Received: 10 July 2023 | Revised: 6 June 2024 | Accepted: 31 July 2024

DOI: 10.1002/rob.22417

SURVEY ARTICLE

WILEY

Assessing the practical applicability of neural-based point clouds registration algorithms: A comparative analysis

Simone Fontana[1] | Federica Di Lauro[2] | Domenico G. Sorrenti[2]

[1]School of Law, Università degli Studi di Milano - Bicocca, Milano, Italy

[2]Department of Informatics, Systems and Communication, Università degli Studi di Milano - Bicocca, Milano, Italy

Correspondence
Simone Fontana, School of Law, Università degli Studi di Milano - Bicocca, Piazza dell'Ateneo Nuovo 1, Milano, Italy.
Email: simone.fontana@unimib.it

Abstract

Point cloud registration is a vital task in three-dimensional (3D) perception, with several different applications in robotics. Recent advancements have introduced neural-based techniques that promise enhanced accuracy and robustness. In this paper, we thoroughly evaluate well-known neural-based point cloud registration methods using the Point Clouds Registration Benchmark, which was developed to cover a large variety of use cases. Our evaluation focuses on the performance of these techniques when applied to real-complex data, which presents a more challenging and realistic scenario than the simpler experiments typically conducted by the original authors. The results reveal considerable variability in performance across different techniques, highlighting the importance of assessing algorithms in realistic settings. Notably, 3DSmoothNet emerges as a standout solution, demonstrating good and consistent results across various data sets. Its efficacy, coupled with a relatively low graphics processing unit (GPU) memory footprint, makes it a promising choice for robotics applications, even if it is not yet suitable for real-time applications due to its execution time. Fully Convolutional Geometric Features also performs well, albeit with greater variability among data sets. PREDATOR and GeoTransformer are promising, but demand substantial GPU memory, when handling large point clouds from the Point Clouds Registration Benchmark. A notable finding concerns the performance of Fast Point Feature Histograms, which exhibit results comparable to the best approaches while demanding minimal computational resources. Overall, this comparative analysis provides valuable insights into the strengths and limitations of neural-based registration techniques, both in terms of the quality of the results and the computational resources required. This helps researchers to make informed decisions for robotics applications.

KEYWORDS

3D robotic mapping, 3D scan registration, SLAM, unstructured environment

Abbreviations: DCP, deep ICP; FCGF, fully convolutional geometric features; FMR, feature metric registration; FPFH, fast point feature histograms; GICP, generalized ICP; ICP, iterative closest point; NDT, normal distributions transform; OOM, out-of-memory; PFH, point feature histograms; RoReg, rotation-guided registration; SLAM, simultaneous localization and mapping; SVD, singular value decomposition.

Simone Fontana and Federica Di Lauro are equally contributed.

1 | INTRODUCTION

Point clouds registration is the problem of finding the rototranslation that best aligns two point clouds, that is, two sets of points with three-dimensional (3D) coordinates.

Point clouds registration algorithms can be used in many different applications. For example, robots typically require maps to accurately locate themselves in an environment; point cloud registration techniques are often used to create these maps. Remote sensing of natural environments is another application, as most 3D sensors produce point clouds that are partial views of the environment and need to be aligned. However, point cloud registration does not necessarily have to be applied to complex scenes, as in the previous applications: it can also be applied to single objects or small scenes.

The requirements of the different types of applications are very different. Therefore, in this work we focus on robotics applications solely, which usually involve large, complex scenes. For the same reason, we focus on rigid point clouds registration, which is most commonly used and most effective when dealing with large and complex scenes, rather than individual objects.

The difference between different fields of application lies in the usage conditions and assumptions. For example, in case a point clouds registration technique is used as part of an architectural reconstruction pipeline, we can expect a static world, carefully calibrated acquisition poses, a large overlap between the point clouds to be aligned, and no occlusion. In addition, poorly acquired or too noisy point clouds can be discarded or acquired again. Of course, the same does not apply to robotics and especially not to mobile robotics. Furthermore, execution times and memory usage are much more critical for robotics applications, which are often ultimately intended to work in real time. At the same time, lower accuracy can be accepted, considering that the sensors themselves are usually less accurate as well (although, with properly aligned point clouds, noise can be reduced with smoothing techniques, another important application of registration techniques in robotics). Of course, there are overlaps between different but closely related fields of study. They are not completely separate worlds, but the benchmark we use is eminently robotic and may not adequately represent the advantages and disadvantages of algorithms for other applications. For example, we test different approaches under very difficult conditions, such as low overlap, high noise, or in highly repetitive environments. These are all challenges that are common in robotics but may be unnecessary for other applications.

The main difficulty in aligning two point clouds is finding correspondences between the two sets of points. With the exception of point clouds created with RGB-D sensors, point clouds are usually sparse. For this reason, it is much more difficult to describe the neighborhood of a point than it is, for example, with images. Therefore, 3D features are usually less effective than the 2D counterpart.

The first approach to registering point clouds was Iterative Closest Point (ICP), independently developed by Zhang (1994), Besl and McKay (1992), and Chen and Medioni (1992). ICP attempts to solve the correspondence problem by approximating the true unknown correspondences with a closest neighbor policy. This means that for each point in the first point cloud, called "source," the match is the closest one (using a Euclidean distance) in the second point cloud, the "target." The generated set of correspondences is used to create an optimization problem that is solved using Singular Value Decomposition (SVD). The whole process is repeated until the convergence criteria are met. ICP, despite its simplicity, works reasonably well. However, two conditions must be met: the two point clouds must already be roughly aligned, and there should be a large overlap between the two point clouds. Nevertheless, even if these two conditions are met, ICP can lead to incorrect solutions, mainly because it tends to fall into local minima and because its data association strategy can lead to a large number of incorrect correspondences.

For this reason, several variants of ICP have been proposed in recent years. These variants may aim, for example, to improve the quality of the result, to speed up the algorithm, or to relax the constraint of overlap or initial misplacement between point clouds. A comprehensive comparison of ICP variants was performed by Pomerleau et al. (2013). Important variations of the original ICP algorithm are those involving the error function to be optimized. For example, the Generalized ICP (GICP) algorithm uses not only the distances between two points, but also the covariances computed in a neighborhood of the points (Segal et al., 2010). In this way, it can better represent the underlying geometry of the clouds and produce much better results than the original version (Fontana et al., 2021). Instead of using a point-to-point association strategy, another option is to associate points to planes, planes to planes, or even one point with many points and weight them probabilistically (Agamennoni et al., 2016).

One of the main problems with the basic version of ICP is that if the overlap between the two point clouds is not complete, which is almost always the case, a large number of false associations are inserted into the optimization problem. To mitigate this issue, a number of outlier rejection strategies have been developed (Babin et al., 2019). There are also registration algorithms that do not use a closest-point-based data association. For example, Normal Distributions Transform aligns two point clouds by representing them as a set of Gaussians and searching for the most likely alignment (Biber & Straßer, 2003).

Despite the improvements, a major drawback of ICP-like approaches is their limitation to small initial misalignments: they were designed to solve the so-called *local* registration problems. These are defined as problems where the point clouds are already roughly aligned or for which there is an initial guess about their alignment. In many real-world cases, an initial guess can be provided by other sensors, such as inertial units, odometry, or Global Positioning System (GPS). However, there is another class of problems: *global* registration problems. In this case, the misalignment between the two point clouds is not bounded. Obviously, global registration is a much more difficult problem, of which local registration is a subset. Traditionally,

however, global registration algorithms produce inferior results, which usually need to be refined with a local technique.

Many global registration algorithms are based on geometric features that aim to represent the geometric structure in the neighborhood of a point by a vector. Of these, Point Feature Histograms (Rusu et al., 2008), its faster variant Fast Point Feature Histograms (FPFH) (Rusu et al., 2009), and Angle Invariant Features (Jiang et al., 2009) are notable examples. In addition, SIFT features extracted from a depth image generated from a point cloud have been used for the same purpose (Sehgal et al., 2010).

Geometric descriptors allow one to estimate a set of correspondences between the two point clouds. From these correspondences, the rototranslation can be estimated using, for example, random sample consensus (RANSAC) (Fischler & Bolles, 1981). However, the number of false correspondences that these descriptors entail usually makes the rototranslation estimation step challenging. Approaches such as Fast Global Registration (Zhou et al., 2016) or TEASER (Yang et al., 2020) address this step and allow the estimation of a rototranslation even in the presence of a high number of outlier correspondences. TEASER is particularly able to produce very good results in challenging scenarios (Fontana et al., 2021).

More recently, neural network-based solutions for point cloud registration have been proposed. Two different types of approaches have been proposed. The first type aims to represent a point cloud with a single feature vector. Two point clouds are then matched by finding the rototranslation that minimizes the distance between the two feature vectors. Examples of the first category include point cloud registration network (PCRNet) (Sarode et al., 2019) or PointNetLK, which implements PointNet and the classical Lukas-Kanade algorithms in a single recurrent deep neural network (Aoki et al., 2019). While representing a point cloud as a single feature vector is an elegant and efficient solution, it is particularly problematic when aligning point clouds with partial overlap or moving objects. This is because in these cases the two feature vectors differ greatly, even when properly aligned. In our opinion, this type of technique is therefore suitable for applications other than robotics, whose use case commonly implies a partial overlap.

The second type of technique is similar to traditional feature-based approaches in that multiple feature vectors are extracted from a set of keypoints or even from each point of the point clouds. These features allow one to estimate a set of correspondences that are then used with algorithms such as TEASER or Fast Global Registration. An example of such an approach is Fully Convolutional Geometric Features (FCGF), which are produced in a single pass by a fully convolutional neural network (Choy et al., 2019).

Another example is 3D Match, a 3D convolutional neural network that uses a 3D patch around a point and computes a feature descriptor (Zeng, 1917). A smaller distance between two descriptors means a higher probability of matching. The descriptor was trained in an unsupervised manner using correspondences from existing RGB-D reconstructions.

3DFeatNet, on the other hand, is a deep-learning approach based on a Siamese architecture (Chicco, 2021) that learns to recognize whether two given point clouds are from the same location (Yew & Lee, 2018). It is trained in a weakly supervised manner using pairs of point clouds with a GPS and inertial-based localization. It uses PointNet (Qi et al., 2017) to represent point clouds and, unlike 3DMatch, combines both a feature detector and a descriptor extractor. PPFNet is another approach based on PointNet. It uses points, normals, and Point Pair Features (PPF) to compute feature descriptors that are highly rotationally invariant (Deng et al., 2018).

A particular end-to-end approach is Feature Metric Registration (FMR). Despite being based on features, it solves the registration problem by minimizing the projection error onto a feature space without requiring an explicit search for correspondences (Huang et al., 2020).

Deep ICP (DCP) revises the classical ICP algorithm from a deep-learning perspective (Wang & Solomon, 2019). The result is a neural network that maps source and target point clouds to transformation invariant embeddings. These are used by an attention module that predicts the match. Finally, a differentiable SVD layer estimates the transformation.

3DSmoothNet is a neural-based descriptor built using a voxelized smoothed density value (SDV) representation with a Siamese deep-learning architecture and fully convolutional layers (Gojcic et al., 2019). It achieves excellent results in the 3DMatch benchmark data set (Zeng, 1917). It also enables the registration of laser scans of outdoor vegetation, even when trained only on indoor RGB-D data.

GenReg uses a completely different approach (Huang, Xu, et al., 2021). Instead of estimating a set of correspondences, it directly generates an aligned point cloud using a deep generative neural network. Other examples of neural-based approaches to point cloud registration include RGM (robust point cloud registration framework based on deep graph matching) (Fu et al., 2021), robust point matching network (RPM-Net) (Yew & Lee, 2020), point Gaussian mixture model (PointGMM) (Hertz et al., 2020), and CorsNet (Kurobe et al., 2020). However, these were only tested on registration problems involving single objects, not complex scenes, which are a much more difficult problem.

In contrast to other approaches, PREDATOR is a machine learning-based technique that explicitly aims at solving registration problems with low overlap (Huang, Gojcic, et al., 2021). Thanks to an overlap attention block, this approach is able to estimate the common areas of the source and target point clouds and therefore concentrate the keypoints in this area. Thanks to this strategy, it can significantly increase the success rate for problems with low overlap. The attention given to problems with low overlap is noteworthy because they are often the most difficult to solve, they are relatively common in practice, and many other approaches assume complete or at least high overlap to work properly. Another approach that targets problems with low overlap is GeoTransformer (Qin et al., 2023). However, it uses a different strategy than PREDATOR. Rather than estimating the overlap region, it searches for correspondences over downsampled superpoints that are able to capture the geometric structure

of a patch of the point cloud and that are transformation invariant. Furthermore, it uses an overlap-aware circle loss to focus on super-point pairs with higher overlap, to obtain a correspondence-free method that can work efficiently in low-overlap scenarios.

