Simone Mosco 2055298

Lab 3: Iterative Closest Point

Task 1: Compute Closest Path

The first task regards the computation of pairs of closest points between the source and target point clouds. As done in the `compute_rmse()` method, the target cloud is represented in a KD Tree for better nearest neighbor search performance. In this way we loop through all the points in the source point cloud, and look for the closest point in the target point cloud, checking if the distance between them is less than the threshold provided in input. In case of success the `source_indices` and `target_indices` arrays are filled with the correspondent indices of the two points and the *error* is updated with the same formula in `compute_rmse()`. Finally the method returns a tuple containig both array `source_indices`, `target_indices` and the square root of the error, representing the Root Mean Square Error.

Task 2: SVD Registration

In the second task, it is implemented the Singular Value Decomposition (SVD) method for the ICP algorithm. First the centroids of both point clouds are extracted using `ComputeMeanAndCovariance()` method in Open3D. Then we construct the matrix W which will be used for the SVD; in particular, we consider each pair of points whose indices are specified in the two vectors `source_indices` and `target_indices`, we normalize the points with respect to their point cloud centroid, perform the product between them which results in a 3×3 matrix that will be added to W. At this point we apply the SVD to the matrix W and obtain two matrices U, V from the equation $W = U\Sigma V^T$, which will be multiply to obtain the rotation matrix R. We handle the special case, checking if $det(UV^T) = -1$, as shown in the equation below.

$$R = \begin{cases} UV^T & if \qquad det(UV^T) \neq -1 \\ Udiag(1,1,-1)V^T & if \qquad det(UV^T) = -1 \end{cases} \tag{1}$$

Regarding the translation vector t, it is computed as in Equation 2.

$$t = target\_centroid - R \times source\_centroid \tag{2}$$

Finally we construct the transformation matrix stacking the matrix R and the vector t.

Task 3: LM Registration

- **Point Distance**: this class represents an auto-differentiable cost function which accepts as input the source and target points. We extract the translation paramteres from the transformation, apply the `ceres::AngleAxisRotatePoint()` function to obtain the rotated point and in the end compute the residual.
- **Get LM ICP Registration**: inside the `get_lm_icp_registration()` function it is created a *Ceres* problem and for each pair of points whose indices are stored in `source_indices` and `target_indices` vectors, it is specified a cost function and added a Residual Block. After the problem is solved, the three angles are retrieved and used for computing the rotation matrix R, along with the translation vector t. Finally the transformation matrix is constructed using both R and t.

Task 4: ICP

The main loop for the ICP algorithm is implemented in the `execute_icp_registration()` function. For each iteration, until the maximum number of iterations, we perform the following operations:

1. Run the `find_closest_point()` function and extract the current *rmse*;
2. Check convergence if the difference between current and previous *rmse* is smaller than a given threshold;
3. Execute the SVD or LM method and obtain the current transformation;

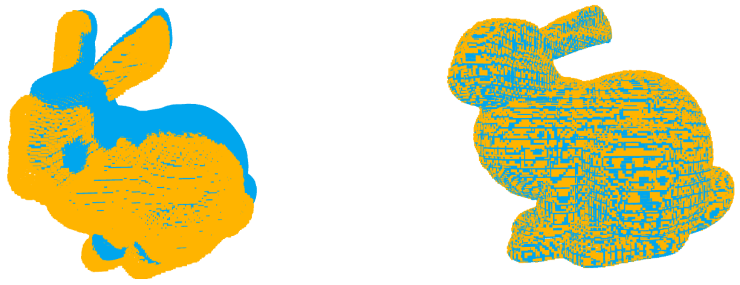
4. Update the global transformation, combining the previous and current transformation;
5. Transform the `source_for_icp` point cloud with the current transformation.

Results

The algorithm is able to learn a good transformation in order to align the two point clouds. The results regarding the *RMSE* are shown in Table 1 as it is possible to notice slightly better performance from the LM method even though it takes more execution time than SVD. The qualitative results are shown below for all the three datasets.

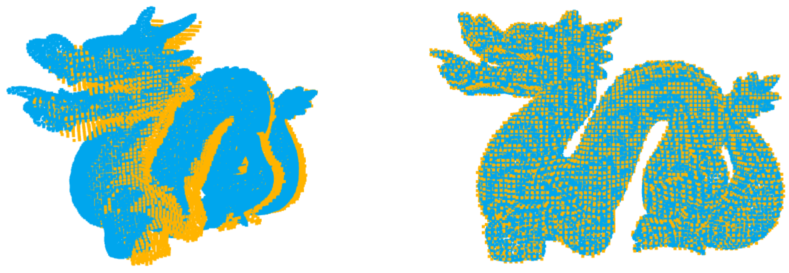
	Bunny	Dragon	Vase
SVD	0.00401621	0.00568867	0.0162243
LM	**0.00341366**	**0.00564134**	**0.0162218**

Table 1: RMSE for the three datasets



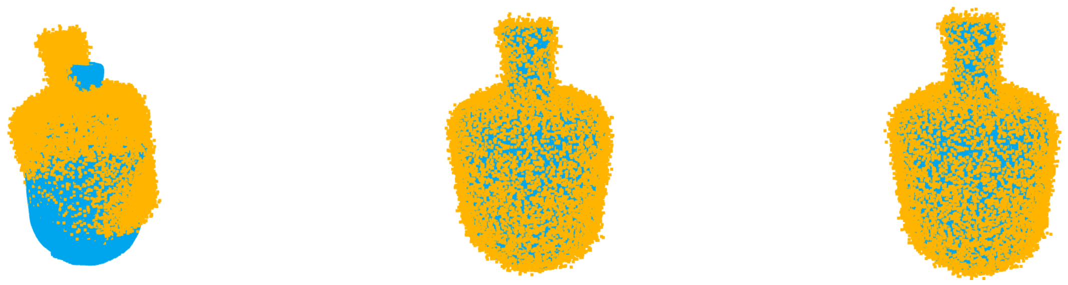
(a) Original Point Clouds (b) SVD (c) LM

Figure 1: Bunny dataset



(a) Original Point Clouds (b) SVD (c) LM

Figure 2: Dragon dataset



(a) Original Point Clouds (b) SVD (c) LM

Figure 3: Vase dataset