Point Cloud Processing

1 POINT CLOUD REGISTRATION

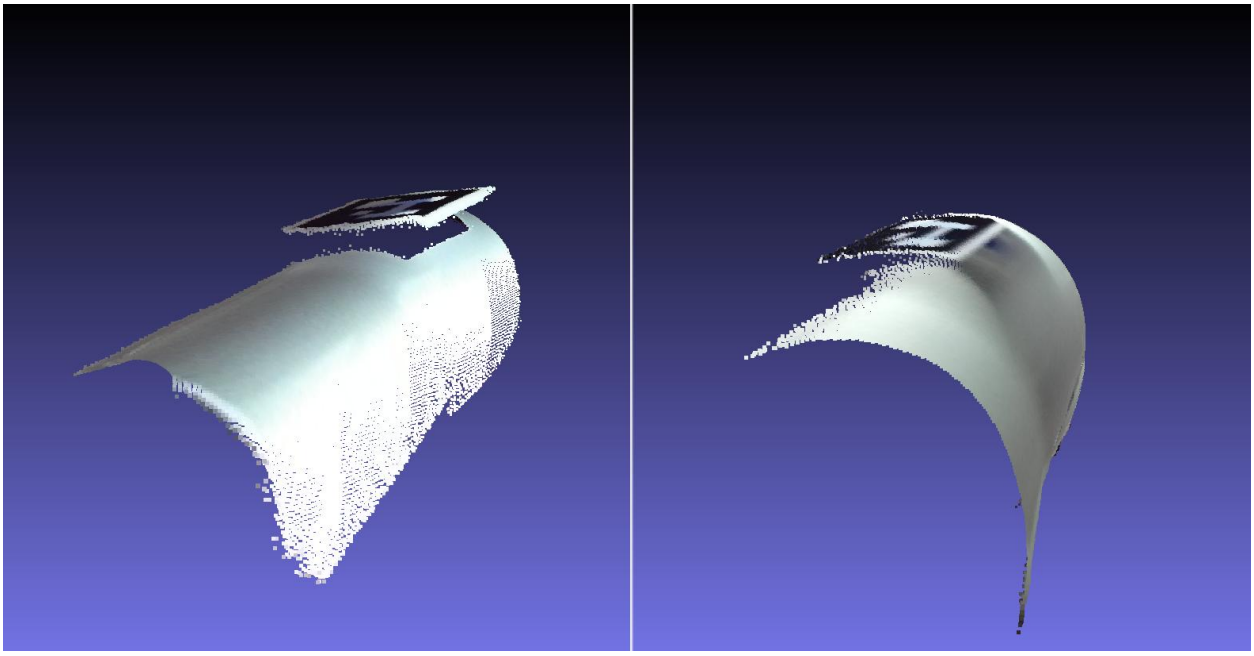
1.1 TEST DATASET CHALLENGES

A set of raw point clouds with color information was provided for two different scenes, likely generated through an image-based 3D reconstruction method. The first set captures a hair dryer held by a human arm, while the second is an outdoor scene. Each point cloud includes detailed color information but also exhibits common reconstruction challenges such as noise, incomplete geometry, and partial overlaps between scans. Although most of the data exhibits smooth transitions between scans, there are also abrupt changes that complicate the registration process.

1.1.1 Hair dryer scene

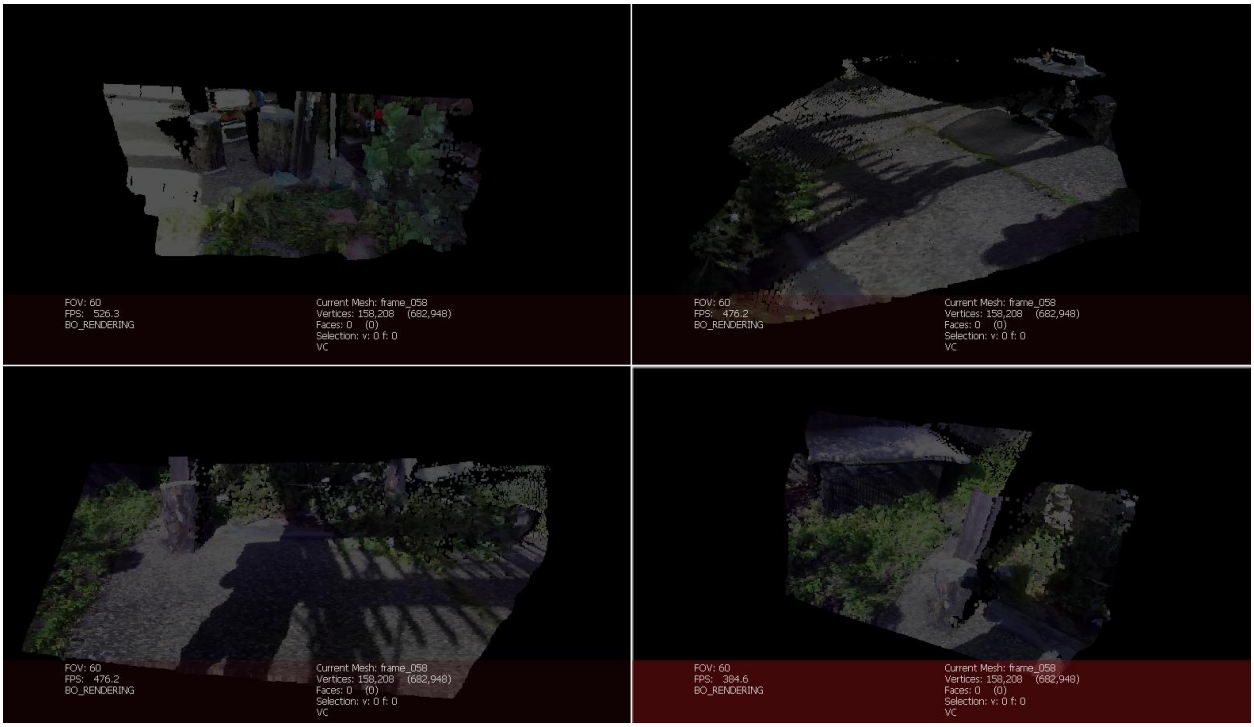
The captures of these scene present certain overlap between them for the first ones; however, at some point, they present hard transitions and noticeable non-rigid motion for the arm. The image below shows some partial scans illustrating these challenges.