In contrast to previous feature-based methods that mainly focus on the extraction of rotation-invariant descriptors, rotation-guided registration (RoReg) is a point cloud registration approach that utilizes oriented descriptors and estimated local rotations (Wang et al., 2023). RoReg utilizes the estimated local rotations to improve the accuracy and robustness of the registration process. The core idea is to incorporate the rotation information into the descriptor matching and transformation estimation phases. This approach should help improve the accuracy of the registration, especially for problems with large rotations.

While many different approaches have been proposed, their fields of application, such as large scenes or single objects, and their advantages are not always clear. Most approaches are tested on very limited data sets or on data sets that are too simple to represent the real-world scenarios of robotics applications. There is usually no indication of how much the parameters of an approach have been tuned to a particular data set. This is especially important given how many parameters most techniques have and that it is often impossible to tune them except by trial and error. We believe that the ability to produce good results even when the parameters have not been fine tuned to a specific scenario is critical to a truly useful point cloud registration technique. In addition, most works do not compare different techniques to each other, or compare only to very old techniques, such as ICP.

We believe that testing point cloud registration approaches against a common benchmark is critical. In particular, we want to test whether recent advances in registration techniques offer real benefits through the use of neural networks. That is, we want to test the actual applicability of neural-based techniques. This means that we do not just want to know whether a technique performs well in a few controlled experiments. Instead, we want to answer the following questions:

- How does it perform on a variety of real-complex scenes with partial overlap?
- How well do the pretrained models provided by the authors perform?
- Is the technique very sensitive to parameter settings?
- Is there an increase in performance compared with traditional and well-consolidated techniques?

All these questions can be summarized in one very important but often neglected question: "Are neural-based techniques for point cloud registration applicable in practice?"

For these reasons, we selected the most remarkable neural-based point cloud registration techniques presented in the literature and compared them on the Point Clouds Registration Benchmark (Fontana et al., 2021). The results are then compared with a known traditional technique. Since we believe that the reproducibility of the experiments is an essential requirement for a sound comparison, we provide all the details to reproduce the tests: the version of the software used, the model, and the values of the parameters. We also provide the raw results of the experiments and instructions to reproduce them in a GitHub repository: https://github.com/iralabdisco/neural_registration_comparison.

2 | MATERIAL AND METHODS

2.1 | Algorithms selection

The choice of techniques for point cloud registration is huge, even if we restrict ourselves only to neural-based approaches. Therefore, it is impossible to compare them all. Moreover, point cloud registration can be used in many different applications. In our opinion, it is not useful to compare techniques developed with different objectives, such as a technique for aligning small objects, with a technique for reconstructing a large outdoor scene.

Since our goal is to compare point cloud registration techniques for robotics applications, we defined the following requirements to decide which techniques to test:

Req1 It must be able to align not only individual objects but also large, complex scenes.
Req2 It must work with partial overlap problems. That is, part of the scene represented in one point cloud might not be represented in the other.
Req3 It must also work in the presence of moving objects. That is, even if the two point clouds represent the same scene, some objects may have moved between the acquisition of the two point clouds. This and the previous requirements may also occur together.
Req4 There must be a public open-source implementation of the technique.
Req5 There must be a pretrained model available for the technique.
Req6 The technique should be neural based. That is, it should use a neural network for at least one step of the registration pipeline.

The first three requirements have led us to discard techniques that align two point clouds by minimizing the distance between two vectors, each of which represents an entire point cloud, such as PCRNet (Sarode et al., 2019) or PointnetLK (Aoki et al., 2019). The embedding vectors of two point clouds with only partial overlap will indeed be very different, even if they are correctly aligned, since they actually represent two very different sets of data. The same is true for moving objects. Another problem is the dimension of the embedding vector, which might be too small to represent a large, complex outdoor scene. To support our conclusions, we observe that these approaches are often tested only on single objects, almost always from the ModelNet40 data set, which consists of 3D computer-aided design models of objects (Wu et al., 2015).

Therefore, this is a use case very different from that of this work, namely, robotics.

Requirement 4 is due in part to the need to test the techniques and adapt them to our chosen benchmark. Although this is not strictly necessary, open-source software definitely facilitates this step. Also, and most importantly, we would like to encourage research in point cloud registration and advocate for the use and development of open software in academic research.

Requirement 5 arises from several considerations. First, we want the comparison to be useful to the potential users of these techniques. Successful training of a neural network depends heavily on a good choice of parameters, which usually requires a lot of trial and error and, most importantly, a lot of computational resources. To choose the right parameters, one needs to know how the network works. Therefore, we think it is highly impractical and unlikely that a user will retrain the network. Furthermore, if we retrain the networks, the poor performance of an approach could be due to our poor training rather than the technique, and we consider this unacceptable. Most importantly, we believe that testing the ability of a network to generalize to environments other than those used in training is fundamental to the practical utility of a network. Because the applications for point cloud registration are so large and training is so impractical, we believe this capability is essential. At a minimum, a network should be able to generalize to environments that are similar, but not identical, to those used in training. For example, an office environment that is different from the one used to train the network. Finally, finding training data for point cloud registration with high-quality ground truth is not an easy task. This is confirmed by the fact that most of the works were trained on few real data sets, such as KITTI (Geiger et al., 2012) or 3DMatch (Zeng, 1917), or on synthetic data sets, such as ModelNet40. Obviously, we cannot use the Point Clouds Registration Benchmark, since it is used in the testing phase.

Regarding Requirement 6, some of the techniques we considered are end-to-end. That is, they implement the entire registration pipeline. Others, however, generate feature descriptors that must be used in conjunction with other techniques to estimate a roto-translation. We have treated all the different types as "neural-based" regardless of whether the entire pipeline is implemented as a neural network.

The algorithms we selected for comparison, together with their implementation, are as follows:

- 3DFeatNet—https://github.com/yewzijian/3DFeatNet
- DCP—https://github.com/WangYueFt/dcp
- 3DSmoothNet—https://github.com/zgojcic/3DSmoothNet
- FCGF—https://github.com/chrischoy/FCGF
- FMR—https://github.com/XiaoshuiHuang/fmr
- PREDATOR—https://github.com/prs-eth/OverlapPredator
- GeoTransformer—https://github.com/qinzheng93/GeoTransformer
- RoReg—https://github.com/HpWang-whu/RoReg

Initially, we also selected other algorithms that we believe are noteworthy, but that we still had to discard. DeepVCP, for example, is an end-to-end registration algorithm that achieves comparable results to conventional nonneural methods but has higher robustness against initial alignment errors (Lu et al., 2019). However, according to the authors' experiments, which involved a random uniform initial misalignment in the interval [0, 1] meters for translation and [0°, 1°] for rotation, it still appears to be a local registration algorithm. We believe that it would be unfair to compare local and global algorithms. On the one hand, it is well known that local algorithms perform much better than global algorithms on problems with smaller initial alignment errors (Fontana et al., 2021); on the other hand, local algorithms are not able to solve global problems. Since all other approaches we selected are global, we therefore had to discard DeepVCP.

We also discarded PPFNet, GenReg, and CorsNet because we could not find an official implementation.

Eventually we did not test RGM, RPM-Net, and PointGMM because, according to the experimental activity reported in the corresponding papers, they are focused on the alignment of single objects rather than complex scenes.

We believe that neural-based approaches should also be compared with traditional techniques. Since the two categories aim at solving the same problem, there is no reason not to compare one with the other. Therefore, we have also added a state-of-the-art geometric descriptor to the comparison, namely, FPFH.

2.2 | Data and methods

Choosing the right data set is an essential step for a comparison. We chose the Point Clouds Registration Benchmark (Fontana et al., 2021), which is very suitable for testing algorithms for robotics applications. It consists of 15 sequences of data collected with different sensors and representing different types of environments. Specifically, the represented outdoor environments are as follows:

- a forest, both summer and fall, a garden with a gazebo in summer and winter, and a plain, from the ETH data sets (Pomerleau et al., 2012);
- a reproduction of a planetary exploration environment, from the planetary data sets (Tong et al., 2013);
- an urban road environment, from the Kaist data sets (Jeong et al., 2019).

The indoor environments, on the other hand, are as follows:

- an indoor house environment and a college building, from the ETH data sets (Pomerleau et al., 2012);
- office environments, from the TUM data sets (Sturm et al., 2012).

Figures 1–3 show some of the sequences used for the experimental activity.

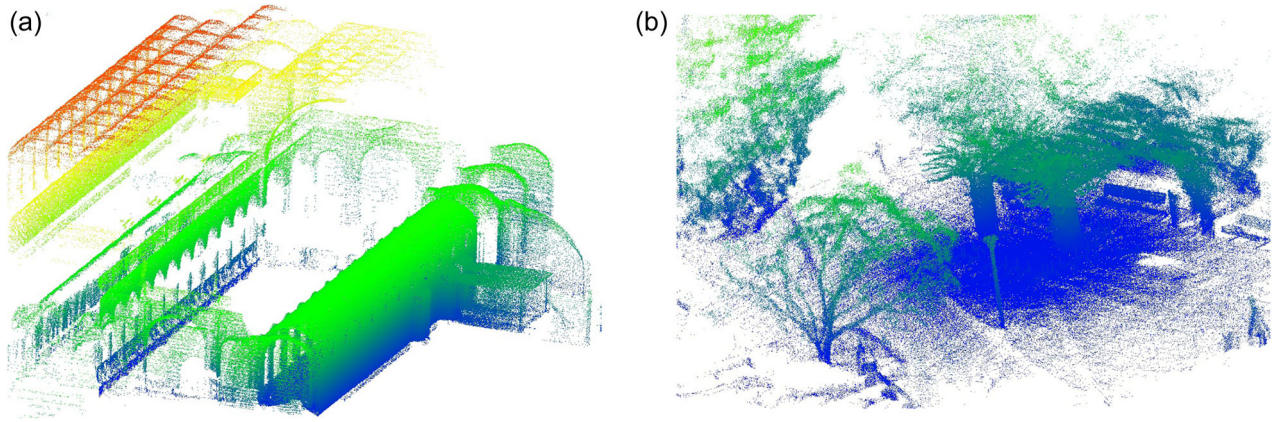

FIGURE 1 Hauptgebaude (a) and gazebo_winter (b) sequences from the ETH data sets.

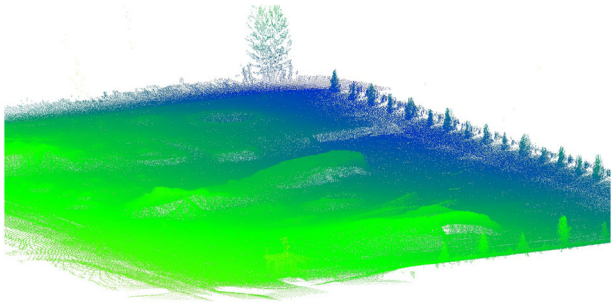

FIGURE 2 The box_met sequence, from the planetary data sets. [Color figure can be viewed at wileyonlinelibrary.com]

FIGURE 3 The pioneer_slam sequence, from the TUM data sets. [Color figure can be viewed at wileyonlinelibrary.com]

The methodology used for the experimental activity is that proposed for the Point Clouds Registration Benchmark. This provides, in addition to the data, sets of registration problems, that is, pair of point clouds to be aligned and initial misalignments to apply to the source point cloud. The overlap between the source and target point clouds, that is, the area of the scene they have in common, strongly affects the difficulty of a registration problem. For this reason, the Point Clouds Registration Benchmark provides problems consisting of pairs of point clouds sampled to cover a range of overlap between 90% and 60%. The same is true for the initial misalignment associated with a problem: it is randomly generated to cover the entire range of possible misalignments.

There are two versions of the benchmark: for local algorithms and for global algorithms. Since all of our chosen methods appear independent of the initial misalignment, they should be able to solve the global version of the benchmark, which includes random rotations between 45° and 180° for rotation and a large data set-specific range of translations.

Further details on the data and the methodology used to generate the registration problems can be found on the work of Fontana et al. (Fontana et al., 2021) and on the original papers of the data sets (Jeong et al., 2019; Pomerleau et al., 2012; Sturm et al., 2012; Tong et al., 2013).

The algorithms we selected are of different types. Some are end-to-end approaches to point clouds registration, others are neural networks used for keypoint detection and descriptor extraction, still others are for descriptor extraction only. Thus, although they serve the same purpose, they need to be tested in different ways. Therefore, we set the parameters of the different approaches to obtain the best results within the limits of the available hardware. Nevertheless, we have ensured the greatest degree of uniformity of testing conditions and thus comparability of results.

