ECE 661 (Fall 2016) - Computer Vision - HW 7

Debasmit Das

November 1, 2016

1 Depth Image to Point Cloud

For 3D registration such as the ICP, we need to to convert the depth image into point cloud. The depth image consists of a 2D map such that the value of the pixel at that 2D map is the depth (or distance from the sensor) of the part of the object in the pixel. The point cloud on the other hand is a collection of 3D points such that these 3D points represent the surface of an object.

The conversion of depth image to a 3-D point cloud representation is very easy. Let $\mathbf{u} = (x, y, 1)$ be the homogeneous representation of an image pixel coordinate (x, y). Let $\mathbf{D}(\mathbf{u})$ be the depth value at $\mathbf{u}$ (obtained from the depth image), and $\mathbf{K}$ be the *intrinsic camera matrix* of the Kinect 2 sensor. Then, the corresponding 3-D point $\mathbf{P}(\mathbf{u})$ or the (x_p, y_p, z_p) coordinates of the point in the point cloud corresponding to $\mathbf{u}$ is given by

$$\mathbf{P}(\mathbf{u}) = (x_p, y_p, z_p) = \mathbf{D}(\mathbf{u})\mathbf{K}^{-1}\mathbf{u} \tag{1}$$

This is done for both of the depth images given in the assignment

2 Iterative Closest Point Algorithm (ICP)

Let $\mathbf{P}$, $\mathbf{Q}$ be the point clouds for the same object from 2 different view-points. The point clouds are in their own reference frame. The goal is to give a euclidean transformation $[R, t]$ to $\mathbf{Q}$ so that it exists in $\mathbf{P}$'s frame of reference. The goal of ICP is to find this transformation. The steps of ICP are the following -

- The first step in ICP is to solve the correspondence problem between the 2 point clouds $\mathbf{P}$ and $\mathbf{Q}$. Here we find points in $\mathbf{P}$ that have the least euclidean distance to points in $\mathbf{Q}$. A threshold (δ) is selected such that pairs of points having euclidean distance greater than δ is rejected. We choose $\delta = 0.1$

  As a result, we will have point clouds of reduced dimensions $\mathbf{P'}$ and $\mathbf{Q'}$ that have dimensions $3 \times N$, where N is the total number of legal correspondence pairs. The points are arranged in the point clouds $\mathbf{P'}$ and $\mathbf{Q'}$ such that the i^{th} point in $\mathbf{P'}$ corresponds to the i^{th} point in $\mathbf{Q'}$.

- The next step is to estimate the rotation and translation matrices which transform the point cloud $\mathbf{Q}$ to the frame of reference of $\mathbf{P}$.

  For that we need to first calculate the centroid of the two point clouds. Therefore, $\mathbf{P}_c = \sum_i^N \frac{\mathbf{P'}(i)}{N}$ and $\mathbf{Q}_c = \sum_i^N \frac{\mathbf{Q'}(i)}{N}$. After this, the centroids are subtracted from the corresponding point cloud and to form resulting point clouds $\mathbf{M_P}$ and $\mathbf{M_Q}$ which are calculated as follows - $\mathbf{M_P} = \{\mathbf{P'}(i) - \mathbf{P}_c\}$ and $\mathbf{M_Q} = \{\mathbf{Q'}(i) - \mathbf{Q}_c\}$ for $i = 1, 2...N$. The dimensions of $\mathbf{M_P}$ and $\mathbf{M_Q}$ equal $3 \times N$. After that correlation matrix C is computed such

that $C = \mathbf{M_Q}{\mathbf{M_P}}^T$. C should be 3×3 matrix. The rotation matrix R can be estimated by using the SVD Singular value Decmposition (SVD) on C yielding $C = U\Sigma V^T$. The matrix R is given by $R = VU^T$. The translation t is defined as the difference between $\mathbf{P}_c$ and rotated $\mathbf{Q}_c$: $t = \mathbf{P}_c - R\mathbf{Q}_c$.

