EMİR ALAATTİN YILMAZ

EE417

POST-LAB #8 REPORT

Pose Recovery of a Calibrated Camera Through Essential Matrix Using 8-Point

In this lab we will recover the pose of a camera from two images taken from different viewpoints by estimating the Essential Matrix (E) using 8-point algorithm. To do that we will use epipolar geometry.

Procedure:

1. We take a 8 points from XY, XZ and YZ planes.
2. We create vector a and X matrix to estimate E

$$a = [x_1x_2 \quad x_1y_2 \quad x_1z_2 \quad y_1x_2 \quad y_1y_2 \quad y_1z_2 \quad z_1x_2 \quad z_1y_2 \quad z_1z_2]^T$$

$$X = [a_1^T;\ a_2^T;\ \dots\ a_8^T]$$

3. After we create X matrix, we apply SVD method and last column of V gives us Es.
4. We apply SVD to Es for curing and make diag(S)=[$\sigma_1 = \sigma_2 = 1, \sigma_3 = 0$] to estimate Essential Matrix E.
5. Then we calculate epipoles and epipolar lines by the following formulas:

$$l_i^T \mathbf{x}_i = 0 \qquad E\mathbf{e}_1 = 0$$
$$l_i^T \mathbf{e}_i = 0 \qquad \mathbf{e}_2 E^T = 0$$

6. After that, we used following formulas for verification purposes

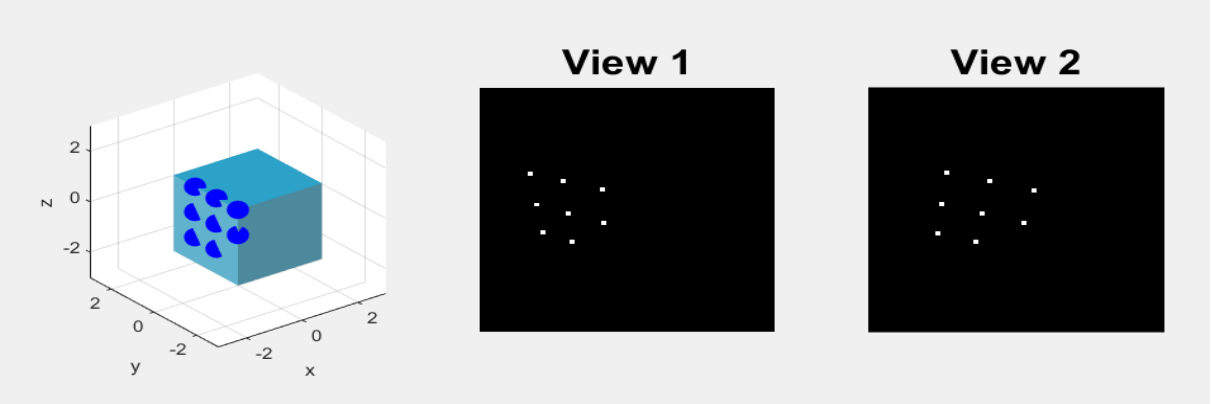
$$l_2 \sim E\mathbf{x}_1$$
$$l_1 \sim E^T\mathbf{x}_2$$

7. Then, to extract rotation and translation matrixes we used the following formulas

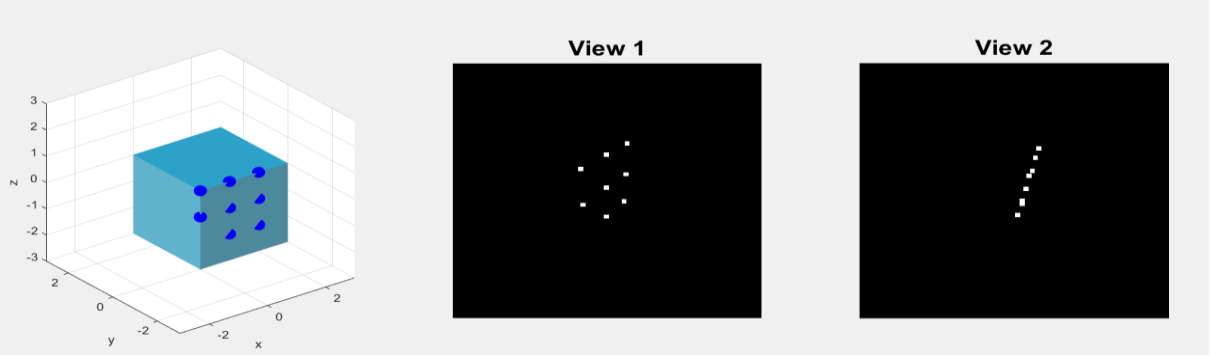
$$(\hat{T}_1, R_1) = (UR_z(\frac{\pi}{2})\Sigma U^T, UR_z^T(\frac{\pi}{2})V^T) \qquad (\hat{T}_2, R_2) = (UR_z(-\frac{\pi}{2})\Sigma U^T, UR_z^T(-\frac{\pi}{2})V^T)$$

