

# GLOBALLY OPTIMAL REGISTRATION OF DENSE TERRESTRIAL LASER SCANS FROM COARSE SAMPLING

Dorian Kempf<sup>1,2</sup>, Guillaume Caron<sup>1,2</sup>, El Mustapha Mouaddib<sup>1</sup>, Fumio Kanehiro<sup>2</sup>

<sup>1</sup>Université de Picardie Jules Verne, MIS Laboratory  
Amiens, France

{dorian.kempf, guillaume.caron, mouaddib}@u-picardie.fr

<sup>2</sup>CNRS-AIST JRL (Joint Robotics Laboratory), IRL,

National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan

{f-kanehiro}@aist.go.jp

## Abstract

*3D point cloud registration is paramount for mapping complex environments. However, aligning overlapping clouds remains computationally demanding due to the increasing size and density of data from modern Terrestrial Laser Scanners. In this paper, we propose a lightweight, memory-efficient registration approach that operates directly on a coarse level of detail extracted from octree structures. Rather than relying on keypoints, we construct full coarse representations and associate each point with a descriptor to form global feature maps. We evaluate several point distribution strategies for building these representations and show that our approach significantly reduces processing time and memory usage while maintaining or improving registration accuracy.*

## 1. Introduction

3D point cloud registration is the cornerstone of creating detailed 3D environments. In robotics, it is essential for navigation and localization, where robots need an accurate representation of their surroundings to move autonomously [1]. Beyond robotics, applications include Virtual Reality and cultural heritage preservation, enabling immersive digital replicas of real-world locations, such as historical buildings, and allowing realistic exploration of remote or inaccessible sites even when scenes vary [2].

When it comes to mapping large-scale industrial environments or capturing complex architectural details, the vast amounts of data generated by recent Terrestrial Laser Scanners (TLS), raise challenges for the point cloud registration, *i.e.* aligning 3D point clouds from unknown positions and different poses. It consists of aligning a source point cloud  $\mathcal{P} = \{\dots, \mathbf{p}_i, \dots\}$ ,  $\mathbf{p}_i \in \mathbb{R}^3$  and a target

point cloud  $\mathcal{Q} = \{\dots, \mathbf{q}_i, \dots\}$ ,  $\mathbf{q}_i \in \mathbb{R}^3$  by minimizing the Euclidean distance between corresponding points:

$$\min_{\mathbf{R}, \mathbf{t}} \sum_i \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|_2, \quad (1)$$

where  $\mathbf{R} \in \text{SO}(3)$  represents the rotation applied to align the two point clouds, while  $\mathbf{t} \in \mathbb{R}^3$  denotes the translation, which are the variables to be optimized. It is classically done with the Iterative Closest Point (ICP) [3] algorithm, which is prone to local minima, particularly in the case of insufficient overlap or differences in the initial positions and orientations of large clouds, leading to misalignment and thus requiring good coarse registration to succeed. The ICP algorithm has a complexity of  $\mathcal{O}(k \cdot n^2)$  where  $k$  is the number of iterations and  $n$  the number of points. The complexity increases quadratically with the number of points in the cloud, resulting in significantly longer computation times for larger point clouds.

Detecting salient points with high geometric significance [4] has improved registration robustness. By matching the most similar parts, one obtains a coarse alignment that can then be refined using ICP. 3D point cloud structures such as octrees improve memory usage and allow for faster processing at varying levels of detail (LoD) by dividing space into octants [5–9]. The right LoD balances speed and accuracy, ensuring relevant features are preserved without overloading the system.

This paper presents a lightweight registration method tailored for efficient processing of dense, high-resolution TLS point clouds on CPU-based systems with limited memory. Rather than relying on traditional keypoint detection, our approach exploits a coarse level of an octree to derive a sparse yet informative representation of the scene. This de-

sign avoids loading the full-resolution cloud and bypasses keypoint extraction, enabling faster and more memory-efficient coarse alignment. We further analyze how the distribution of points across octree levels affects the quality of the initial registration prior to optional ICP-based refinement.

In what follows, Section 2 reviews common LoD techniques used in octree structures. Section 3 then details the point cloud registration methods considered, before introducing the evaluation metrics in Section 4. Section 5 presents the experimental results, followed by a dedicated discussion in Section 6, and concluding remarks in Section 7.

## 2. Related works

### 2.1. Point cloud distributions

An octree structures a dense 3D point cloud by subdividing the volume it spans in recursive layers of octants, each point having to be assigned to only one layer for memory efficiency. This operation can be performed by different approaches: (i) Random distribution, (ii) Voxel grid-based distribution, and (iii) KD-tree-based distribution, as shown in Figure 1.

**Random distribution** is a fast and simple method where a fixed number of points are randomly sampled from the cloud. While efficient, it leads to uneven spatial coverage: regions near the TLS are often overrepresented, while distant or low-density areas are under-sampled, potentially degrading registration robustness.

**Voxel grid distribution** [10] improves uniformity by dividing space into voxels and retaining one point per occupied cell. This ensures better spatial balance, especially in sparse regions, and offers a good trade-off between runtime and coverage quality.

**KD-tree distribution** enforces uniformity through a hierarchical partitioning scheme. Seed points are selected and neighbors are grouped recursively using fixed-radius searches, producing a Poisson-like distribution [9]. Though effective, this method is more computationally expensive due to repeated searches and updates.

