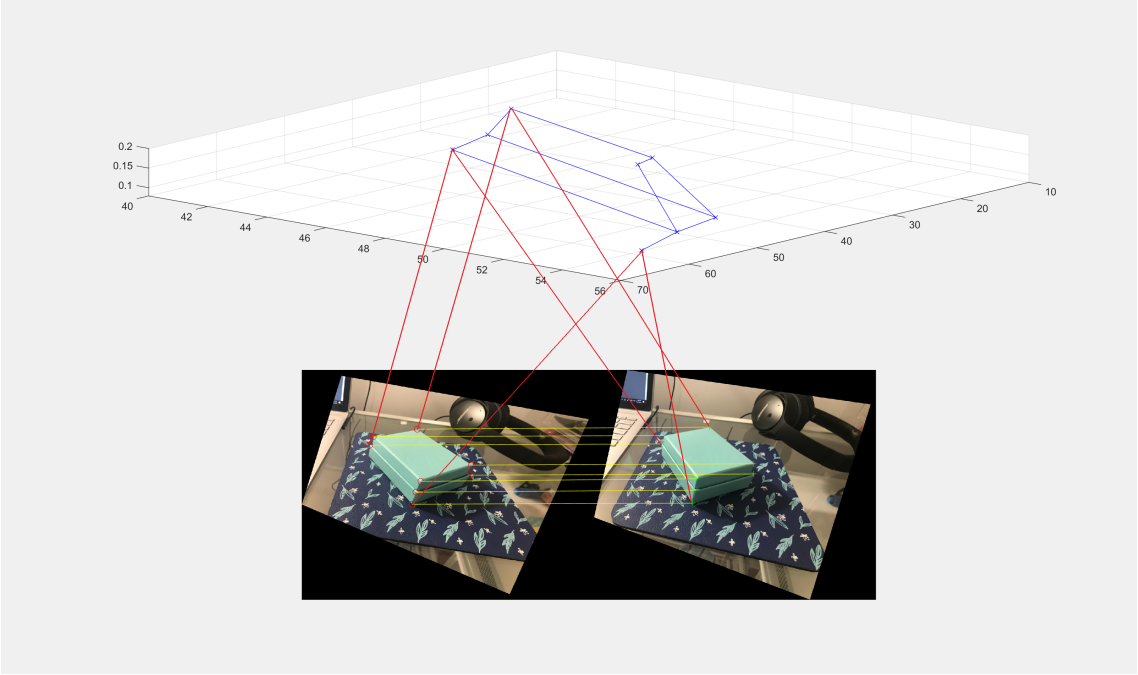

Figure 5: 3D Reconstruction with Corresponding Points after Rectification

Fig.6, Fig.7 and Fig.8 intuitively show the 3D reconstruction distortion and matching points.

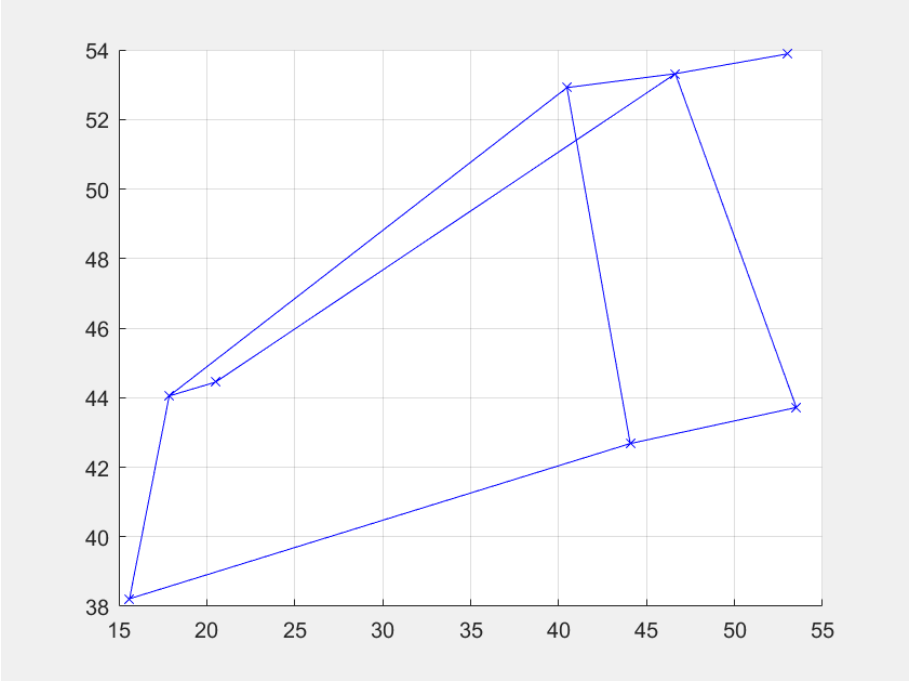

Figure 6: Distortion of 3D Reconstruction before refinement and before Rectification