8. Lastly, we converted T skewed matrixes into T. (see the code)

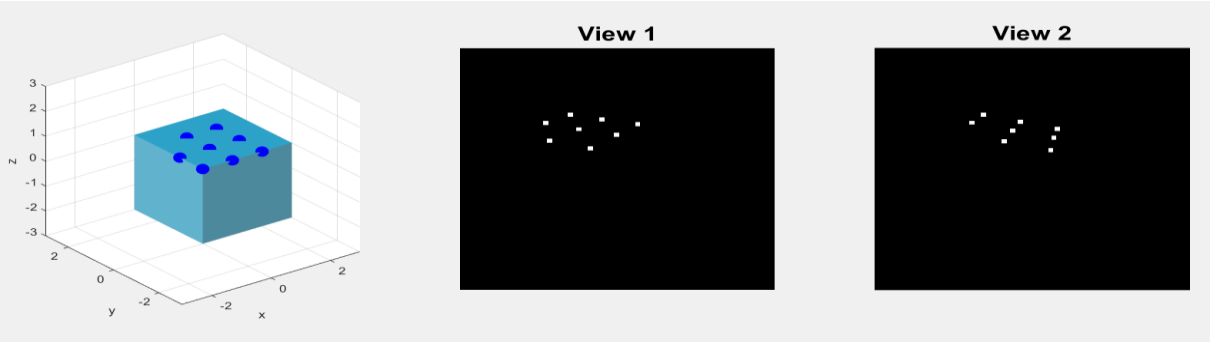
Selecting 8 points on YZ Plane



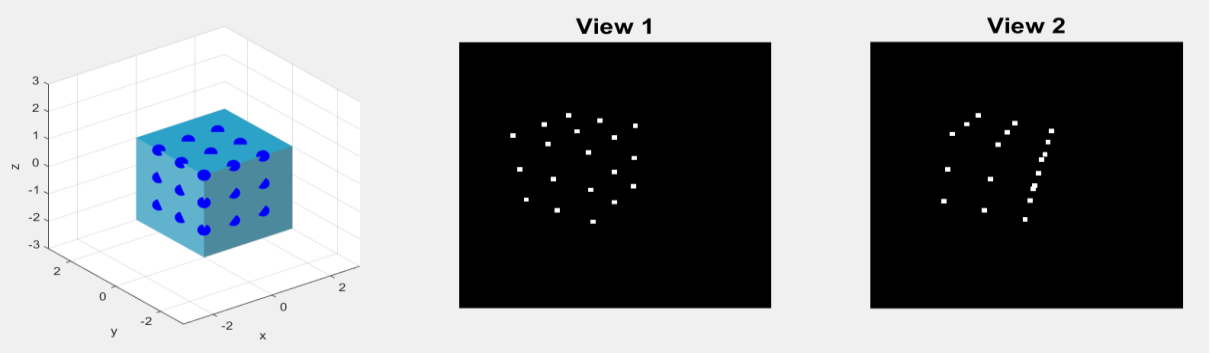
Selecting 8 points on XZ Plane



Selecting 8 points on XY Plane



Selecting All Points



YZ PLANE

```
-------------
True E =
         0   -1.0000         0
   -0.3615         0   -3.1415
         0    3.0000         0
Estimated E =
   -0.1093   -0.8964    0.0162
    0.9074   -0.1359    0.3919
   -0.1201   -0.4171   -0.0217
-------------
True R =
    0.9063         0   -0.4226
         0    1.0000         0
    0.4226         0    0.9063
Estimated R =
    0.6589   -0.0668    0.7492
    0.1761    0.9821   -0.0674
    0.7313   -0.1764   -0.6589
-------------
True T =
     3
     0
     1
Estimated T =
   -0.4293
    0.0675
    0.9006
-------------
```

XZ PLANE

```
-------------
True E =
         0   -1.0000         0
   -0.3615         0   -3.1415
         0    3.0000         0
Estimated E =
   -0.6130   -0.7382   -0.0485
   -0.3949    0.2259   -0.7906
    0.0094    0.3422   -0.3574
-------------
True R =
    0.9063         0   -0.4226
         0    1.0000         0
    0.4226         0    0.9063
Estimated R =
    0.1496   -0.1880    0.9707
   -0.2548   -0.9559   -0.1459
   -0.9553    0.2256    0.1909
-------------
True T =
     3
     0
     1
Estimated T =
   -0.2773
    0.4099
   -0.8690
-------------
```