Octrees have long been essential for efficiently handling and rendering large-scale point clouds, especially in contexts such as TLS, where individual scans may contain tens or even hundreds of millions of points. Their hierarchical nature implements LoD to load only relevant subsets of data based on the view frustum [6]. This makes octrees a foundational data structure for enabling scalable processing on systems with limited computational and memory resources. Over the years, various approaches have considered spherical representations [5], nested octrees [7] or Modifiable Nested Octrees [8] thanks to using a regular grid to organize points within each node. The concept was later re-

vised in the octree construction by assigning each point to a unique octree level while maintaining a minimum distance between them, enabling uniform subsampling [9] and for dynamic octrees [11]. This naturally leads to the use of octrees for efficient point cloud registration in the context of 3D reconstruction and robotics.

### 2.2. Octree-based registration

In the context of robotics, octrees are widely used in Simultaneous Localization And Mapping (SLAM) to manage large-scale global maps by modeling the environment memory-efficiently. This approach enables the rapid removal of dynamic obstacles while maintaining real-time performance [12, 13]. Octree-based fusion techniques have contributed in facilitating map merging for multi-robot systems, resulting in more accurate shared environment representations [14]. Vlamincq et al. [15] introduced a coarse-to-fine octree-based ICP algorithm, effectively reducing computational costs during initial stages. Efficient nearest neighbor search has also been addressed using cached KD-trees to compute closest points iteratively [16]. Kim et al. [17] developed a method to align 3D point cloud data from building projects with 3D CAD models. Han et al. [18] further improved registration by combining a hierarchical search with an octree-based ICP algorithm and introducing a heuristic escape strategy to avoid local minima. A Gaussian Octree-based GICP leveraged its pruning capabilities to accelerate nearest neighbor search and applied incremental Gaussian distribution updates to reduce tree construction time [19]. These contributions highlight the versatility of

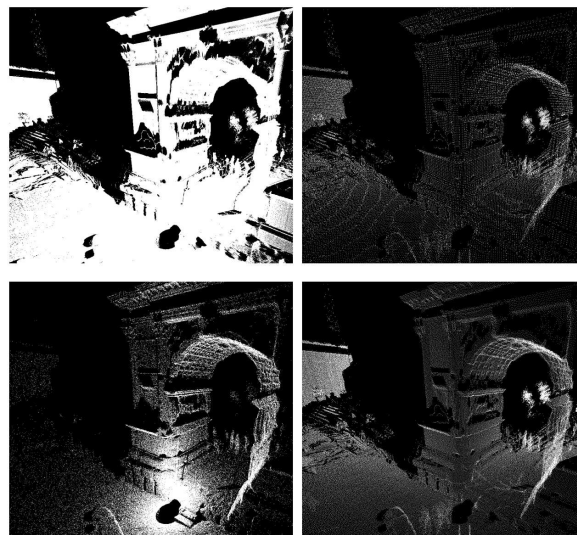


Figure 1. Full cloud (top left) with 23 million points and octree level 0 with about 400k points for distributions Random (Bottom left), Voxel (Top right) and KD-tree (Bottom right).

octrees in point cloud registration, particularly for managing levels of detail and optimizing efficiency. While effective in robotics, these methods are not optimized for large TLS point clouds. Using multiple LoDs in octrees helps avoid memory saturation and accelerates the processing.

### 2.3. Point cloud features

In point cloud registration, converging to the optimal solution while avoiding local minima is a major challenge. Keypoints with distinctive characteristics, such as those obtained with the so-called SIFT detector adapted to point clouds [20], and local descriptors help achieve initial alignment, refined with methods like ICP. Descriptors are either handcrafted or trained. Handcrafted descriptors, like Fast Point Feature Histograms (FPFH) [21], extensively evaluated [22], have been used for coarse alignment [4, 23, 24], improving registration success.

Learning-based descriptors [25, 26] show greater accuracy but often face generalization issues, not only due to sensor variability but also because of scene variability between training and real-world environments. Recent work [27] has attempted to address these limitations by improving the generalization capabilities of learned descriptors. This limitation becomes critical when dealing with dense and diverse TLS datasets. Moreover, training such models typically demands massive amounts of data, hundreds of gigabytes for LiDAR or RGB-D clouds, and potentially several terabytes for high-resolution TLS scenes, resulting in significant computational and memory burdens.

After feature extraction, initial alignment relies on feature matching, a crucial step in the registration pipeline. To improve robustness, some methods [28, 29] assign multiple candidate matches per point, increasing the chances of finding correct correspondences. Among them, PointDSC [29] integrates a Neural Spectral Matching (NSM) module that enforces spatial consistency, reducing the impact of outlier correspondences. It supports standard 3D local descriptors such as FPFH as well as learning-based descriptors like Fully Convolutional Geometric Features (FCGF) [25]. However, these strategies greatly increase the number of correspondences to process and store, exacerbating memory consumption and computational cost.

## 3. Proposed registration method

To align dense and high-resolution TLS scans, we propose a registration pipeline that leverages a coarse LoD from an octree representation. Instead of relying on full-resolution point clouds and computationally expensive keypoint detection, we directly operate on a subsampled representation extracted from a specific octree level. This approach significantly reduces memory usage by avoiding the need to load tens of millions of points into RAM, while still preserving sufficient geometric information for robust and scalable

alignment. Both the feature-based matching and the ICP refinement are performed at the same coarse octree level, further limiting computational and memory costs. The overall pipeline enables rapid data access and efficient registration on CPU-based systems with limited resources.

