Assignment 3 - ICP Cloud Registration

Introduction

This assignment involves implementing the Iterative Closest Point (ICP) Cloud Registration algorithm to find the rigid body transformation that aligns two given point clouds.

Implementation

The assignment focuses on the implementation of the ICP main loop.

The `execute_icp_registration` function implements the loop:

- it determines the correspondences between data (source) and model (target) point clouds, via the `find_closest_point` function;
- it computes the new transformation matrix using the correspondences, via the `get_svd_icp_registration` or `get_lm_icp_registration` functions, based on the input mode;
- it updates the current estimation of the total transformation by multiplying the previous estimation by the newly found matrix;
- it checks for convergence by comparing the mean squared error of the alignment to the previous value (using a relative threshold) and stops after a given number of iterations.

Finding correspondences

To find the correspondences, the `find_closest_point` function uses a K-D Tree for efficiency. It considers the current estimation of the transformed source points and the original target points. Looping through all of the source points, if the minimum distance to a target point is within a provided threshold, it registers the correspondence and updates the RMSE computation.

SVD registration

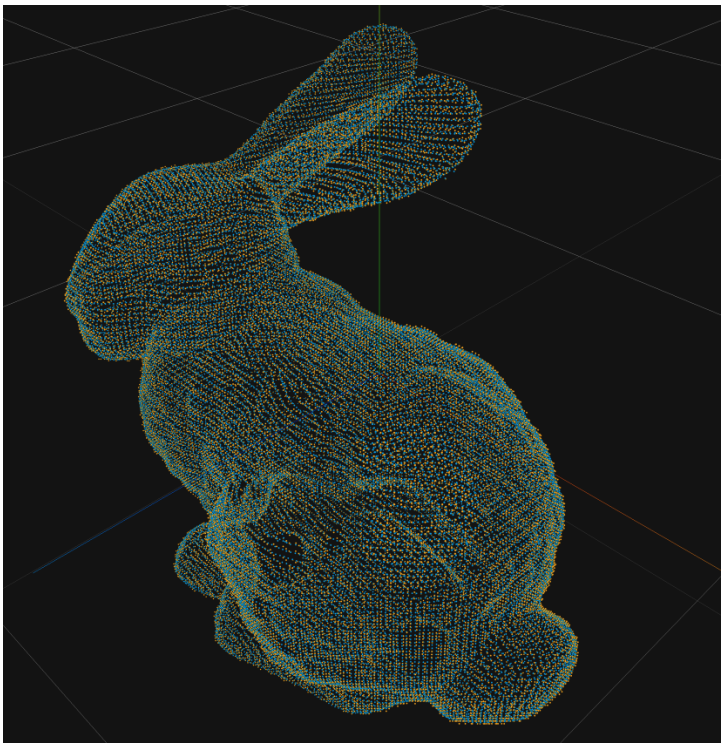
The `get_svd_icp_registration` function finds the transformation (R, t) between the current estimation of the transformed source points and the original target points, using the SVD decomposition. This method decouples the computation of the rotation matrix and translation vector and finds them explicitly.
To find the matrix W to decompose, the source and target centroids are first computed. Starting from the SVD decomposition, R and t are recovered, also accounting for the special reflection case.

LM registration

The `get_lm_icp_registration` function finds the transformation (R, t) between the current estimation of the transformed source points and the original target points, using the Levenberg–Marquardt algorithm. This method implicitly finds the transformation by directly solving the nonlinear optimization problem of minimizing the registration error.
The Ceres problem is given a cost function that computes the residuals between the transformed data and the model point clouds. That is obtained as the difference between the two.
The standard squared loss function is used, and the residuals are not weighted. These simplifications provide very good results, as illustrated in the next section, and no trivial changes have proved to optimize further (Cauchy loss and inverse point-to-point distance weighing don't improve performances while slowing down the execution).