XY Plane

```
-------------
True E =
         0   -1.0000         0
   -0.3615         0   -3.1415
         0    3.0000         0
Estimated E =
   -0.4719    0.8680    0.1038
   -0.8427   -0.4578   -0.2108
   -0.2187    0.0127   -0.0288
-------------
True R =
    0.9063         0   -0.4226
         0    1.0000         0
    0.4226         0    0.9063
Estimated R =
   -0.8791   -0.4660   -0.1002
    0.4589   -0.8843    0.0859
    0.1286   -0.0295   -0.9913
-------------
True T =
     3
     0
     1
Estimated T =
   -0.1140
   -0.1893
    0.9753
-------------
```

All Points

```
-------------
True E =
         0   -1.0000         0
   -0.3615         0   -3.1415
         0    3.0000         0
Estimated E =
    0.0025   -0.3198    0.0025
   -0.1249    0.0035   -0.9922
   -0.0059    0.9475    0.0054
-------------
True R =
    0.9063         0   -0.4226
         0    1.0000         0
    0.4226         0    0.9063
Estimated R =
    0.9001    0.0037   -0.4356
   -0.0023    1.0000    0.0038
    0.4356   -0.0024    0.9001
-------------
True T =
     3
     0
     1
Estimated T =
    0.9475
    0.0041
    0.3198
-------------
```

DISCUSSION

To get a proper Essential matrix which contains then translation and rotation information, points should be selected from different planes. In the first cases, we get 8 points from YZ, XZ, XY planes but we did not get robust results because we only get the points from one plane.

On the other hand, if we use all the points (19 points from 3 different planes XY,YZ,XZ) we get more robust result. These results are only a scaled version of the real values which can be seen on the previous page. This scale is approximately 3.

Based on these results, I would choose to select all points from different planes method in order to get a more robust Essential matrix.