The coarse LoD can be generated using different strategies such as voxel grid, KD-tree, or random sampling (see Sec. 2.1). Each strategy results in a different spatial distribution of points, which has a direct influence on the quality of registration. For successful matching, it is crucial that the LoD provides a representative overview of the entire scene, with descriptors extracted from well-distributed points across both near and distant regions. We select a coarse octree level that significantly reduces the number of points and accelerates computation while aiming to preserve a representative spatial distribution when the chosen strategy allows it. This is especially relevant in large-scale TLS scenarios, such as outdoor scans over 100 meters, where sensor positions are far apart and overlap between scans may be minimal.

Random sampling tends to over-represent regions near the sensor, where the point density is highest, and under-represent distant regions. This imbalance can degrade matching robustness in scenes with sparse overlap or large viewpoint changes, as fewer common points remain for correspondence estimation. Additionally, descriptor computations become biased toward geometrically redundant areas, adding unnecessary computational overhead. In contrast, voxel grid and KD-tree strategies produce more uniform spatial coverage, allowing for improved geometric diversity and more stable matching.

Figure 2 illustrates the overall pipeline. We compare the classical approach based on keypoint detection and matching with our strategy that computes descriptors directly on a uniformly subsampled cloud obtained from the octree.

### 3.1. Keypoints versus coarse LoD

In our convention, *Level 0* of the octree refers to the root node, encompassing the entire point cloud and yielding the coarsest resolution. Extracting points from this level provides a natural form of geometric subsampling. We denote the reduced clouds as  $\mathcal{P}' = \{\dots, \mathbf{p}'_i, \dots\}$ ,  $\mathbf{p}'_i \in \mathbb{R}^3$  for the source, and  $\mathcal{Q}' = \{\dots, \mathbf{q}'_i, \dots\}$ ,  $\mathbf{q}'_i \in \mathbb{R}^3$  for the target.

While classical registration pipelines often rely on handcrafted keypoint detectors such as SIFT [20], these methods are typically sensitive to parameter tuning and may fail to extract a sufficient number of reliable features in sparse or unstructured scenes. In dense outdoor point clouds, thousands of keypoints can be detected but the process remains computationally expensive, often requiring several seconds per scan and the parameters must be carefully adjusted to achieve satisfactory performance.

In contrast, directly using all points from  $\mathcal{P}'$  and  $\mathcal{Q}'$

avoids the variability of keypoint detectors and ensures spatial information is retained across the entire scene. This is particularly advantageous in scenes with large depth variations or limited overlapping areas, where keypoint-based methods may lead to poor initial coarse alignment for registration. By operating on the coarse LoD, we ensure rapid access to a uniformly distributed set of features, improving robustness and efficiency.

Additionally, since the LoD is constructed via an explicit and controllable strategy (voxel, KD-tree, or random), we gain full control over the resulting point distribution, an advantage not afforded by learned or handcrafted keypoint detectors. This flexibility allows us to adapt the LoD generation to the geometry of each dataset, improving generalization and consistency across diverse environments.

### 3.2. Descriptor computation to make the feature map

To extract geometric information, we compute the FPFH descriptor at each point of the coarse clouds  $\mathcal{P}'$  and  $\mathcal{Q}'$ . FPFH computes relationships between a point and its neighbors within a radius  $r \in \mathbb{R}^+$ , resulting in complexity  $\mathcal{O}(n \cdot k)$ , where  $n \in \mathbb{N}$  is the number of points and  $k \in \mathbb{N}$  the number of neighbors.

Computing descriptors on full-resolution clouds is expensive both in time and memory. By computing them at the coarsest level, we achieve a significant speedup while preserving essential local geometric information. The resulting feature maps are denoted  $\mathbf{P}\mathbf{f}$  and  $\mathbf{Q}\mathbf{f}$ .

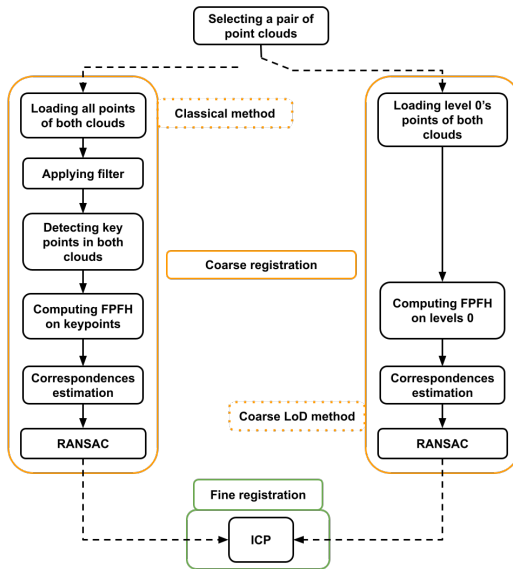


Figure 2. Pipeline of the registration process. Classical method (left) uses keypoints, while our method (right) operates directly on a coarse octree level.

### 3.3. Registration

We establish initial correspondences between  $\mathbf{P}\mathbf{f}$  and  $\mathbf{Q}\mathbf{f}$  using nearest-neighbor search and reciprocal filtering, as in [30]. These correspondences are then refined using RANSAC, which estimates a rigid transformation  $\mathbf{T}_f \in \text{SE}(3)$  by minimizing distances between matched points. RANSAC randomly samples triplets of correspondences, estimates a transformation via SVD, and selects the one yielding the largest number of inliers.

