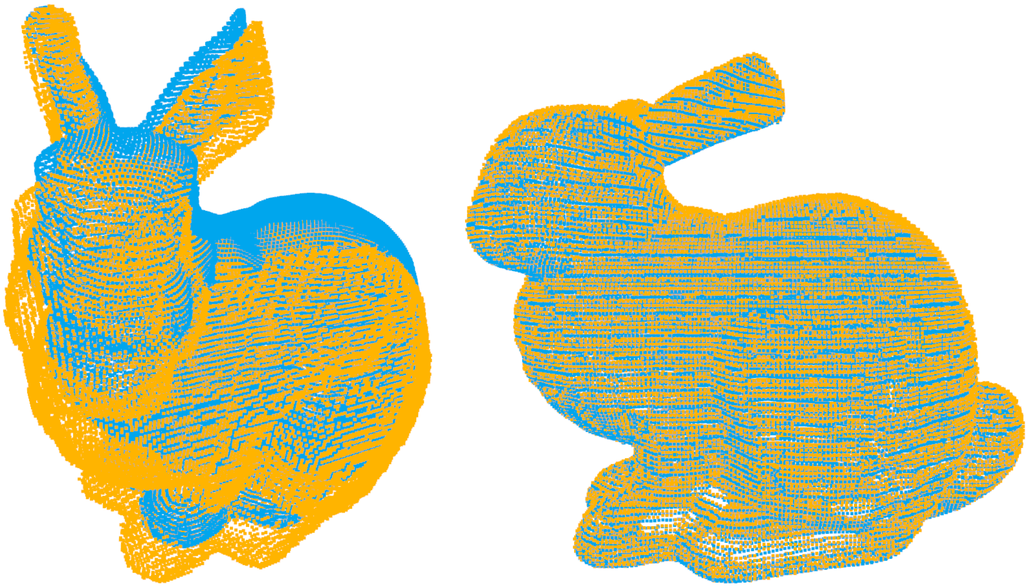
*3D DATA PROCESSING - LAB 3 (**Individual assignment**)*



Topic: Iterative Closest Point Cloud Registration
Goal: Given a source and a target point cloud roughly aligned, find the fine alignment transformation of the source to the target cloud.

Instructions

Extend the provided C++ software by implementing the ICP main loop, the closest point matching and the transformation matrix estimation .

The provided software already implements the following methods:

- **`Registration(...)`**
  - Initialize the source and target point cloud to be processed.
- **`draw_registration_result()`**
  - Visualize source and target point cloud.
- **`get_transformation()`**
  - Get the current transformation matrix needed to align the source to the target cloud.
- **`compute_rmse()`**
  - Compute the RMSE between the points of the source and the target point cloud.

Instead, the the following methods must be completed in order to successfully perform ICP:

- **`find_closest_point(...)`**
  - For each point in the source point cloud find the closest one in the target (look at **`compute_rmse()`**).

- **`get_svd_icp_registration(...)`**
  - First extract the centroid for each of the two point clouds, after subtracting it use `Eigen::JacobiSVD<Eigen::MatrixXd>` on the matrix obtained by multiplying the two Nx3 point matrices, ordered following the results of the `find_closest_point(...)` to successfully perform SVD decomposition.

- **`get_lm_icp_registration(...)`**
  - Remember to define a templated functor that computes the distance error/residual (defined as `PointDistance`). See Ceres Solver tutorial for a better understanding. Differently from the Bundle Adjustment only the 6-dimensional array (rx, ry, rz, tx ,ty, tz) must be optimized (instead of jointly optimizing camera and 3D point positions). As in `get_svd_icp_registration(...)` use the point correspondences extracted previously using `find_closest_point(...)`. Remember to convert from the euler axis-angle representation to rotation matrix.

- **`execute_icp_registration(...)`**
  - Main ICP loop, check convergence criteria and call `find_closest_point(...)` followed by either `get_svd_icp_registration(...)` or `get_lm_icp_registration(...)`. Feel free to use the class variable `source_for_icp_` to store transformed source points.

Compilation instruction

```
- mkdir build && cd build
- cmake ..
- make
```

To execute:

```
./registration path/to/source path/to/target mode
```

where mode could be either **`svd`** `or` **`lm`**.

What you need to deliver

- Source code (without objects and executables)
- A .ply file for the two provided datasets, representing the registered clouds
- A short written report with:
    - A brief description of the work done;
    - Some qualitative results (screenshots of aligned point clouds) for the two provided datasets.
    - Quantitative results in `RMSE` for the two provided datasets.