Assessing the Practical Applicability of Neural-Based Point Clouds Registration Algorithms: A Comparative Analysis*

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

Domenico G. Sorrenti
Department of Informatics, Systems and Communication
Viale Sarca 336, Milano, Italy
domenico.sorrenti@unimib.it

Federica Di Lauro
Department of Informatics, Systems and Communication
Viale Sarca 336, Milano, Italy
federica.dilauro@unimib.it

Abstract

Point cloud registration is a vital task in 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. With few exceptions, the results reveal a significant performance gap, with most neural-based methods performing poorly on real-complex data.

However, amidst the underwhelming results, 3DSmoothNet stands out as an exception, demonstrating excellent performance across the majority of benchmark sequences. Remarkably, it outperforms a state-of-the-art feature extractor, namely FPFH, showcasing its effectiveness in capturing informative and discriminative features from point clouds.

Given these results, we assert that while neural-based techniques could have advantages over conventional methods, further research is necessary to fully unlock their practical utility. We advocate for more extensive studies employing unbiased and realistic testing procedures, akin to the rigorous evaluation framework employed in this work.

This paper represents a crucial step towards establishing a more robust literature in the field of point cloud registration. By exposing the limitations and advantages of existing neural-based methods, we aim to inspire future research and drive the development of more effective techniques for real-world applications.

***Data availability:** the data used is available at https://github.com/iralabdisco/neural_registration_comparison
Funding statement: this research received no additional funding.
Conflict of interest: the authors declare no conflict of interest.

1 Introduction

Point clouds registration is the problem of finding the rototranslation that best aligns two point clouds, *i.e.* two sets of points with 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 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 *et al.* (Zhang, 1994), Besl and McKay (Besl and McKay, 1992), and Chen and Medioni (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.* (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 algorithm uses not only the distances between two points, but also the covariances computed in a neighborhood of the points (Segal et al., 2009). 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 which do not use a closest-point based data association. For example, NDT aligns two point clouds by representing them as a set of Gaussians and searching for the most likely alignment (Biber and 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 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, PFH (Rusu et al., 2008), its faster variant 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 to estimate a set of correspondences between the two point clouds. From these correspondences, the rototranslation can be estimated using, for example, RANSAC (Fischler and 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 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, where partial overlap is the most common case.

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 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, 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 et al., 2017). 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.

3DFeat-Net, 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 and Lee, 2018). It is trained in a weekly 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. 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 and 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 singular value decomposition layer estimates the transformation.

3DSmoothNet is a neural-based descriptor built using a voxelized smoothed density value representation with a Siamese deep learning architecture and fully convolutional layers (Gojcic et al., 2019). It achieves excellent results in the 3DMatch benchmark dataset. 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 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 (Fu et al., 2021), RPM-Net (Yew and Lee, 2020), 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.

While many different approaches have been proposed, their field of application, such as large scenes or single objects, and their advantages are not always clear. Most approaches are tested on very limited datasets or on datasets 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 dataset. 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 pre-trained models provided by the authors perform?
- Is the technique very sensitive to parameter settings?
- Is there an increase in performance compared to 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 to 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 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 pre-trained 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 dataset, which consists of 3D CAD 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 datasets, such as Kitti (Geiger et al., 2012) or 3DMatch (Zeng et al., 2017), or on synthetic datasets, 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 rototranslation. 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:

- 3DFeat-Net - `https://github.com/yewzijian/3DFeatNet`
- DCP - `https://github.com/WangYueFt/dcp`
- 3DSmoothNet - `https://github.com/zgojcic/3DSmoothNet`
- FCGF - `https://github.com/chrischoy/FCGF`
- Feature Metric Registration - `https://github.com/XiaoshuiHuang/fmr`

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 non-neural 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 dataset 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:

- a forest, both summer and fall, a garden with a gazebo in summer and winter, and a plain, from the ETH datasets (Pomerleau et al., 2012);

- a reproduction of a planetary exploration environment, from the Planetary datasets (Tong et al., 2013);
- an urban road environment, from the Kaist datasets (Jeong et al., 2019).

The indoor environments, on the other hand, are:

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

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, *i.e.*, 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, *i.e.*, 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 dataset-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 datasets (Sturm et al., 2012; Tong et al., 2013; Jeong et al., 2019; Pomerleau et al., 2012).