To further refine alignment, we apply ICP between  $\mathcal{P}'$  and  $\mathcal{Q}'$ , optimizing the following objective:

$$\min_{\mathbf{R}_{icp}, \mathbf{t}_{icp}} \sum_i \|\mathbf{p}'_i - \mathbf{R}_{icp}\mathbf{q}'_i - \mathbf{t}_{icp}\|_2, \quad (2)$$

where  $\mathbf{R}_{icp} \in \text{SO}(3)$  and  $\mathbf{t}_{icp} \in \mathbb{R}^3$ . The transformation  $\mathbf{T}_f$  obtained from the initial matching step is used to initialize ICP by pre-aligning  $\mathcal{P}'$  and  $\mathcal{Q}'$  before optimization. The final transformation  $\mathbf{T} \in \text{SE}(3)$  is then obtained by composing both steps:

$$\mathbf{T} = \mathbf{T}_f \cdot \mathbf{T}_{icp}.$$

To ensure robust convergence during fine alignment, we adopt a multi-stage ICP strategy in which the distance threshold for nearest-neighbor correspondence search is progressively reduced across iterations. We typically start with a threshold of 1.5 meters to tolerate moderate initial misalignments, and gradually reduce it in subsequent iterations to refine correspondences and improve final accuracy. This hierarchical approach stabilizes convergence and enhances precision, particularly in scenes with significant initial misalignment.

## 4. Evaluation Metrics

To evaluate the quality of point cloud registration, we use several metrics. The Root Mean Square Error (RMSE) evaluates the quality of the estimated transformation  $\mathbf{T} \in \text{SE}(3)$  by comparing the transformed source points with the ground truth source points. We used the same formula as [26]:

$$\text{RMSE}(\mathcal{P}, \mathbf{T}, \mathbf{T}^*) = \frac{1}{|\mathcal{P}|} \sqrt{\sum_{\mathbf{p} \in \mathcal{P}} \|\mathbf{T}(\mathbf{p}) - \mathbf{T}^*(\mathbf{p})\|^2}, \quad (3)$$

where  $|\mathcal{P}|$  represents the total number of points in the source cloud,  $\mathbf{T}$  the estimated transformation and  $\mathbf{T}^*$  the ground-truth transformation.

The Registration Recall (RR) counts, for a set of point cloud pairs, the number of successful registrations, *i.e.*, the number of times the RMSE is below  $\tau_1 \in \mathbb{R}_+$ . The Registration Recall expression is:

$$\text{RR}(\mathcal{H}) = \frac{1}{|\mathcal{H}|} \sum_{i=1}^{|\mathcal{H}|} \text{RMSE}(\mathcal{P}_i, \mathbf{T}_i, \mathbf{T}_i^*) < \tau_1, \quad (4)$$

where  $\mathcal{H}$  represents the scan pairs,  $|\mathcal{H}|$  the number of pairs that can be registered and  $\mathcal{P}_i$  the source cloud number  $i$ .

To assess transformation accuracy, we compute the relative transformation  $\Delta\mathbf{T}$ :

$$\Delta\mathbf{T} = \mathbf{T}(\mathbf{T}^*)^{-1} = \begin{bmatrix} \Delta\mathbf{R} & \Delta\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \quad (5)$$

where  $\Delta\mathbf{R}$  is the relative rotation matrix and  $\Delta\mathbf{t}$  is the relative translation vector.

From this we extract the Relative Rotation Error (RRE) and the Relative Translation Error (RTE). The RRE quantifies the angular difference between the estimated and the ground-truth orientations:

$$\text{RRE} = \arccos\left(\frac{\text{tr}(\Delta\mathbf{R}) - 1}{2}\right), \quad (6)$$

where  $\text{tr}(\Delta\mathbf{R})$  is the trace of the relative rotation matrix.

The RTE measures the Euclidean distance between translations:

$$\text{RTE} = \|\Delta\mathbf{t}\|, \quad (7)$$

where  $\|\Delta\mathbf{t}\|$  is the Euclidean norm of the relative translation vector.

RRE and RTE are standard metrics for evaluating rigid transformation accuracy [23, 27].

## 5. Experiments

### 5.1. Datasets

The experimental data were acquired by TLS and include three distinct datasets to evaluate the proposed approach. The first dataset, the *ETH TLS Arch dataset*<sup>1</sup>, captures a Roman arch with five scans taken from different positions around the structure. Each scan spans approximately  $150 \text{ m} \times 150 \text{ m} \times 100 \text{ m}$ , containing from 25 to 30 million points, with a 30–40% overlap ratio. Challenges include vegetation, dynamic artifacts (e.g., moving people), significant occlusions, and transformations, with Euclidean distances between scans ranging from 10 to 30 meters. The dataset includes 20 pairwise alignments with ground truth transformations provided.

The second dataset, called *Office*, is captured in an indoor laboratory setting. This dataset features five scans of an office space  $20 \text{ m} \times 18 \text{ m} \times 6 \text{ m}$  with 37 million points per scan and higher overlap compared to the Arch dataset. The scans, taken at an average Euclidean distance of 3.5 meters between each pose, provide a controlled environment with minimal occlusions. As with the Arch dataset, there are 20 pairwise alignments. Figure 3 illustrates the scanning configuration.

<sup>1</sup>[prs.igp.ethz.ch/research/completed\\_projects/automatic\\_registration\\_of...](https://prs.igp.ethz.ch/research/completed_projects/automatic_registration_of...)