In the case of techniques that extract keypoints and generate feature descriptors, we used three different algorithms to estimate the rototranslation from a set of correspondences: RANSAC, FastGlobalRegistration, and TEASER. The latter two algorithms are state-of-the-art and have proven to be very effective in estimating a transformation even when there are a large number of outliers. RANSAC, on the other hand, is still one of the most popular algorithms and we therefore think that its results can be valuable to the community. Regardless of the network used to generate the

descriptors, we used the same parameters for the transformation estimation algorithms. These are as follows:

- To search for the most similar descriptors in the other point cloud, that is, to estimate correspondences, we used the L2 distance. That is, each descriptor in one point cloud is associated to the one in the other cloud that has the smallest Euclidean distance. To speed up the calculation, we used a KDtree data structure.
- correspondences were filtered out based on a "mutual distance" filter. That is, a pair (descriptorS, descriptorT) is considered a true correspondence and used for the transformation estimation step only if descriptorT is closest to descriptorS and vice versa.
- For RANSAC, we used as "max_correspondence_distance" a value equal to *VOXEL_SIZE* · 1.5.
- For Fast Global Registration, we used as "max_correspondence_distance" a value equal to *VOXEL_SIZE* · 0.5.
- For TEASER we used as "noise_bound" a value equal to *VOXEL_SIZE*.

VOXEL_SIZE is the leaf size of the voxel grid filter we used to downsample the point clouds. This parameter is specific to each approach and specified below.

The values of these parameters have been chosen according to suggestions of the Open3D framework we used for the experiments (Zhou et al., 2018).

For approaches that generate feature descriptors but do not extract keypoints, we used 5000 uniformly sampled points as keypoints.

As a preprocessing step, we applied a voxel grid filter with a leaf size of 0.1 m to each point cloud. This step is the same for most of the approaches. Exceptions are PREDATOR, GeoTransformer, FCGF, and RoReg. PREDATOR and GeoTransformer require a specific voxel size, depending on the data set used for the training. Therefore, we used a leaf size of 0.025 m with the model trained on 3DMatch and of 0.30 m with that trained on KITTI, as recommended by the authors. FCGF has an internal voxel filter; therefore, we did not use an external one. The leaf size we used is the same of the other approaches, hence 0.1 m. The voxel size for RoReg was determined using a specific procedure described in the corresponding section.

Other parameters are algorithm specific and are described below.

2.2.1 | 3DSmoothNet

3DSmoothNet uses a voxelized SDV representation. Therefore, we generated it for 5000 randomly selected points. The voxel grid has a size of 16 × 16 × 16, with a radius of 0.5 and a smoothing kernel width of 1.75. It should be noted that these are mostly default values provided by the developers. As we will see in the experimental section, they work quite well for a variety of data. We only changed the voxel radius because most of the point clouds we used were too sparse to use the default value. Still, we used the same value for each point cloud, regardless of density.

The feature extraction network does not require any special parameters except the feature size, which we set to 64.

2.2.2 | Feature metric registration

For FMR we used 10 iterations as suggested by the authors. We performed experiments with two different pretrained models, trained with the ModelNet40 and 7Scenes data sets.

2.2.3 | FPFH

FPFH is a nonneural feature. We did not change any parameter compared with the default settings specified by the authors.

2.2.4 | 3DFeatNet

With 3DFeatNet we used the following parameters:

- Radius for sampling clusters = 2.0
- Maximum number of points to consider per cluster = 64
- Feature dimension size = 32
- Radius for nonmaximal suppression = 0.5
- Minumum response ratio = 10^{-3}.
- Maximum number of keypoints = 1024.

2.2.5 | Fully convolutional geometric features

There are various pretrained models for FCGF. We used the two models that gave the best results according to the original work. These are the models trained on 3DMatch and KITTI.

The parameters used with the model pretrained on 3DMatch are summarized below:

- model = 3dmatch_normalized_5cm_32,
- output features = 32,
- normalized = true,
- conv 1 kernel size = 7,
- *D* = 3.

Those used with the KITTI pretrained model, instead, are as follows:

- model = kitti_notnormalized_20cm_32,
- output features = 32,
- normalize = False,
- conv 1 kernel size = 7,
- *D* = 3.

2.2.6 | DCP

Although we wanted to test the benchmark with DCP as well, we could not use it with the larger point clouds because we kept running out-of-memory (OOM). This also happened with a very aggressive down-sampling step. Therefore, we unfortunately had to discard this approach.

For the experiments, we used an NVIDIA GeForce GTX 1080 Ti graphics processing unit (GPU) with 11 Gbytes of memory. DCP stores point clouds as PyTorch tensors with float32 elements. Therefore, with our hardware we can use point clouds with a maximum of 5350 points, that is, 64,200 bytes. In Table 1 we show the mean number of points which composes the point clouds of the sequences of the benchmark. As can be seen, even if we doubled the available memory, we could not apply DCP to most sequences (take as an example the sequence wood_autumn with an average size of 36,352 points or the sequence box_met with an average size of 41,452 points).

2.2.7 | PREDATOR

The parameters used with the model pretrained on 3DMatch are summarized below:

- voxel size = 0.025 m.

TABLE 1 Statistics on the number of points composing the point clouds in the various sequences.

Sequence	Mean number of points	Maximum	Minimum
plain	10,961	12,656	8884
stairs	12,685	20,243	6873
apartment	8355	12,275	3966
hauptgebaude	32,111	36,767	28,622
wood_autumn	34,660	38,786	31,601
wood_summer	36,352	46,369	31,436
gazebo_summer	23,626	34,458	15,221
gazebo_winter	26,397	33,054	21,270
box_met	41,452	56,521	23,909
p2at_met	18,113	32,751	6369
pioneer_slam	3802	7834	261
pioneer_slam3	3379	7467	1286
long_office_household	1834	5596	147
urban05	12,170	20,126	5581

Note: The point clouds have been downsampled using a voxel grid filter with a leaf size of 0.1 m.

Those used with the KITTI pretrained model, instead, are as follows:

- voxel size = 0. 3 m.

2.2.8 | GeoTransformer

The parameters used with the model pretrained on 3DMatch are summarized below:

- voxel size = 0.025 m.

Those used with the KITTI pretrained model, instead, are as follows:

- voxel size = 0.3 m.

2.2.9 | RoReg

RoReg also requires a specific voxel size to function properly. We have calculated it according to the following methodology.

The model we used was trained on 3DMatch with a voxel size of 0.025 m; on average, the spatial extent of the point clouds in the 3DMatch data set is 4 m. To determine the voxel size we use for testing each sequence, we found the bounding box of each point cloud and used this to calculate the extent of the point cloud. We averaged these values over each sequence and used as voxel size the value:

$$\frac{avg\_extent}{4} \cdot 0.025.$$

The actual voxel size we used for each sequence is shown in Table 2.

3 | RESULTS

We measure the quality of the results using the normalized distance defined by the Point Clouds Registration Benchmark (Fontana et al., 2021). That is, given a source point cloud S, composed of n points S_i, the rototranslation T estimated by a registration technique, and the ground truth rototranslation G, the quality of the results is measured by the distance between the source point cloud aligned with the estimated rototranslation and the source point cloud aligned with the ground truth rototranslation. The distance is defined as follows:

$$D(G \cdot S, T \cdot S) = \frac{\sum_i \frac{\| G \cdot S_i - T \cdot S_i \|}{\| S_i - \bar{S} \|}}{n}, \quad (1)$$

where $\overline{S}$ is the centroid of S, $T \cdot S$ is the application of the roto-translation T to S, and $\|x\|$ is the L_2 norm of the vector x.

The distance D represents the average distance between the ground truth and the estimated positions of a point, normalized by the average distance to the centroid of the point cloud. We chose this distance first to allow comparison with other experiments on the same benchmark, as described in the work of Fontana et al. Second, normalization using the distance to the centroid allows meaningful calculation of aggregate statistics, such as the median. The details and complete rationale for this distance can be found in the original paper (Fontana et al., 2021). It is not always easy to visualize how large an error is just by looking at the metric, as it is dimensionless. For this reason, we show images of example results corresponding to different degrees of error along with the ground truth for better visualization. These are shown in Figures 4–7. Note that an error of 0.01 corresponds to an essentially perfect alignment, while errors of more than 0.5 lead to unusable results.

TABLE 2 The voxel size we used to test rotation-guided registration.

Sequence	Voxel size (m)
apartment	0.07
gazebo_summer	0.2
gazebo_winter	0.2
hauptgebaude	0.2
plain	0.2
stairs	0.1
wood_autumn	0.2
wood_summer	0.2
pioneer_slam	0.04
pioneer_slam3	0.06
long_office_household	0.03
box_met	0.2
p2at_met	0.3
planetary_map	0.7
urban05	0.7

Tables 3-13 report the median and the 0.75 and 0.95 quantiles of the experiments performed with the different algorithms. The statistics were computed both for the results of the registration problem in each sequence and for all sequences together (the row named "Total"). Feature extractors require other approaches to match the extracted features and estimate a rototranslation from them. For this step, we used RANSAC, FastGlobal Registration, and TEASER, all of whose results are shown in the tables. FMR, RoReg, and Geo-Transformer, on the other hand, are end-to-end approaches; for this reason they have only a single set of results.

The metric proposed by the Point Clouds Registration Benchmark is a dimensionless quantity, since each error is scaled by the average distance from the centroid of the point cloud. Therefore, it is not always easy to identify how large an error is. For this reason, we also give the residual errors, with respect to the initials. These are shown in Table 14, which depicts the residual error in percent with respect to the initial error. Given an initial error err_i and a final error err_f, the residual res is defined as

$$res = \frac{err_f}{err_i} \cdot 100.$$

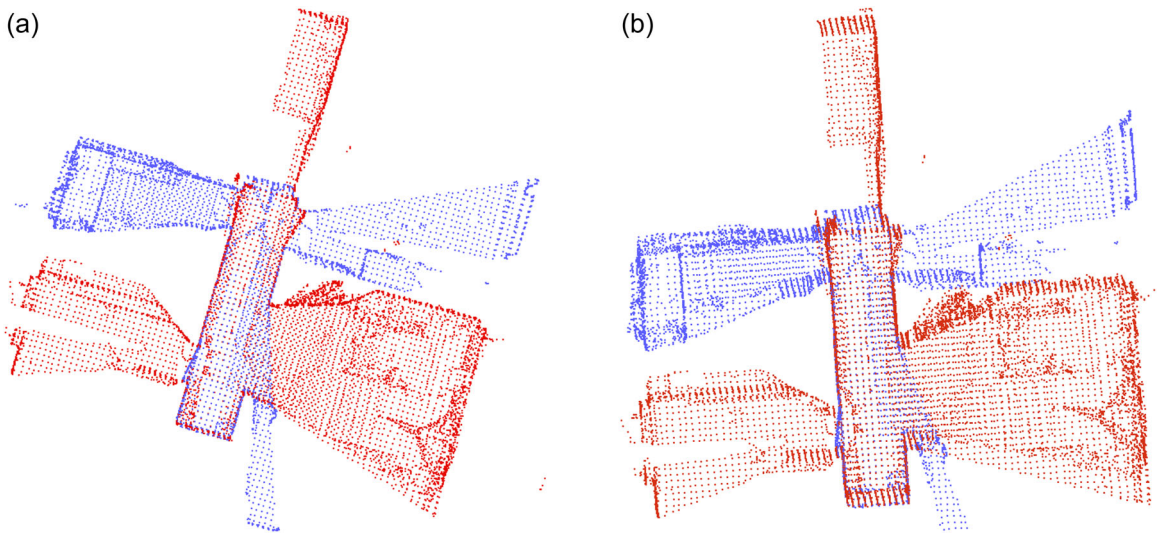

FIGURE 4 A pair of point clouds from the apartment sequence of the ETH data sets, aligned obtaining a final error of 0.01. Notice how the alignment is substantially perfect. (a) Aligned with the ground truth and (b) result with an error of 0.01. [Color figure can be viewed at wileyonlinelibrary.com]