Also, the scans are not consistent maybe due to acquisition limitations. For example, in the image below, we can see one scan with a gap between the marker and the rounded surface, while in the other scan we can see a continuous rounded surface that connects smoothly with the marker.



1.1.2 Outdoor scene

This scene captures multiple objects encompassing a complex and large environment difficult to register. The image below shows some of the scans, presenting several and partial shapes of objects. Further, it presents some shadows that can hinder color-based alignment.



1.2 TESTED APPROACHES

Several methods and combinations of methods were tested, including the following methodologies:

- ICP point to point
- ICP point to plane
- ICP point to plane + color matching
- Coarse to fine registration considering ICP-based methods
- Coarse to fine registration using feature-based RANSAC for coarse alignment (FPFH features)
- Multiway registration using pose graph estimation and optimization, and considering the different registration methodologies
- Multiway registration using sequential alignment considering different registration methodologies
- Global and sliding window processing for multiway registration
- Multiscale registration using different tolerance and down sampling parameters
- Double-sided registration
-
- Preprocessing using outlier removal, largest component selection, down voxel, repeated points removal, radius-based normal estimation, tangent plane normal consistency, fixed parameters using average distance between points, etc.
- During registration: normal orientation correction, normal error measurement,

1.3 SELECTED APPROACH

The proposed approach relies on certain assumptions about the input data: each point cloud should contain color information and be part of a sequential acquisition, ensuring substantial overlap with preceding scans. This sequence order is represented by the corresponding positions of an input list of point clouds. The method can be summarized by the following pseudocode:

ALGORITHM 1: MULTIWAY REGISTRATION

Input: a list of point clouds with color information (without coupled normals)

Output: the list of registered point clouds

Compute the average distance between points for all the point clouds
Compute a voxel size using the average distance: voxel size = average distance * 1.5
Compute a maximum coarse correspondence distance: voxel size * 15
Compute a maximum fine correspondence distance: voxel size * 1.5
Preprocess each point cloud:

1. Remove repeated points (very small distance)
2. Outlier removal
3. Select maximum connected component (with high tolerance for connectivity)
4. Estimate normal using regular neighborhoods
5. Consistent normal orientation using tangent planes

For each point cloud:

1. Register the current point cloud with the previous registered point clouds. The preferred alignment direction is to set as source the smaller point cloud and as target the larger point cloud. For all the steps, we use the source point cloud with the current and inverted normals. From these possible results, we pick the registration that better fits the target points and target normals. A normal error function was designed for this purpose. The final score is composed as follows: score = 0.2 distance-based score (fitness) + 0.8 normal-based score.
    a. Coarse registration using point to plane ICP and considering multiple initializations (translations of the source on the axes x, y and z). Optionally, we can simply apply a point to point ICP.
    b. Fine geometric registration using point to plane ICP.
    c. Fine point to plane + color matching ICP
    d. If the final fitness (distance-based overlap) is below a given threshold, the registration is not acceptable.
2. Correct normal orientation of the registered point cloud by checking the normal similarity regarding the previously aligned point clouds.
3. Combine the registered point cloud with the previous one (if the registration is acceptable)
4. Downsample this combination using voxel size * 0.1
5. Attach the current registered point cloud to a list of registered point clouds (if the registration is acceptable). This list will contain all the registered point clouds as independent sets.

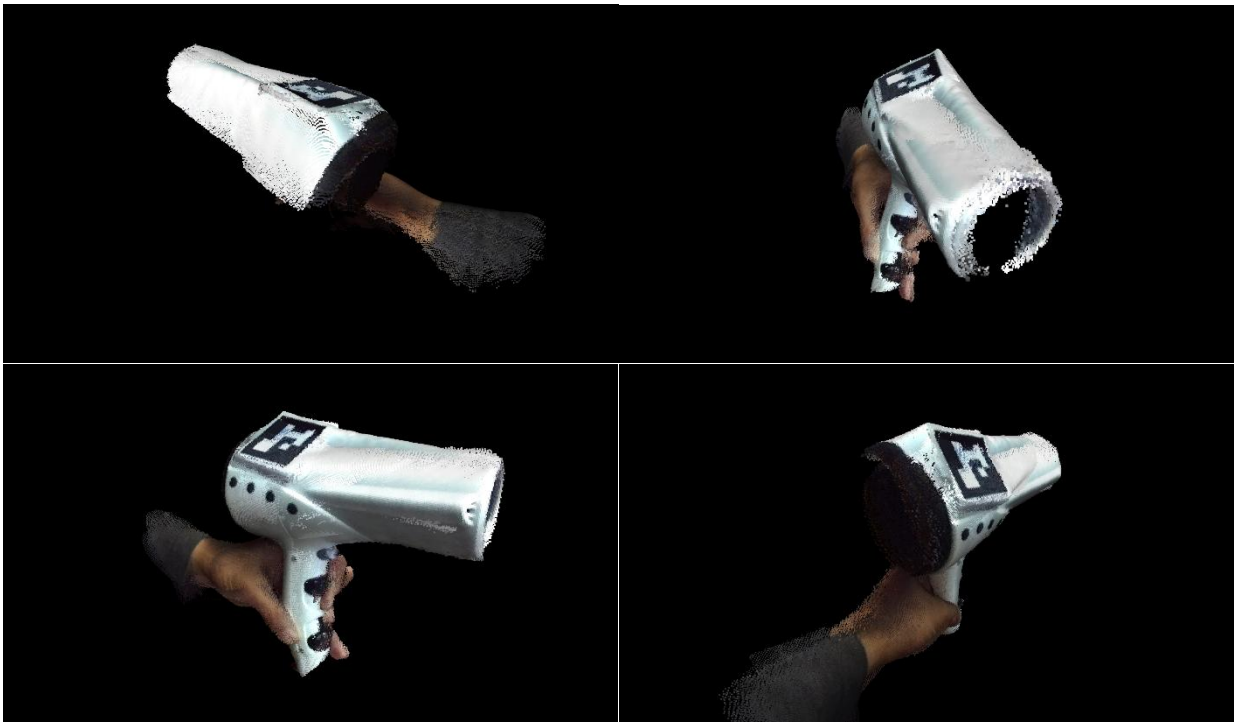
Return the merged point cloud and a list of point clouds of all the registered point clouds

In our implementation, the combined point cloud and the isolated registered point clouds will be exported to the folder data/results/<name>. Notice that just the point clouds that presented an acceptable registration will be considered.

