

# REPORT LAB 3:

## Cloud Registration

Boscolo Simone

simone.boscolo.2@studenti.unipd.it

N.Matricola: 2026826

---

### 1. INTRODUCTION

This assignment regards point cloud registration, that is the process of finding a rigid body transformation that aligns two or more 3D shapes. With this work, it is explored the Iterative Closest Point Algorithm (ICP) by matching two objects and it is tested with two test datasets. Also, I have personally tried to implement 2 methods for the initial guess of the transformation matrix.

### 2. IMPLEMENTATION

In the provided code is left to complete the following functions:

- A. 2 constructors of the provided class
- B. `draw_registration_result`
- C. `preprocess`
- D. `execute_global_registration`
- E. `execute_icp_registration` method that implement the ICP algorithm

**All the code has been developed by using the Open3D library.**

#### A. Constructors

The implementation of the 2 constructors is straightforward, they both require a source and target point cloud, the only difference is that one requires two objects of `PointCloud` type, instead the other one requires the paths to the file that store the point clouds.

#### B. `draw_registration_result`

This method shows the two point clouds in the same window, after the ICP algorithm the two clouds should overlap.

#### C. `preprocess`

In this function the input cloud is downsampled by the voxel size, given as parameter, and there are estimated its normals and the FPFH features (of the downsampled data). The downsampled data and its features are returned by using pointer, in fact two pointer to this kind of data structure are required as function parameters.

#### D. `execute_global_registration`

This class method performs the initial guess of the transformation matrix, that it is used later by our ICP algorithm, by using global matching.

The source cloud is transformed according to the actual transformation matrix and then source and target clouds are preprocessed (by using the `preprocess` function), in order to obtain the downsampled data and their features.

By following the Open3D tutorial (at this [link](#)), the library provides two methods to implement global registration and a boolean variable named `fast` has been created to select one of the two methods:

- (a) `true` = Fast Global Registration
- (b) `false` = RANSAC Registration Method

This variable needs to be changed in the code and by default it is set to `true`, as final implementation.

The downsampled data and their features (of source and target data cloud computed before during the preprocessing step) are passed as parameters to both methods.

### (a) Fast Global Registration

This method is performed by

`open3d::pipelines::registration::FastGlobalRegistrationBasedOnFeatureMatching.`

It is possible to change some algorithm parameters by passing as argument to the previous function a `FastGlobalRegistrationOption` object. In this implementation everything is left to default, but the `maximum_correspondence_distance` is set to `distance_threshold = voxel_size * 0.5`; as it is used in the linked tutorial.

### (b) RANSAC Registration Method

This method is performed by

`open3d::pipelines::registration::RegistrationRANSACBasedOnFeatureMatching.`

It is possible to change some algorithm parameters by passing some value as argument to the previous function.

In this case everything is set to default, but (always by following the linked tutorial) a parameter and some checkers are passed, in particular:

- `maximum_correspondence_distance = distance_threshold`
- `checkers_1: CorrespondenceCheckerBasedOnEdgeLength(0.9)`
- `checkers_2: CorrespondenceCheckerBasedOnDistance(distance_threshold)`

Where `distance_threshold = voxel_size * 1.5`.

For both method it is set a timer in order to have also a time comparison among the two methods, rather than only a quantitative results.

Both methods return a `RegistrationResult` object, from where it is retrieved the new transformation matrix that is used to compute the transformation among the source and the target cloud. The new matrix is stored in class member named `transformation_`.

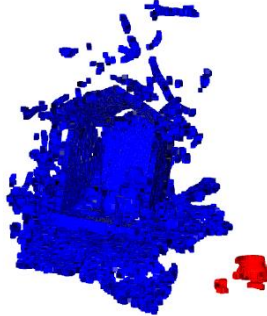
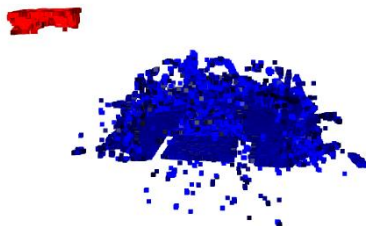
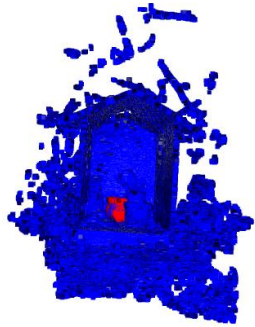
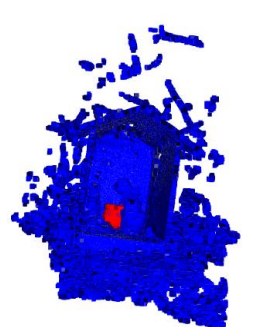
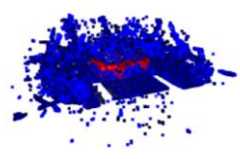
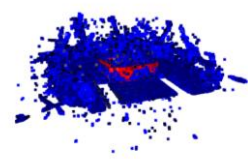
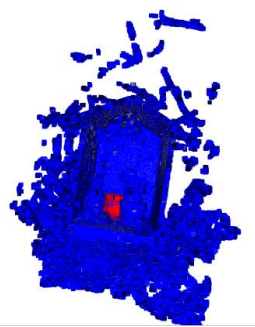
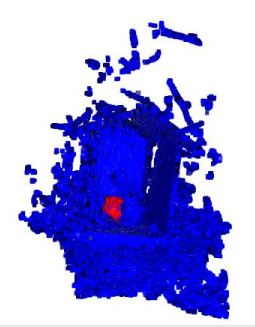
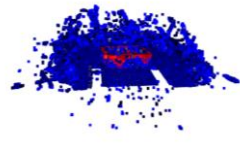
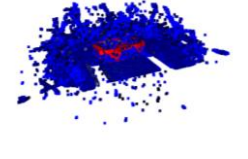
### E. `execute_icp_registration`

This method implements the `open3d::pipelines::registration::RegistrationICP` and in this case there are passed as parameters:

- `source = source_`
- `target = target_`
- `max_correspondence_distance = threshold`
- `init = transformation_`
- `estimation = TransformationEstimationPointToPoint(false)`
- `criteria = ICPConvergenceCriteria(relative_fitness, relative_rmse, max_iteration)`, which parameters are passed as parameters to our `execute_icp_registration` function

And, as in the previous method, `transformation_` is updated by retrieving the new transformation matrix provided by the result of the `RegistrationICP` function.

### 3. RESULTS AND COMMENTS

DATASET	ONE		TWO	
METHOD	FAST GLOBAL	RANSAC	FAST GLOBAL	RANSAC
INITIAL				
GLOBAL				
ICP				
fitness	0.869718	0.869718	1	1
inlier_rmse	0.00396817	0.00396819	0.00185056	0.00185062
time	29527.6 ms	34837.5 ms	1302.84 ms	1100.41 ms

As it can be seen, the results are almost identical in terms of fitness and `inlier_rmse`, but the Fast Global method is faster than the RANSAC one with the dataset “one”, that is the data cloud with more points to compute.

Instead, the two methods are almost identical in terms of time as well with the second data cloud.

Therefore, it has been chosen to keep as final implementation the Fast Global method (variable `fast = true`), because it can deal big data cloud in less time than the RANSAC Global method.

**N.B.:** the images in the table can be found in the folder named “report\_imgs” and the final results can be found in the “results” folder