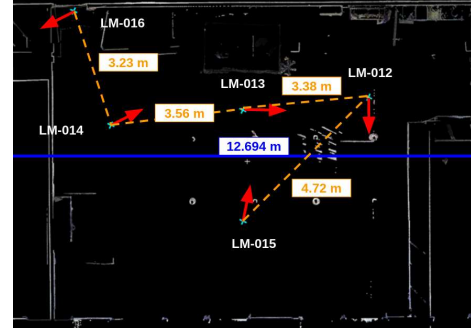


Figure 3. Overview of the Office, with initial position and orientation of scans and scale

The third dataset, called *Outside*, includes four scans of a large-scale static scene, covering  $130 \text{ m} \times 100 \text{ m} \times 27 \text{ m}$ . Two scans are high-resolution ( $\sim 70$  million points), and two are lower-resolution ( $\sim 22$  million points). The four scans, with 4 to 12 meters of positional differences and significant overlap, yield 12 pairwise alignments. Figure 4 shows the datasets used.

For the Arch dataset, ground truth transformations are provided. For the Office and Outside datasets, scanned with a Leica TLS RTC 360 (range up to 130 m), Leica Cyclone software’s registration is considered as the ground truth.

### 5.2. Octree Levels-of-Detail

Operating directly on a low LoD extracted from the octree drastically reduces memory usage compared to full-resolution loading, while also eliminating the time typically required for downsampling and keypoint extraction. By bypassing these steps, we accelerate the preprocessing phase and reduce the memory footprint, which is particularly advantageous when hardware resources are limited.

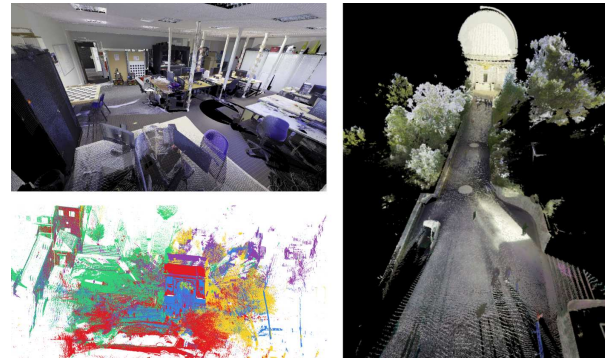


Figure 4. Used datasets: Office (Top left), Arch (Bottom left) and Outside (Right)

While classical methods require loading the full-resolution cloud and allocating additional memory for intermediate downsampled clouds or keypoint structures, our approach operates directly on the octree-extracted LoD, providing a lightweight yet descriptive subset for registration without additional preprocessing. This explains the significant reductions observed in Table 1, where memory consumption is consistently reduced by an order of magnitude across datasets of varying size and density. Importantly, this efficiency gain does not compromise the quality of the initial alignment, since the octree-level points remain sufficiently informative to guide robust registration, as further sections show.

Table 1. Memory and point count comparison between full-resolution clouds and their Level 1 octree representations (second coarsest level). Values correspond to the combined loading of two clouds in each case, as typically required during pairwise registration.

Dataset	Full (pts)	Lv 1 (pts)	Memory Full (GB)	Memory Lv 1 (GB)
Arch	23 M	350 k	3.0	0.3
Office	38 M	500 k	4.6	0.5
Outside	70 M	1.0 M	7.0	0.6

### 5.3. Multi-stage ICP refinement

We evaluate the multi-stage ICP strategy against a standard single-pass ICP on the Arch dataset, the most challenging dataset with initial RMSEs of 1–3 meters from the ground truth (see Sec. 5.1). As shown in Table 2, using a fixed threshold of 1.5 m often yields imprecise alignment, while reducing it to 0.5 m improves accuracy but may miss valid matches. Setting it too low, such as 0.05 m, degrades convergence by discarding correct correspondences. In contrast, the multi-stage ICP consistently achieves lower RMSE by progressively refining correspondences without additional computational cost. The threshold starts at 1.5 m and is halved at each stage down to 0.01 m, allowing up to 10k iterations per stage, with early stopping triggered if convergence is reached.

## 5.4. Registration results

### 5.4.1. Experimental setup

We compare three point distribution methods: Voxel, KD-tree, and Random (see Sec. 2). We perform registration tests on the 20 configurations of the Arch dataset as well as on the 20 from the Office and the 12 from the Outside datasets. In order to find a balance between keeping calculation times as low as possible and maintaining satisfac-

Table 2. Average RMSE (in meters) obtained for different ICP strategies on the Arch dataset, with initial RMSEs ranging between 1 and 3 meters.

ICP strategy	Average RMSE (m)
Fixed threshold 1.5 m	0.210
Fixed threshold 0.5 m	0.072
Fixed threshold 0.05 m	1.204
Multi-stage (progressive threshold)	0.020

tory results, tests were carried out on the lowest LoD, representing less than 1% of the initial cloud. Loading the lowest LoD allows obtaining a subsampled point cloud in just a few milliseconds, compared to several seconds required to load the full point cloud and apply a filtering process for subsampling. The tests are conducted with an equivalent number of points for the three methods to ensure a fair comparison and are performed as described in Section 3. All experiments were conducted on a workstation equipped with an Intel i9-13950HX processor and 32 GB of RAM. However, to simulate a more constrained computational environment, the available memory was deliberately limited to 16 GB during all tests. The features are extracted using the PCL library [31] in C++. Once the initial alignment is obtained, results are refined using our multi-stage ICP refinement strategy, which leverages the ICP implementation provided by the Open3D library [32], to improve fine alignment precision. All RR metrics are computed with  $\tau_1 = 0.05$  m. Results on the Arch, Office, and Outside datasets, including the RR, average run-time, RRE, and RTE for each method, are reported in Table 3.

