Two View Sparse Reconstruction

By: Gulshan Kumar

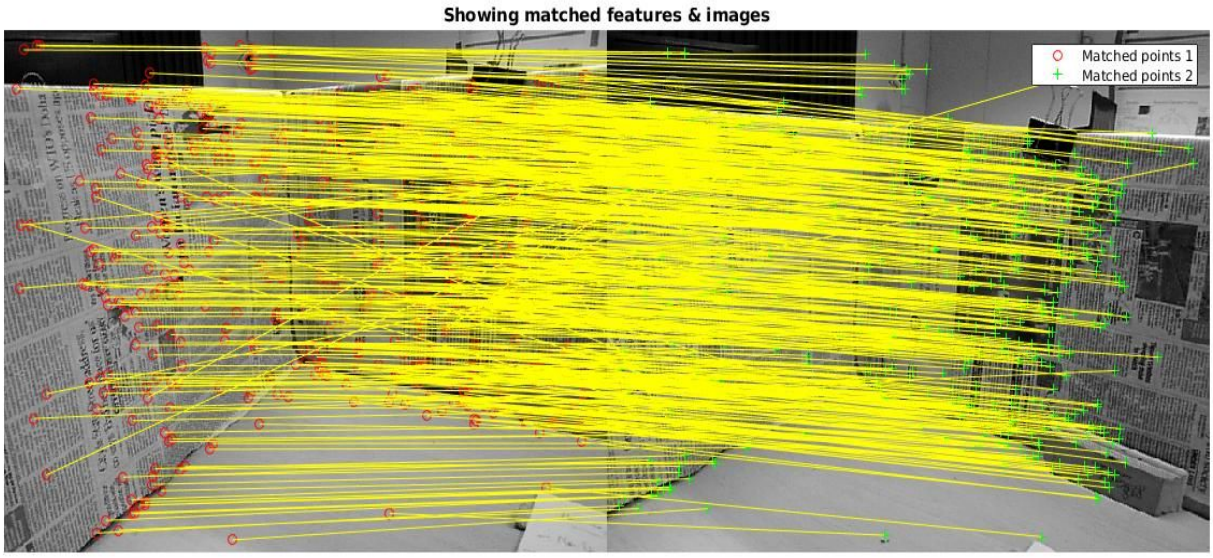
Calibration Matrix:

$$K = \begin{pmatrix} 558.7087 & 0.0000 & 310.3210 \\ 0.0000 & 558.2827 & 240.2395 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix}$$

1. Extract corresponding points:

Read the two images and extract globally unique interest points (SURF)
and establish correspondences between the points of each image. The function used is `detectSURFFeatures(img1,'Metric Threshold',10)`

The value 10 for Metric Threshold variable allows for a large number of not so strong features to be detected and helps us to get rid of degeneracy caused when all features lie on the same plane



Metric Threshold = 10

2. Preconditioning the system:

Normalize the 2D points of each image to avoid ill conditioning of system used for Fundamental matrix estimation. To normalize subtract the mean from the points and scale them such that the average distance to the points is equal sqrt(2).

Function for this is

```
function [normPts2D, T] = normalize2DPoints(pts2D)
```

3. Compute Fundamental matrix:

Estimate the Fundamental matrix (F) using the normalized 8-point algorithm in a **RANSAC framework.**

```
function [norm_F] = estimateFundamentalMatrixRANSAC(corresponding_pts1,
corresponding_pts2)
```

Note: The rank of the F matrix is 2 and it could be so that due to noise or other factors the estimated F matrix is full rank i.e. rank3. In order to make the F matrix as rank 2 matrix do the following steps:

```
[u d v] = svd(F);
new_d = diag([d(1,1) d(2,2) , 0]);
F = u * new_d * v' ;
```

Fundamental Matrix Matrix:

[0.0000 0.0000 -0.0124
-0.0000 0.0000 -0.0680
0.0153 0.0582 1.0000] (3*3)