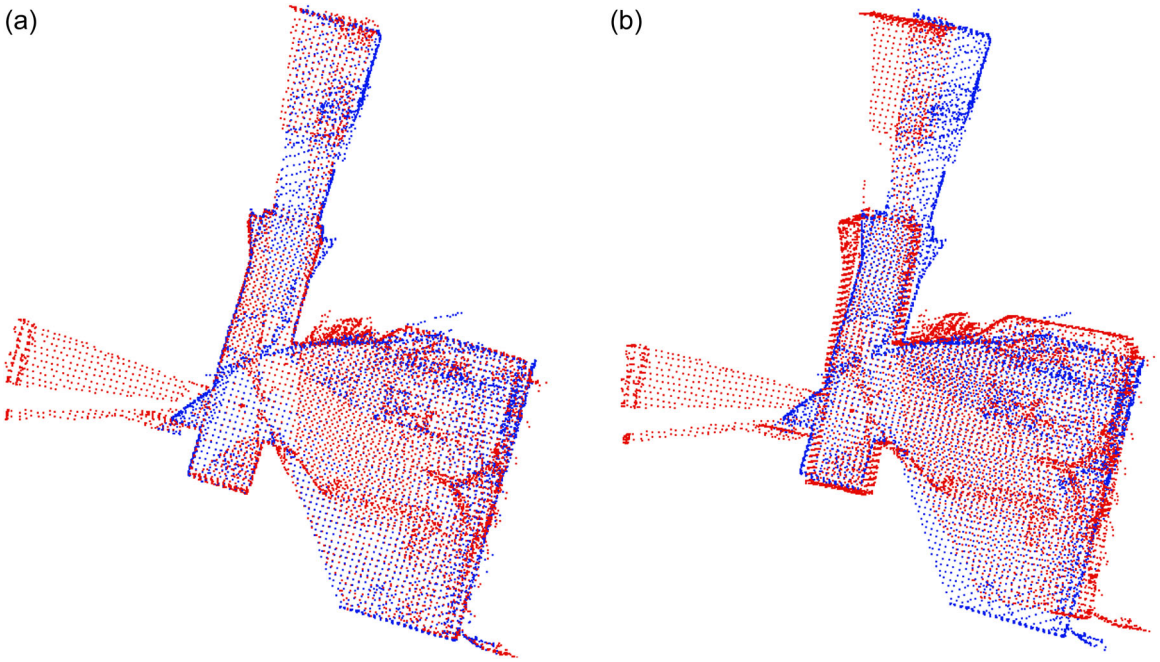

FIGURE 5 A pair of point clouds from the apartment sequence of the ETH data sets, aligned obtaining a final error of 0.1. (a) Aligned with the ground truth and (b) result with an error of 0.1. [Color figure can be viewed at wileyonlinelibrary.com]

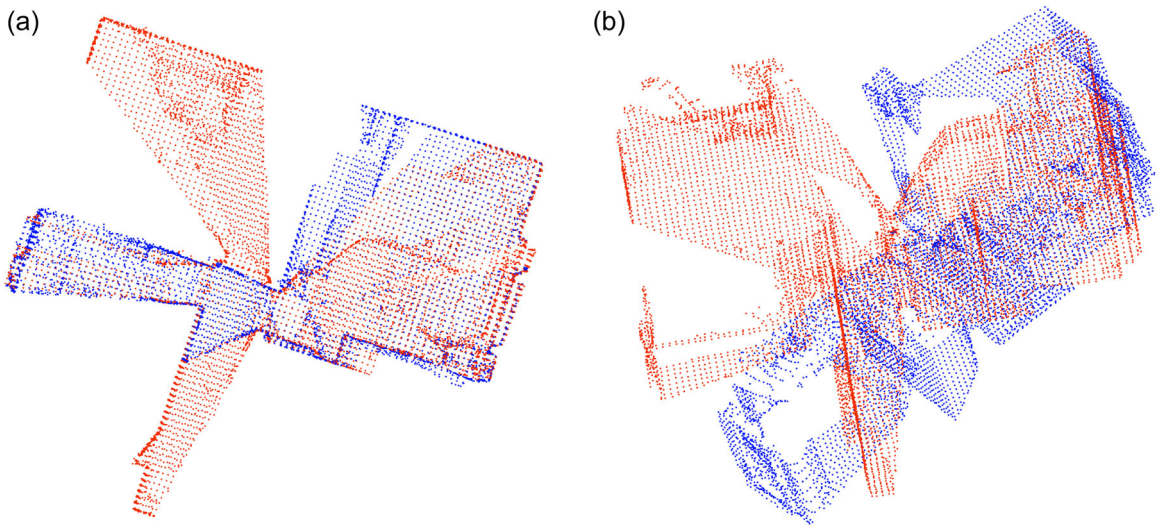

FIGURE 6 A pair of point clouds from the apartment sequence of the ETH data sets, aligned obtaining a final error of 0.5. (a) Aligned with the ground truth and (b) result with an error of 0.5. [Color figure can be viewed at wileyonlinelibrary.com]

Similar to the other columns, this shows the median of the residuals in each sequence and among the entire benchmark. Therefore, a residual of *X*% means that for half of the problems, the final error was *X*% of the initial error or better. We believe that these tables can help better identify how effective an algorithm is.

To facilitate the comparison of the different results, we also show the median residual error of the various approaches in Figures 8 and 9. The y-axis is truncated at 20%, as otherwise the smaller residuals, which are the most interesting, would be indistinguishable from each other.

In addition to the quality of the results, we also show statistics on the computational resources required by each approach. These have been collected using a machine equipped with an Intel Core i7 - 6850K central processing unit (CPU), with 64 Gbyte of random access memory and a NVIDIA GeForce GTX 1080 Ti with 11 Gbyte of memory GPU.

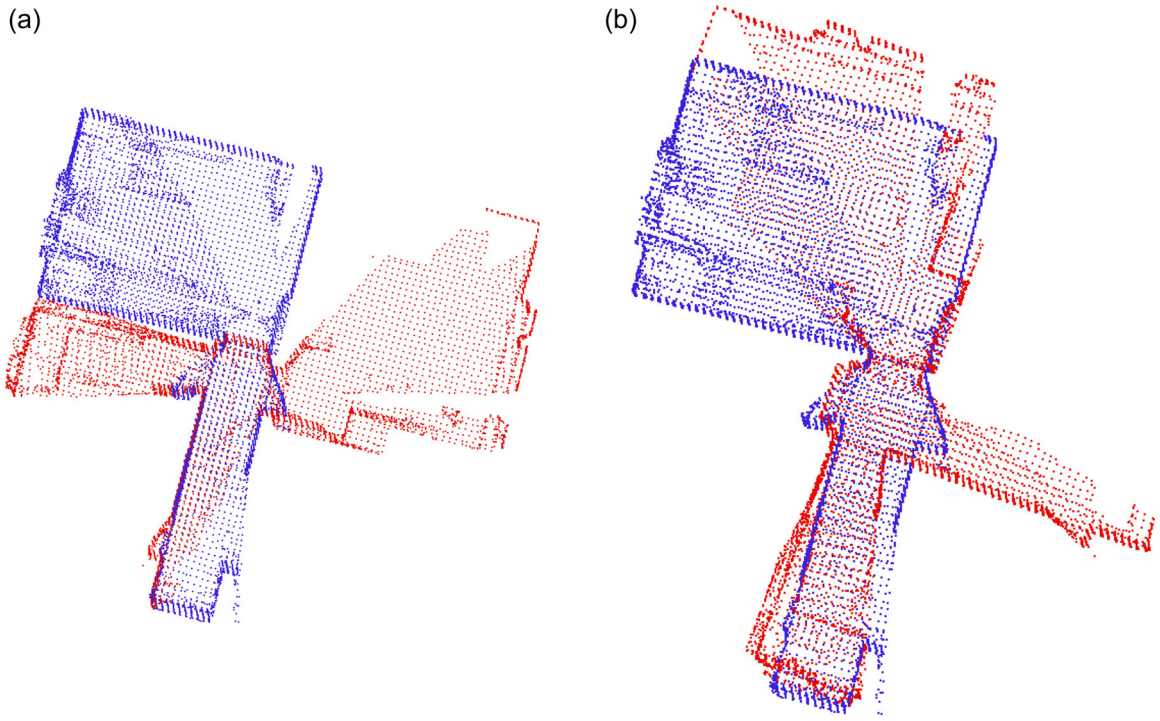

FIGURE 7 A pair of point clouds from the apartment sequence of the ETH data sets, aligned obtaining a final error of about 1.1. Notice how the rotation of the red point cloud is completely wrong. (a) Aligned with the ground truth and (b) result with an error of 1.1. [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 3 Results obtained by employing 3DFeatNet features in conjunction with RANSAC, FastGlobal, and TEASER.

	TEASER			RANSAC			FastGlobal		
Sequence	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	median	0.75 Q	0.95 Q
plain	6.96	11.76	17.23	4.19	7.76	14.28	10.92	14.03	17.00
stairs	1.91	2.58	5.48	2.06	2.72	4.15	2.58	3.65	7.94
apartment	1.54	1.76	2.35	1.42	1.76	2.17	1.78	2.04	2.48
hauptgebaude	1.50	2.15	7.13	1.91	2.66	5.62	2.43	3.36	4.60
wood_autumn	2.28	3.38	5.14	2.88	3.98	5.92	2.23	2.77	3.67
wood_summer	2.05	2.69	3.90	2.54	3.68	5.24	2.20	2.72	3.72
gazebo_summer	2.25	3.59	6.26	2.66	4.04	6.32	2.63	3.44	5.33
gazebo_winter	1.97	3.63	6.37	2.51	3.75	7.10	2.76	3.45	5.60
box_met	14.35	32.12	79.55	9.57	14.19	21.64	8.70	10.87	15.33
p2at_met	9.15	17.00	29.90	8.46	15.42	26.89	7.34	12.90	19.80
planetary_map	42.42	63.00	103.86	35.98	52.49	97.70	27.64	42.94	59.72
pioneer_slam	1.45	2.32	9.62	1.33	1.91	7.64	4.60	6.18	9.85
pioneer_slam3	3.44	6.59	10.58	1.89	4.58	9.02	4.60	6.26	9.43
long_office_household	1.61	2.21	13.86	1.62	2.05	9.71	2.09	3.73	9.26
urban05	2.00	3.15	5.98	2.13	3.20	6.86	2.70	3.42	4.65
Total	2.36	6.34	39.62	2.52	5.50	25.46	3.15	6.92	22.77

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

Abbreviation: RANSAC, random sample consensus.

TABLE 4 Results obtained by employing FCGF features with the model trained on 3DMatch in conjunction with RANSAC, FastGlobal, and TEASER.

	TEASER			RANSAC			FastGlobal		
Sequence	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	median	0.75 Q	0.95 Q
plain	0.34	2.38	3.33	1.51	6.44	12.33	0.60	1.66	2.38
stairs	0.02	0.04	4.97	0.05	0.08	3.07	0.04	0.09	2.83
apartment	0.03	0.06	1.91	0.04	0.06	1.31	0.04	0.10	1.46
hauptgebaude	0.75	1.56	2.11	0.58	1.41	2.12	0.55	1.11	2.04
wood_autumn	0.03	0.05	0.43	0.93	3.01	4.61	0.11	0.18	0.50
wood_summer	0.03	0.04	0.13	0.11	2.13	4.19	0.11	0.23	0.60
gazebo_summer	0.02	0.04	0.17	0.03	0.06	2.65	0.05	0.19	1.59
gazebo_winter	0.02	0.03	0.06	0.05	0.14	2.90	0.08	0.17	0.68
box_met	7.80	10.25	14.58	7.44	10.04	13.79	7.72	9.86	12.89
p2at_met	5.70	8.43	14.19	8.81	12.50	17.48	4.30	7.60	11.21
planetary_map	52.37	71.92	104.37	11.08	15.63	21.71	30.21	41.74	67.31
pioneer_slam	0.11	0.32	1.60	0.13	0.27	1.51	0.11	0.30	10.35
pioneer_slam3	0.06	0.09	0.18	0.07	0.11	0.15	0.05	0.08	0.18
long_office_household	0.18	0.86	2.55	0.15	0.48	12.39	0.22	1.88	10.46
urban05	3.05	4.19	7.47	2.57	3.75	5.35	1.17	1.83	3.80
Total	0.09	2.48	31.10	0.21	4.00	13.03	0.22	1.95	18.42

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

Abbreviations: FCGF, fully convolutional geometric features; RANSAC, random sample consensus.

TABLE 5 Results obtained by employing FCGF features with the model trained on KITTI in conjunction with RANSAC, FastGlobal, and TEASER.

	TEASER			RANSAC			FastGlobal		
Sequence	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	median	0.75 Q	0.95 Q
plain	9.37	13.93	23.33	6.93	9.58	16.63	3.17	6.44	11.51
stairs	2.93	4.48	8.08	2.44	3.33	7.78	2.21	3.33	5.51
apartment	1.68	2.06	3.22	1.42	1.78	2.18	1.45	1.79	2.23
hauptgebaude	3.16	5.02	8.79	2.74	3.36	4.36	1.26	2.77	6.03
wood_autumn	2.65	3.69	5.94	2.98	3.98	5.39	0.71	1.67	3.24
wood_summer	1.98	2.76	3.54	2.93	3.73	5.23	0.69	1.55	3.07
gazebo_summer	2.93	4.37	7.16	2.73	3.75	5.23	0.95	2.06	4.30
gazebo_winter	3.05	4.55	7.17	3.62	4.78	5.87	0.67	1.84	4.48
box_met	12.96	17.29	28.75	9.08	13.99	22.34	7.79	9.86	13.37
p2at_met	10.59	16.31	27.74	10.67	13.45	21.27	7.15	9.21	14.53
planetary_map	42.37	62.96	99.80	11.41	16.00	28.81	28.93	40.79	67.32
pioneer_slam	6.22	11.74	17.24	2.79	9.89	16.96	5.53	7.98	12.64
pioneer_slam3	9.76	13.51	17.73	4.38	9.36	13.58	4.34	6.04	7.31
long_office_household	2.36	8.18	18.02	1.68	3.55	15.09	3.49	6.93	11.31
urban05	3.35	5.25	12.04	2.68	4.04	6.43	1.70	2.62	4.61
Total	3.78	10.05	29.48	3.36	6.98	16.66	2.28	6.11	19.47

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

Abbreviations: FCGF, fully convolutional geometric features; RANSAC, random sample consensus.