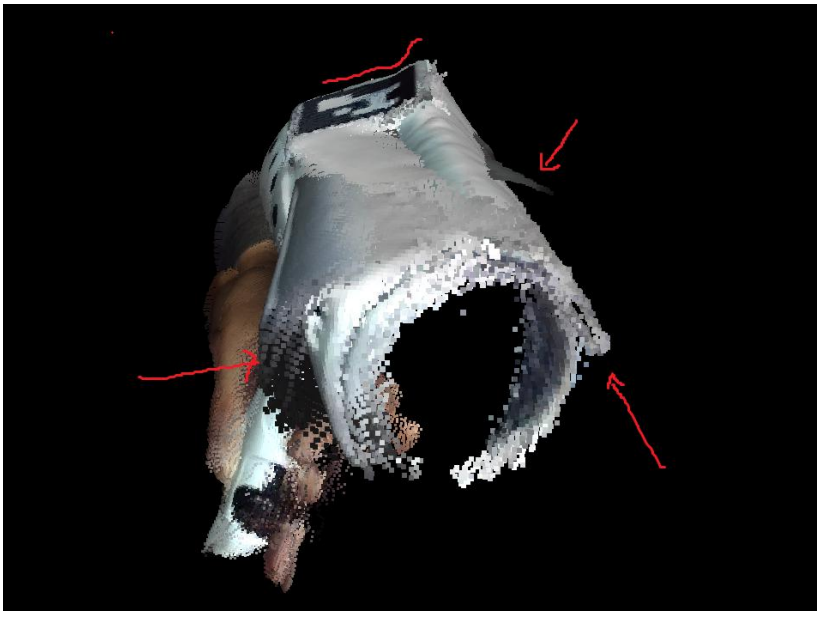
1.4 RESULTS

For the dryer scans, we noticed a stable behavior from _frame002.ply to _frame021.ply. The other scans are not stable, continuous, and accurate. So, we decided to split this sample into two datasets: one

containing the full scans, and the other containing just the samples from _frame002.ply to _frame021.ply. For these sets, we consider acceptable registrations when the fitness is over 0.7. The images below show the registration result considering the stable captures, which are named hair_dryer_part_1.



The image below shows the result of using all the captures, where we can notice some misalignments that are introduced due to the inconsistency with the first set of scans.



For the outdoor scene, we use the simple point to point ICP with a tolerance fitness of 0.4. In the image below, we show some captures of these scans.

These are the recommendations for the parameter setting:

```
# Set registration options
    fitness_threshold = 0.7 # 0.7 for objects with high overlap like the dryer. 0.4 for large scenes with low overlap between sections.
    use_simple_coarse = False # False for objects. True for scenes that present several regions of floor.
```

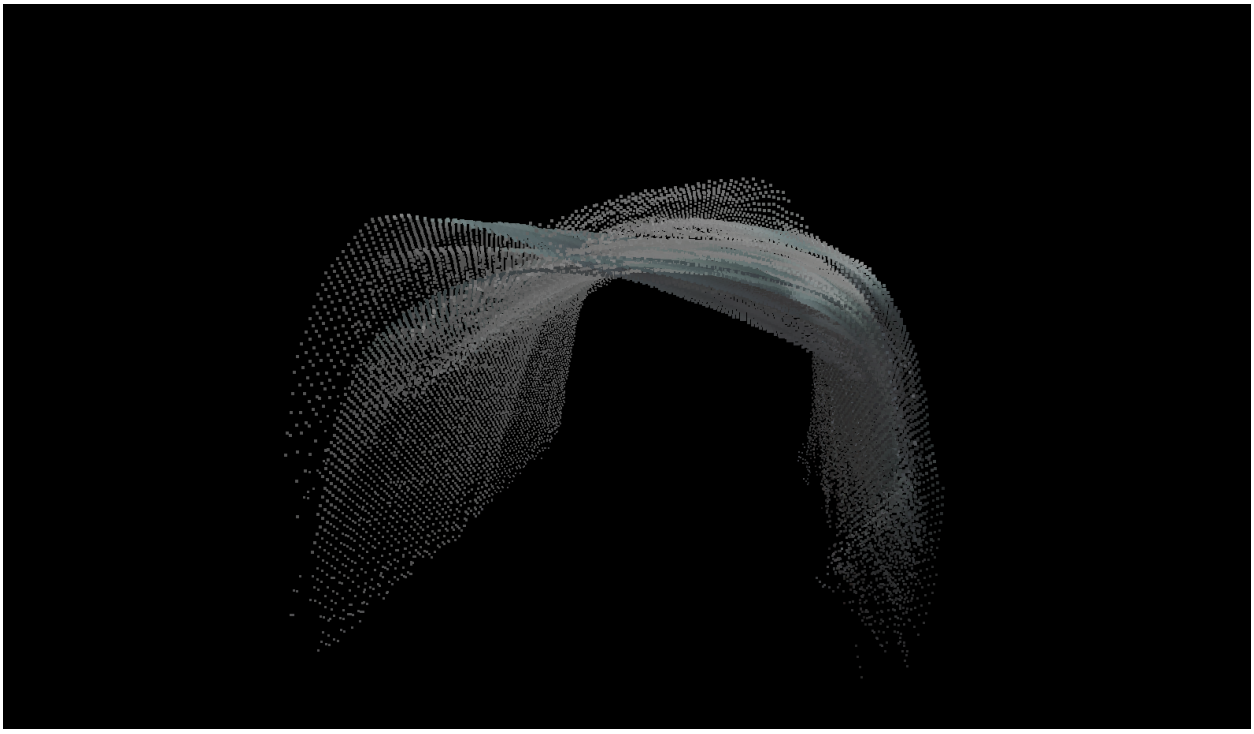
These parameters and the input specification at hardcoded at this moment; however, the idea is to align the implementation in the following milestones.