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. 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 trasformation 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:

- To search for the most similar descriptors in the other point cloud, *i.e.*, 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 $VOXEL\_SIZE \cdot 1.5$

- For Fast Global Registration, we used as "max_correspondence_distance" a value equal to $VOXEL\_SIZE \cdot 0.5$
- 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.

Other parameters are algorithm specific and are described below.

2.2.1 3DSmoothNet

Before extracting features using 3DSmoothNet, we applied a voxel grid filter with a leaf size of $0.05m$. Moreover, we applied an outlier removal step to the point clouds. Specifically, we used a Statistical Outlier Removal from the PCL library (Rusu and Cousins, 2011): For each point P, we search for the K closed neighbors N_i and calculate the mean distance

$$\sum_{i}^{K} \frac{\|P - N_i\|_2}{K}$$

and the standard deviation. Any neighbor whose distance is greater than J times the standard deviation is considered an outlier and removed. The parameters we use are $K = 10$ and $J = 3$.

3DSmoothNet uses a voxelized smoothed density value representation. Therefore, we generated it for 5000 randomly selected points. The voxel grid has a size of $16x16x16$, 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 Feature Metric Registration (FMR), we downsampled the point clouds using a voxel grid with a leaf size of 0.1 meters rather than 0.05, because otherwise we kept running out of memory on our GPU (a NVidia GeForce GTX 1080 Ti with 11 GByte of memory). The iteration number used is 10, as suggested by the authors. We performed experiments with two different pre-trained models, trained with the ModelNet40 and 7Scenes datasets.

2.2.3 FPFH

FPFH is a non-neural feature. We changed only one parameter compared to the default settings, namely the voxel size (we used 0.2 meters).

2.2.4 3DFeatNet

For 3DFeatNet, we downsampled the original point clouds with a voxel grid with leaf size of 0.1 The pameters of the feature extraction method we used are:

- Radius for sampling clusters = 2.0
- Maximum number of points to consider per cluster= 64
- Feature dimension size = 32
- Radius for non-maximal suppression = 0.5
- Minumum response ratio= 1e-3
- Maximum number of keypoints = 1024

2.2.5 Fully Convolutional Geometric Feature

There are various pre-trained models for Fully Convolutional Geometric Feature. 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 pre-trained on 3DMatch are summarized below:

- model = 3dmatch_normalized_5cm_32
- voxel size = 0.05
- output features = 32
- normalized = true
- conv 1 kernel size = 7
- D = 3

Those used with the Kitti pre-trained model, instead, are:

- model = kitti_notnormalized_20cm_32
- voxel size = 0.2
- output features = 32
- normalize = False
- conv 1 kernel size = 7
- D = 3

Table 1: Statistics on the number of points composing the point clouds in the various sequences. The point clouds have been downsampled using a voxel grid filter with leaf size of 0.1m.

sequence	Mean number of points	Max	Min
plain	10961	12656	8884
stairs	12685	20243	6873
apartment	8355	12275	3966
hauptgebaude	32111	36767	28622
wood_autumn	34660	38786	31601
wood_summer	36352	46369	31436
gazebo_summer	23626	34458	15221
gazebo_winter	26397	33054	21270
box_met	41452	56521	23909
p2at_met	18113	32751	6369
pioneer_slam	3802	7834	261
pioneer_slam3	3379	7467	1286
long_office_household	1834	5596	147
urban05	12170	20126	5581

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. 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 GPU with 11Gbytes 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, 64200 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 36352 points or the sequence box_met with an average size of 41452 points).

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:

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

where $\overline{S}$ is the centroid of S, $T \cdot S$ is the application of the rototranslation 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 firstly to allow comparison with other experiments on the same benchmark, as described in the work of Fontana *et al.*. Second, because 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 on the original paper (Fontana et al., 2021).

Tables 2, 4, 6, 8, 9, 12 and 14 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. Feature Metric Registration, on the other hand, is an end-to-end approach; for this reason it has 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 tables 3, 5, 7, 13 and 15. Columns "Median of Final Errors" have the same meaning as columns Medians" in the previous tables. Columns Median of Initial Errors" is the median of the distances between the misaligned source point clouds and the source point clouds aligned with the ground truth. Thus, it represents the error that an algorithm must correct. Column Residual" represents 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:

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

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 to better identify how effective an algorithm is.

Figure 1 compares the results obtained with 3DSmoothNet and FPFH features with TEASER. We chose to show only the results of these features because they are the best among those compared. Moreover, the difference in the magnitude of errors between the best and the worst algorithm is so large that it would be problematic to show all algorithms in one plot.