TABLE 6 Results obtained by employing FMR with a model trained on Modelnet40 and a model trained on 7scene.

	7scene			Modelnet40		
Sequence	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q
plain	2.14	2.63	3.54	2.03	2.37	2.88
stairs	2.70	4.89	11.05	1.90	3.72	8.81
apartment	1.64	1.90	2.33	1.65	1.85	2.16
hauptgebaude	1.79	2.24	2.81	1.72	2.12	2.50
wood_autumn	2.01	2.39	3.17	1.95	2.33	2.81
wood_summer	1.78	2.31	3.44	1.76	2.06	2.64
gazebo_summer	1.75	2.28	3.43	1.63	2.00	2.90
gazebo_winter	3.04	4.19	7.21	2.71	3.90	6.62
box_met	2.42	2.99	3.35	2.45	2.90	3.29
p2at_met	2.51	3.36	7.20	2.28	2.91	5.18
planetary_map	11.96	18.09	24.69	7.06	10.38	13.64
pioneer_slam	2.13	2.83	4.13	2.26	3.81	9.04
pioneer_slam3	2.14	2.94	3.69	1.93	2.78	3.58
long_office_household	2.03	3.46	18.11	1.94	3.53	13.89
urban05	2.50	4.27	8.64	2.08	3.47	7.96
Total	2.17	3.12	12.20	2.03	2.83	8.42

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

Abbreviations: FMR, feature metric registration; RANSAC, random sample consensus.

TABLE 7 Results obtained by employing FPFH features in conjunction with RANSAC, FastGlobal, and TEASER.

	TEASER			RANSAC			FastGlobal		
Sequence	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	median	0.75 Q	0.95 Q
plain	0.14	1.22	4.81	5.93	8.09	12.32	0.78	1.59	2.66
stairs	0.05	0.51	4.47	1.91	3.86	7.46	0.24	0.72	4.86
apartment	0.02	0.04	1.64	0.10	1.62	4.16	0.05	0.14	1.55
hauptgebaude	0.56	1.59	2.15	1.63	2.82	3.86	0.55	1.16	2.02
wood_autumn	0.06	0.32	2.26	3.02	4.07	5.39	0.31	0.53	1.78
wood_summer	0.05	0.22	1.21	2.88	3.73	5.23	0.34	0.60	1.55
gazebo_summer	0.03	0.07	1.37	2.65	3.56	4.79	0.24	0.57	1.49
gazebo_winter	0.03	0.08	0.68	3.03	4.43	5.85	0.23	0.50	2.10
box_met	7.34	10.29	14.35	7.57	10.84	13.79	7.71	10.14	13.75
p2at_met	4.21	7.64	11.72	8.85	12.57	17.48	4.42	7.55	11.22
planetary_map	45.92	66.15	97.65	11.41	15.67	28.66	29.63	41.77	67.47
pioneer_slam	0.09	0.20	2.04	0.10	0.24	1.45	0.09	0.23	10.37
pioneer_slam3	0.05	0.08	0.28	0.06	0.12	0.35	0.06	0.11	0.33
long_office_household	0.10	0.26	2.38	0.13	0.31	9.25	0.12	1.34	11.68
urban05	2.64	3.71	6.34	2.46	3.62	5.17	1.47	2.27	3.12
Total	0.11	1.95	24.58	2.55	5.00	13.19	0.41	2.10	18.17

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

Abbreviations: FPFH, fast point feature histograms; RANSAC, random sample consensus.

TABLE 8 Results obtained by employing 3DSmoothNet features in conjunction with RANSAC, FastGlobal, and TEASER.

	TEASER			RANSAC			FastGlobal		
Sequence	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	median	0.75 Q	0.95 Q
plain	0.03	0.05	2.51	0.05	0.09	7.37	1.28	1.96	2.91
stairs	0.01	0.02	0.08	0.03	0.05	4.61	0.98	2.47	5.10
apartment	0.01	0.01	0.02	0.03	0.04	0.05	0.56	1.41	1.85
hauptgebaude	0.79	1.03	2.37	0.01	0.80	1.70	0.66	1.10	2.02
wood_autumn	0.01	0.02	0.02	0.02	0.03	0.06	1.22	1.66	2.91
wood_summer	0.01	0.01	0.02	0.02	0.02	0.04	0.85	1.43	1.97
gazebo_summer	0.01	0.01	0.02	0.01	0.02	0.07	0.73	1.71	3.17
gazebo_winter	0.01	0.01	0.02	0.01	0.02	0.03	0.46	1.39	2.78
box_met	6.95	10.97	16.18	7.03	10.48	14.03	8.79	10.80	15.63
p2at_met	0.12	0.73	9.59	3.73	9.03	15.92	5.46	8.04	13.84
planetary_map	39.20	62.76	103.36	10.80	15.22	20.95	29.88	41.41	68.35
pioneer_slam	0.04	0.09	0.19	0.12	0.24	0.78	0.04	0.10	0.13
pioneer_slam3	0.02	0.03	0.06	0.05	0.08	0.15	0.01	0.03	0.07
long_office_household	0.05	0.12	0.25	0.09	0.20	0.41	0.04	0.10	0.92
urban05	0.62	1.84	3.55	2.24	3.61	4.83	1.70	2.21	3.29
Total	0.02	0.18	23.29	0.04	1.34	12.21	0.97	2.37	18.47

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.
Abbreviation: RANSAC, random sample consensus.

TABLE 9 Results obtained by employing PREDATOR, trained on 3DMatch, in conjunction with RANSAC, FastGlobal, and TEASER.

	TEASER			RANSAC			FastGlobal		
Sequence	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	median	0.75 Q	0.95 Q
plain	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
stairs	0.27	1.96	3.59	0.80	3.28	6.64	2.13	2.56	3.45
apartment	0.15	1.23	2.03	2.78	4.03	5.01	1.50	1.70	2.00
hauptgebaude	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
wood_autumn	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
wood_summer	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
gazebo_summer	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
gazebo_winter	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
box_met	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
p2at_met	0.07	0.08	0.09	5.85	8.61	10.81	2.50	2.64	2.76
planetary_map	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
pioneer_slam	0.05	0.07	0.86	0.03	0.07	0.77	0.22	1.43	5.80
pioneer_slam3	0.02	0.03	0.09	0.02	0.03	0.13	1.21	2.35	3.79
long_office_household	0.03	0.06	0.57	0.04	0.08	0.32	0.32	1.39	6.57
urban05	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
Total	0.04	0.12	1.91	0.05	1.64	4.57	1.21	1.80	3.99

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.
Abbreviation: n.a., not available; RANSAC, random sample consensus.

TABLE 10 Results obtained by employing PREDATOR, trained on KITTI, in conjunction with RANSAC, FastGlobal, and TEASER.

	TEASER			RANSAC			FastGlobal		
Sequence	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	median	0.75 Q	0.95 Q
plain	0.21	0.66	2.77	0.20	0.27	1.22	0.44	1.08	2.60
stairs	0.24	1.52	5.71	0.19	0.36	3.23	0.53	1.81	4.23
apartment	0.37	1.40	1.90	0.13	0.19	1.66	0.34	1.35	1.77
hauptgebaude	0.11	1.59	6.18	0.10	0.88	4.18	4.90	6.17	7.52
wood_autumn	0.05	0.07	0.15	0.06	0.10	0.25	2.73	3.08	3.82
wood_summer	0.04	0.07	0.13	0.07	0.10	0.97	2.41	3.07	3.83
gazebo_summer	0.04	0.07	2.03	0.07	0.10	0.82	0.68	2.63	4.37
gazebo_winter	0.03	0.05	0.33	0.06	0.09	0.79	2.21	3.35	5.01
box_met	6.90	10.37	13.44	7.71	10.26	15.52	8.46	10.69	15.92
p2at_met	0.48	5.08	13.54	0.50	6.72	14.14	5.83	9.53	15.92
planetary_map	58.77	85.97	112.36	13.96	21.05	118.67	38.36	53.04	77.98
pioneer_slam	0.62	6.32	10.06	0.42	0.99	3.72	1.62	6.31	7.52
pioneer_slam3	0.33	0.59	2.17	0.23	0.37	0.55	0.31	0.99	6.57
long_office_household	0.64	1.18	3.78	0.29	0.38	3.47	1.14	5.28	578,845.55
urban05	0.07	0.28	2.33	0.16	2.06	4.42	0.71	1.68	4.40
Total	0.14	0.99	10.41	0.14	0.59	9.77	1.89	3.97	10.97

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

Abbreviation: RANSAC, random sample consensus.

TABLE 11 Results obtained by employing GeoTransformer when trained on 3DMatch.

Sequence	Median	0.75 Q	0.95 Q
plain	n.a.	n.a.	n.a.
stairs	n.a.	n.a.	n.a.
apartment	n.a.	n.a.	n.a.
hauptgebaude	n.a.	n.a.	n.a.
wood_autumn	n.a.	n.a.	n.a.
wood_summer	n.a.	n.a.	n.a.
gazebo_summer	n.a.	n.a.	n.a.
gazebo_winter	n.a.	n.a.	n.a.
box_met	n.a.	n.a.	n.a.
p2at_met	n.a.	n.a.	n.a.
planetary_map	n.a.	n.a.	n.a.
pioneer_slam	0.02	0.03	0.05
pioneer_slam3	0.02	0.02	0.06
long_office_household	0.03	0.04	0.08
urban05	n.a.	n.a.	n.a.
Total	0.02	0.03	0.07

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

Abbreviation: n.a., not available.

TABLE 12 Results obtained by employing GeoTransformer when trained on KITTI.

Sequence	Median	0.75 Q	0.95 Q
plain	0.02	0.04	0.14
stairs	0.03	0.05	1.29
apartment	0.05	0.07	1.56
hauptgebaude	0.01	0.02	0.38
wood_autumn	0.01	0.01	0.02
wood_summer	0.01	0.01	0.02
gazebo_summer	0.01	0.02	0.04
gazebo_winter	0.01	0.01	0.03
box_met	0.16	1.86	2.86
p2at_met	0.03	0.07	1.90
planetary_map	5.28	8.06	12.84
pioneer_slam	n.a.	n.a.	n.a.
pioneer_slam3	n.a.	n.a.	n.a.
long_office_household	n.a.	n.a.	n.a.
urban05	1.47	1.81	2.42
Total	0.03	0.19	4.67

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

Abbreviation: n.a., not available.

TABLE 13 Results obtained by employing RoReg when trained on 3DMatch.

Sequence	Median	0.75 Q	0.95 Q
plain	5.60	12.87	25.96
stairs	0.02	0.03	0.06
apartment	0.25	1.51	2.56
hauptgebaude	0.03	0.13	10.73
wood_autumn	0.09	0.17	9.31
wood_summer	0.07	0.19	7.04
gazebo_summer	0.05	0.14	9.84
gazebo_winter	0.04	0.09	12.52
box_met	9.93	14.77	26.33
p2at_met	19.58	30.44	40.42
planetary_map	48.62	68.85	105.97
pioneer_slam	0.03	0.07	0.24
pioneer_slam3	0.01	0.03	0.05
long_office_household	0.03	0.05	0.10
urban05	5.22	9.82	14.94
Total	0.11	6.33	37.53

Note: Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

Since neural networks are almost exclusively used on GPUs nowadays, we show the GPU memory required by each technique. This is a critical requirement as GPUs with large memory are still very expensive and, if the required memory is not available, the approach simply cannot be used. Table 15 and Figure 10 show the maximum GPU memory used by the different approaches in each sequence of the benchmark. It should be noted that the displayed value was calculated considering only the cases when the required memory did not exceed the available memory, so that the network did not receive OOM errors. However, there are cases where OOM errors actually occurred; for PREDATOR and GeoTransformer (other technique never got OOM errors) these are indicated in Table 16. Moreover, we also show the time used by each approach to align a pair of point cloud, divided into different phases.

The acronym "n.a." in Tables 9, 11, and 12 stands for "not available." This appears when we were unable to produce results, mostly due to OOM errors. The exception is GeoTransformer on sequences from the TUM data sets (pioneer_slam, pioneer_slam3 and long_office_household), which could not produce results due to the required voxel-based subsampling (0.3 m), which in these particular sequences did not allow one to find enough neighbors in each voxel.

Table 17 and Figure 11 show the execution time of the different approaches, averaged over all sequences. The times are divided into Voxel Grid, SDV Voxelization, Features and Registration times, to distinguish between the different steps of the registration pipeline.

TABLE 14 Median of residual errors (%).