2 POINT CLOUD OUTLIER REMOVAL

2.1 REGISTRATION RESULTS

During the registration process, a per-point cloud outlier removal step is applied using a fitness threshold. This parameter helps prevent the inclusion of severely misaligned point clouds that could negatively affect the final combination. Despite this initial filtering, we also apply an additional outlier removal

algorithm to address minor misalignments. This second step targets points with poor overlap relative to those from other aligned point clouds, as well as points affected by acquisition limitations, such as inconsistencies in surface capture. The image below illustrates an example from the hair dryer model, where different surface reconstructions lead to discrepancies that cannot be resolved through rigid transformations. These artifacts can interfere with the mesh generation process, often resulting in conflicting triangle sets that represent multiple overlapping surfaces.



2.2 SELECTED APPROACH

To remove outliers, we propose a three step-based method. Let us define each registered point cloud as a cluster.

1. First, we remove points that are not in close proximity to points from other clusters. The proximity criteria are based on a Euclidean radius equal to 2.5 times the same average distance between points computed for the registration pipeline. However, if an entire cluster shows substantial overlap with other clusters (i.e., greater than 60%), all points within that cluster are retained, even those that do not directly overlap with other clusters. This indicates that this cluster has high confidence. Conversely, if a cluster exhibits poor overlap (i.e., less than 30%) with others, all its points are considered outliers and are removed. This per cluster evaluation is optional and controlled by the parameter param_outlier_removal_use_per_point_cloud_checking in the implementation. If you don't use this evaluation, the behavior is the suggested one (point-based overlap confidence).
2. Second, we remove points that present poor visibility from external points of view. This step is optional and is controlled by the parameter param_outlier_removal_use_visibility_confidence.

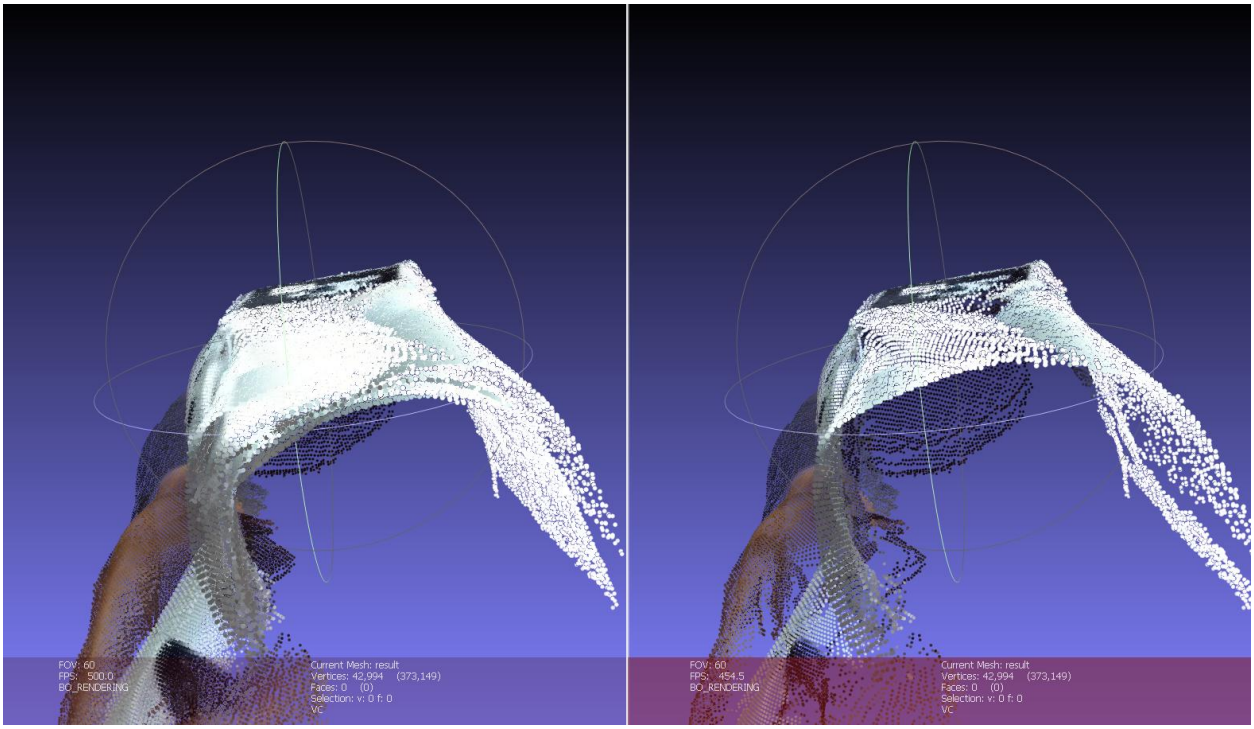
The core idea is to sample multiple viewpoints from a surrounding sphere and determine which points are visible from each view. Visibility is assessed not only based on point positions but also on whether the points are oriented toward the views, using surface normals. We adopt the method proposed in “Direct Visibility of Point Sets” (Katz et al.), which relies on computing the convex hull of a shape’s projection to identify visible points. In a post-processing step, we refine the selection by evaluating the dot product between each point’s normal and the view-to-point direction vector to ensure the surface is facing the view point. This visibility filtering is particularly useful for object scans, such as the hair dryer model, to exclude internal surfaces and focus on the outer shell. However, for large-scale or outdoor scans, this method may inadvertently remove some hidden regions in highly concave areas.