We set the parameters of the different approaches to obtain the best results within the limits of the available hardware. Although most parameters depend on the particular method and are therefore not comparable, we always applied a voxel grid filter as a preprocessing step. The leaf size was adjusted to obtain the best results, resulting in different values being used for different approaches. We are aware that using a different subsampling could be considered unfair. Therefore, we also performed experiments using the same granularity of subsampling for each approach. That is, a voxel grid filter with a leaf size of 0.1 meters. However, the results of this second set of experiments are virtually identical to those of the others and do not change the discussion. Therefore, we do not report them here, but provide them as supplementary material.

4 Discussion

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 datasets 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

Table 2: Results obtained by employing 3DFeatNet features in conjunction with RANSAC, FastGlobal, and TEASER. Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

	RANSAC			FastGlobal			TEASER		
	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q
sequence									
plain	8.40	12.24	18.73	4.99	6.17	7.61	9.57	13.26	22.57
stairs	2.17	3.09	7.07	2.31	3.36	5.26	2.30	3.78	7.39
apartment	1.92	2.78	4.81	2.03	2.53	3.73	2.22	3.27	5.10
hauptgebaude	2.72	6.47	9.28	2.79	4.91	6.31	2.44	5.57	8.72
wood_autumn	3.20	4.42	5.92	2.53	3.62	5.10	2.87	4.10	5.95
wood_summer	2.79	3.73	5.34	2.21	3.52	4.17	2.49	3.83	5.06
gazebo_summer	3.51	4.81	8.41	2.33	3.30	4.17	3.13	4.98	7.15
gazebo_winter	3.72	5.47	9.10	2.15	3.34	4.51	3.14	4.92	8.09
box_met	10.14	12.76	19.82	8.30	10.53	13.98	13.01	28.65	81.78
p2at_met	10.43	17.48	32.84	6.27	10.27	15.89	10.22	18.40	89.83
planetary_map	41.00	60.25	100.93	27.08	42.20	61.59	40.76	62.61	103.20
pioneer_slam	10.25	17.67	35.64	6.74	8.91	10.64	13.87	27.94	43.60
pioneer_slam3	4.73	9.13	13.84	5.86	7.98	9.47	7.47	12.44	21.51
long_office_household	14.93	27.53	41.75	4.13	7.77	11.24	9.68	28.31	72.94
urban05	2.27	3.54	6.79	1.33	2.03	3.47	2.17	3.71	7.09
Total	4.21	9.96	37.31	3.53	6.53	21.15	4.17	10.55	50.44

Table 3: 3DFeatNet median of residual errors. Transformation estimated using TEASER

	Median of Final Errors	Median of Initial Errors	Residual (%)
sequence			
plain	9.57	7.05	127.27
stairs	2.30	4.58	55.48
apartment	2.22	5.35	42.03
hauptgebaude	2.44	2.60	90.14
wood_autumn	2.87	3.09	91.15
wood_summer	2.49	2.99	87.81
gazebo_summer	3.13	2.76	111.50
gazebo_winter	3.14	3.54	97.81
box_met	13.01	7.61	156.58
p2at_met	10.22	9.98	114.94
planetary_map	40.76	10.80	343.98
pioneer_slam	13.87	13.22	100.00
pioneer_slam3	7.47	8.97	85.41
long_office_household	9.68	17.40	87.23
urban05	2.17	2.51	87.11
Total	4.17	4.98	97.47

Table 4: Results obtained by employing FCGF features with the model trained on 3dMatch in conjunction with RANSAC, FastGlobal, and TEASER. Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error

	RANSAC			FastGlobal			TEASER		
	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q
sequence									
plain	7.05	8.90	13.55	7.22	9.19	13.95	7.22	9.34	13.82
stairs	5.77	7.45	9.15	5.85	7.43	9.28	5.85	7.50	9.28
apartment	5.93	7.38	10.94	6.15	7.62	10.33	6.15	7.63	10.33
hauptgebaude	2.53	3.08	3.71	2.59	3.22	3.78	2.62	3.18	3.78
wood_autumn	3.19	4.21	5.64	3.92	5.12	5.79	3.84	5.12	5.79
wood_summer	2.96	3.73	4.98	2.60	3.44	4.55	2.54	3.51	4.63
gazebo_summer	2.84	4.07	5.06	3.11	4.35	5.20	3.14	4.36	5.44
gazebo_winter	3.50	4.79	5.85	3.60	4.77	5.85	3.61	4.78	5.68
box_met	7.61	10.84	13.79	8.40	12.07	17.89	8.92	11.43	17.93
p2at_met	9.82	12.75	17.48	8.43	13.13	21.25	8.69	12.97	19.59
planetary_map	11.33	15.52	21.16	31.09	46.49	67.82	31.92	47.25	67.94
pioneer_slam	11.74	18.76	30.26	11.77	18.76	30.29	11.77	18.76	30.30
pioneer_slam3	7.07	12.86	16.62	7.07	12.86	16.64	7.06	12.85	16.63
long_office_household	9.21	20.70	41.05	9.41	20.66	41.05	9.68	20.65	41.05
urban05	2.51	3.76	5.30	1.41	1.93	3.16	2.27	3.68	11.04
Total	4.85	8.73	18.11	4.92	9.53	30.27	5.08	9.71	30.35

