MM 805 Computer Vision and 3DTV

Programing Assignment

Code base can be found on: https://github.com/dharanUoA/MM805-Programing-Assignment

1. 3D Transformations

a. Transformation Matrix for 1.1

```
Transformation matrix:
     1     0     0     4
     0     1     0    -5
     0     0     1     6
     0     0     0     1
```

b. Transformation Matrix for 1.2

```
Transformation matrix:
    0.6017   -1.2543    1.6526   -0.0000
    2.0746    0.3870   -0.4616         0
   -0.0281    1.7157    1.3124   -0.0000
         0         0         0    1.0000
```

c. Transformation Matrix for 1.3

```
Transformation matrix:
    0.6000   -0.8000         0    0.0000
    0.8000    0.6000         0         0
         0         0    1.0000    0.0000
         0         0         0    1.0000
```

d. Matlab Codebase:

```
Unset
function [transformationMatrix, status]= findTransformationMatrix(A, B)
  [tformEst, ~, status] = estgeotform3d(A, B,"similarity");
  transformationMatrix = zeros(4,4);
  if (status == 0)
    disp("Transformation matrix:")
    disp(tformEst.A);
    transformationMatrix = tformEst.A;
  elseif (status == 1)
    disp("Inputs do not contain enough points");
  else
    disp("Not enough inliers found");
  end
end
```

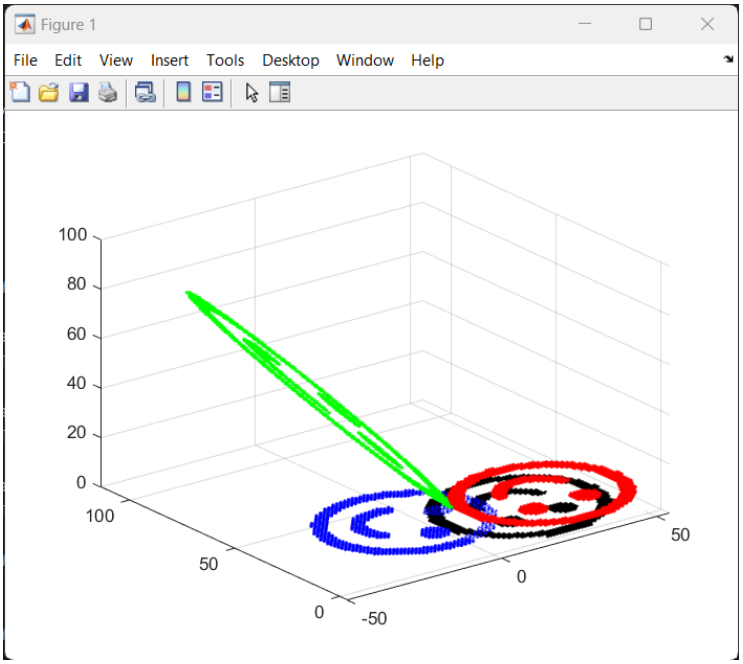
e. Code Explanation

- I used the estgeotform3d function to estimate the geometric transformation between two 3D points (A and B). The resulting tformEst contains the estimated transformation.
- We were suppose to find scaling, rotation, and translation transformation. Hence, I used "similarity" in function parameter.

2. Experiments

Results after modifying the program. Applied transformation matrices captured earlier and plot the data.

- `Red` dots represents transformed points after applying transformation matrix for 1.1
- `Green` dots represents transformed points after applying transformation matrix for 1.2
- `Blue` dots represents transformed points after applying transformation matrix for 1.3



Updated Matlab Codebase

Unset

```
im = double (imread('./smile.png'));
[row_im, column_im] = size(im);

figure
set (gcf, 'Color', [1 1 1])
for x = 1:column_im
  for y = 1:row_im
    if status1 == 0
      temp1 = transformationMatrix1 * [x;y;1;1];
    end
    if status2 == 0
      temp2 = transformationMatrix2 * [x;y;1;1];
    end
    if status3 == 0
      temp3 = transformationMatrix3 * [x;y;1;1];
    end
    if im(y, x) == 255
      plot3(x,y,1,'w.')
      grid on
    else
      plot3(x,y,1,'k.')
      if status1 == 0
```

```
        plot3(temp1(1), temp1(2), temp1(3), 'k.', 'Color', 'red');
      end
      if status2 == 0
        plot3(temp2(1), temp2(2), temp2(3), 'k.', 'Color', 'green');
      end
      if status3 == 0
        plot3(temp3(1), temp3(2), temp3(3), 'k.', 'Color', 'blue');
      end
      grid on
    end

    hold on
    drawnow
  end
end
```

Explanation of updation in the above function

```
Unset
    if status1 == 0
      temp1 = transformationMatrix1 * [x; y; 1; 1];
    end
    if status2 == 0
      temp2 = transformationMatrix2 * [x; y; 1; 1];
    end
    if status3 == 0
      temp3 = transformationMatrix3 * [x; y; 1; 1];
    end
```

Applied transformations (transformationMatrix1, transformationMatrix2, transformationMatrix3) found in Section 1, to the image coordinates if the corresponding status is 0 (indicating a successful transformation).