Sequence	3DFeatNet	FMR	FCGF	FPFH	3DSmoothNet	PREDATOR	GeoTransformer	RoReg
plain	98.35	83.05	4.48	2.17	0.46	3.19	1.14	70.32
stairs	37.29	68.36	0.59	1.21	0.35	6.01	1.19	0.43
apartment	45.75	49.02	1.10	0.86	0.28	11.39	1.79	9.41
hauptgebaude	59.93	103.86	26.95	15.75	21.23	4.77	0.67	1.35
wood_autumn	69.64	98.16	1.09	2.07	0.38	1.49	0.58	3.13
wood_summer	69.48	82.93	0.92	1.71	0.31	1.44	0.58	2.43
gazebo_summer	79.68	83.44	0.62	0.91	0.31	1.13	0.77	1.45
gazebo_winter	55.95	131.91	0.55	0.81	0.24	0.86	0.46	1.13
box_met	198.54	99.37	107.92	94.62	76.15	82.34	7.25	134.99
p2at_met	103.11	78.68	56.16	40.13	1.47	4.65	1.55	188.41
planetary_map	364.79	247.83	432.42	381.33	374.04	441.33	228.20	423.60
pioneer_slam	10.53	34.27	1.03	0.75	0.35	5.23	n.a.	0.27
pioneer_slam3	37.50	29.06	0.61	0.54	0.17	3.77	n.a.	0.18
long_office_household	9.17	26.44	1.12	0.69	0.32	8.59	n.a.	0.16
urban05	71.28	122.05	107.04	100.60	22.85	2.99	100.00	180.18
Total	63.39	82.04	1.73	1.87	0.47	3.20	1.27	3.10

Note: For non-end-to-end approaches, the transformation has been estimated using TEASER.

Abbreviations: FCGF, fully convolutional geometric features; FMR, feature metric registration; FPFH, fast point feature histograms; n.a., not available; RoReg, rotation-guided registration.

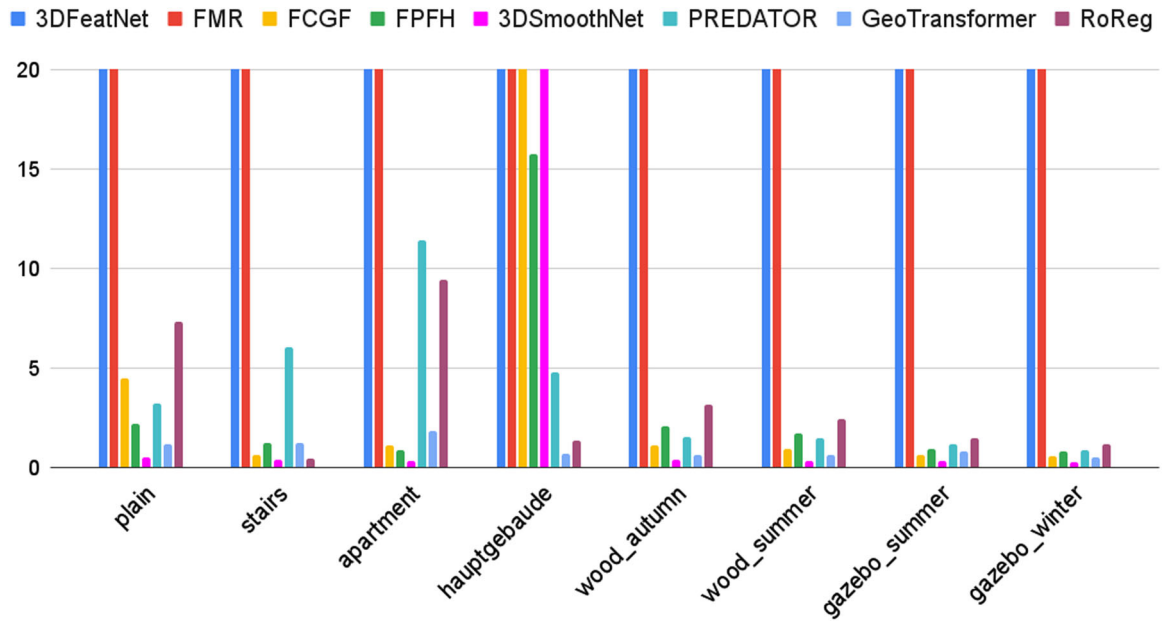

FIGURE 8 The median residual errors of the different approaches on sequences from the ETH data sets, in percentage w.r.t. the initial error. FCGF, fully convolutional geometric features; FMR, feature metric registration; FPFH, fast point feature histograms; RoReg, rotation-guided registration. [Color figure can be viewed at wileyonlinelibrary.com]

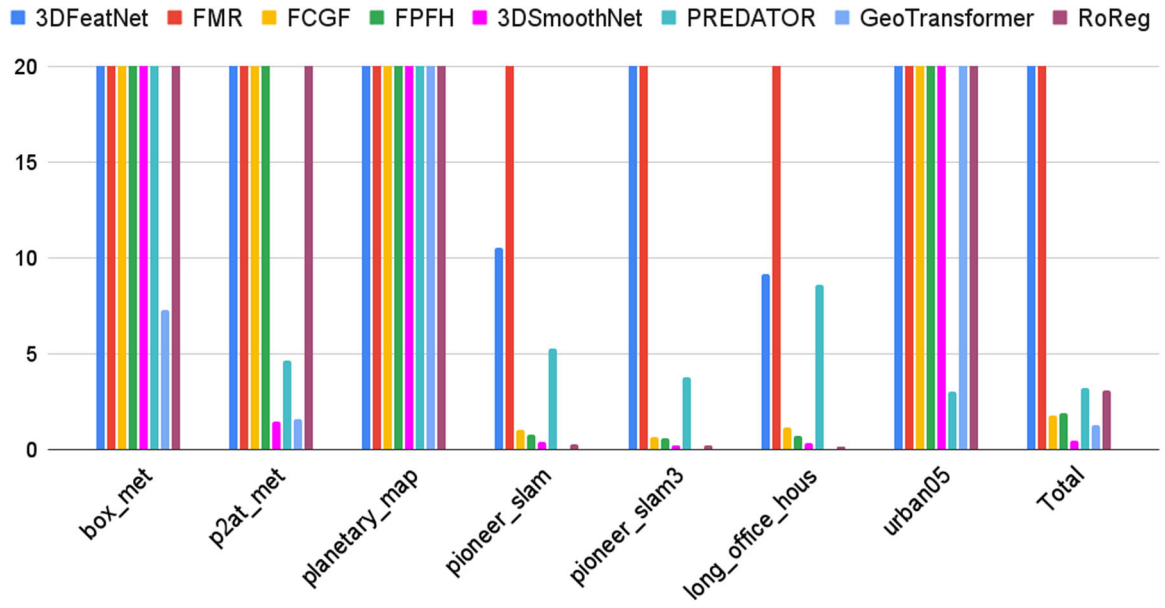

FIGURE 9 The median residual errors of the different approaches on sequences from the Planetary, TUM and Kaist data sets and among the whole Benchmark ("Total"), in percentage w.r.t. the initial error. FCGF, fully convolutional geometric features; FMR, feature metric registration; FPFH, fast point feature histograms; RoReg, rotation-guided registration. [Color figure can be viewed at wileyonlinelibrary.com]

"SDV Voxelization" refers to the SDV voxelization of 3DSmoothNet and does not include the time required by the voxel grid filter, which is shown in the column named "Voxel Grid." FCGF already includes a voxel grid filter and therefore we did not use an external one. For this reason, its execution time also includes the application of its internal voxel grid filter. "Feature" refers to the time required for feature extraction. It is only given for the approaches that have separate feature extraction and registration steps, not for the end-to-end approaches. "Registration" refers to the remaining steps, that is, the estimation of rototranslation from the set of correspondences using TEASER, for the techniques that require it, or the execution time of the end-to-end approaches.

TABLE 15 The maximum memory usage for each sequence, in MiB, of the various approaches.

Sequence	3DFeatNet	FCGF	FMR	3DSmoothNet	PREDATOR	GeoTransformer	RoReg
plain	8484	1038	1404	5660	1398	1160	10,626.0
stairs	8486	1148	1880	5658	1462	1208	11,160.0
apartment	4354	1034	1318	5658	1166	1006	9494.0
hauptgebaude	8486	1268	2970	5658	2160	1884	10,826.0
wood_autumn	8486	1288	3072	5660	2318	2126	10,788.0
wood_summer	8486	1336	3470	5664	2662	2034	11,060.0
gazebo_summer	8484	1246	2798	5660	1992	1590	11,024.0
gazebo_winter	8484	1248	2696	5658	2190	1758	11,166.0
box_met	8486	1380	3972	5660	2664	2000	10,626.0
p2at_met	8500	1184	2576	5664	2230	1920	10,626.0
planetary_map	8500	1300	4936	5664	11,168	6970	10,626.0
pioneer_slam	8500	996	920	9248	1076	914	11,168.0
pioneer_slam3	8486	986	826	5664	1056	936	10,862.0
long_office_household	8486	996	950	5664	1052	850	10,736.0
urban05	8492	1096	1734	5658	10,646	8806	10,626.0
Total	8500	1380	4936	9248	11,168	8806	11,168.0

Abbreviations: FCGF, fully convolutional geometric features; FMR, feature metric registration; MiB, Mebibyte.

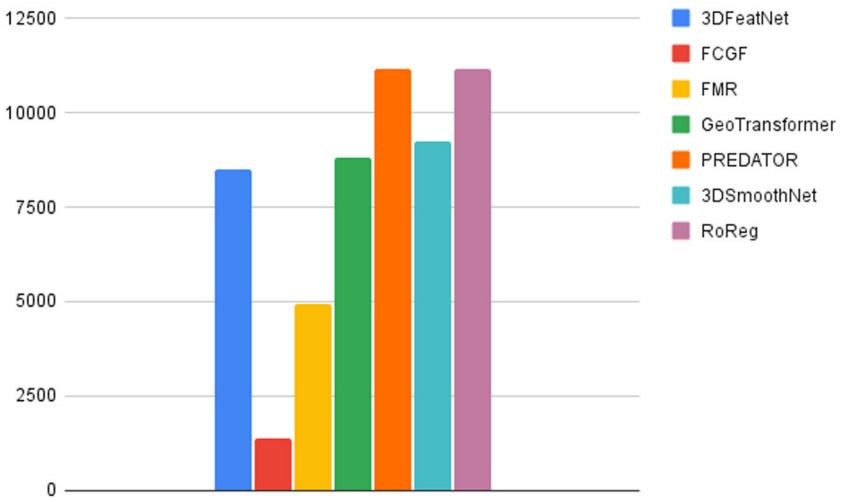

FIGURE 10 The maximum memory usage, in MiB, of the different approaches. FCGF, fully convolutional geometric features; FMR, feature metric registration; MiB, Mebibyte. [Color figure can be viewed at wileyonlinelibrary.com]

Finally, we also reproduced a very robotic problem. The sequences from the TUM data sets were originally recorded by a robot navigating in an office-like environment. Therefore, we decided to reconstruct the robot's trajectory by aligning each point cloud with the previous one, as is done in simultaneous localization and mapping (SLAM), but without loop closure, which is beyond the scope of this paper. We then compared the estimated poses with the ground truth and computed the errors and the corresponding statistics according to Equation (1). For these experiments, we selected the best approaches that emerged from the previous results in terms of the quality of the alignment. These are 3DSmoothNet, RoReg, FPFH, PREDATOR, and FCGF. Table 18 shows the results of these experiments.

4 | DISCUSSION

4.1 | Quality of the registration

Before discussing the results, we need to make some considerations. First, the Point Clouds Registration Benchmark is very challenging, especially in its *global* version that we used. In particular, the data sets

in the *planetary* section, the *box_met*, *p2at_met*, and *planetary_map* sequences, are very challenging due to the high repetitiveness of the environment. The *urban05* sequence is also very challenging in the global version of the benchmark because the point clouds are very small (they contain only a few points) and sparse.

The second consideration is that most of the approaches we have considered were originally tested on much simpler problems and sometimes on single objects rather than on complex scenes. Therefore, it is not surprising that some of them do not perform well on real-complex data sets.

Nevertheless, we still consider very valuable a fair comparison of challenging data sets. The Point Clouds Registration Benchmark was originally developed to represent problems encountered in real-world robotics applications. This is in contrast to the tendency we analyzed to test algorithms on simple data sets, often comparing them only to very old traditional algorithms, such as ICP. On the other hand, we are interested in finding out whether neural-based registration algorithms are really useful in practice and whether they have any advantages over effective traditional techniques, especially considering the additional computational resources required.

TABLE 16 Number of out-of-memory errors of PREDATOR and GeoTransformer, using models pretrained on KITTI and 3DMatch.

