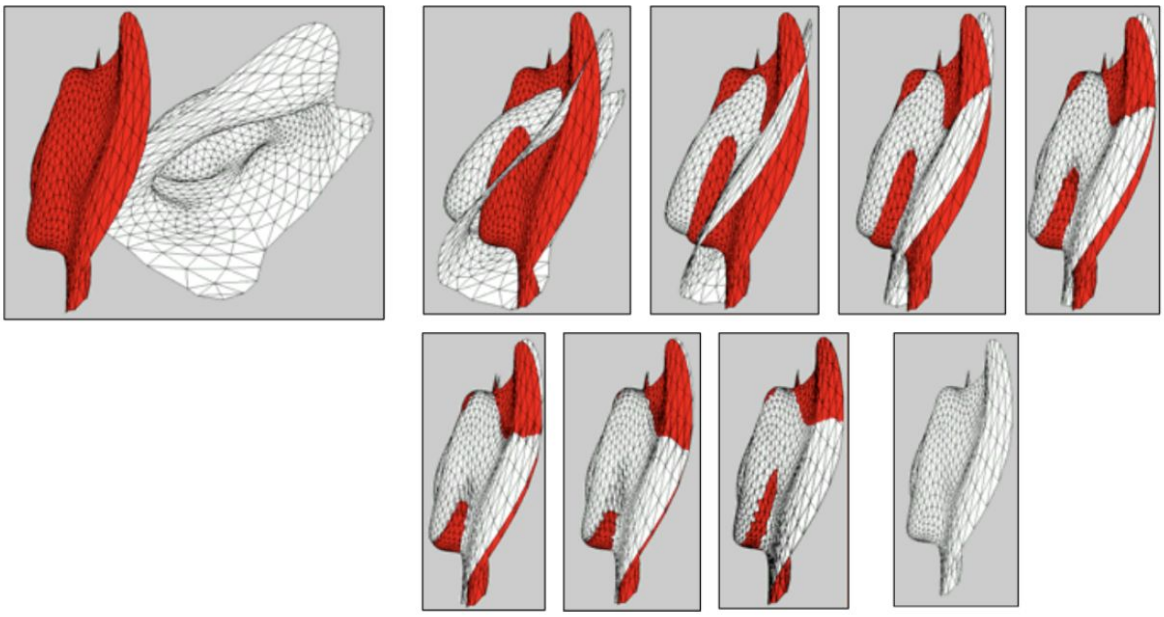


3D Semantic Mapping

CMSC426 Computer Vision Final Project

Mingbo Gu, Hsin Chen, Mingxin Ye

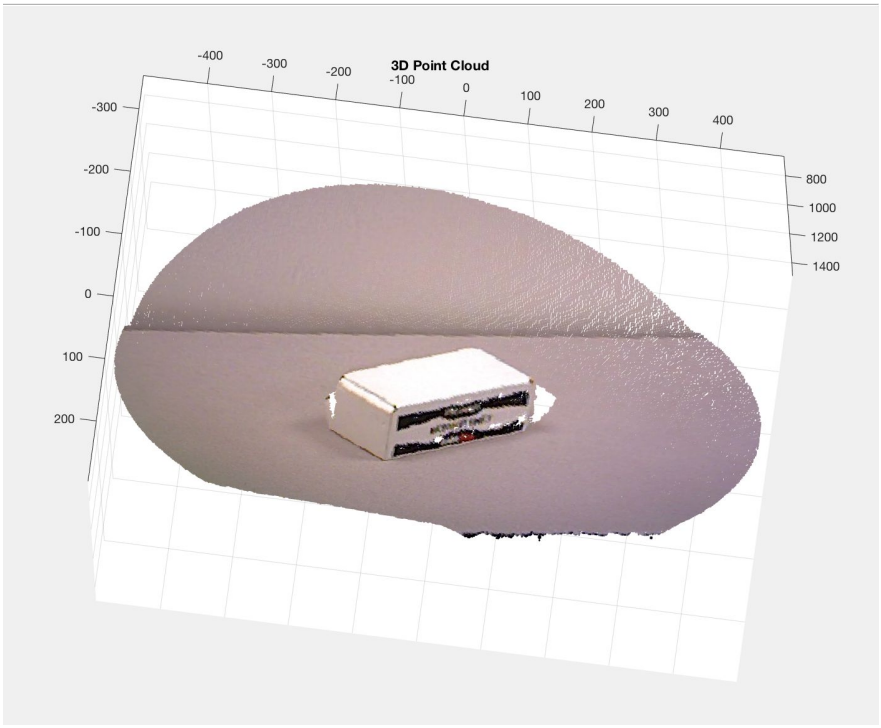
09/04/2018

Introduction

In this project, our goal is to create a 3D semantic mapping for a object given a set of frames that include the object. To do that, we need to remove the background of the object and noises and then combine the result of ICP(Iterative Closest Point) to get the 3D mapping of the object. Below is the step-to-step report of our implementation.