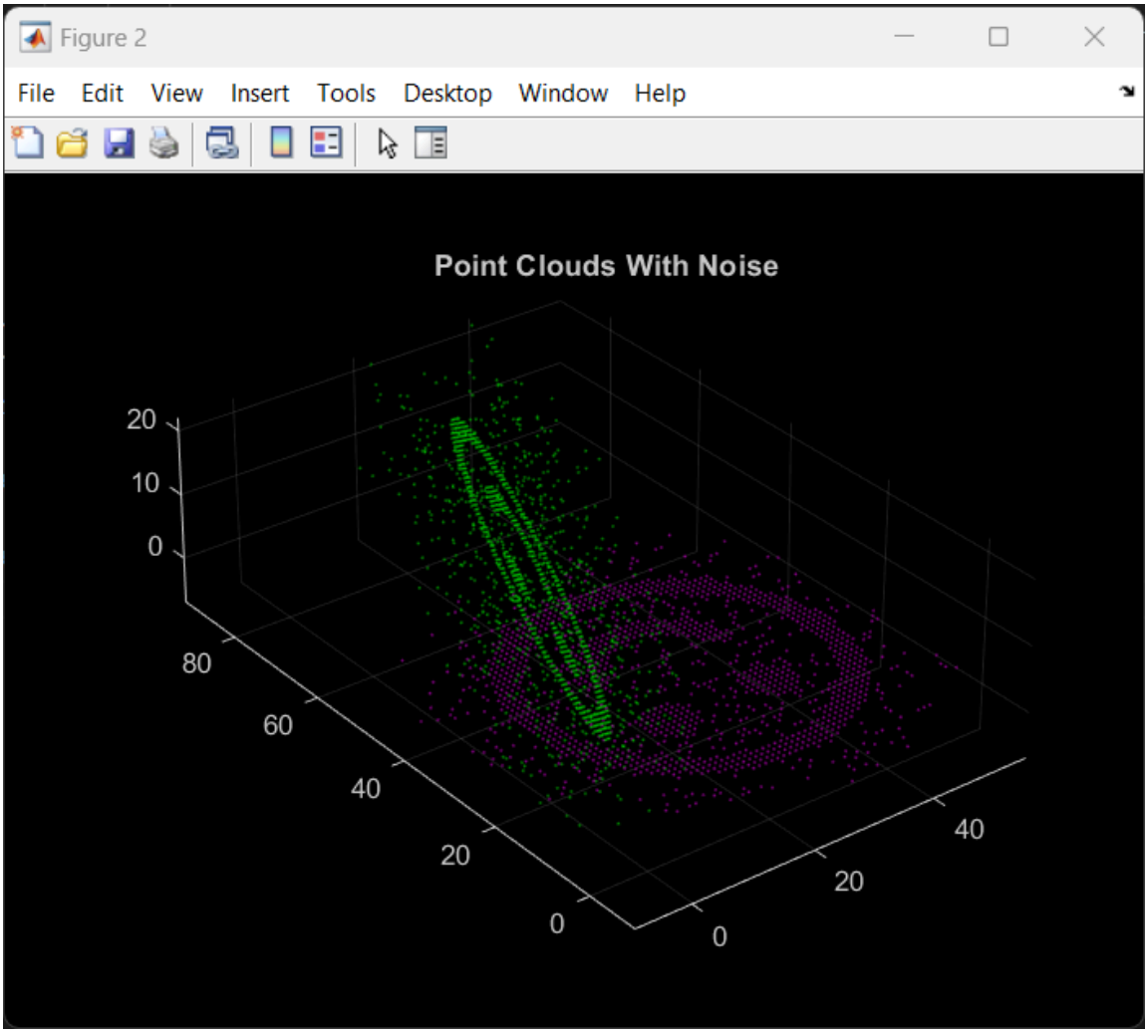
```
Unset
if status1 == 0
  plot3(temp1(1), temp1(2), temp1(3), 'k.', 'Color', 'red');
end
if status2 == 0
```

```
  plot3(temp2(1), temp2(2), temp2(3), 'k.', 'Color', 'green');
end
if status3 == 0
  plot3(temp3(1), temp3(2), temp3(3), 'k.', 'Color', 'blue');
end
```