	PREDATOR		GeoTransformer	
Sequence	KITTI	3DMatch	KITTI	3DMatch
plain	0	100	0	100
stairs	0	48	0	100
apartment	0	5	0	100
hauptgebaude	0	100	0	100
wood_autumn	0	100	0	100
wood_summer	0	100	0	100
gazebo_summer	0	100	0	100
gazebo_winter	0	100	0	100
box_met	0	100	0	100
p2at_met	0	98	0	100
planetary_map	74	102	0	100
pioneer_slam	0	0	0	11
pioneer_slam3	0	0	0	30
long_office_household	0	0	0	13
urban05	0	100	0	100
Total	74	1053	0	1256

Note: Other techniques never got out-of-memory errors and are not shown.

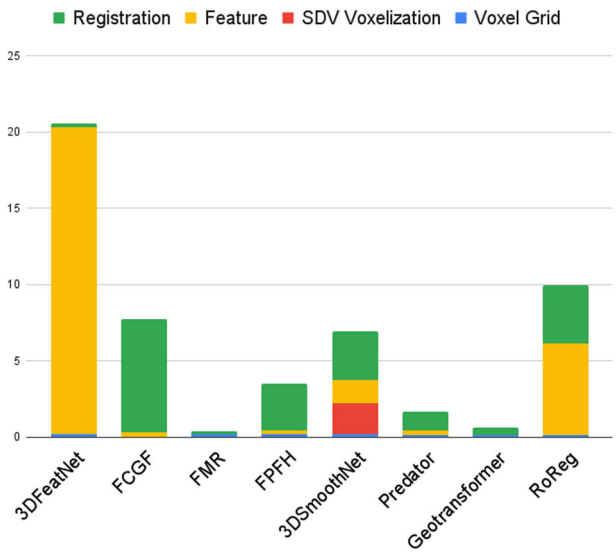

FIGURE 11 Average execution time, in seconds, of the various approaches, divided into the different phases. FCGF, fully convolutional geometric features; FMR, feature metric registration; FPFH, fast point feature histograms; RoReg, rotation-guided registration; SDV, smoothed density value. [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 17 Average execution time, in seconds, of the different approaches.

	Voxel grid	SDV voxelization	Feature	Registration	Total
3DFeatNet	0.19	n.a.	20.133	0.231	20.554
FCGF	n.a.	n.a.	0.357	7.393	7.750
FMR	0.19	n.a.	n.a.	0.230	0.42
FPFH	0.19	n.a.	0.277	3.079	3.546
3DSmoothNet	0.19	2.028	1.560	3.187	6.965
PREDATOR	0.17	n.a.	0.272	1.213	1.655
GeoTransformer	0.17	n.a.	n.a.	0.465	0.635
RoReg	0.17	n.a.	6.000	3.786	9.956

Abbreviations: FCGF, fully convolutional geometric features; FMR, feature metric registration; FPFH, fast point feature histograms; n.a., not available; RoReg, rotation-guided registration; SDV, smoothed density value.

TABLE 18 Results of the best approaches on sequences from the TUM data sets.

	3DSmoothNet			FPFH			RoReg		
Sequence	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q
pioneer_slam	0.068	0.13	0.52	0.087	0.16	1.17	**0.047**	0.11	1.27
pioneer_slam3	0.047	0.073	0.17	0.059	0.12	0.77	**0.032**	0.057	0.53

	PREDATOR			FCGF		
	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q
pioneer_slam	0.054	0.12	1.11	0.075	0.33	1.14
pioneer_slam3	0.036	0.063	0.55	0.059	0.096	0.64

Note: Each point cloud was aligned with the previous one to reconstruct the trajectory of the robot.

Abbreviations: FCGF, fully convolutional geometric features; FPFH, fast point feature histograms; Q, quantile; RoReg, rotation-guided registration.

Of the approaches we have tested, 3DSmoothNet, FCGF, PREDATOR, GeoTransformer, RoReg, and FPFH (which is not based on a neural network) have achieved satisfactory results. This becomes particularly clear when looking at Table 14, which compares the residuals of the algorithms we tested. Most of these algorithms are actually feature extractors. Therefore, we need another algorithm to estimate a rototranslation from a set of features. We tested each approach with RANSAC, FastGlobal and TEASER. However, TEASER consistently achieved the best results, hence, in Table 14 and in the rest of the discussion we consider only results obtained using TEASER (if necessary). GeoTransformer, PREDATOR, FMR, and FCGF were tested with different pretrained models. While all results are listed in the previous tables, in Table 14 we only show the results of the model with the best performance. This is FCGF trained on 3DMatch, FMR trained on modelnet40, and GeoTransformer and PREDATOR trained on KITTI.

The performance of the different algorithms on sequences from different data sets varies greatly; therefore it is worth looking at each data set individually. On sequences from the ETH data sets (plain, stairs, apartment, hauptgebaude, wood_autumn, wood_summer, gazebo_summer, and gazebo_winter), 3DSmoothnet achieved the best results, with the median of the residuals almost always below 1%, that is, the alignment was almost perfect. RoReg also performed very well, with the exception of the plain sequence, where we achieved significantly poorer results. It should be noted that this sequence represents an environment that is very different from the others and more similar to that of the planetary data sets, which could be the reason for this difference. With the exception of 3DFeatNet and FMR, the other approaches obtained worse, but still very good results. It should be noted that FCGF, FPFH, and 3DSmoothNet performed much worse on the hauptgebaude sequence than on others of the ETH data sets. However, a solution with a residual error of 20% can still be considered usable as global alignment (which is usually refined later). On the other hand, PREDATOR and especially GeoTransformer performed well on this sequence too. A characteristics which distinguish the hauptgebaude sequence is the high degree level of repetitivity. Therefore, it could be that GeoTransformer and PREDATOR perform properly under such conditions too. However, this is preliminar hypothesis only and should be confirmed by further experiments in highly repetitive environments. The relatively worse performance of some algorithms on the hauptgebaude sequence could also be caused by a non-optimal choice of parameters. It should be noted that comparing approaches without fine-tuning the parameters specifically for each sequence is an objective of the Point Clouds Registration Benchmark and this work. Indeed, in a real-world application, it is often impossible to tune the parameters with exactly the same data that will be used when the system is deployed. Therefore, we believe that fine-tuning leads to experiments with unrealistically good results that do not correspond to the practical experience of the end user.

On sequences from the planetary data sets, the performances are very different. While on the p2at_met sequence 3DSmoothNet, PREDATOR and GeoTransformer achieved good results, only the latter approach was able to obtain usable solutions on the box_met sequence, which is remarkable. It is quite surprising that some approaches performed well on the p2at_met sequence and much worse on the box_met, as the two sequences are very similar: they depict the same environment and were recorded under very similar conditions. This is an indication of how susceptible most approaches might be to even small changes in the data. On the other hand, the planetary_map sequence is so hard (probably to much) in its global variant that it is not surprising that none of the approaches worked properly on it.

The sequences from the TUM data sets (pioneer_slam, pioneer_slam3, and long_office_household) represent an indoor office environment and are therefore a good indicator of the performance of the different approaches in indoor robotic applications. We can see that the same observations we made for the ETH data sets hold. On the one hand, this is to be expected, since indoor environments are also represented in some ETH sequences. On the other hand, it should be noted that the point clouds of the TUM data sets were created with an RGB-D camera, while those of the ETH were created with a laser scanner. We believe that the ability of some approaches to generalize to other types of sensors, without retraining, is remarkable. Moreover, also 3DFeatNet and FMR, which produced

mostly unusable results on other sequences, were able to produce usable results, albeit worse than those of the other approaches.

As for the urban05 sequence from the Kaist data set, this is certainly very challenging, especially because of the sparsity of the point clouds. Only 3DSmoothNet and PREDATOR were able to correctly align the point clouds, with the latter approach achieving quite good results, with a residual of about 3%. It should be noted that PREDATOR was trained on KITTI, which is quite similar to the Kaist data set. However, it is also true that other approaches trained on KITTI, such as GeoTransformer, could not achieve the same results.

To summarize, we can conclude that 3DSmoothNet obtains the best results overall with a median residual error of 0.47%. However, other approaches are able to achieve similar results, with a high variability between different sequences. We believe that the performances of the FPFH features are remarkable as they are very close to those of the other approaches, while they are not based on a neural network and therefore, they do not require any GPU and are much less computationally intensive.

We expected that the degree of overlap between the source and target point clouds would influence the registration accuracy. However, our analysis showed no correlation between the degree of overlap and the quality of the results. This lack of correlation may be attributed to the presence of other influencing factors, such as initial misalignment and scene geometry. In particular, the scene geometry has a significant influence that may mask the effects of other parameters. Due to the lack of a clear correlation, we chose not to include the correlation with the degree of overlap in the results. Nonetheless, our full results are publicly available, including the overlap values according to the methodology described in Fontana et al. (2021), so that interested readers can independently reproduce the correlation statistics.

4.2 | Memory usage

Regarding the GPU memory usage, there is lot of variability among different approaches. 3DSmoothNet, which performs very well in terms of accuracy, has a relatively low memory usage, with a maximum of less than 6GiB, except for the usage for the pioneer_slam sequence. Also noteworthy are the results of FCGF, which requires very little GPU memory (with a maximum of 1380 MiB [Mebibyte]), but can still achieve very good results.

On the contrary, the memory consumption of RoReg, and especially of PREDATOR and GeoTransformer is very high when they are trained on 3DMatch. As can be seen in Table 16, the latter two got 100 OOM errors on many sequences. Considering that there are 100 registration problems per sequence, we can conclude that they are hardly usable in such a configuration. This is the reason for the "n.a." results in Tables 9 and 11. On the other hand, these two networks, when trained on KITTI, have a lower memory footprint (except for the planetary_map sequence, which however consists of very large point clouds) and have few or no OOM errors, while performing very well in terms of quality of the results. This leads us to believe that the data used for training can strongly influence not only the accuracy of the technique but also its memory consumption. This is probably due to the different subsampling used for the internal representation of the point clouds.

4.3 | Execution time

Of the various approaches, only GeoTransformer and FMR show a low average execution time (less than 0.5 s). In contrast, 3DSmoothNet, which achieved very good results and has a low memory usage, has a very high execution time of about 6.8 s. As expected, there seems to be a trade-off between GPU memory consumption and execution time, as approaches, such as GeoTransformer and PREDATOR, which have very high memory consumption, take much less time than others with low memory usage, such as 3DSmoothNet or FCGF. 3DFeatNet seems to be an outlier, with an average execution time of more than 20 s. Therefore, given the quality of its results, we would not recommend it for general-purpose point cloud registration. The performance of FPFH is remarkable: although it is already a few years old and not based on a neural network, it is still able to achieve almost state-of-the-art results with an execution time that is still below that of the other best approaches. In addition, it does not require GPU memory and can therefore be used on a much broader variety of machines.

As can be seen from Figure 11, the execution time is dominated by the "Registration" phase, with the sole exceptions of 3DFeatNet and RoReg. For approaches that are not end-to-end, and therefore have separate phases for feature extraction and transformation estimation, the "Registration" time is that required by TEASER to estimate a roto-translation from the set of correspondences. Therefore, we can state that the bottleneck, at least on our test machine, is the CPU rather than the GPU, since TEASER is executed on the CPU. Moreover, the "Registration" times of GeoTransformer and FMR are much lower, which confirms our hypothesis, as these are end-to-end approaches that run entirely on a GPU. On the contrary, the execution time of RoReg is very high, although it is not the highest. It is dominated by the "Feature" phase, which is to be expected given the way RoReg works. Indeed, it has to extract features from the source and target point cloud after applying several different rotations, hence the high computational cost of this step.

4.4 | SLAM-like registration

Considering the results in Table 18, we can say that the selected approaches got very good comparable median results when trying to reconstruct the trajectory of a robot by aligning consecutive point clouds. This is expected, since consecutive point clouds usually have a larger overlap and smaller misalignment w.r.t. the problems in the benchmark we used in the previous test. The only significative difference regards the 0.95th quantile. Here 3DSmoothNet significantly outperforms all the other three approaches, with a much smaller error. This means that, the worst alignments of 3DSmoothNet are better than those

of the other techniques. While this may seem a lesser advantage, it has to be noted that when recostructing a trajectory, a single large error can cause serious problems, for example, by making harder to detect loop closures that are essential in SLAM applications. For these reasons, 3DSmoothNet appears to be the best choice when considering merely the quality of the result. However, it has much larger execution times (see Table 17); hence, its use for real-time trajectory estimation appears unfeasible at the moment, although it is a great choice for off-line mapping. RoReg, which performs best in terms of median error, has a similar issue: its use does not currently seem feasible for this type of problem due to its large execution time.

4.5 | Future directions

An initial consideration to note is that the majority of existing approaches have been trained on two data sets: either 3DMatch or KITTI. However, neither of these data sets adequately represents natural environments. While KITTI captures outdoor scenes, it focuses on road environments, which significantly differ from the natural settings found, for example, in the planetary data sets (box_met, p2at_met, and planetary_map). This disparity may explain the comparatively poorer performances on sequences from these sequences. Of course, training neural networks on data sets that better reflect the testing environment holds the potential for substantial performance enhancements.