4. Compute rigid body transform between cameras:

First denormalize the estimated F matrix and compute compute the Essential matrix (E) using the calibration matrix and the F matrix. Essential matrix, just like F matrix, is a rank 2 matrix with both its singular values same. So again due to noise or other factors, the E matrix might not have above properties. So to convert E matrix into a valid Essential matrix do the following:

```
[u d v] = svd(E);
new_d = diag([(d(1,1)+d(2,2))/2, (d(1,1)+d(2,2))/2 , 0]);
E = u * new_d * v' ;
```

Essential Matrix:

[**0.1176** **0.9296** **-0.0719**
-0.6167 **0.1684** **-3.2072**
0.4608 **3.0998** **0.1034**] (**3*3**)

Decompose the E matrix into R and t using the function which has been provided.

```
function [R t] = decomposeEssentialMatrix(E, pts2D_1, pts2D_2, K)
```
(where, `E = K'*F*K` and `F = T2*F_norm*T1`)

Rotational Matrix:

[**0.9885** **-0.1177** **0.0951**
0.1154 **0.9929** **0.0299**
-0.0979 **-0.0186** **0.9950**] (**3*3**)

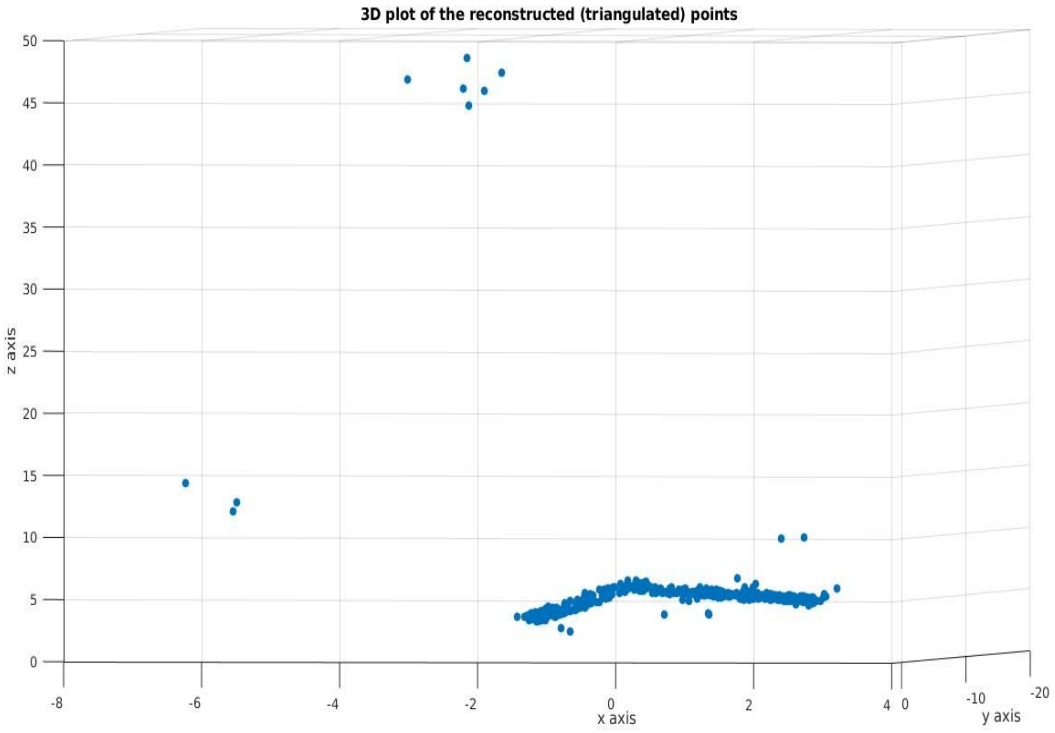
Translation Matrix:

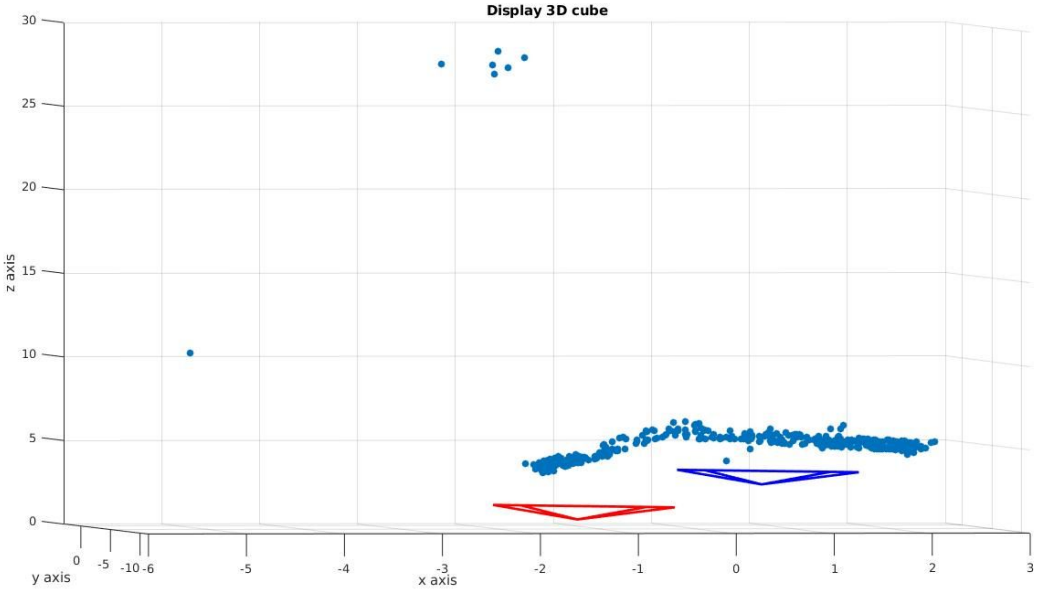
[**-0.9579**
0.0307
0.2856] (**3*1**)

5. Reconstruct the scene:

A function to algebraically triangulate (reconstruct) the interest points:

```
function [pts3D] = algebraicTriangulation(pts2D_1, pts2D_2, ProjMat_1, ProjMat_2)
```
(Where `ProjMat_1 = K*[eye(3,3) [0 0 0]']` and `ProjMat_2 = K*[R t]`)


3D plot of the reconstructed (triangulated) points
z axis
x axis
y axis


Display 3D cube
z axis
y axis
x axis

All functions are documented and arranged structurally.

Normalizing 2D points:

When we directly use image coordinates in the linear system to solve for fundamental matrix estimates, SVD has numerical stability issues. To address this, we typically transform the points (coordinates of features) in each image such that they are centered at the origin and their mean distance from the origin is sqrt(2).

A step-by-step approach will be something similar to the following.

1. Compute the mean of all points. Call this *mu* = [*mu*(1); *mu*(2)].
2. Compute the average distance of all points from the origin. Call this *d*.
3. Compute the scaling factor. Call this *scale*. Mathematically, *scale* = *sqrt(2) / d*
4. Construct a homogeneous transform matrix *T* that will be applied to each point. In Matlab notation, *T* = [*scale*, 0, -*scale* * *mu*(1); 0, *scale*, -*scale* * *mu*(2); 0, 0, 1]
5. (Pre-)Multiply each image feature (in homogeneous coordinates) by *T* to obtain the transformed image coordinates.
6. Perform fundamental matrix estimation using these transformed image coordinates.
7. **Important**: You will have to *undo* this transformation after computation of F. Specifically, assume that your F computation is from the equation x2^T * F * x1 = 0 (T here is transpose, not the *T* matrix from step 4). If *T1* and *T2* are the normalizing transform matrices for images 1 and 2 respectively, then the fundamental matrix F obtained from solving the normalized system must be modified as *T2*^(transpose) * F * *T1*, to obtain the fundamental matrix for the original system. This is a more numerically stable way to compute F.