Report of the Project

Computer Vision Course

Two-view (or two camera) setup

Hanieh Shabanian

Instructor:

Dr. M. Balasubramanian

Spring 2017

Problem Statement:

- **Get two views (images) of an object or a scene of your choice**

- **Manually choose 8 corresponding points from the two views.**

- **Using the normalized 8 point algorithm discussed in class, estimate the fundamental matrix F that establishes the epipolar geometry of the two views.**

- **Assuming that the essential matrix E is same as the fundamental matrix F, draw epipolar lines of these 8 corresponding points in both views (or images). For example, to draw epipolar lines in the first view or image, use the 8 points chosen in the second view or image and the estimated fundamental matrix F; and vice versa to draw epipolar lines on the second image / view.**

- **Assuming E = F, estimate the translation vector tau and the rotation matrix omega. These estimates will tell the relative orientation of the two views of the scene based on the two images that you captured.**

Solution:

I will describe what I did through Matlab code for each step.

First and second Step: is getting two views (images) of an object. For this step, I drew a chessboard in A4 paper and put it on my desk at the office. Then I took two photos in 2 various positions. After that, I manually choose 8 corresponding points from the two views. The size of both images are 800*600 pixels.

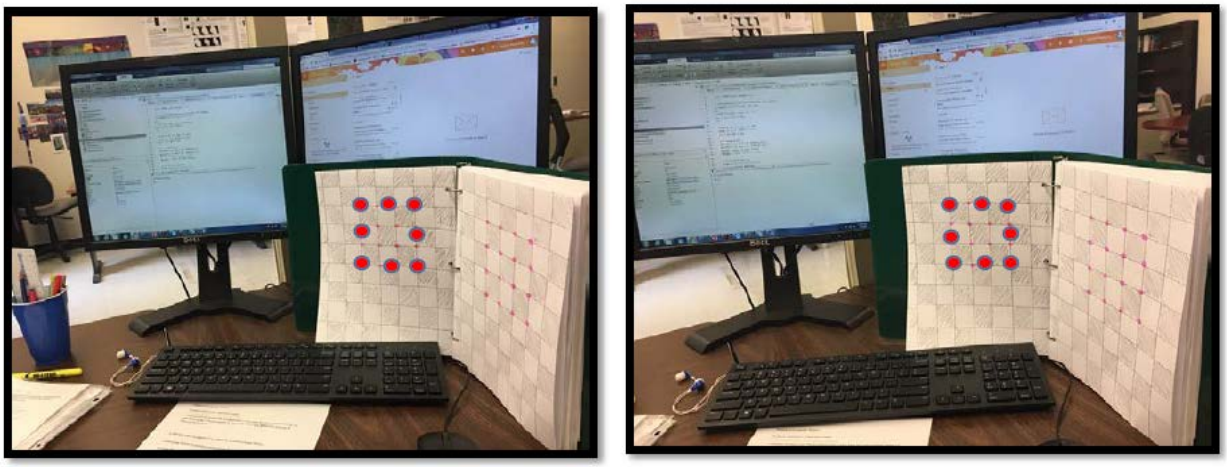

Image 1 Image 2

Figure 1: Captured two images with different positions. 8 points from image1 and image2 are selected.

As you see in the Figure 1, I selected 8 points from image1 and image2. Matlab Code for this step:

```
%% First Step : Getting two views (images) of an object or a scene of your
choice ==> Generate two view  points
% Reading two captured images
% Image 1 and Image II
img1=imread('Image1.jpg');
img2=imread('Image2.jpg');
figure(1);
subplot(1,2,1); imshow(img1); title('Image1'); axis image;axis off;
subplot(1,2,2); imshow(img2);  title('Image2'); axis image;axis off;

%% Second Step :Manually choose 8 corresponding points from the two views.

Point_Image1= [482 263; 483 307; 488 345; 527 345; 563 347; 561 305;558 267;
516 264 ]
figure
imshow(img1)
hold on
m1=Point_Image1;
plot(Point_Image1(:,1),
Point_Image1(:,2),'go','LineWidth',4,'MarkerEdgeColor','r')

Point_Image2= [ 440 264;440 305; 445 348; 486 351; 522 351; 522 310; 522
268; 478 267 ]
figure
imshow(img2)
hold on
m2=Point_Image2;
plot(Point_Image2(:,1),
Point_Image2(:,2),'go','LineWidth',4,'MarkerEdgeColor','r')
```

Third Step: After normalization and applying 8 point algorithm, I got F matrix. Then I drew epipolar lines of 8 points. In both views(images). I used 8 points chosen from the second image and estimated fundamental matrix F, I did the same for the second image. The result has been shown here. (To run this portion

of the code you need computer vision toolbox, if you do not have it, this function of the code ('lineToBorderPoints') would not be recognized by Matlab).