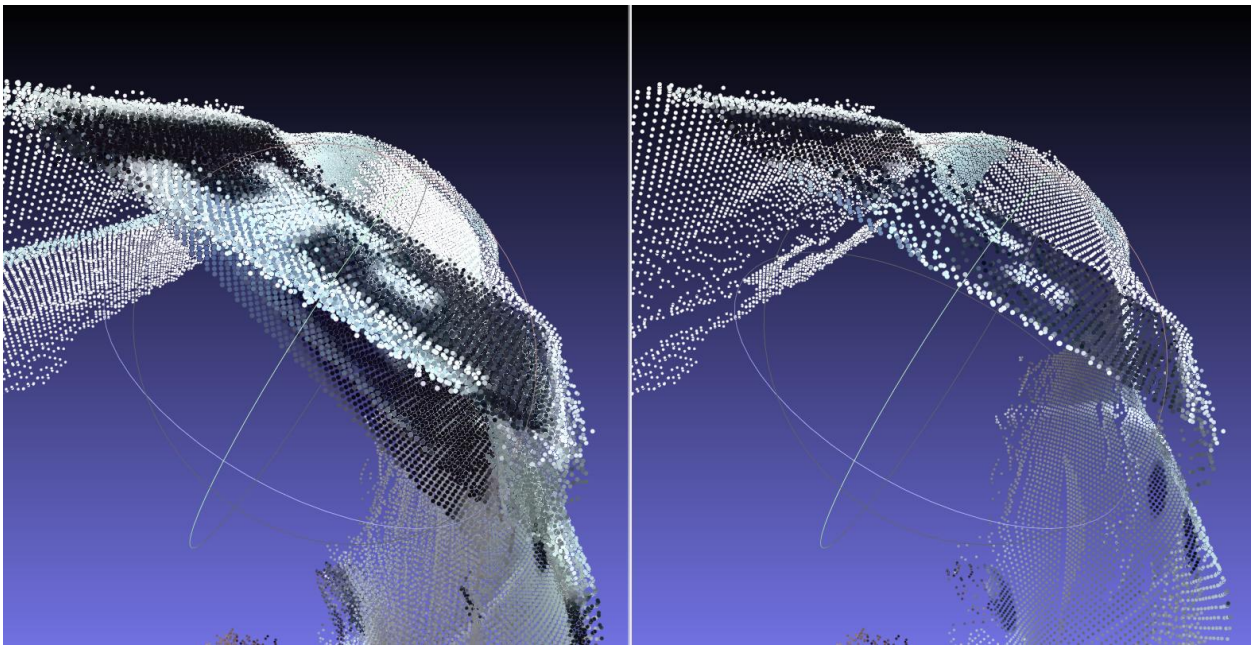
3. Finally, we apply a statistical outlier removal filter to the last results, using the same parameters considered to preprocess the registered point clouds. Also, we apply a radius outlier removal.

(*) At the first step, we force the input normals to point outwards by applying an heuristic ray-tracing-based approach.

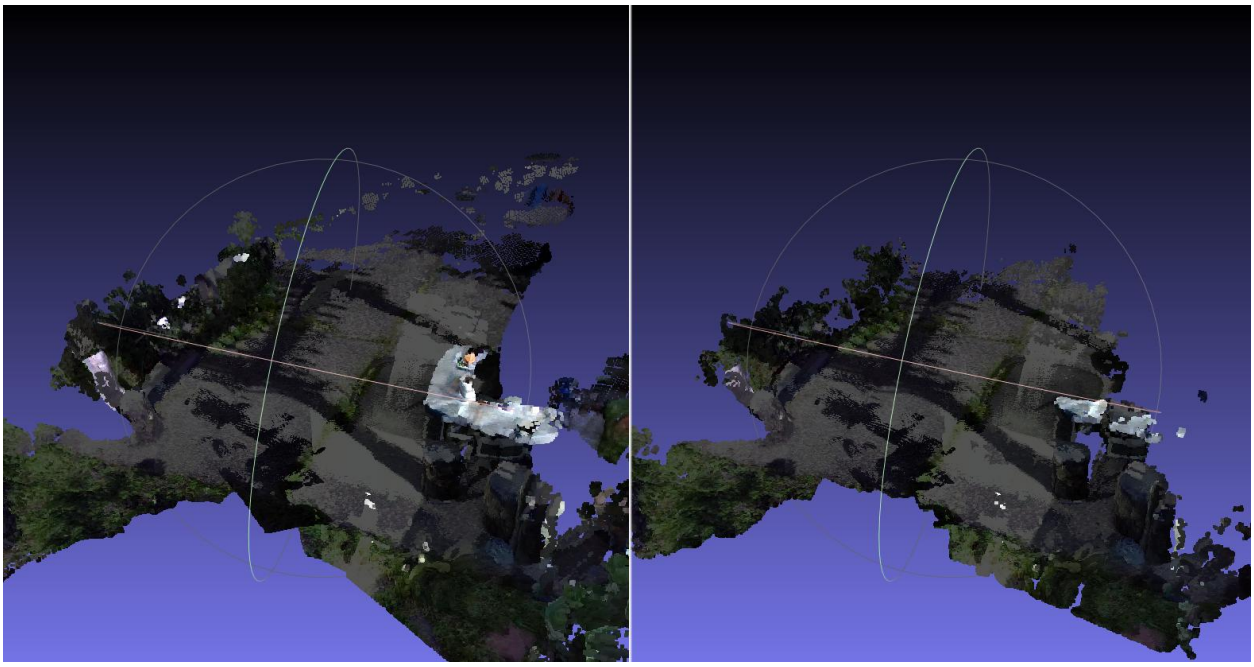
2.3 RESULTS

For object scans, we recommend to use all the steps and checkings. The images below show examples of the input merged point cloud (left) and the result after the full outlier removal (right). Notice how the inner surfaces are removed while preserving the outer one





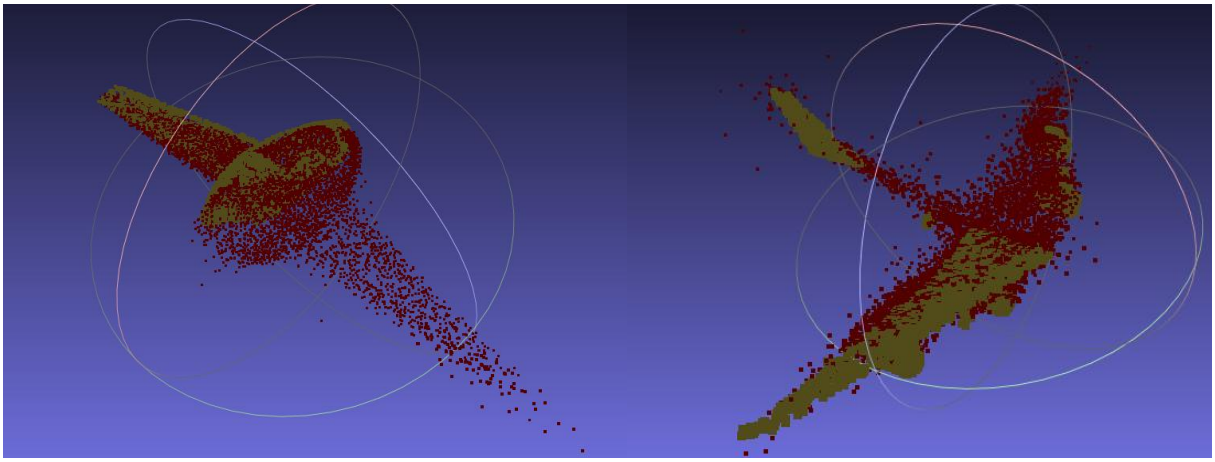
If we just use the simple overlap-based outlier removal, without the per-cluster checking and visibility checking, we can remove several points from scenes that multiple points are represented by a single cluster. The image below shows the behavior of this configuration:



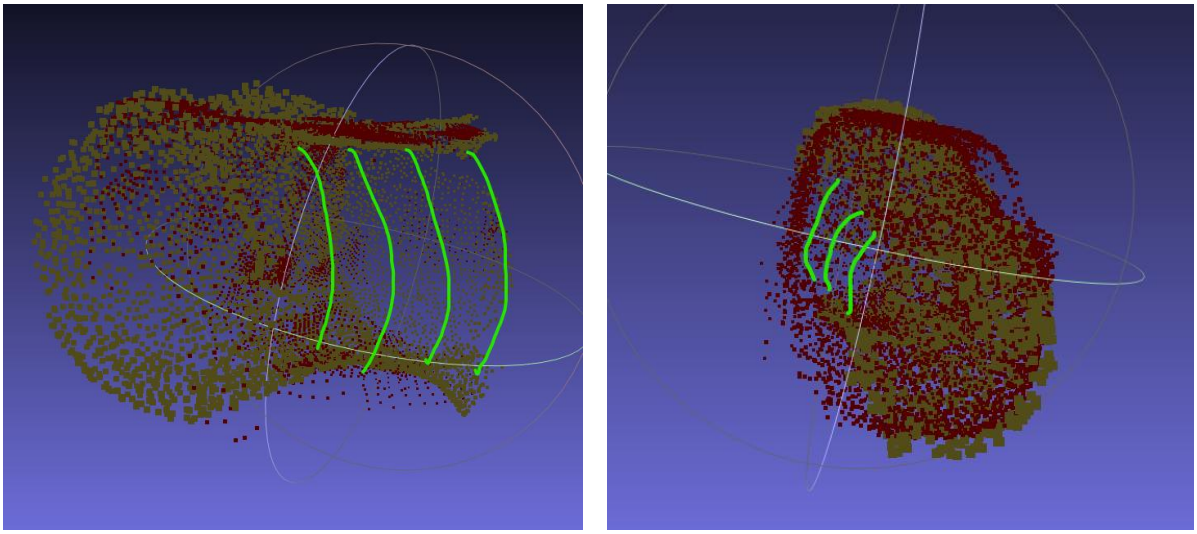
3 SHAPE COMPLETION

3.1 MACHINE LEARNING METHODS LIMITATION

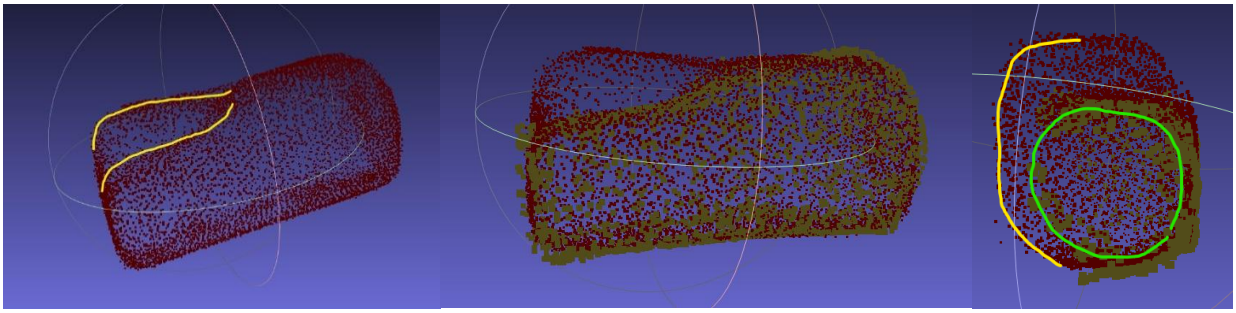
Shape completion is a very complex problem in the literature, where most methods consider domain-limited scenarios and rely heavily on strong priors or assumptions about object categories, symmetry, or structure. While it is tempting to assume that deep learning models trained on large datasets are sufficiently general for real-world applications, this often proves untrue. Recent methods such as PoinTr and AdaPoinTr combine PointNet-based feature extractors with transformer architectures to infer missing geometry from partial point clouds, generating complete 3D shape representations. However, these models still tend to generalize poorly outside the distribution of the training data, limiting their effectiveness in practical scenarios. For example, when we consider partial shapes of airplanes, which were included in the training dataset, we can achieve acceptable performance, as shown in the images below. (input points in yellow and completion points in red).



However, when we use the same model (trained on projected ShapeNet55) on our real-world input, the prediction introduces several artifacts. In the images below some of these artifacts are illustrated, where the green lines represent the expected surface to be completed.



Further, the prediction can create unexpected surfaces that can hinder the mesh reconstruction process. Some of these unexpected are highlighted in yellow in the images below.



In the paper titled “Revisiting Point Cloud Completion: Are We Ready For The Real-World?” some experiments and discussions are presented to explain the poor generalization of deep learning completion approaches.