Table 5: FCGF median of residual errors using the model trained on 3dmatch. Transformation estimated using TEASER.

	Median of Final Errors	Median of Initial Errors	Residual (%)
sequence			
plain	7.22	7.05	100.00
stairs	5.85	4.58	100.00
apartment	6.15	5.35	100.00
hauptgebaude	2.62	2.60	100.00
wood_autumn	3.84	3.09	100.00
wood_summer	2.54	2.99	100.00
gazebo_summer	3.14	2.76	100.00
gazebo_winter	3.61	3.54	100.00
box_met	8.92	7.61	100.00
p2at_met	8.69	9.98	100.00
planetary_map	31.92	10.80	290.64
pioneer_slam	11.77	13.22	96.91
pioneer_slam3	7.06	8.97	100.00
long_office_household	9.68	17.40	74.19
urban05	2.27	2.51	89.55
Total	5.08	4.98	100.00

Table 6: Results obtained by employing FCGF features with the model trained on Kitti in conjunction with RANSAC, FastGlobal, and TEASER. Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error

	RANSAC			FastGlobal			TEASER		
	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q
sequence									
plain	10.21	12.56	19.85	7.78	10.10	14.51	11.63	16.89	32.94
stairs	6.04	7.56	10.13	6.02	7.92	10.38	6.17	7.83	11.73
apartment	5.94	7.35	10.88	6.07	7.65	11.77	6.06	7.69	12.04
hauptgebaude	2.79	3.51	7.78	3.02	3.93	6.82	3.51	5.38	8.63
wood_autumn	3.88	5.09	6.76	3.89	5.29	7.85	4.15	5.43	8.67
wood_summer	3.30	4.16	6.03	3.31	4.21	6.36	3.42	4.90	6.75
gazebo_summer	3.73	4.93	7.34	3.45	4.35	5.75	4.43	6.10	9.47
gazebo_winter	4.03	5.43	8.85	4.06	4.85	6.34	4.67	5.85	10.72
box_met	10.46	14.21	22.49	9.00	12.32	18.95	12.12	16.54	29.09
p2at_met	12.73	18.62	30.77	8.98	13.79	21.26	14.22	23.06	38.71
planetary_map	32.19	47.00	66.02	31.18	46.69	64.70	33.58	49.06	71.22
pioneer_slam	10.84	19.36	30.27	10.62	18.48	29.57	11.09	21.03	33.51
pioneer_slam3	9.08	13.85	17.85	9.51	13.39	19.63	11.97	15.67	22.10
long_office_household	12.96	24.14	41.05	11.56	21.19	42.18	12.65	23.06	41.05
urban05	2.78	4.01	8.11	1.61	2.92	6.08	3.93	7.69	12.94
Total	5.91	11.73	30.85	5.55	10.29	29.74	6.71	13.29	35.32

Table 7: FCGF median of residual errors using the model trained on 3dmatch. Transformation estimated using TEASER.

	Median of Final Errors	Median of Initial Errors	Residual (%)
sequence			
plain	11.63	7.05	158.19
stairs	6.17	4.58	101.03
apartment	6.06	5.35	100.00
hauptgebaude	3.51	2.60	110.41
wood_autumn	4.15	3.09	111.79
wood_summer	3.42	2.99	105.34
gazebo_summer	4.43	2.76	137.91
gazebo_winter	4.67	3.54	120.31
box_met	12.12	7.61	137.43
p2at_met	14.22	9.98	123.53
planetary_map	33.58	10.80	327.53
pioneer_slam	11.09	13.22	100.00
pioneer_slam3	11.97	8.97	100.19
long_office_household	12.65	17.40	85.93
urban05	3.93	2.51	159.84
Total	6.71	4.98	114.00