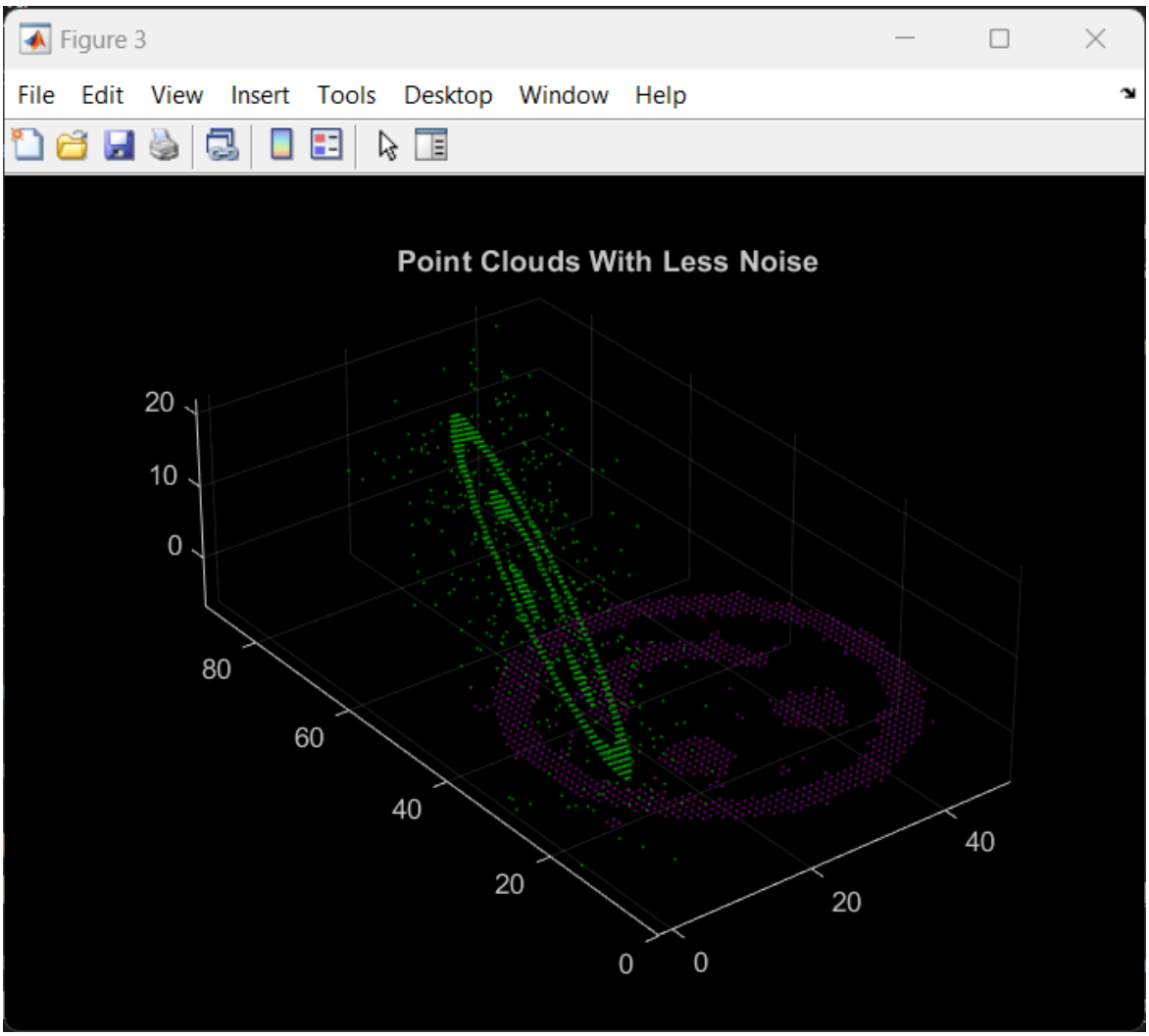
If transformations are applied successfully, plotd the transformed coordinates as additional colored points (red, green, blue).

3. Advance

a. Close to 400 outliers are removed from the noisy data.
b. Positions of points before and after transformation with noisy data



c. Positions of points before and after transformation with reduced noise



d. Code for reducing noise

```
Unset
data_A = readFile('A.txt');
data_B = readFile('B.txt');

ptCloud_A = pointCloud(data_A);
ptCloud_B = pointCloud(data_B);

[ptCloud_A, inliersIndices]= pcdenoise(ptCloud_A, "Threshold", 0.01);
ptCloud_B = pointCloud(ptCloud_B.Location(inliersIndices, :));
```

```
function [data] = readFile(file_path)
  fileID = fopen(file_path, 'r');
  data = textscan(fileID, '%f %f %f %f');
  fclose(fileID);
  data = cell2mat(data);
  data = data(:,1:3);
end
```

Explanation: Applied point cloud denoising to ptCloud_A using the pcdenoise function. "Threshold", 0.01 specifies a threshold for denoising. Points with a distance greater than 0.01 from their estimated normals are considered outliers and hence, removed. Extracted the inliers from ptCloud_B based on the indices obtained from the denoising process done in ptCloud_A.

e. Capturing transformation matrix in less noisy data

0.9064	0.1766	0.3837	-9.5249
-0.0867	0.9668	-0.2404	27.7425
-0.4135	0.1846	0.8916	11.9871
0	0	0	1.0000

Function used to find transformation matrix

```
Unset
[tform, ~, ~] = pcregistericp(ptCloud_A, ptCloud_B);
```

I used pcregistericp function, which performs point cloud registration using the Iterative Closest Point (ICP) algorithm. It aligns ptCloud_A to ptCloud_B and estimates the transformation matrix.

f. Visualizing transformation in less noisy data