CSE 275 HW2 Problem 2: ICP for 6D Pose Estimation

Arth Shukla
University of California, San Diego
arshukla@ucsd.edu

1. ICP Method

1.1. Formulation

Iterative Closest Point (ICP) [1] requires a source point cloud (e.g. object model) and target point cloud (e.g. transformation applied). In our case, we have the model point cloud which represents pose $[I \mid \mathbf{0}]$, and we have the depth-lifted point cloud which represents the target pose.

However, the depth cloud has heavy occlusions, as the RGB-D camera can only see part of the object. Since the depth cloud is half-occluded, and ICP finds the minimal average distance for *all* points on the model cloud to the depth cloud, we end up with the depth cloud spliced in between the transformed model cloud. An example is seen below in Fig. 1:

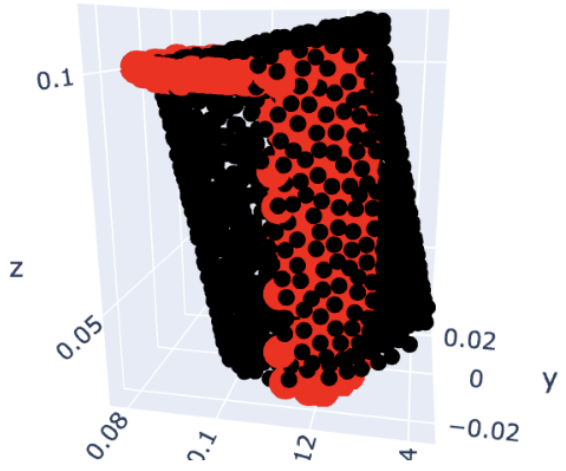

Figure 1. An example of failure when using model as source and depth cloud as target. ICP minimizes distances of *all* points on source cloud to target cloud. So, we end up with this odd result.

However, if we run ICP with the depth cloud as the source, then we avoid this issue since the global optimum only exists when the depth cloud is matched with the correct part of the model. An example is seen below in Fig. 2:

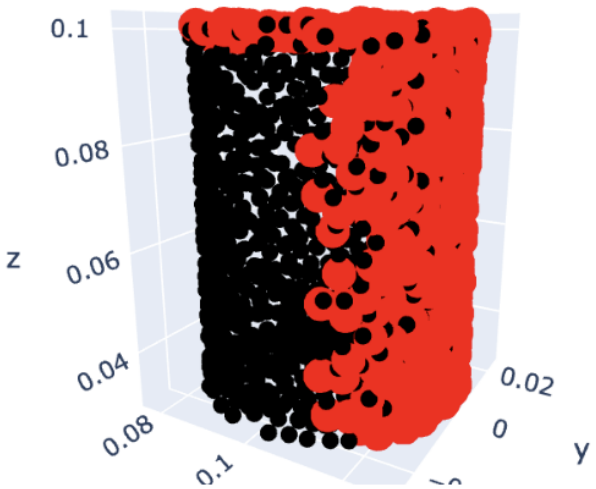

Figure 2. Correct result from using depth cloud as source and model as target. ICP finds the minimal distance between depth cloud points and model points by placing depth cloud *on* model cloud.

So, we run ICP to find the transformation $[R^* \mid t^*]$ which moves the depth cloud to match the model. Then, the estimated pose will be the opposite, $[R \mid t] = [R^{*T} \mid -t^*]$.

1.2. Random Initialization

The goal of using the depth cloud as the source is to have it "cover" part of the model. This only works if we initialize with the depth cloud "facing" the model in the correct way. To accomplish this, we try many random initializations of the point cloud by applying a random rotation and translation.

First, we apply the random translation. For each model point cloud, we define the model center as the mean of all points and the model radius as the distance between the center and the further point in the model. Then, we create a

ball centered at the model center and with radius 2 times the model radius. We then move the depth cloud to random points on this ball by randomly sampling spherical coordinates θ and ϕ.

Next, we apply the random rotation. To do this, we sample a random 3x3 matrix with values over $[0, 1)$ which we convert to a rotation matrix with SVD decomposition. Finally, we apply this random rotation to the depth cloud as well.

An example of this random initialization is seen below in Fig. 3:

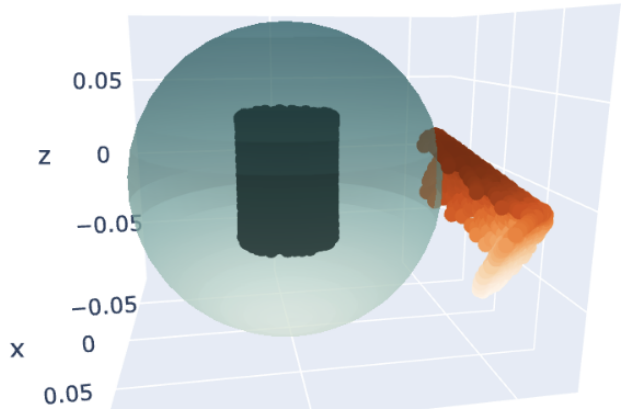

Figure 3. An example of our random initialization strategy. We have a ball centered at the model's center with 2 times model radius. We move our depth cloud to somewhere on the ball, then apply a random rotation before running ICP.

When ICP moves the depth cloud to the model cloud, we can use the closed-form solution to find $[R^* \,|\, t^*]$, and thus $[R \,|\, t]$.

2. Experiments

We consider one initialization as one "attempt". In our experiments, we run versions with 1, 5, and 500 attempts per object. Each attempt gets 1000 max iters each, and early return if the average distance in points changes by less than 10^{-10}. Test set accuracy and runtime results can be seen in Tab. 1. Runtime measures amount of time spent running the algorithm when processing one object at a time (no parallelization between objects/scenes).

The only run which has runtime feasible for real-time pose estimation is 1 attempt per object. However, this attempt only achieves 65.40% accuracy. 5 attempts per object can achieve 90% accuracy with somewhat reasonable runtime, but not fast enough for real-time pose estimation. Running 500 attempts per object acheives excellent pose accuracy of 98.13%, but requires an extremely long runtime, nearly 10 hours.

Certainly optimizations can be made to reduce this overall runtime metric (e.g. parallelizing reading different objects), but this does not change the real-world circumstance which requires fast pose estimation when first observing an object.

Model	Test Acc	Runtime
ICP (1 / Obj)	65.40	**59s**
ICP (5 / Obj)	90.00	5m50s
ICP (500 / Obj)	**98.13**	9h39m46s

Table 1. ICP test set accuracy and runtime for HW2. Best accuracy and best runtime are **bolded**.

References

[1] Paul Besl and H.D. McKay. A method for registration of 3-d shapes. ieee trans pattern anal mach intell. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14:239–256, 03 1992. 1