Natural environments, in particular, are inadequately represented in current point cloud registration data sets. While acquiring high-quality ground truth data sets in natural outdoor environments may pose challenges, it is a necessary endeavor to advance research in this domain. For instance, the rise of robotic agriculture is likely to necessitate more data to improve existing techniques. However, it is noteworthy that certain approaches exhibit remarkable generalization capabilities across diverse conditions, such as registering point clouds recorded with light detection and rangings, even when trained on RGB-D data sets.

Recent methods like PREDATOR and GeoTransformer have demonstrated good performance in terms of residual error. Nevertheless, their considerable memory requirements pose challenges, making them less practical for aligning large point clouds, such as those encountered in the Point Clouds Registration Benchmark. This problem becomes even more important when considering robotic applications, which usually have hardware limitations in terms of size, weight and power consumption.

A critical point is the remarkable discrepancy between the performance of some approaches reported in the original papers and the actual results of our testing activity, which we believe reflects practical, albeit difficult, use cases well. This, together with the very large memory requirements of some approaches, such as DCP, which we were unable to test, highlights the need for more realistic experiments when proposing new techniques. Although some proposals have indeed been tested under very realistic conditions, this is still not common. By "more realistic" we mean, for example, testing on complex data representing different types of environments, so that the generalization capabilities of an approach are properly emphasized. It also means that a model trained on data generated with one type of sensor should be tested on data produced with another one, which was already the case with only a few of the best proposals. Finally, specifying the memory and computational requirements of the proposals can also help in evaluating the techniques and assessing when they are truly applicable.

Another critical point is not evident from the results we have shown, but is nevertheless important: it is the time needed to make some approaches work with experiments other than those of the original papers. We were able to reproduce all the original experiments without any problems. However, there were relevant problems when we had to adapt the approaches to solve problems from the Point Clouds Registration Benchmark. Some were easy to adapt, but others required a lot of work, often on undocumented code. To make a technique truly applicable in practise, it is therefore important to provide an easy-to-use interface. An example could be a function or script to align any two point clouds, similar to what is done for traditional approaches, like ICP or GICP in the Point Cloud Library (Rusu & Cousins, 2011). In contrast, with most machine learning-based approaches, we had to write a custom data loader for our problems, which in some cases meant a significant effort.

5 | CONCLUSIONS

We compared well-known neural-based point cloud registration techniques on the Point Clouds Registration Benchmark, to assess their practical applicability to robotic applications. The results show that there is a lot of variability in their performances when applied to real-complex data, in contrast to the much simpler experiments usually performed by the original authors. The results varies, of course, between different approaches, but also when the same approach is applied to different data, even from similar environments. 3DSmoothNet appears to be the best solution available, considering its excellent results, which are very consistent even between different kinds of data. This is especially true if we consider its relatively low GPU memory requirements. FCGF and RoReg are also capable of obtaining very good results, albeit with a larger variability among different data sets. PREDATOR and GeoTransformer seems very promising, however they requires a large GPU memory, at least when applied to large point clouds, such as those from the Point Clouds Registration Benchmark. Special considerations must be done regarding FPFH features. Their performance is very close to that of the best approaches. This is remarkable, especially considering the little computational resources they require.

ACKNOWLEDGMENT

Open access publishing facilitated by Universita degli Studi di Milano-Bicocca, as part of the Wiley - CRUI-CARE agreement.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in Neural-based Point Clouds Registration Comparison at https://github.com/iralabdisco/neural_registration_comparison.

ORCID

Simone Fontana https://orcid.org/0000-0001-7823-8973
Federica Di Lauro https://orcid.org/0000-0001-6327-5931

REFERENCES

Agamennoni, G., Fontana, S., Siegwart, R.Y. & Sorrenti, D.G. (2016) Point clouds registration with probabilistic data association. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4092–4098.

Aoki, Y., Goforth, H., Srivatsan, R.A. & Lucey, S. (2019) PointNetLK: robust and efficient point cloud registration using pointnet. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Babin, P., Giguère, P. & Pomerleau, F. (2019) Analysis of robust functions for registration algorithms. In: *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada. pp. 1451–1457. https://doi.org/10.1109/ICRA.2019.8793791

Besl, P.J. & McKay, N.D. (1992) Method for registration of 3-D shapes. In: *Sensor fusion IV: control paradigms and data structures*, vol. 1611. SPIE, pp. 586–606. https://doi.org/10.1117/12.57955

Biber, P. & Straßer, W. (2003) The normal distributions transform: a new approach to laser scan matching. In: *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No. 03CH37453)*, Las Vegas, NV, USA, vol. 3. IEEE, pp. 2743–2748. https://doi.org/10.1109/IROS.2003.1249285

Chen, Y. & Medioni, G. (1992) Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3), 145–155.

Chicco, D. (2021) Siamese neural networks: an overview. *Artificial Neural Networks*, 2190, 73–94.

Choy, C., Park, J. & Koltun, V. (2019) Fully convolutional geometric features. In: *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South). pp. 8958–8966. https://doi.org/10.1109/ICCV.2019.00905

Deng, H., Birdal, T. & Ilic, S. (2018) PPFNet: global context aware local features for robust 3D point matching. In: *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA. pp. 195–205. https://doi.org/10.1109/CVPR.2018.00028

Fischler, M.A. & Bolles, R.C. (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381–395.

Fontana, S., Cattaneo, D., Ballardini, A.L., Vaghi, M. & Sorrenti, D.G. (2021) A benchmark for point clouds registration algorithms. *Robotics and Autonomous Systems*, 140, 103734.

Fu, K., Luo, J., Luo, X., Liu, S., Zhang, C., & Wang, M. (2021) Robust point cloud registration framework based on deep graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5), 6183–6195. https://doi.org/10.1109/TPAMI.2022.3204713

Geiger, A., Lenz, P. & Urtasun, R. (2012) Are we ready for autonomous driving? The KITTI vision benchmark suite. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Providence, RI, USA: IEEE, pp. 3354–3361. https://doi.org/10.1109/CVPR.2012.6248074

Gojcic, Z., Zhou, C., Wegner, J.D. & Andreas, W. (2019) The perfect match: 3D point cloud matching with smoothed densities. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA. pp. 5540–5549. https://doi.org/10.1109/CVPR.2019.00569

Hertz, A., Hanocka, R., Giryes, R. & Cohen-Or, D. (2020) PointGMM: a neural GMM network for point clouds. In: *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA. pp. 12054–12063. https://doi.org/10.1109/CVPR42600.2020.01207

Huang, S., Gojcic, Z., Usvyatsov, M., Wieser, A. & Schindler, K. (2021) PREDATOR: Registration of 3D point clouds with low overlap. In: *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, TN, USA. pp. 4267–4276.

Huang, X., Mei, G. & Zhang, J. (2020) Feature-metric registration: a fast semi-supervised approach for robust point cloud registration without correspondences. In: *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA. pp. 11363–11371. https://doi.org/10.1109/CVPR42600.2020.01138

Huang, X., Xu, Z., Mei, G., Li, S., Zhang, J., Zuo, Y. et al. (2021) GenReg: deep generative method for fast point cloud registration. [Preprint] Available from: https://doi.org/10.48550/arXiv.2111.11783

Jeong, J., Cho, Y., Shin, Y.-S., Roh, H. & Kim, A. (2019) Complex urban dataset with multi-level sensors from highly diverse urban environments. *The International Journal of Robotics Research*, 38(6), 642–657.

Jiang, J., Cheng, J. & Chen, X. (2009) Registration for 3-D point cloud using angular-invariant feature. *Neurocomputing*, 72(16–18), 3839–3844.

Kurobe, A., Sekikawa, Y., Ishikawa, K. & Saito, H. (2020) CorsNet: 3D point cloud registration by deep neural network. *IEEE Robotics and Automation Letters*, 5(3), 3960–3966.

Lu, W., Wan, G., Zhou, Y., Fu, X., Yuan, P. & Song, S. (2019) DeepVCP: an end-to-end deep neural network for point cloud registration. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, pp. 12–21. https://doi.org/10.1109/ICCV.2019.00010

Pomerleau, F., Colas, F., Siegwart, R. & Magnenat, S. (2013) Comparing ICP variants on real-world data sets. *Autonomous Robots*, 34(3), 133–148.

Pomerleau, F., Liu, M., Colas, F. & Siegwart, R. (2012) Challenging data sets for point cloud registration algorithms. *The International Journal of Robotics Research*, 31(14), 1705–1711.

Qi, C.R., Yi, L., Su, H. & Guibas, L.J. (2017) PointNet++: deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30. https://proceedings.neurips.cc/paper/2017/hash/d8bf84be3800d12f74d8b05e9b89836f-Abstract.html

Qin, Z., Yu, H., Wang, C., Guo, Y., Peng, Y., Ilic, S. et al. (2023) GeoTransformer: fast and robust point cloud registration with geometric transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8), 9806–9821. https://doi.org/10.1109/TPAMI.2023.3259038

Rusu, R.B., Blodow, N. & Beetz, M. (2009) Fast point feature histograms (FPFH) for 3D registration. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, Kobe, Japan: pp. 3212–3217.

Rusu, R.B., Blodow, N., Marton, Z.C. & Beetz, M. (2008) Aligning point cloud views using persistent feature histograms. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nice, France: IEEE, pp. 3384–3391. https://doi.org/10.1109/IROS.2008.4650967

Rusu, R.B. & Cousins, S. (2011) 3D is here: point cloud library (PCL). In: *2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*. pp. 1–4. https://doi.org/10.1109/ICRA.2011.5980567

Sarode, V., Li, X., Goforth, H., Aoki, Y., Srivatsan, R.A., Lucey, S. et al. (2019) PCRNet: point cloud registration network using pointnet encoding. [Preprint] Available from: https://doi.org/10.48550/arXiv.1908.07906

Segal, A.V., Haehnel, D. & Thrun, S. (2010) Generalized-ICP. In: Trinkle, J.F., Matsuoka, Y. & Castellanos J.A. (Eds.) *Robotics: science and systems*, vol. 2. Seattle, WA, p. 435.

Sehgal, A., Cernea, D. & Makaveeva, M. (2010) Real-time scale invariant 3D range point cloud registration. In: *International Conference Image Analysis and Recognition*. Springer, pp. 220–229.

Sturm, J., Engelhard, N., Endres, F., Burgard, W. & Cremers, D. (2012) A benchmark for the evaluation of RGB-D SLAM systems. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura-Algarve, Portugal: IEEE, pp. 573–580. https://doi.org/10.1109/IROS.2012.6385773

Tong, C.H., Gingras, D., Larose, K., Barfoot, T.D. & Dupuis, É. (2013) The Canadian planetary emulation terrain 3D mapping dataset. *The International Journal of Robotics Research*, 32(4), 389–395.

Wang, H., Liu, Y., Hu, Q., Wang, B., Chen, J., Dong, Z. et al. (2023) RoReg: pairwise point cloud registration with oriented descriptors and local rotations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8), 10376–10393. https://doi.org/10.1109/TPAMI.2023.3244951

Wang, Y. & Solomon, J.M. (2019) Deep closest point: learning representations for point cloud registration. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South). pp. 3522–3531. https://doi.org/10.1109/ICCV.2019.00362

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X. et al. (2015) 3D ShapeNets: a deep representation for volumetric shapes. In: *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA. pp. 1912–1920. https://doi.org/10.1109/CVPR.2015.7298801

Yang, H., Shi, J. & Carlone, L. (2020) TEASER: fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2), 314–333.

Yew, Z.J. & Lee, G.H. (2018) RPM-Net: weakly supervised local 3D features for point cloud registration. In: Ferrari, V., Hebert, M., Sminchisescu, C. & Weiss, Y. (Eds.) *Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science*, vol. 11219. Cham: Springer. https://doi.org/10.1007/978-3-030-01267-0_37

Yew, Z.J. & Lee, G.H. (2020) RPM-Net: robust point matching using learned features. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA. pp. 11821–11830. https://doi.org/10.1109/CVPR42600.2020.01184

Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J. & Funkhouser, T. (2017) 3DMatch: learning local geometric descriptors from RGB-D reconstructions. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 199–208. https://doi.org/10.1109/CVPR.2017.29

Zhang, Z. (1994) Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2), 119–152.

Zhou, Q.-Y., Park, J. & Koltun, V. (2016) Fast global registration. In: Leibe, B., Matas, J., Sebe, N. & Welling, M. (Eds.) *Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science*, vol. 9906. Cham: Springer. https://doi.org/10.1007/978-3-319-46475-6_47

Zhou, Q.-Y., Park, J. & Koltun, V. (2018) Open3D: a modern library for 3D data processing. [Preprint] Available from: https://doi.org/10.48550/arXiv.1801.09847

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Fontana, S., DiLauro, F. & Sorrenti, D. G. (2024) Assessing the practical applicability of neural-based point clouds registration algorithms: A comparative analysis. *Journal of Field Robotics*, 1–24. https://doi.org/10.1002/rob.22417