- Here, we form the transformation matrix T using R and t found in the previous step and apply the transformation to $\mathbf{Q}$. The transformed $\mathbf{Q}$ is give by $\mathbf{Q}_{trans} = T\mathbf{Q}$.

$$T = \begin{bmatrix} R & t^T \\ 0 & 1 \end{bmatrix} \tag{2}$$

  The point cloud $\mathbf{Q}$ is transformed using the transformation matrix T. We should make sure that the homogeneous co-ordinates are converted to physical points.

- The above steps are iterated for the original point cloud $\mathbf{P}$ and $\mathbf{Q}_{trans}$ until the maximum no. of iterations. The maximum no. of iterations for our case is $M = 20$.

3 Observations

Since the Point Clouds are very close we just need very few iterations and very low threshold to register the 2 point clouds

4 Experimental Results

The following parameters are set -
$\delta = 0.1$
$M = 20$

4.1 Depth Images

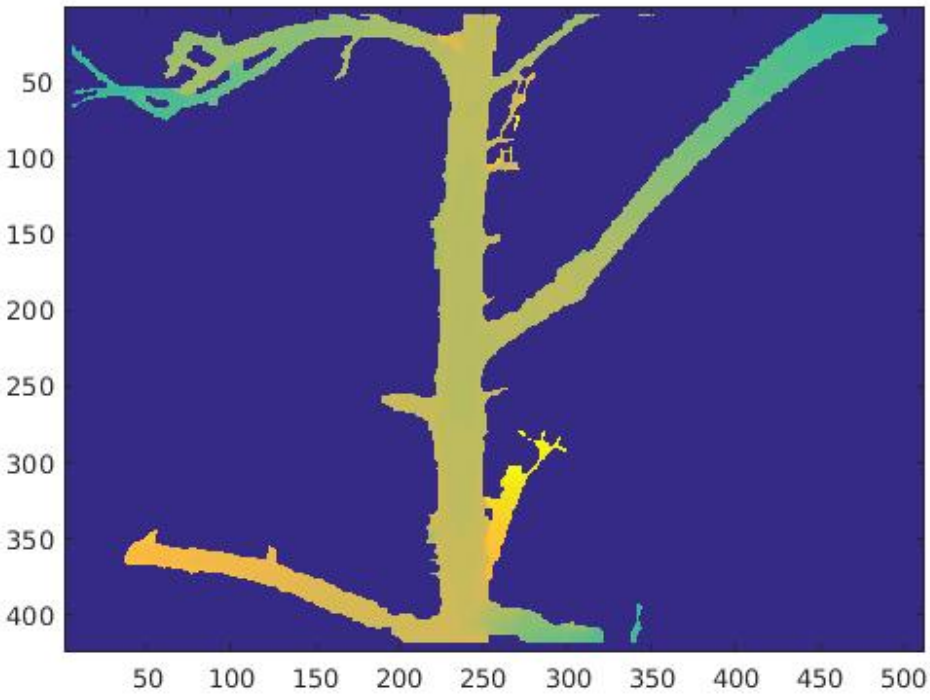

Figure 1: Depth Image for the reference **P**

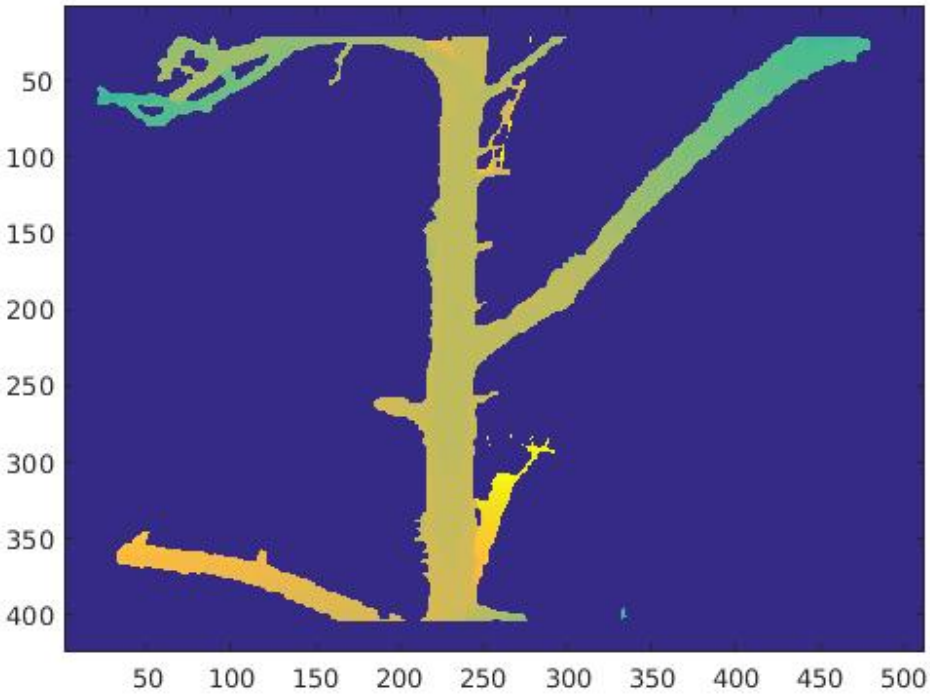

Figure 2: Depth Image for the target **Q**

4.2 Point Clouds

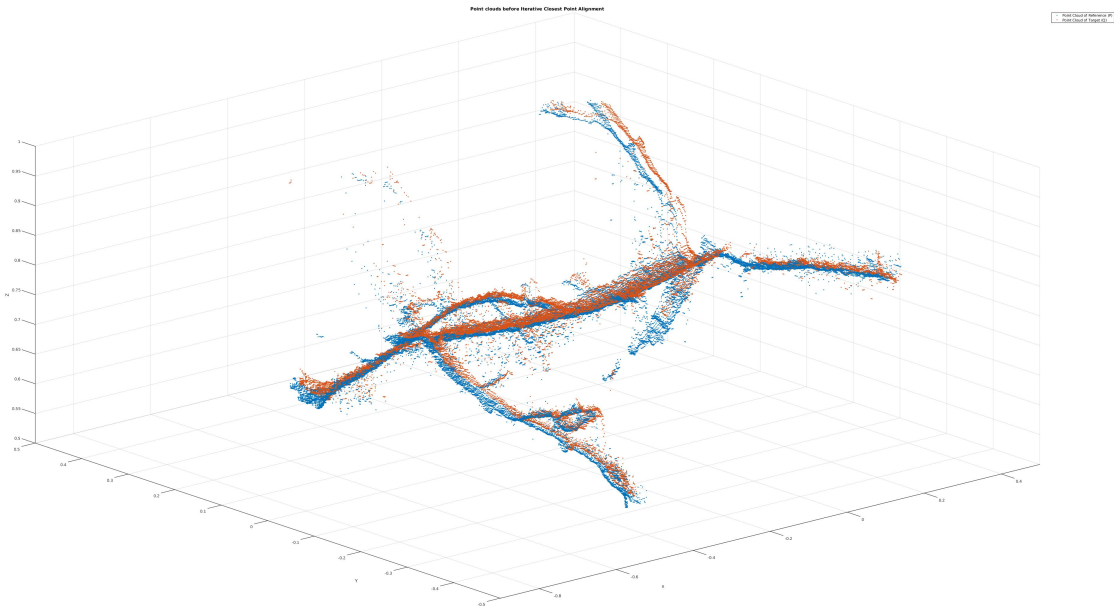

Figure 3: Point Cloud before ICP (Blue - Reference (P) Red - Target (Q))