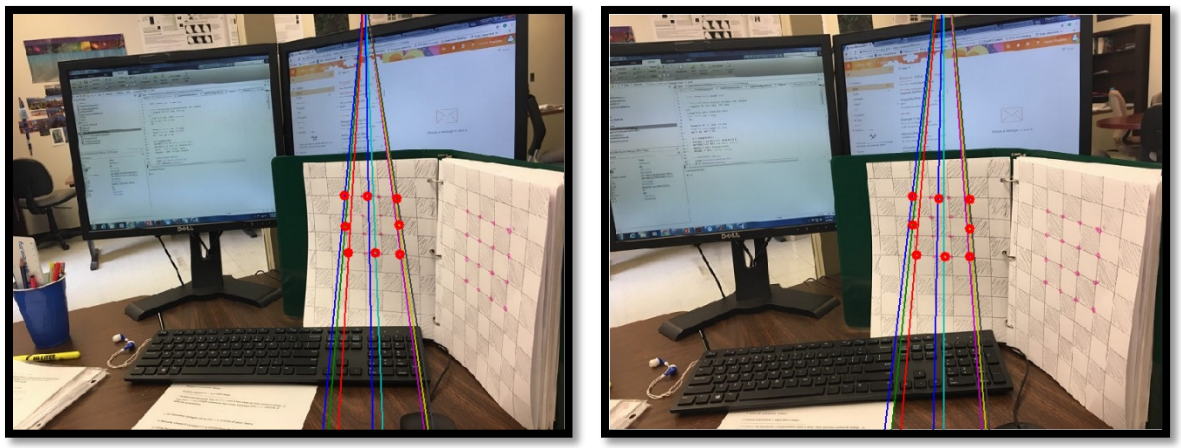

Image1 **Image2**

Figure 2: Epipolar lines through center of the selected points.

As you see from these result (see Figure 2), epipolar lines has been pass through each selected points. Matlab code of this step is:

```
%% Getting Matrix F

F = EightPointAlgorithm(Point_Image1,Point_Image2);

function [ F ] = EightPointAlgorithm( points1, points2)
% getting Essential matrix by 8 pont algorithm
points1=points1(1:8,:);
points2=points2(1:8,:);
[normPoints1,T1] = Normalization( points1);
[normPoints2,T2] = Normalization( points2);
X1= normPoints1(:,1);
Y1= normPoints1(:,2);
X2= normPoints2(:,1);
Y2= normPoints2(:,2);

A = [ X2.*X1 X2.*Y1 X2 Y2.*X1 Y2.*Y1 Y2 X1 Y1 ones(8,1)];
[~,~,V] = svd(A,0);
temp = V(:,end);
F = reshape( temp,[3 3])';
% make it rank two
[U,L,V] = svd(F);
L(3,3) = 0;
L = L / L(1,1);
F = U * L * V';

F = T2' * F * T1;
End

%% EpipolarLines In Image 1
figure
imshow(img1)
hold on
epiLines1 = GenerateEpipolarLines(F',Point_Image2);
edgePoints = lineToBorderPoints(epiLines1,size(img1));
line(edgePoints(:,[1,3])',edgePoints(:,[2,4])','LineWidth',2);
plot(Point_Image1(:,1),
Point_Image1(:,2),'go','LineWidth',4,'MarkerEdgeColor','r')

hold off;

%% EpipolarLines In Image 2 >>>
figure
imshow(img2)
hold on
epiLines2 = GenerateEpipolarLines(F,Point_Image1);
edgePoints = lineToBorderPoints(epiLines2,size(img2));
line(edgePoints(:,[1,3])',edgePoints(:,[2,4])','LineWidth',2);
plot(Point_Image2(:,1),
Point_Image2(:,2),'go','LineWidth',4,'MarkerEdgeColor','r')

hold off;
```

```
function [normPoints, T] = Normalization( points )
% returns a transform for column vectors to normalize the points


% the results with top normalization was not good so I searched for other
% method, this methods results is better
mu = sum(points(:,1:2)) / size(points,1);
centeredPoints = points(:,1:2) - repmat(mu,size(points,1),1);
averageDistance = sum(sqrt(centeredPoints(:,1).^2 + centeredPoints( :,2).^2))
/ size(points,1);
s = sqrt(2) / averageDistance;
T = diag([s s 1]) * [eye(3,2) [-mu 1]'];
normPoints = points;
normPoints(:,3) = 1;
normPoints = normPoints * T';
normPoints = normPoints ./ repmat( normPoints(:,3), 1, 3);
normPoints(:,3) = [];


function [ lines ] = GenerateEpipolarLines(F,points)

homoPoints=[points,ones(size(points,1),1)];
lines=homoPoints*F';
end
```

Last Step: By assumption E = F, I estimated the translation vector tau and the rotation matrix omega. These estimates tell the relative orientation of the two views based on the two images that I captured. Based on the Page 431 of Simon's book, I calculated Omega and Tau :

Estimated Omega:

0.9622 -0.0221 -0.2714

-0.2723 -0.0679 -0.9598

-0.0028 -0.9974 0.0713

Estimated Taue Cross:

-0.0020 -0.0072 -0.0000

0.0091 0.0325 -0.0003

2.2192 7.9102 -0.0305

Matlab code for calculating Tau and Omega:

```
%% Getting Omega and Tau
[ omega, tauCross ] = DecomposeEssential( F );

function [ omega, tauCross ] = DecomposeEssential( E )
% Decomposses Essential matrix to rotation and translation matrix
W=[0 -1 0; 1 0 0; 0 0 1];
[U,L,V]=svd(E);

if L(3,3)> 0.00001
    L=diag([1 1 0]);
end

tauCross=U*L*W*(U');
omega=U*(inv(W))*V;
end
```