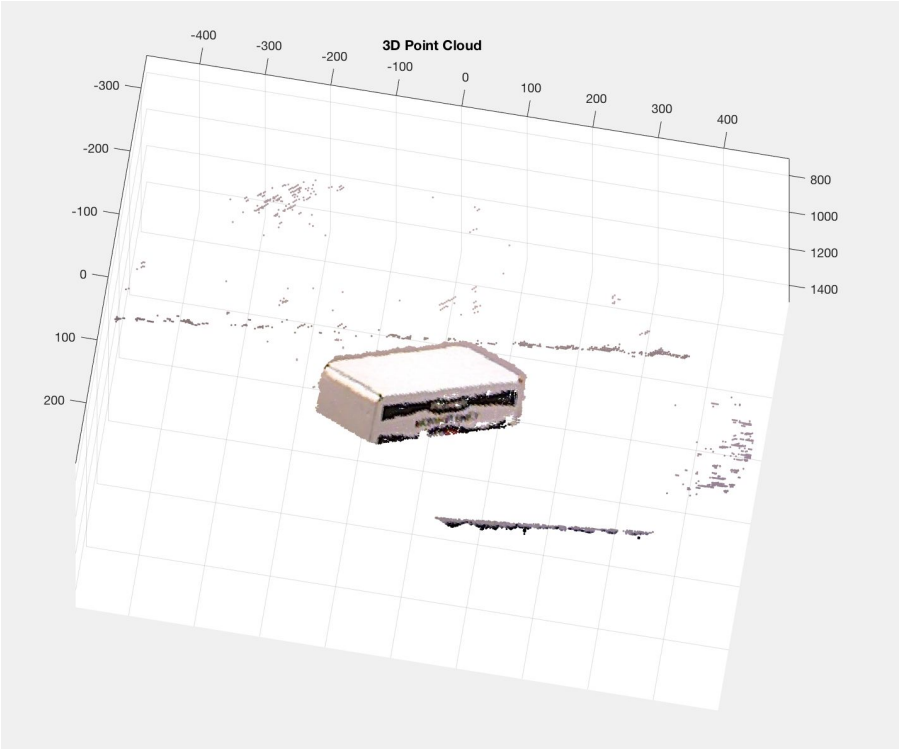
Part 1: Object Segmentation

We are first given a point cloud of the first frame in a series of images. With the point cloud, we want to find a set points that represents the object in the frame. Because the quantity of the points in the point cloud is very large, we first reduce the number of points by cropping the frame (point cloud). Since the center of the object is roughly the center of the frame, We did a sphere cropping from the center (0,0,0). The final image is shown in the picture below:

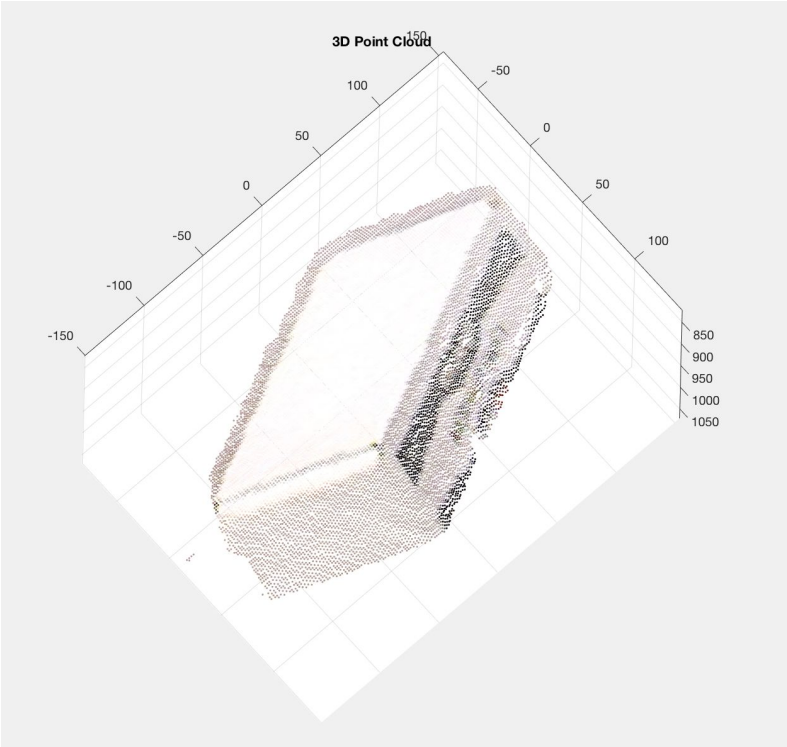


After getting the cropped image, we move forward to remove the background of the frame, using RANSAC. In RANSAC, we select 3 random points and compute the cross product between these points to attain a normal vector of some plane. We then

measure the distance between one of the random points and all the points, and take the dot product of the distances and the normal vector. Doing so gives us the distances between all points and the plane formed by the 3 random points. We threshold the distance to be 7; we keep all the points whose distances to the plane are within the threshold. We go over this process 100 times and take the set of points with the most inliers, the result of which is a set of points that make/ up the plane we aim to remove. We then remove this set of points from all points and we have the something looking like this:

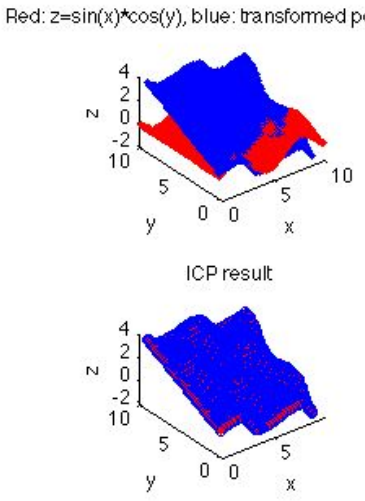


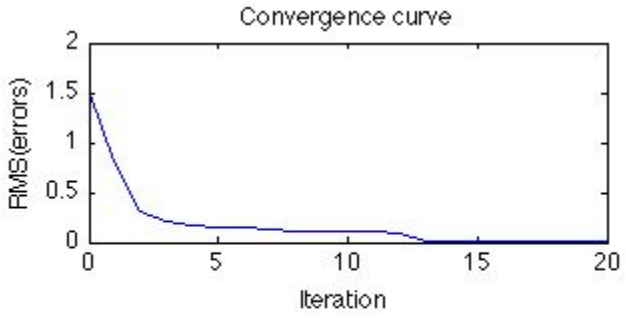
After removing the noise, by removing points not within two standard deviation, we have something that looks a little cleaner:



Iterative Closest Point

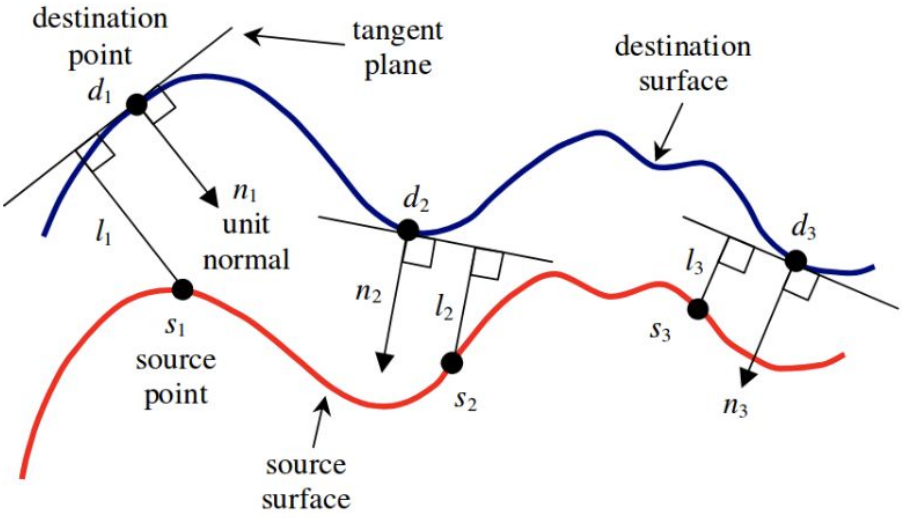
Next, we will be using Point-to-Plane ICP(Iterative Closest Point) to build the modal for our object. Iterative Closest Point is an algorithm used to align two planes, with a set of corresponding points. The process was illustrated below.