3.2 SELECTED METHOD

Due to the limitations of machine learning methods, we decided to implement a method purely based on non-machine learning geometry processing. The proposed solution is inspired in our paper “Enveloping CAD models for visualization and interaction in XR applications”, which introduces a shrink-wrapping method to adapt a coarse surrounding surface to a given target surface. More precisely, we introduce a point cloud-based adaptation that uses less operations to generate surfaces in regions with missing data. The algorithm assumes moderate partiality and a degree of convexity in the input shape. The steps of the algorithm can be summarized in the following points:

1. Compute a Convex Hull mesh for the input points
2. Apply isotropic remeshing to the Convex Hull mesh
3. Expand the Convex Hull mesh and make it as the initial envelope
4. Execute n times:
    a. Project the vertices of the envelope onto the input points, considering a step size
    b. Smooth the envelope vertices that are far away from the input points
    c. Apply remeshing based on Screened Poisson Surface Reconstruction
    d. Apply isotropic remeshing to the envelope mesh
5. Expand those regions that are far away from the input point, generating curved surfaces

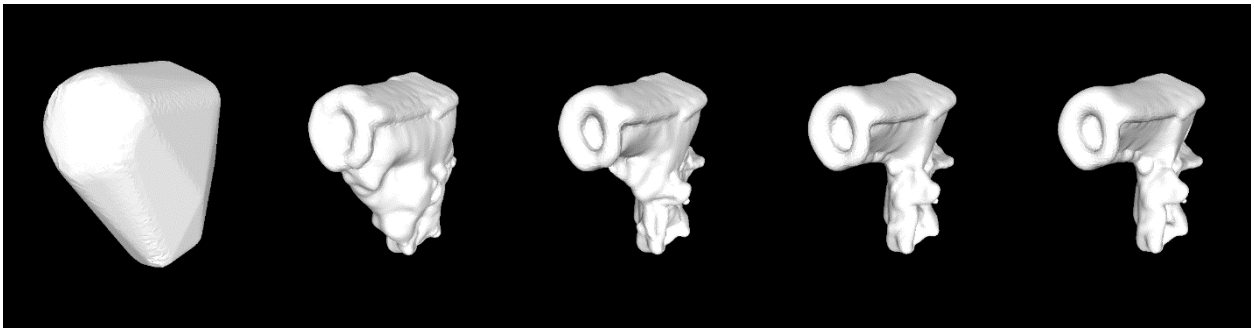
The number of envelope generation iterations is defined by the parameter param_completion_envelope_iterations.

Once we have an adapted envelope, we use its vertices to complete the point cloud, ignoring those that are close to the input points and applying a down voxel operation. To approximate color information, we assign to the completion points the color of the corresponding closest input points.

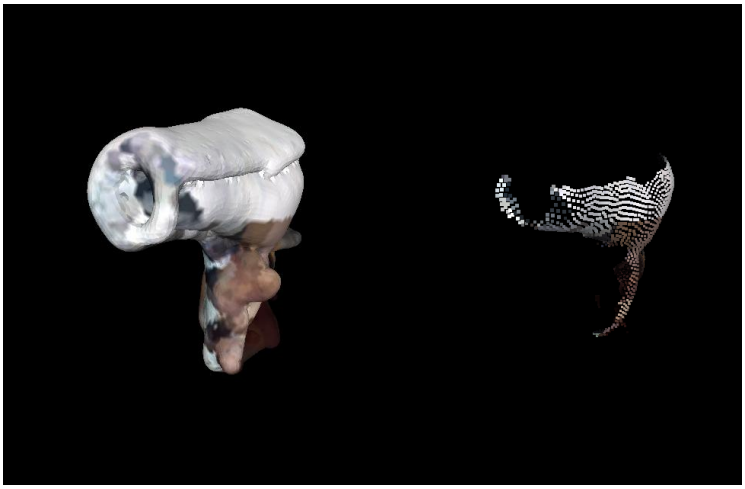
3.3 RESULTS

Notice that the completion points are sparse intentionally. This behavior helps to minimize overfitting of the meshing method to the non-existent surfaces.

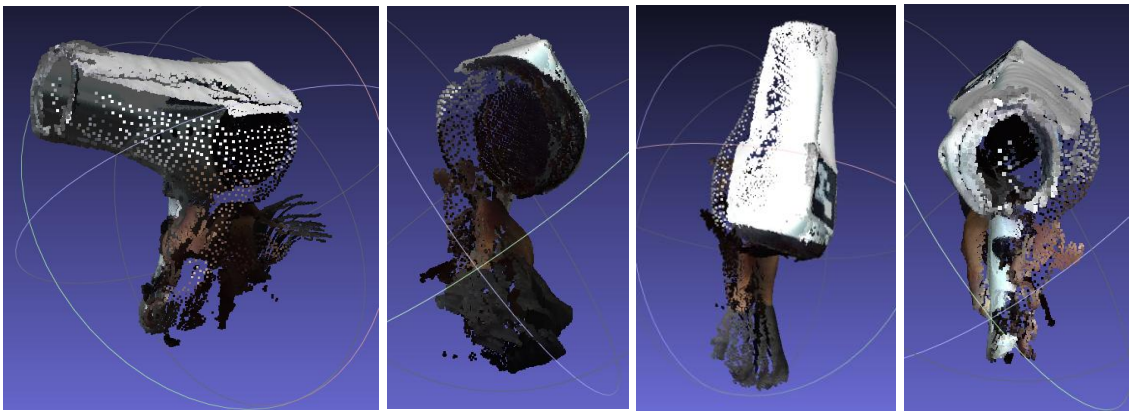
In the images below, we show the input point cloud and the iterative process of the envelope generation.



The images below show the expanded envelope with color mapping and the selected points from the envelope. These points are then down sampled and merged with the input point cloud, preserving normal and color information.



The following images show some captures of the completed point cloud, specifically computed for the following meshing operation.



4 MESHING

The entire pipeline was designed to produce point clouds optimized for meshing algorithms. To this end, we preserve color information and ensure normal consistency. Combined with our completion strategy, this enables the generation of point clouds that closely approximate watertight surfaces, ideal inputs for a variety of mesh reconstruction methods. For this reason, we propose applying the Screened Poisson Surface Reconstruction algorithm, which enables the construction of a smooth, watertight surface from noisy point clouds. This method balances data fidelity and smoothness by incorporating both point

