Report Laboratory 3:

Iterative Closest Point

Christian Francesco Russo 2087922

Task 1: Find closest point

To find the closest target point for each source point, I iterated over all source points to find their nearest neighbor target point. To speed up the search, I created a K-D tree with the points in the target point cloud and then called the function `open3D::geometry::KDTreeFlann::SearchKNN()` on the K-D tree to find the closest target point to each source point. If the distance between the matched points was lower than a threshold, I stored the indices of the source and target matched points in the vectors `source_indices` and `target_indices`, respectively, and updated the MSE estimation. Upon completing the association of the two point clouds, I returned the tuple `(source_indices, target_indices, RMSE)`.

Task 2: Compute SVD transformation

To estimate the rigid body transformation $\boldsymbol{R}, \boldsymbol{t}$ between the two point clouds using the SVD method, I decoupled the problem by first estimating $\boldsymbol{R}$, for then I used $\boldsymbol{R}$ to estimate $\boldsymbol{t}$.

- To estimate $\boldsymbol{R}$:
  - I computed the centroids of the two point clouds using the function `open3D::geometry::PointCloud::ComputeMeanAndCovariance()`.
  - I computed the matrix $\boldsymbol{W}$ as:
$$\boldsymbol{W} = \sum_{i=1}^{n} \boldsymbol{m}_i' \boldsymbol{d}_i'^T$$
    where n is the number of points contained in the two point clouds, and $\boldsymbol{m}_i'$ and $\boldsymbol{d}_i'$ are the points in the target and source point clouds, respectively, with their centroids subtracted:
$$\boldsymbol{d}_i' = \boldsymbol{d}_i - \boldsymbol{d}_c, \quad \boldsymbol{m}_i' = \boldsymbol{m}_i - \boldsymbol{m}_c$$
  - I computed the Single Value Decomposition (SVD) of the matrix $\boldsymbol{W}$, and use it to compute an optimal solution for the rotation matrix:
$$\widehat{\boldsymbol{R}} = \arg\min_{\boldsymbol{R}} \sum_{i=1}^{n} \|\boldsymbol{m}_i' - \boldsymbol{R}\boldsymbol{d}_i'\|^2 = \boldsymbol{U}\boldsymbol{V}^T$$
  - As required by the assignment, I handled the special reflection case (due to possibly corrupted data). If:
$$\det(\boldsymbol{U}\boldsymbol{V}^T) = -1$$
    I fixed the rotation matrix as follows:
$$\widehat{\boldsymbol{R}} = \boldsymbol{U}\,\mathrm{diag}(1,1,-1)\boldsymbol{V}^T$$
- To estimate $\boldsymbol{t}$:
  - I computed the optimal translation vector $\boldsymbol{t}$ (in closed form) as:
$$\hat{\boldsymbol{t}} = \boldsymbol{m}_c - \widehat{\boldsymbol{R}}\boldsymbol{d}_c$$

Finally, I built and returned the transformation matrix $[\boldsymbol{R} \quad \boldsymbol{t}]$.

Task 3: Compute LM transformation

To estimate the rigid body transformation $\boldsymbol{R}, \boldsymbol{t}$ between the two point clouds using the LM method I used Ceres.

- I implemented the Ceres auto-differentiable cost function, where, by the theory, the registration error is defined as:

$$E(\boldsymbol{R},\boldsymbol{t}) = \sum_{i=1}^{n_{\mathcal{D}}} \left\| \boldsymbol{m}_{j(i)} - (\boldsymbol{R}\boldsymbol{d}_i + \boldsymbol{t}) \right\|^2 = \sum_{i=1}^{n_{\mathcal{D}}} \boldsymbol{e}_i(\boldsymbol{R},\boldsymbol{t})^T \boldsymbol{e}_i(\boldsymbol{R},\boldsymbol{t})$$

- I created a Ceres problem where the cost function was evaluated over all the pairs of source and target matched points. The solution to this problem provides the optimal transformation parameters $\left[r_x, r_y, r_z, t_x, t_y, t_z\right]^T$.
- From this solution, I extracted and built the rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{t}$.

Finally, I built and returned the transformation matrix $[\boldsymbol{R} \quad \boldsymbol{t}]$.

Task 4: ICP registration

To implement the ICP main loop I followed the ICP algorithm steps:

1. **Search new correspondences $\boldsymbol{d}_i \leftrightarrow \boldsymbol{m}_{j(i)}$:**
   - I called the function `find_closest_point()` implemented earlier to associate the points of the two point clouds.
2. **Check convergence:**
   - I estimated the relative RMSE error as:

$$\mathrm{RELATIVE\_RMSE} = |\mathrm{CURR\_RMSE} - \mathrm{PREV\_RMSE}|$$

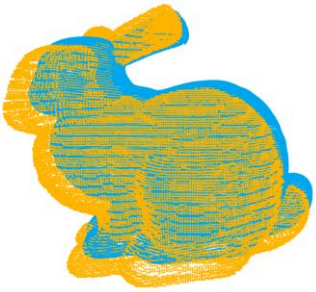
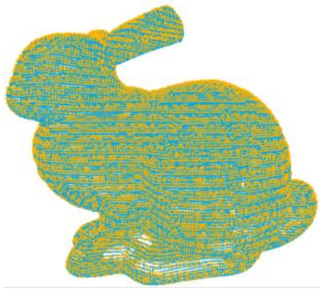
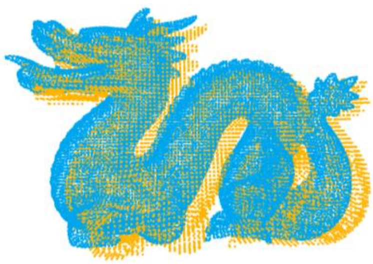
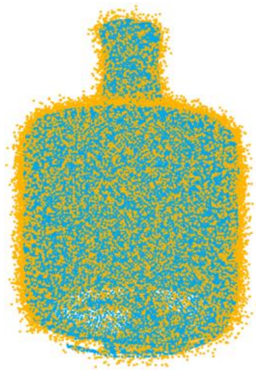
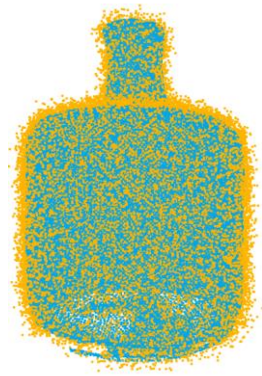
   - If this relative RMSE error was lower than a certain threshold, convergence was reached, and the algorithm could terminate.
3. **Estimate the transformation parameters $\boldsymbol{R}, \boldsymbol{t}$ using the new correspondences:**
   - Depending on the required mode (SVD or LM), the function `get_svd_icp_transformation()` or `get_lm_icp_registration()` was called to estimate the current transformation parameters.
   - I integrated this current transformation with the previous transformations by pre-multiplying it with the cumulative rigid body transformation computed so far.
4. **Transform the points using the estimated parameters $\boldsymbol{R}, \boldsymbol{t}$:**
   - I used the estimated transformation parameters to transform the source point cloud `source_for_icp_`.

Results

Table 1 - RMSE errors

Data Item	Bunny	Dragon	Vase
RMSE with SVD	0.00401621	0.00568867	0.0162243
RMSE with LM	0.00341366	0.00564134	0.0162218

Table 2 – Images of ICP reconstruction results with SVD and LM

Original Point Cloud	SVD	LM
Bunny		
		
Dragon		
		
Vase		
		

Note: the results are really good for both SVD and LM based approaches, however, as we can see, the performance are slightly better for the LM version of the ICP algorithm.