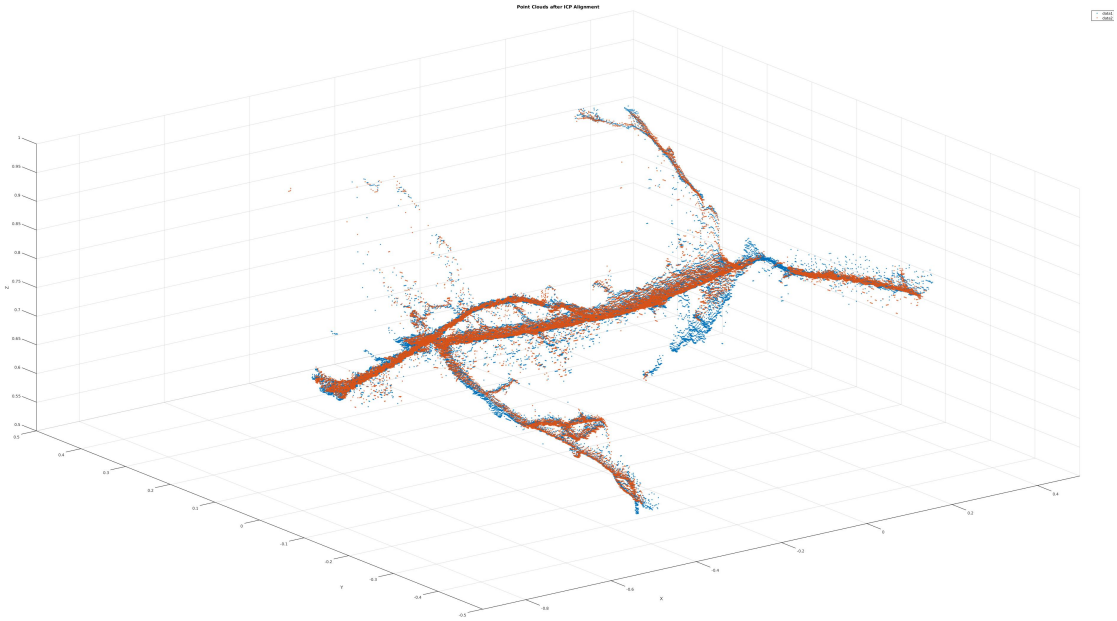

Figure 4: Point Cloud after ICP (20 iterations) (Blue - Reference (P) Red - Target (Q))

Code

The script is in MATLAB and is self-explanatory

ICP Script

```
function Qt = ICP(Q, P, th)

%Output is the transformed target point cloud for one iteration

Pp = [];
Qp = [];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Find the closest points
for k = 1:size(Q,2)
    [D,I] = pdist2(P',Q(:,k)','euclidean','Smallest',1);
    if D < th
        Pp = [Pp,P(:,I)];
        Qp = [Qp,Q(:,k)];
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Estimate Transformation Matrix
N = size(Pp,2);

Pc = sum(Pp,2)/N;
Qc = sum(Qp,2)/N;

Mp = Pp-repmat(Pc,1,N);
```

```
Mq = Qp-repmat(Qc,1,N);

C = Mq*Mp';

[U S V] = svd(C);

R = V*U';
t = Pc - R * Qc;

T = [R t;0 0 0 1];

Qt = T*[Q;ones(1,size(Q,2))];
Qt = Qt./repmat(Qt(4,:),4,1);
Qt(4,:) = [];

end
```

Main Script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read Depth Data
Dp = dlmread('depthImage1ForHW.txt');
Dq = dlmread('depthImage2ForHW.txt');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plotting Depth Image
figure;
imagesc(Dp)
figure;
imagesc(Dq)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Convert Depth Image to Point Cloud
% This is the camera calibration matrix
K = [365 0 256; 0 365 212; 0 0 1];
%This is the threshold for the ICP algorithm
th = 0.1;
%This is the number of iterations
M=20;

w = size(Dp,2);
l = size(Dp,1);

P = [];
Q = [];

for i = 1:w
    for j = 1:l
        if Dp(j,i) ~= 0
            P = [P,Dp(j,i)*inv(K)*[j;i;1]];
        end
```

```
        if Dq(j,i) ~= 0
            Q = [Q,Dq(j,i)*inv(K)*[j;i;1]];
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot Pointcloud Before ICP
figure;
scatter3(P(1,:),P(2,:),P(3,:),'.')
hold on
scatter3(Q(1,:),Q(2,:),Q(3,:),'.')
hold off
xlim([-0.9 0.5])
ylim([-0.5 0.5])
zlim([0.5 1])
view(-80,9)

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Conduct ICP
%

for i = 1:M
    %Do the iterative ICP
    Q = ICP(Q,P,th);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot Pointcloud after ICP
figure;
scatter3(P(1,:),P(2,:),P(3,:),'.')
hold on
scatter3(Q(1,:),Q(2,:),Q(3,:),'.')
hold off
xlim([-0.9 0.5])
ylim([-0.5 0.5])
zlim([0.5 1])
view(-80,9)
```