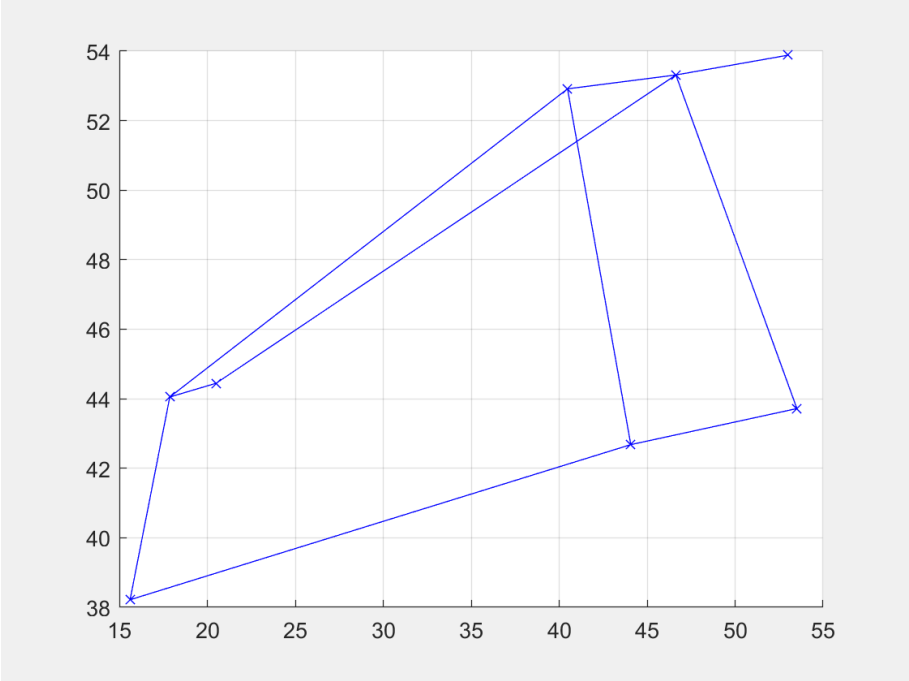

Figure 7: Distortion of 3D Reconstruction after refinement and before Rectification

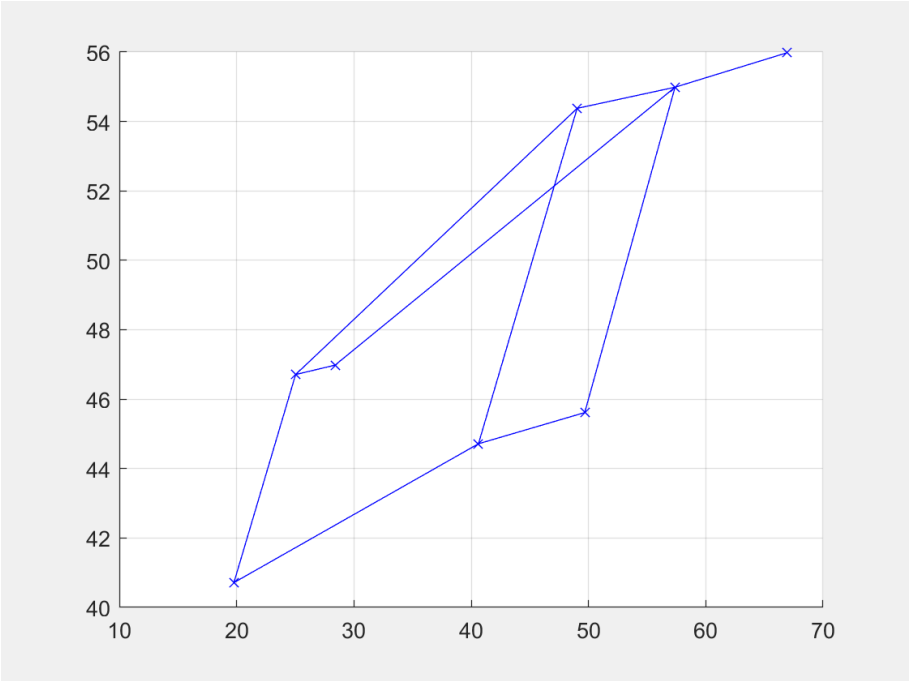

Figure 8: Distortion of 3D Reconstruction after Rectification

3 Source Code

The code is written in Matlab 2017b and Windows 10.