### 5.4.2. Impact of the sampling method

Table 3 presents the registration results across the Arch, Office, and Outside datasets, confirming the clear advantage of structured sampling methods over random selection, which often fails to produce meaningful registrations while incurring higher computation times. Among the structured methods, KD-tree sampling achieves slightly higher accuracy than voxel sampling, as reflected by lower RRE and RTE values, likely due to its adaptive selection better preserving geometric details relevant for alignment. However, voxel-based sampling remains marginally faster while maintaining comparable accuracy, making it a compelling choice when computational speed is prioritized. Overall, both voxel and KD-tree strategies yield robust, scalable registration while balancing precision and efficiency. The reported RR, RRE, and RTE reflect the full pipeline, including final alignment from our multi-stage ICP refinement (see Sec. 3.3).

Table 4 shows that the random method increases FPFH computation time due to uneven point density, while the

Table 3. Registration results, with about 45k points per cloud for Arch, 35k points for Office and about 55k points per cloud for Outside dataset. For each dataset, the best result is in **blue**, and the second-best in **bold**.

Dataset	Method	RR (%)	RRE (mdeg)	RTE (mm)	Time (s)
Arch	Voxel	<b>85</b>	<b>31</b>	<b>11.6</b>	<b>9.8</b>
	KD-tree	<b>85</b>	<b>33</b>	<b>10.1</b>	<b>13.7</b>
	Random	0	/	/	47
Office	Voxel	100	<b>73</b>	<b>9</b>	<b>7.3</b>
	KD-tree	100	<b>45</b>	<b>7.7</b>	<b>8.9</b>
	Random	0	/	/	14.5
Outside	Voxel	100	26	<b>3</b>	<b>11.7</b>
	KD-tree	100	<b>14</b>	<b>3.2</b>	<b>13.7</b>
	Random	50	<b>23</b>	10	108

more uniform distributions of voxel and KD-tree sampling avoid this overhead (see Sec. 3).

#### 5.4.3. Keypoint-based registration performances

The impact of the number of SIFT keypoints on RR is shown in Table 5 for the challenging Arch dataset. In these tests, SIFT keypoints are extracted from a voxel-filtered cloud, with FPFH descriptors computed at the keypoint locations before correspondence search and RANSAC, following the pipeline in Section 3 but using keypoints instead of a uniformly subsampled LoD. As expected, increasing the number of detected keypoints leads to longer computation times, due to the higher complexity of the correspondence search process. However, this increase does not translate into improved registration results: the highest RR is achieved with a moderate number of keypoints, while both lower and higher counts result in degraded performance. This can be attributed to the difficulty of ensuring consistent keypoint detection across source and target clouds, especially in the presence of partial overlap, noise, or surface variations. In contrast, voxel sampling ensures uniform spatial coverage, enhancing matching reliability and efficiency.

Table 4. Average Run-time for FPFH descriptor calculation on Arch dataset on two different levels with about 40k and 300k points. The two best results are shown in **blue** and **bold**.

Method	40 k points (s)	300 k points (s)
Voxel	<b>0.747</b>	<b>30.67</b>
KD-tree	<b>0.836</b>	<b>32.69</b>
Random	9.24	1187.53

Table 5. Registration results on the Arch dataset ( $\sim 400k$  points) using the classical pipeline on voxel-filtered clouds, with SIFT keypoints (PCL) extracted under varying parameters, yielding different keypoint counts.

Average SIFT keypoints	RR (%)	Average time (s)
5k	0	/
15k	70	18.5
35k	55	27.4

#### 5.4.4. Comparison with the state-of-the-art keypoint-based approach

Table 6 compares our method with HL-MRF [24] across different input resolutions on the evaluated datasets. HL-MRF requires loading high-resolution clouds to extract keypoints for processing, whereas our method operates directly on an appropriately subsampled cloud for each dataset. Despite operating on a larger number of points for matching, our method achieves accuracy comparable to or better than HL-MRF while being significantly faster. For instance, on the Arch dataset, our method achieves a higher RR while running more than ten times faster. Similar trends are observed on Office and Outside, with high precision and lower runtimes. These results confirm that our strategy effectively balances accuracy, scalability, and computational efficiency across diverse scenes.

#### 5.4.5. Comparison with the state-of-the-art learning-based method

To further assess the generalization and scalability of our method, we compare it with PointDSC [29], a deep learning-based registration approach, as shown in Table 7. For our experiments, we used PointDSC in combination with FCGF [25], employing the pretrained weights provided by the authors without retraining to test generalization. We used the 3DMatch-trained weights for the Office dataset and the KITTI-trained weights for the KITTI dataset on sequences 8, 9, and 10 and for the Arch dataset, with all evaluations performed on CPU to ensure a fair comparison. Due to memory constraints, PointDSC could only process point clouds limited to 15k features, restricting its use on large-scale scenes. In contrast, our method handled much larger inputs, processing up to 30k points on Office and KITTI and 45k on Arch without memory issues, while remaining faster in practice despite using more points for matching. Performance results reflect the training distribution of PointDSC. The method performs well on the Office dataset, which shares characteristics with 3DMatch, and on the KITTI dataset, which was also included in its training data. However, its performance drops significantly on the Arch dataset, which differs in structure and density and was

Table 6. Comparison with HL-MRF [24] across three datasets and varying numbers of input points. Best values per dataset are in **blue**, second-best in **bold**.

