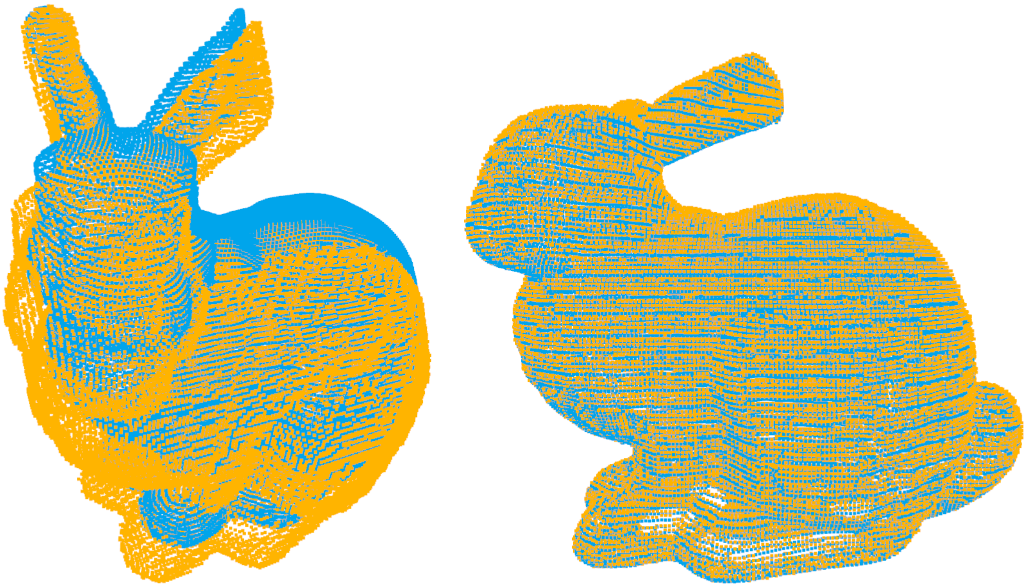
*3D DATA PROCESSING - LAB 3 (**Individual assignment**)*



Topic: Point Cloud Registration
Goal: Given an unaligned source and target point cloud, find the final alignment transformation from the source to the target cloud.

Instructions

You are required to extend the provided C++ software by implementing a global registration step based on feature descriptors, followed by an ICP refinement stage.

STEP1: Descriptor (Initial Alignment)

- You must detect keypoints and compute feature descriptors **manually**, based on techniques introduced during the course
- **Do not use high-level Open3D functions** such as **`ComputeFPFHFeature`** or **`RegistrationRANSACBasedOnFeatureMatching.`**
- However, **you may use low-level tools** (e.g., **`KDTreeFlann`**) to assist in implementing your descriptor-based pipeline.
- Once you have computed descriptors and found correspondences **by yourself**, you **may use `RegistrationRANSACBasedOnCorrespondence()`** to robustly estimate the initial transformation matrix.

STEP2: ICP Refinement

- Implement the full ICP algorithm manually, including:
    - Implement the full ICP algorithm manually, including:
    - Nearest-neighbor matching (e.g., using KDTreeFlann)
    - Transformation estimation via SVD
    - RMSE-based convergence checking

- **You must not use** Open3D's **`RegistrationICP()`** function in your implementation.
- **You may compare your result to Open3D's built-in ICP as a benchmark, but only your custom ICP logic will be graded.**

Important Notes

- Open3D's **high-level registration functions** must not be used for final results.
- Open3D's **basic data structures and utilities** (e.g., **`PointCloud,`** **`KDTreeFlann`**, **`visualization`**) are allowed.
- If your final code, report, and results rely solely on Open3D's built-in registration functions, they **will not be considered for grading**.

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_transformation(...)`**
  - 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.

- **`execute_icp_registration(...)`**

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

- **`execute_descriptor_registration()`**
  Implement a feature-based global registration method from scratch:
    - Preprocess the point clouds (e.g., downsampling, normal estimation).
    - Detect keypoints.
    - Compute local geometric descriptors around each keypoint.
    - Do not use any built-in descriptor extractors.
    - Implement from scratch: histogram-based, surface geometry, etc.
    - Match descriptors between source and target.
    - Estimate a robust initial rigid transformation
    - Store the result in transformation_.
    - Built-in registration functions or descriptor APIs are not allowed. Only self-implemented methods are permitted.

Compilation instruction

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

To execute:

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

What you need to deliver

- Source code (<u>without</u> objects and executables)
- A .ply file for the three 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 three provided datasets.
    - Quantitative results in `RMSE` and timing for the three provided datasets.