3.1 Select Corresponding Points

```
1 function points=getpoints(CI)
2 figure
3 imshow(CI)
```

```
[y x] = getpts;
points = round([y x]);
end
```

3.2 Main

```
%% ECE 661 2018 Fall Homework 9
% Ye Shi
% shi349@purdue.edu

close all;clear; clc;
L =@(l) latex(vpa(l.*(abs(l)>1e-3),4)) % quick latex generator
%% Import images and resize them
CI1 = imread('IMG_1809.JPG');
CI2 = imread('IMG_1810.JPG');
CI1 = imresize(CI1,[600,800]);
CI2 = imresize(CI2,[600,800]);
I1 = rgb2gray(CI1);
I2 = rgb2gray(CI2);
%% Points Preset
points1 =[   360   147
     213   217
     434   332
     566   224
     573   259
     431   377
     210   254
     433   426];

points2 =[   371   153
     214   216
     370   338
     540   241
     549   273
     372   379
     212   248
     378   425];
%% Visualization of matching points
if 1
     f = figure;
     showMatchedFeatures(CI1,CI2,points1,points2,'montage');
     hold off;
     img = getframe();
     imwrite(img.cdata,['selection.png']);
     close(f);
end



%% Find F, In the 1st round, we assume I1 is in the world plane
F0 = findF(points1,points2,1);
[P10, P20] = findP(F0);
X0 = findX(P10,P20,points1,points2);
[F,P1,e1,P2,e2,X] = refineF(F0,points1,points2)

%% Rectify the image
[H1,H2] = rectify(P1,P2,e1,e2,I1,I2,points1,points2);

rpoints1 = H1*[points1 ones(8,1)]'
rpoints1 = [rpoints1(1:2,:)./rpoints1(end,:)]'
rpoints2 = H2*[points2 ones(8,1)]'
rpoints2 = [rpoints2(1:2,:)./rpoints2(end,:)]'


%% Visualization of Rectification

if 1
     f  = figure
     t1 = maketform('projective',H1');
     [J1,xdata1,ydata1]= imtransform(CI1,t1);
     % Add offset for the projection
     dpoints1= rpoints1;
     dpoints1(:,1) = dpoints1(:,1) - xdata1(1);
```

```
    dpoints1(:,2) = dpoints1(:,2) - ydata1(1);
    t2 = maketform('projective',H2');
    [J2,xdata2,ydata2] = imtransform(CI2,t2);
    dpoints2= rpoints2;
    dpoints2(:,1) = dpoints2(:,1) - xdata2(1);
    dpoints2(:,2) = dpoints2(:,2) - ydata2(1);
    showMatchedFeatures(J1,J2,dpoints1,dpoints2,'montage');
    hold off;
    img = getframe();
    imwrite(img.cdata,['rectification.png']);
    close(f);
end
%% Find F, In the 1st round, we assume I1 is in the world plane
rF0 = findF(dpoints1,dpoints2,1);
[rF,rP1,re1,rP2,re2,rX] = refineF(F0,dpoints1,dpoints2)

%% Visualize All 3D plots
f = WorldPlot(X,CI1,CI2,points1,points2)
r = WorldPlot(rX,J1,J2,dpoints1,dpoints2)
```

3.3 Fundamental Matrix

```
%% ECE 661 2018 Fall Homework 9
% Ye Shi
% shi349@purdue.edu

function F = findF(points1,points2,Nflag)
% This function is used to find the fundemental matrix
if nargin <3
    Nflag = 0;
end

l = length(points1);
if l <8
    error("Please input at least 8 pairs of points!");
end

% Normalize
if Nflag
    T1 = findT(points1);
    T2 = findT(points2);
    points1 = T1*[points1 ones(l,1)]';
    points1 = [points1(1:2,:)./points1(end,:)]';
    points2 = T2*[points2 ones(l,1)]';
    points2 = [points2(1:2,:)./points2(end,:)]';
end

A = zeros(l,9);
for i = 1:l
    A(i,:) = [points1(i,1)*points2(i,1) ,
        points1(i,2)*points2(i,1) ,
        points2(i,1) ,
        points1(i,1)*points2(i,2) ,
        points1(i,2)*points2(i,2) ,
        points2(i,2),
        points1(i,1) ,
        points1(i,2) ,
        1];
end
[~,~,V] = svd(A);
F = reshape(V(:,end),3,3)';
% Reduce rank(F) to 2
if rank(F) == 3
[U,S,V] = svd(F);
S(end) = 0;
F = U*S*V';
end
if Nflag
    F = T2'*F*T1;
end
F = F./F(end);
end
```