Dataset	Method	# Points	# Keypoints	RR (%)	RRE (mdeg)	RTE (mm)	Time (s)
Arch	Ours (Voxel)	100k	–	75	34	8.2	24
		45k	–	<b>85</b>	31	11.6	9.8
		25k	–	<b>80</b>	<b>29</b>	8	<b>7.3</b>
		<b>15k</b>	–	75	<b>31</b>	8	<b>5.1</b>
	HL-MRF	430k	30k	80	39.6	<b>7.4</b>	903
		430k	5k	70	44.2	<b>7.9</b>	144
		215k	2.5k	50	60	21	58.2
		150k	1.5k	15	225	62	36.4
Office	Ours (Voxel)	120k	–	100	<b>32</b>	<b>4.3</b>	<b>19.7</b>
		55k	–	100	71	7.8	<b>5.2</b>
	HL-MRF	140k	1.5k	100	<b>52</b>	<b>7.2</b>	35.5
		30k	1k	100	64	8.6	34.7
Outside	Ours (Voxel)	15k	–	100	37	9.4	<b>5.6</b>
		7k	–	100	<b>26</b>	<b>5.1</b>	<b>3.7</b>
	HL-MRF	120k	650	100	<b>26</b>	<b>5.8</b>	45
		120k	300	100	27	6.2	110

not seen during training. In contrast, our method maintains high accuracy across all datasets, including the challenging Arch scenes, without requiring learned features or retraining, while supporting larger inputs at lower computational cost.

Table 7. Registration recall (%) and computation time (s) for PointDSC and our method across three datasets. Best recall values are shown in **blue**.

Dataset	PointDSC		Ours (voxel)	
	RR (%)	Time (s)	RR (%)	Time (s)
Office	100	6.4	100	<b>5.2</b>
KITTI	89	10.7	<b>99</b>	<b>6.1</b>
Arch	0	/	<b>85</b>	<b>9.8</b>

## 6. Discussion

Our results highlight the importance of point distribution during octree construction: voxel and KD-tree-based sampling consistently outperform random selection by ensuring uniform scene coverage, crucial for large-scale registration. A key advantage of our approach is avoiding full-resolution cloud loading, reducing memory usage and preprocessing time. As shown in Table 1, it achieves up to 10× RAM savings, making it suitable for large outdoor datasets and low-memory platforms. The proposed multi-stage ICP enhances accuracy and robustness by progressively refining alignment with decreasing thresholds, mitigating local minima while remaining efficient, and enabling sub-centimeter

errors on challenging datasets despite downsampling. However, the current implementation relies on exhaustive correspondence matching, whose quadratic complexity is a bottleneck beyond 100k points. Future work will explore hierarchical multi-resolution registration with multiple octree levels for better robustness in low-overlap cases and more efficient correspondence search to cut computation without sacrificing accuracy.

## 7. Conclusion

We presented a lightweight, memory-efficient method for registering high-resolution TLS point clouds using a coarse octree representation. By avoiding keypoint extraction and full-resolution loading, our pipeline lowers computational cost without compromising alignment quality. Experiments show that KD-tree and voxel-based point distribution offer the best trade-off between speed and accuracy. Compared to keypoint and learning-based methods, our approach achieves competitive results on large-scale outdoor scenes with low memory usage, making it a practical, scalable solution for 3D reconstruction in resource-constrained settings and suitable for integration into robotics or long-term mapping workflows.

## 8. Acknowledgement

The authors thank the ANR - FRANCE (French National Research Agency) for its financial support of the SAMURAI project n°ANR-21-CE33-0014.