The blue plane represents the source points and the red plane represents the destination points. The idea is to reduce the distance between the plane each time we

run the ICP. In this case, “the ”distance” to be minimized is the sum of squared distances between each source point and the tangent plane at its corresponding destination point”(As shown below)



We will run the algorithm for a certain number of iteration until the distance(error) between the plane is smaller than 0.001. The error equation can be found below:

$$E = \sum_{i=1}^{m} [(s_i - d_i) \cdot n_i + t \cdot n_i + r \cdot c_i]^2$$

In the equation above ””

si, di are the source and destination points; ni is the surface’s normal at di;

To estimate the the transformation from frame to frame, we need to solve for t and r in the equation above. To simplify the equation above within each iteration, we keep track on two factors within each iteration. The first factor is the r the rotation and the second factor is t the linear translation. We update these two factors iteration by iteration. To speed up the process, we can solve the equation above by using matrix. We need to construct two matrix b and C as shown below:

$$b = -\sum_{i=1}^{m} \begin{bmatrix} c_{ix}(s_i - d_i) \cdot n_i \\ c_{iy}(s_i - d_i) \cdot n_i \\ c_{iz}(s_i - d_i) \cdot n_i \\ n_{ix}(s_i - d_i) \cdot n_i \\ n_{iy}(s_i - d_i) \cdot n_i \\ n_{iz}(s_i - d_i) \cdot n_i \end{bmatrix}$$

$$C = \sum_{i=1}^{m} \begin{bmatrix} c_{ix}c_{ix} & c_{ix}c_{iy} & c_{ix}c_{iz} & c_{ix}n_{ix} & c_{ix}n_{iy} & c_{ix}n_{iz} \\ c_{iy}c_{ix} & c_{iy}c_{iy} & c_{iy}c_{iz} & c_{iy}n_{ix} & c_{iy}n_{iy} & c_{iy}n_{iz} \\ c_{iz}c_{ix} & c_{iz}c_{iy} & c_{iz}c_{iz} & c_{iz}n_{ix} & c_{iz}n_{iy} & c_{iz}n_{iz} \\ n_{ix}c_{ix} & n_{ix}c_{iy} & n_{ix}c_{iz} & n_{ix}n_{ix} & n_{ix}n_{iy} & n_{ix}n_{iz} \\ n_{iy}c_{ix} & n_{iy}c_{iy} & n_{iy}c_{iz} & n_{iy}n_{ix} & n_{iy}n_{iy} & n_{iy}n_{iz} \\ n_{iz}c_{ix} & n_{iz}c_{iy} & n_{iz}c_{iz} & n_{iz}n_{ix} & n_{iz}n_{iy} & n_{iz}n_{iz} \end{bmatrix}$$

Since Cx = b, We can use mid division to solve for x: x = c\b

$$x = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ t_x \\ t_y \\ t_z \end{bmatrix}$$

After getting the rotation and translation, we can update the modal with following equation:

$$s = R_i * s' + t_i$$
$$R = R_i * R$$
$$t = R_i * t_i + t$$

Ri is the current rotation matrix and t is the current translation matrix.

Finally we will concatenate the transformed source point with the destination point. By concatenating the ICP results from all frames, we will get a estimated 3D model of object.

Part 2: Multi-Object Segmentation

Moving on to part two, now we have more objects in one frame but the goal doesn't change. The idea is to build estimated model for each object in the scene, We need to first segment the objects and run the point cloud of individual objects through ICP in each frame.

Segmentation with SURF Matching & ICP

Since we didn't know how many objects are present in the scene, first thing we need to do is to find each present object. Fortunately, we do know that all the objects in part 2 can also be seen in part 1. Therefore, we know we already have the individual images of all the objects that might be present in the multi-object scene. All we need to do is to run feature matching between these individual images and the Multi-Object image to find the present objects. Here, we use SURF to find the matching points. With these matching points, we can roughly estimate a bounding box over each present object. With the x and y coordinates within these bounding boxes, we can determine a point cloud for each project. With these point clouds, we can then reuse our algorithm for part one to remove unwanted background and use ICP to estimate the a 3D model for each object.