3.4 Normalization Matrix

```
%% ECE 661 2018 Fall Homework 9
% Ye Shi
% shi349@purdue.edu


function T = findT(points)
% Find the normalizion matrix T
% Find mean of the interest points
mu = mean(points);
d = points-repmat(mu,size(points,1),1);
d = sqrt(d(:,1).^2+d(:,2).^2);
dmean = mean(d);
k = sqrt(2)/dmean;
x = -k*mu(1);
y = -k*mu(2);
T = [k 0 x; 0 k y ; 0 0 1];
end
```

3.5 Refinement Fundamental Matrix

```
%% ECE 661 2018 Fall Homework 9
% Ye Shi
% shi349@purdue.edu

function [F,P1,e1,P2,e2,X] = refineF(F0,points1,points2)
% Refine F by LM
[P1,P20] = findP(F0);
p20 = P20(:);
X0 = findX(P1,P20,points1,points2);
xl = size(X0,1);
x0 = X0(:);
p0 = [p20;x0];
options.Algorithm = 'levenberg-marquardt';
options.FunctionTolerance =1e-8;
% options = optimoptions('lsqnonlin','Algorithm','levenberg-marquardt'...
% ,'TolX',0.00000000001,'TolFun',0.00000000001);
p = lsqnonlin(@PXcost,p0,[],[],options,points1,points2);
p2 = p(1:12);
P2 = reshape(p2,3,4);
P2 = P2./P2(end);
x = p(13:end);
X =reshape(x,xl,3);
e2 = P2(:,end);
e2x= [0 -e2(3) e2(2);e2(3) 0 -e2(1);-e2(2) e2(1) 0];
% F = reshape(f,3,3);
F = e2x*P2*pinv(P1);
F = F./F(end);
e1 = null(F);
end

function err = PXcost(p,points1,points2)
% Cost function for refine P and X
nt = size(points1,1);
p2 = p(1:12);
P2 = reshape(p2,3,4);
x = p(13:end);
X =reshape(x,nt,3);
P1 = [eye(3),zeros(3,1)];
P2 = reshape(p2,3,4);
estPoints1 = P1*[X';ones(1,nt)];
estPoints1 = estPoints1(1:2,:)./ estPoints1(end,:);
estPoints2 = P2*[X';ones(1,nt)];
estPoints2 = estPoints2(1:2,:)./ estPoints2(end,:);
err  = sum(vecnorm(estPoints2 - points2')) + ...
    sum(vecnorm(estPoints1 - points1'));
end
```

3.6 Camera Matrix

```
%% ECE 661 2018 Fall Homework 9
% Ye Shi
```

```
% shi349@purdue.edu

function [P1,P2] = findP(F)
% This function is to find P and P' as P1,P2
P1 = [eye(3),zeros(3,1)];
e2 = null(F');
e2 = e2./e2(end);
e2x= [0 -e2(3) e2(2);e2(3) 0 -e2(1);-e2(2) e2(1) 0];
P2 = [e2x*F,e2];
end
```

3.7 Reconstruct World Frame Points

```
%% ECE 661 2018 Fall Homework 9
% Ye Shi
% shi349@purdue.edu

function X = findX(P1,P2,points1,points2)
% This is to find world frame X
n = size(points1,1);
X = zeros(n,3);
for i = 1:n
    A = [ (points1(i,1)*P1(3,:) -P1(1,:));
          (points1(i,2)*P1(3,:) -P1(2,:));
          (points2(i,1)*P2(3,:) -P2(1,:));
          (points2(i,2)*P2(3,:) - P2(2,:)) ];

    [~ ,~, V]=svd(A);
    x=V(:,4);
    x=x(1:3)/x(end);
    X(i,:)=x;
end
end
```