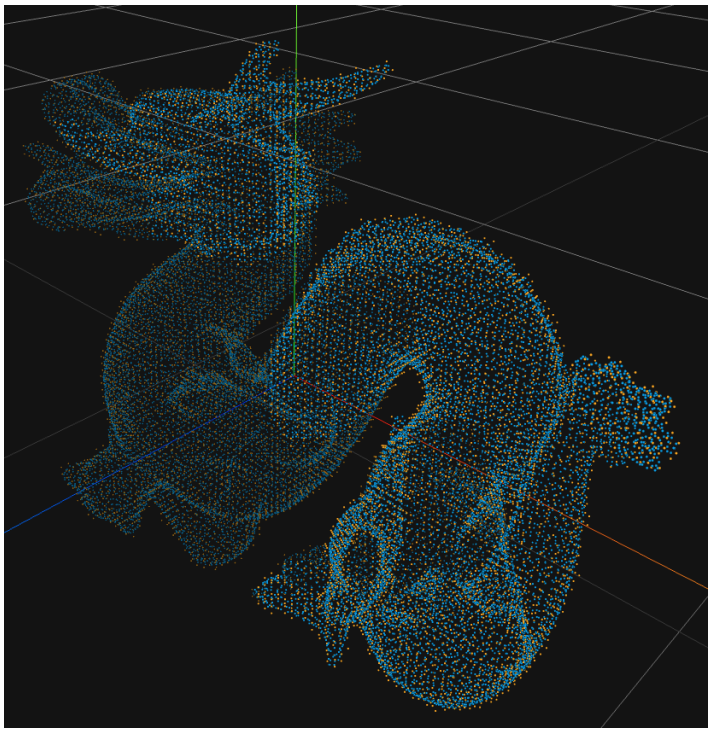
Results

Bunny dataset



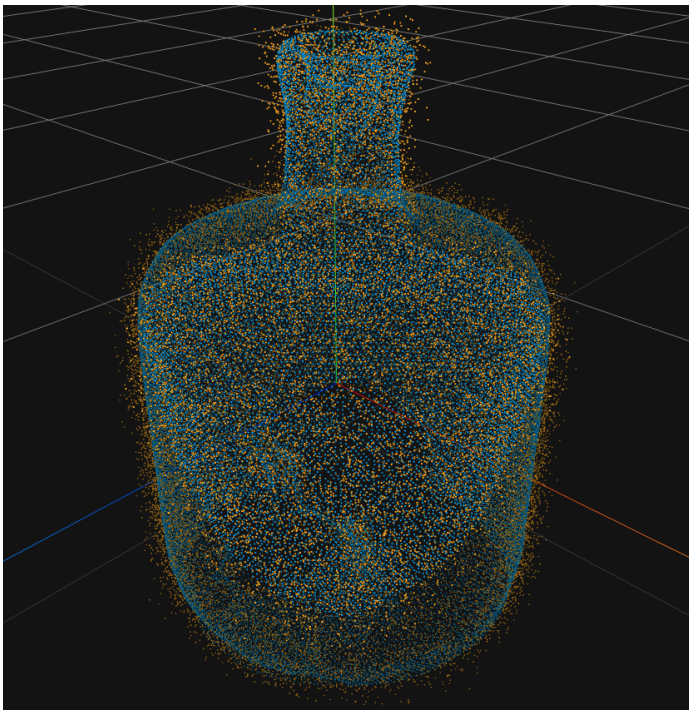
	SVD	LM
runtime [s]	8.207	58.034
iterations	34	23
RMSE	0.00401559	0.00341361

Dragon dataset



	SVD	LM
runtime [s]	1.274	18.176
iterations	13	20
RMSE	0.00568868	0.0056415

Vase dataset



	SVD	LM
runtime [s]	6.891	70.071
iterations	26	29
RMSE	0.0162231	0.016221

Analysis

The results show a very good alignment of the source with the target cloud.
In all cases, the algorithm reaches convergence before the maximum number of iterations.
Comparing SVD with LM, we notice that the performance is slightly better for the latter, which in turn requires much longer to execute (this is especially evident on the Bunny dataset).
The more challenging dataset is the Vase one, where the source seems highly affected by noise. Even though the quantitative result suffers from this, qualitatively the alignment is extremely satisfactory.
For the other two datasets, both qualitative and quantitative results are excellent.