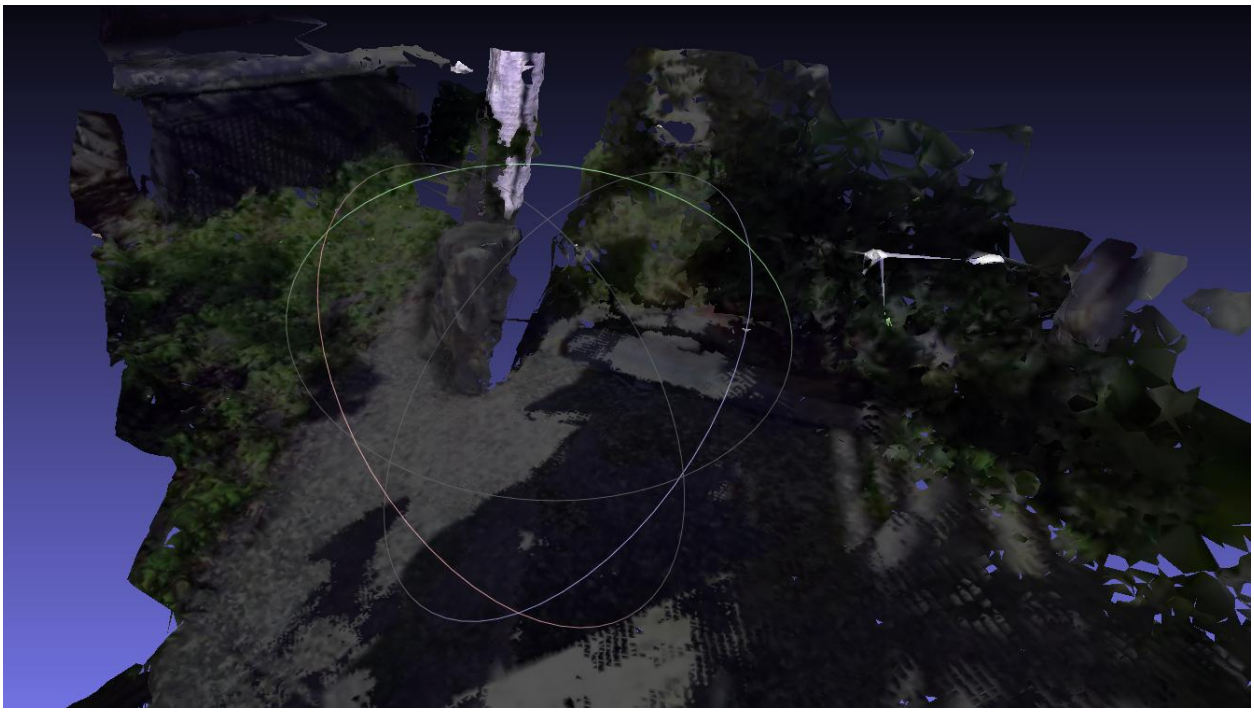
positions and normal vectors, making it particularly effective for producing high-quality meshes. More precisely, we use the MeshLab implementation with the corresponding default parameters. Some captures of the reconstructed mesh using the hair dryer are shown below.

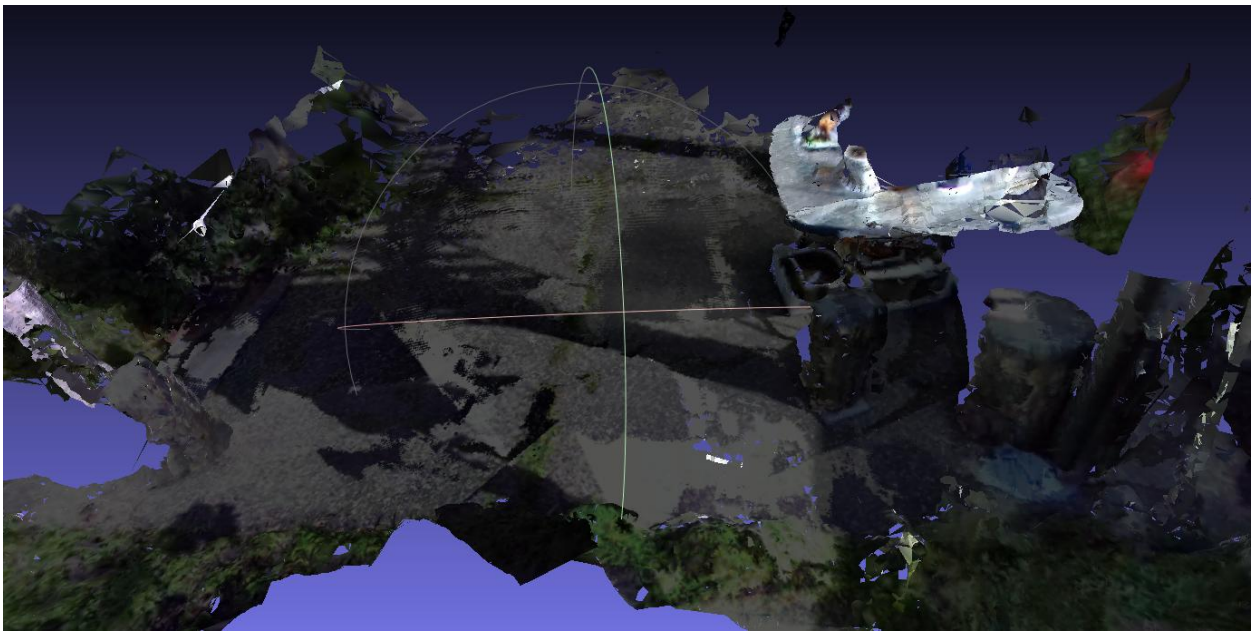
5 ADDITIONAL CONSIDERATIONS

The full pipeline was specifically designed to reconstruct objects. However, the implementation has an additional param to process outdoor scene, such as the yard model. This param is named param_is_outdoor_scene. When it is enabled the pipeline suffers the following modifications:

- For outlier removal, just the third step is considered.
- No completion operation is considered
- For meshing, we use the result of the outlier removal operation as input and the Ball Pivoting as the algorithm for reconstruction. The radius sizes we considered are 2x, 4x, 8x, and 16x, where x represents the average distance between points.

Some captures of the meshing result on the yard model are shown below:





6 EXECUTION COMMANDS

For the hair dryer model:

python pcp_app.py data/input_data/hair_dryer_part_1

For the yard model:

python pcp_app.py data/input_data/20250403_yard_10fps_vslam --param_registration_fitness_threshold 0.4 --param_registration_use_simple_coarse 1 --param_outlier_removal_use_per_point_cloud_checking 0 --param_outlier_removal_use_visibility_confidence 0 --param_is_outdoor_scene 1