## References

- [1] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, “KISS-ICP: In Defense of Point-to-Point ICP-Simple, Accurate, and Robust Registration If Done the Right Way,” *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1029–1036, 2023. 1
- [2] E. Mouaddib, A. Pamart, M. Pierrot-Deseilligny, and D. Girardeau-Montaut, “2D/3D data fusion for the comparative analysis of the vaults of Notre-Dame de Paris before and after the fire,” *J. of Cultural Heritage*, vol. 65, pp. 221–231, Jan. 2024. 1
- [3] P. J. Besl and N. D. McKay, “Method for registration of 3-D shapes,” in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606. 1
- [4] A. Gressin, C. Mallet, J. Demantké, and N. David, “Towards 3D lidar point cloud registration improvement using optimal neighborhood knowledge,” *ISPRS J. of Photogrammetry and Remote Sensing*, vol. 79, pp. 240–251, 2013. 1, 3
- [5] S. Rusinkiewicz and M. Levoy, “QSplat: A multiresolution point rendering system for large meshes,” in *Proc. of Annual Conf. on Computer Graphics and Interactive Techniques*, 2000, pp. 343–352. 1, 2
- [6] C. Dachsbacher, C. Vogelgsang, and M. Stamminger, “Sequential point trees,” *ACM Trans. on Graphics*, vol. 22, no. 3, pp. 657–662, 2003. 2
- [7] M. Wimmer and C. Scheiblauer, “Instant Points: Fast Rendering of Unprocessed Point Clouds,” in *PBG@ SIG-GRAPH*, 2006, pp. 129–136. 2
- [8] C. Scheiblauer, “Interactions with Gigantic Point Clouds,” Ph.D. dissertation, Technische Universität Wien, 2014. 2
- [9] M. Schütz, S. Ohrhallinger, and M. Wimmer, “Fast Out-of-Core Octree Generation for Massive Point Clouds,” *Computer Graphics Forum*, vol. 39, no. 7, pp. 155–167, 2020. 1, 2
- [10] L. Kang, J. Jiang, Y. Wei, and Y. Xie, “Efficient Randomized Hierarchy Construction for Interactive Visualization of Large Scale Point Clouds,” in *Proc. of IEEE Int. Conf. on Data Science in Cyberspace*, 2019, pp. 593–597. 2
- [11] J. Zhu, H. Li, Z. Wang, S. Wang, and T. Zhang, “i-Octree: A Fast, Lightweight, and Dynamic Octree for Proximity Search,” in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2024, pp. 12 290–12 296. 2
- [12] N. Fairfield, G. Kantor, and D. Wettergreen, “Real-time SLAM with Octree Evidence Grids for Exploration in Underwater Tunnels,” *J. of Field Robotics*, vol. 24, no. 1-2, pp. 03–21, 2007. 2
- [13] A. Hornung, K. M. Wurm, and M. Bennewitz, “Humanoid robot localization in complex indoor environments,” in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010, pp. 1690–1695. 2
- [14] J. Jessup, S. N. Givigi, and A. Beaulieu, “Merging of octree based 3D occupancy grid maps,” in *Proc. of IEEE Int. Systems Conf.*, 2014, pp. 371–377. 2
- [15] M. Vlaminck, H. Luong, and W. Philips, “Multi-resolution ICP for the efficient registration of point clouds based on octrees,” in *Proc. of IAPR Int. Conf. on Machine Vision Applications*, 2017, pp. 334–337. 2
- [16] A. Nuchter, K. Lingemann, and J. Hertzberg, “Cached k-d tree search for ICP algorithms,” in *Proc. of Sixth Int. Conf. on 3-D Digital Imaging and Modeling (3DIM 2007)*, 2007, pp. 419–426. 2
- [17] C. Kim, H. Son, and C. Kim, “Fully automated registration of 3D data to a 3D CAD model for project progress monitoring,” *Automation in Construction*, vol. 35, pp. 587–594, 2013. 2
- [18] J. Han, P. Yin, Y. He, and F. Gu, “Enhanced ICP for the registration of large-scale 3D environment models: An experimental study,” *Sensors*, vol. 16, no. 2, p. 228, 2016. 2
- [19] Z. Yu, W. Yuan, H. Zhao, H. Zhuang, and M. Yang, “GOG-ICP: A Real-time Gaussian Octree-based GICP Method for Faster Point Cloud Registration,” in *Proc. of IEEE Int. Conf. on Real-time Computing and Robotics (RCAR)*, 2024, pp. 112–117. 2
- [20] R. Hänsch, T. Weber, and O. Hellwich, “Comparison of 3D interest point detectors and descriptors for point cloud fusion,” *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-3, pp. 57–64, 2014. 3
- [21] R. B. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (FPFH) for 3D registration,” in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 3212–3217. 3
- [22] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, “A comprehensive performance evaluation of 3D local feature descriptors,” *Int. J. of Computer Vision*, vol. 116, pp. 66–89, 2016. 3
- [23] Y. Pan, B. Yang, F. Liang, and Z. Dong, “Iterative global similarity points: A robust coarse-to-fine integration solution for pairwise 3D point cloud registration,” in *Proc. of Int. Conf. on 3D Vision*, 2018, pp. 180–189. 3, 5
- [24] H. Wu, L. Yan, H. Xie, P. Wei, and J. Dai, “A hierarchical multiview registration framework of TLS point clouds based on loop constraint,” *ISPRS J. of Photogrammetry and Remote Sensing*, vol. 195, pp. 65–76, 2023. 3, 7, 8
- [25] C. Choy, J. Park, and V. Koltun, “Fully convolutional geometric features,” in *Proc. of IEEE/CVF Int. Conf. on Computer Vision*, 2019, pp. 8958–8966. 3, 7
- [26] H. Yu, Z. Qin, J. Hou, M. Saleh, D. Li, B. Busam, and S. Ilic, “Rotation-invariant transformer for point cloud matching,” in *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2023, pp. 5384–5393. 3, 4
- [27] F. Poiesi and D. Boscaini, “Learning General and Distinctive 3D Local Deep Descriptors for Point Cloud Registration,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3979–3985, 2022. 3, 5
- [28] L. Yan, P. Wei, H. Xie, J. Dai, H. Wu, and M. Huang, “A New Outlier Removal Strategy Based on Reliability of Correspondence Graph for Fast Point Cloud Registration,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, p. 1–17, 2022. 3
- [29] X. Bai, Z. Luo, L. Zhou, H. Chen, L. Li, Z. Hu, H. Fu, and C.-L. Tai, “PointDSC: Robust point cloud registration using deep spatial consistency,” in *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2021, pp. 15 859–15 869. 3, 7

- [30] Q.-Y. Zhou, J. Park, and V. Koltun, “Fast global registration,” in *Proc. of European Conf. on Computer Vision*. Springer, 2016, pp. 766–782. [4](#)
- [31] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 1–4. [6](#)
- [32] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A modern library for 3D data processing,” *arXiv preprint arXiv:1801.09847*, 2018. [6](#)