CODE

lab8.m

```
clear all; close all; clc;
%% Definitions
rng(1);
L = 300;
I1 = zeros(L,L);

f=L;
u0 = L/2;
v0 = L/2;

K = [f 0 u0;
    0 f v0;
    0 0 1];

DEG_TO_RAD = pi/180;

%% World Coordinates
P_W=[0  2   0   1;
    0   1   0   1;
    0   0   0   1;
    0   2   -1  1;
    0   1   -1  1;
    0   0   -1  1;
    0   2   -2  1;
    0   1   -2  1;
    0   0   -2  1;
    1   0   0   1;
    2   0   0   1;
    1   0   -1  1;
    2   0   -1  1;
    1   0   -2  1;
    2   0   -2  1;
```

```
    1   1   0   1;
    2   1   0   1;
    1   2   0   1;
    2   2   0   1];

P_W = P_W';
NPTS = size(P_W,2); %Number of points

%% Visualization
figure;
subplot(1,3,1)
wally = meshgrid(0:0.1:3);
wallz = meshgrid(-3:0.1:0);
wallx = 0*ones(size(wallz,1));
surf(wallx, wally, wallz','FaceColor',(1/255)*[97 178
205],'EdgeColor','none')
hold on
wallx = meshgrid(0:0.1:3);
wallz = meshgrid(-3:0.1:0);
wally = 0*ones(size(wallz,1));
surf(wallx, wally, wallz','FaceColor',(1/255)*[77 137
157],'EdgeColor','none')
wallx = meshgrid(0:0.1:3);
wally = meshgrid(0:0.1:3);
wallz = zeros(size(wally,1)); % Generate z data
surf(wallx, wally', wallz,'FaceColor',(1/255)*[45 162
200],'EdgeColor','none')
plot3(P_W(1,:),P_W(2,:),P_W(3,:),'b.','MarkerSize',36);
axis equal;
grid on
axis vis3d;
axis([-3 3 -3 3 -3 3])
grid on
xlabel('x')
ylabel('y')
zlabel('z')

%% Camera Transformation for View 1
ax = 120 * DEG_TO_RAD;
ay = 0 *DEG_TO_RAD;
az = 60 * DEG_TO_RAD;

Rx = [1 0 0;
      0 cos(ax) -sin(ax);
      0 sin(ax) cos(ax)];
Ry = [cos(ay)  0  sin(ay);
           0   1     0;
      -sin(ay) 0  cos(ay)];
Rz = [cos(az) -sin(az) 0;
      sin(az) cos(az)  0;
      0          0     1];

Rc1 = Rx*Ry*Rz;
Tc1 = [0;0;5];
M = [Rc1 Tc1];
```

```
p1 = K*(M * P_W);
noise1 = 4*rand(3,NPTS)-2;
noise1(3,:)=1;
p1 = p1 + noise1;

u1(1,:) = p1(1,:) ./ p1(3,:);
u1(2,:) = p1(2,:) ./ p1(3,:);
u1(3,:) = p1(3,:) ./ p1(3,:);

for i=1:length(u1)
    x = round(u1(1,i)); y=round(u1(2,i));
    I1(y-2:y+2, x-2:x+2) = 255;
end

subplot(1,3,2), imshow(I1, []), title('View 1', 'FontSize',20);

%% Camera Transformation for View 2
ax = 0 * DEG_TO_RAD;
ay = -25 *DEG_TO_RAD;
az = 0 * DEG_TO_RAD;

Rx = [1 0 0;
      0 cos(ax) -sin(ax);
      0 sin(ax) cos(ax)];
Ry = [cos(ay)  0  sin(ay);
           0   1     0;
      -sin(ay) 0  cos(ay)];
Rz = [cos(az) -sin(az) 0;
      sin(az) cos(az)  0;
      0          0     1];

Rc2c1 = Rx*Ry*Rz;
TrueR = Rc2c1;
Tc2c1 = [3;0;1];
TrueT = Tc2c1;
Hc1 = [Rc1 Tc1; 0 0 0 1];
Hc2c1 = [Rc2c1 Tc2c1; 0 0 0 1];
Hc2 = Hc2c1*Hc1;

Rc2 = Hc2(1:3,1:3);
Tc2 = Hc2(1:3,4);

M = [Rc2 Tc2];

I2 = zeros(L,L);
p2 = K*(M*P_W);

noise2 = 4*rand(3,NPTS)-2;
noise2(3,:)=1;
p2 = p2 + noise2;

u2(1,:) = p2(1,:) ./ p2(3,:);
u2(2,:) = p2(2,:) ./ p2(3,:);
u2(3,:) = p2(3,:) ./ p2(3,:);
```

```
for i=1:length(u2)
    x = round(u2(1,i)); y=round(u2(2,i));
    I2(y-2:y+2, x-2:x+2) = 255;
end

subplot(1,3,3), imshow(I2, []), title('View 2', 'FontSize',20);

t = Tc2c1;
T_skew = [0 -t(3) t(2); t(3) 0 -t(1); -t(2) t(1) 0];
Etrue = T_skew*Rc2c1;

%% Displaying the information
disp('u1: Pixel coordinates in view 1')
u1info = ['Size of u1 is ' num2str(size(u1,1)) 'x'
num2str(size(u1,2))];
disp(u1info)
disp('u2: Pixel coordinates in view 2')
u2info = ['Size of u2 is ' num2str(size(u2,1)) 'x'
num2str(size(u2,2))];
disp(u2info)
disp('-------------')
%% Lab#8 Assignment starts here.
%% Transform pixel coordinates and construct X matrix using
Equations 1 and 2

b1 = inv(K)*u1;
b2 = inv(K)*u2;

x1 = b1(1,1:19);
y1 = b1(2,1:19);
z1 = b1(3,1:19);

x2 = b2(1,1:19);
y2 = b2(2,1:19);
z2 = b2(3,1:19);

a = [x1.*x2; x1.*y2; x1.*z2; y1.*x2; y1.*y2; y1.*z2; z1.*x2; z1.*y2;
z1.*z2];
X = [];
for i = 1:1:19
    X = [X; transpose(a(:,i))];
end

%% Estimate E, cure it and check for Essential Matrix
Characterization

[U S V] = svd(X'*X);
Es = V(:,end);
Es = reshape(Es,[],3);

[U S V] = svd(Es);
S = zeros(3) + diag([1 1 0]);
Ecure = U*S*V';

%% Find epipoles and epipolar lines
e1 = null(Ecure);
e2 = null(Ecure');
```

```
%% Verify epipoles and epipolar lines
l1 = Ecure'*b2;
l2 = Ecure*b1;

ver11 = l1'*b1;
ver12 = l2'*b2;

ver21 = l2'*e1;
ver22 = l2'*e2;

%% Recover the rotation and the translation
az = 90 * DEG_TO_RAD;

Rz = [cos(az) -sin(az) 0;
      sin(az) cos(az)  0;
      0          0     1];
T1skewed = U*Rz*S*U';
R1 = U*Rz'*V';

az = 270 * DEG_TO_RAD;
Rz = [cos(az) -sin(az) 0;
      sin(az) cos(az)  0;
      0          0     1];
T2skewed = U*Rz*S*U';
R2 = U*Rz'*V';

estT1 = [T1skewed(3,2); T1skewed(1,3); T1skewed(2,1)];
estT2 = [T2skewed(3,2); T2skewed(1,3); T2skewed(2,1)];


%% Compare your results with ground truth
disp('True E =')
disp(Etrue)
disp('Estimated E = ')
disp(Ecure);
% disp(your estimated E variable)
disp('-------------')

disp('True R =')
disp(TrueR)
disp('Estimated R = ')
disp(R1);
% disp(your estimated R variable)
disp('-------------')

disp('True T =')
disp(TrueT)
disp('Estimated T = ')
disp(estT1);
% disp(your estimated T variable)
disp('-------------')
```