PROGRAMMING ASSIGNMENT
REAL TIME AUTONOMOUS SYSTEMS

OSWAL VINIT JITENDRA (B18ME040)
TARUN MEENA (B18ME056)

Problem

Consider two images I1 (“im1.jpg”) and I2 (“im2.jpg”) of a static scene captured from a single camera with the given intrinsic camera matrix K (“Intrinsic Matrix K.txt”).

- Find a set of ground-truth correspondences {(pi, p0i)} n i=1 using any of the existing implementations. Ensure that there are at least n = 100 true correspondences. Assume that the world-coordinate system is aligned with the coordinate-system of the camera location.
- Implement the algorithm taught in the class to find the Essential matrix E.
- Decompose the obtained Essential matrix E into the camera motion rotation matrix R and the translation vector t.
- Let Pi be the corresponding 3D point for the pixel pair (pi, p0i). Find Pi ∀ i ∈ {1, 2, . . . , n} using the triangulation approach learned in the class.
- Plot the obtained Pi, ∀ i ∈ {1, 2, . . . , n} and the camera center t.

Solution Approach

System Settings

1. MATLAB R2020b
2. Computer Vision Toolbox (Add-on)

Ground Truth Correspondences

We load the two JPG images into our MATLAB environment and convert them to grayscale for the ground truth correspondence calculations.

We find the features in both the images using the **detectHarrisFeatures()** function available in the computer vision toolbox in MATLAB. This function detect unique features in the images like corners, intensity changes, etc. and stores their locations in a matrix.

We then extract these features using the **extractFeatures()** function and use the **matchFeatures()** function to find the pairs of correspondences between the two images.

Then we extract the locations of the corresponding pairs from the data and store them in the corresponding matrices for image 1 and 2.

```
% Given data
% Intrinsic Matrix
load Intrinsic_Matrix_K.txt
K = Intrinsic_Matrix_K;


% Q1. Ground truth corresponences calculations
I1 = rgb2gray(imread('im1.jpg'));
I2 = rgb2gray(imread('im2.jpg'));

points1 = detectHarrisFeatures(I1);
points2 = detectHarrisFeatures(I2);

[features1,valid_points1] = extractFeatures(I1,points1);
[features2,valid_points2] = extractFeatures(I2,points2);

indexPairs = matchFeatures(features1,features2);

matchedPoints1 = valid_points1(indexPairs(:,1),:);
matchedPoints2 = valid_points2(indexPairs(:,2),:);

m1 = matchedPoints1.Location;
m2 = matchedPoints2.Location;
```

Essential Matrix “E”

We first convert the points in the ground truth correspondence matrices calculated earlier into a homogeneous coordinate system by adding an extra column of ones at the end of the matrices.

We then use these homogeneous coordinate matrices to calculate the concatenated A matrix in the equation below.

$$\mathbf{E} = \mathbf{U}\mathrm{diag}\left(\begin{bmatrix}\sigma & \sigma & 0\end{bmatrix}\right)\mathbf{V}^{\top} \text{ with } \sigma = \frac{\sigma_1 + \sigma_2}{2}.$$

We then use the **svd()** function to find out the singular values for the calculation of elements of the essential matrix. From the values returned by the function, the least 9 values are chosen for the formation of the 3x3 Essential matrix.

We then apply the condition for the essential matrix by first decomposing it using the **svd()** function and then plugging in the modified values of sigma to get the updated E matrix.

```
% Q2. Essential matrix "E" calculations

% m1 and m2 are ground truth correspondence matrics
s = length(m1);
m1=[m1(:,1) m1(:,2) ones(s,1)];
m2=[m2(:,1) m2(:,2) ones(s,1)];

x1=m1'; x2=m2';
x1=[x1(1,:)' x1(2,:)'];
x2=[x2(1,:)' x2(2,:)'];

A=[x1(:,1).*x2(:,1) x1(:,2).*x2(:,1) x2(:,1) x1(:,1).*x2(:,2) x1(:,2).*x2(:,2) x2(:,2) x1(:,1) x1(:,2), ones(s,1)];

[U D V] = svd(A);
E=reshape(V(:,9), 3, 3)';

[U D V] = svd(E);
d = (D(1,1) + D(2,2))/2;
E=U*diag([d d 0])*V';
disp(E)
```

Rotation “R” and Translation “t” matrices

We use the U, V, D features of the Essential matrix calculated in the previous question for the calculation of the “R” and “t” matrices. We calculate the values for both the cases viz. Rotation by +180 degrees (+pi/2) and -180 degrees (-pi/2)..

Hence, we get two values for “R” matrix and two values for the “t” vector each.

We then use these values to calculate the extrinsic and the camera matrices for both the cameras.

The Rotation matrix for the first camera is Identity matrix and the translation matrix is null vector since we are considering the camera coordinate system and the world coordinate system are the same.

For the matrix for the second camera, we get four cases. We then choose the best matrix by hit and trial method.

Then we calculate the camera matrices using these values.

$$
\begin{aligned}
\mathbf{R} &= \mathbf{U}\mathbf{R}_z^\top\left(\pm\frac{\pi}{2}\right)\mathbf{V}^\top \\
[\mathbf{t}]_\times &= \mathbf{U}\mathbf{R}_z\left(\pm\frac{\pi}{2}\right)\Sigma\mathbf{U}^\top.
\end{aligned}
$$

```
R1 = U*transpose(Rz1)*transpose(V);
R2 = U*transpose(Rz2)*transpose(V);

% tx1 = U*Rz1*D*transpose(U);
% tx2 = U*Rz2*D*transpose(U);

% t1 = tx1(:,3);
% t2 = tx2(:,3);
t1 = U(:,3);
t2 = -U(:,3);

% Calculate the corresponding camera matrices for camera 1 and camera 2

% The world co-ordinate system is assumed to be at first camera center
P1 = [eye(3) zeros(3,1)];

% Calculating for rotation and translation w.r.t. camera 1
% Choosing the correct matrix by hit and trial of the 4 combinations

% P2 = [R1 t1];
% P2 = [R1 t2];
P2 = [R2 t2];
% P2 = [R2 t1];

% calculating camera matrics
M = K*P1 ;
N = K*P2 ;
```

3D point calculations by Triangulation approach

The ground correspondence matrices and the camera matrices are used for the calculation of the 3D points using the triangulation approach.

We calculate the 3D coordinates for all the points using a for loop and calculating the values using SVD.

3D points Plotting

The 3D points calculated by the triangulation approach are plotted using the **plot3()** function in MATLAB..

Results

1. **Essential Matrix E**

[-0.4736   -0.1283   0.0032

0.0926   0.0251   0.0001

-0.0032   -0.0003   -0.5000]

2. **R and t matrices**

R1 = [0.0567 -0.1822   -0.9816

0.2568   -0.9474   0.1907

-0.9647   -0.2628   0.0069]

R2 = [0.0436   -0.1880   0.9811

0.2565   -0.9470   -0.1928

0.9655   0.2601   0.0069]

t1 = [-0.1918

-0.9814

-0.0014]

t2 = [0.1918

0.9814

0.0014]

3. 3D points graph