3.8 Rectification

```
%% ECE 661 2018 Fall Homework 9
% Ye Shi
% shi349@purdue.edu

function [H1,H2] = rectify(P1,P2,e1,e2,I1,I2,points1,points2)
% fucntion to find H and H'
[h,w]=size(I1);
%% Find H2
e2=e2/e2(end);
% rotation angle
theta=atan(-(e2(2)-h/2)/(e2(1)-w/2));
f=cos(theta)*(e2(1)-w/2)-sin(theta)*(e2(2)-h/2);
% rotation matrix
R=[cos(theta) -sin(theta) 0;sin(theta) cos(theta) 0; 0 0 1];
% Translate origin to center
T=[1 0 -w/2;0 1 -h/2;0 0 1];
% Translate baseline to infinity
G=[1 0 0;0 1 0;-1/f 0 1];
H2=G*R*T;
% Construct T2 to send origin back to original center
center=[w/2 h/2 1]';
newCen=H2*center;
newCen=newCen/newCen(end);
T2=[1 0 w/2-newCen(1);0 1 h/2-newCen(2);0 0 1];
H2=T2*H2;
H2 = H2./H2(end);

%% Find H1
if 1
    e1=e1/e1(end);
    % rotation angle
    theta=atan(-(e1(2)-h/2)/(e1(1)-w/2));
    f=cos(theta)*(e1(1)-w/2)-sin(theta)*(e1(2)-h/2);
    % rotation matrix
    R=[cos(theta) -sin(theta) 0;sin(theta) cos(theta) 0; 0 0 1];
    % Translate origin to center
    % T=[1 0 -w/2;0 1 -h/2;0 0 1];
```

```
38      T=[1 0 -w/2;0 1 -h/2;0 0 1];
39      % Translate baseline to infinity
40      G=[1 0 0;0 1 0;-1/f 0 1];
41      H1=G*R*T;
42      % Construct T2 to send origin back to original center
43      center=[w/2 h/2 1]';
44      newCen=H1*center;
45      newCen=newCen/newCen(end);
46      T2=[1 0 w/2-newCen(1);0 1 h/2-newCen(2);0 0 1];
47      H1=T2*H1;
48      H1 = H1./H1(end);
49  end
50
51  %% Method from past solution doesn't work
52  if 1
53      ptsNum = size(points1,1);
54      % M=P2*pinv(P1);
55      % H0=H2*M;
56      H0 = H1;
57      % Construct for LSE
58
59      A = H0* [points1,ones(ptsNum,1)]';
60      A =( A./A(end,:))';
61
62      b = H2* [points2,ones(ptsNum,1)]';
63      % b =( b(1,:)./b(end,:))';
64      b =( b(2,:)./b(end,:))';
65
66      p=pinv(A)*b;
67      % HA=[p(1) p(2) p(3); 0 1 0; 0 0 1];
68      HA=[1 0 0; p'; 0 0 1];
69      H1=HA*H0;
70      %% Recenterlize the H1
71      newCen=H1*center;
72      newCen=newCen./newCen(end);
73      T1=[1 0 w/2-newCen(1);0 1 h/2-newCen(2);0 0 1];
74      H1=T1*H1;
75      H1 = H1./H1(end);
76  end
77
78
79
80  end
```

3.9 RANSAC Fundamental Matrix

This part is never used and removed from homework requirement.

```
1   %% ECE 661 2018 Fall Homework 9
2   % Ye Shi
3   % shi349@purdue.edu
4
5   function [bestIdx,bestF] = ransacF(points1,points2)
6   % This function can find the most fitting F from pairs by RANSAC
7   % pairs is the set of all matched points.
8   nt= size(points1,1); % Total number of matches
9   % parameters:
10  p = .8; % The probability that at least one of the N trials will be free of
        outliers.
11  m = 8; % the set of correspondences for calculating F in each trial.
12  esp = 0.2; % the rough estimation of false correspondences from the total set.
13  del = 20; % The decision threshold to construct the inlier set.
14  N  = ceil(log(1-p)/log(1-(1-esp)^m));% Number of trials
15  M = floor((1-esp)*nt); % A minimum value for the size of the inlier set for it to
        be acceptable
16
17  Maxcount = 0;
18  bestF  = zeros(3);
19  bestIdx = zeros(nt,1);
20
21  for i = 1:N
22      randRowIdx = randsample(nt,m);
23      F = findF(points1(randRowIdx,:),points2(randRowIdx,:));
24      if rank(F) == 2
```

```
25          [~,P2] = findP(F);
26          estPoints2 = P2*[points1';zeros(1,nt);ones(1,nt)];
27          estPoints2 = estPoints2(1:2,:)./ estPoints2(end,:);
28          dist  = sum(abs( (estPoints2 - points2')));
29          inIdx = find(dist<del);
30          count = length(inIdx); % use a naive cheese board distance
31          if count > M &&  count > Maxcount
32              bestF = F;
33              bestIdx = inIdx;
34              Maxcount = count;
35          end
36      end
37  end
38  if sum(bestIdx) == 0
39      error('No inline pairs found!');
40  end
41  end
```