Table 8: Results obtained by employing FMR with the model trained on 7Scene. Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

sequence	Median	0.75 Q	0.95 Q
plain	6.91	9.54	14.18
stairs	5.91	9.88	30.64
apartment	2.48	3.82	5.78
hauptgebaude	5.45	7.94	10.18
wood_autumn	4.15	5.53	7.84
wood_summer	3.23	4.95	6.99
gazebo_summer	3.04	4.15	5.63
gazebo_winter	6.20	10.62	17.56
box_met	8.00	9.82	14.07
p2at_met	8.82	13.23	21.07
planetary_map	54.23	82.47	131.94
pioneer_slam	4.50	6.66	10.52
pioneer_slam3	5.96	9.36	10.59
long_office_household	5.66	12.60	20.02
urban05	6.59	12.04	20.23
Total	5.33	9.33	36.02

Table 9: Results obtained by employing FMR with the model trained on ModelNet40. Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error.

sequence	Median	0.75 Q	0.95 Q
plain	6.37	8.03	10.81
stairs	3.63	6.95	19.36
apartment	2.44	3.47	5.33
hauptgebaude	4.59	7.16	8.49
wood_autumn	4.07	4.98	6.51
wood_summer	2.68	4.11	5.32
gazebo_summer	2.67	3.89	5.28
gazebo_winter	5.51	8.78	14.51
box_met	7.80	9.63	12.74
p2at_met	7.66	10.29	16.97
planetary_map	30.33	43.25	59.60
pioneer_slam	6.13	8.45	12.87
pioneer_slam3	4.62	9.82	10.60
long_office_household	3.36	9.48	14.40
urban05	3.78	8.63	19.69
Total	4.84	8.50	23.85

Table 10: Feature Metric Registration median of residual errors with 7scene pre-trained model residual.

sequence	Median of Final Errors	Median of Initial Errors	Residual (%)
plain	6.91	7.05	94.85
stairs	5.91	4.58	109.02
apartment	2.48	5.35	46.62
hauptgebaude	5.45	2.60	221.21
wood_autumn	4.15	3.09	127.04
wood_summer	3.23	2.99	108.58
gazebo_summer	3.04	2.76	106.52
gazebo_winter	6.20	3.54	186.24
box_met	8.00	7.61	99.10
p2at_met	8.82	9.98	79.21
planetary_map	54.23	10.80	487.49
pioneer_slam	4.50	13.22	30.20
pioneer_slam3	5.96	8.97	59.76
long_office_household	5.66	17.40	39.98
urban05	6.59	2.51	256.98
Total	5.33	4.98	100.91

Table 11: Feature Metric Registration median of residual errors with 7scene pre-trained model residual.

sequence	Median of Final Errors	Median of Initial Errors	Residual (%)
plain	6.37	7.05	83.06
stairs	3.63	4.58	73.68
apartment	2.44	5.35	41.58
hauptgebaude	4.59	2.60	164.49
wood_autumn	4.07	3.09	114.36
wood_summer	2.68	2.99	104.69
gazebo_summer	2.67	2.76	100.16
gazebo_winter	5.51	3.54	169.84
box_met	7.80	7.61	99.20
p2at_met	7.66	9.98	75.91
planetary_map	30.33	10.80	291.77
pioneer_slam	6.13	13.22	44.97
pioneer_slam3	4.62	8.97	49.06
long_office_household	3.36	17.40	24.65
urban05	3.78	2.51	171.84
Total	4.84	4.98	89.95

Table 12: Results obtained by employing FPFH features in conjunction with RANSAC, FastGlobal, and TEASER. Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error

	RANSAC			FastGlobal			TEASER		
	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q
sequence									
plain	0.41	3.74	6.87	0.49	1.11	1.89	0.12	0.31	1.83
stairs	0.21	2.37	4.45	0.28	1.95	3.89	0.08	0.58	3.57
apartment	0.21	1.33	3.57	0.21	1.15	2.67	0.13	1.46	3.47
hauptgebaude	0.19	1.61	3.34	0.36	1.05	1.86	0.02	0.97	2.04
wood_autumn	2.90	3.75	5.43	0.32	0.51	0.89	0.05	0.08	0.24
wood_summer	2.39	3.69	5.23	0.30	0.47	1.11	0.04	0.07	0.17
gazebo_summer	1.90	2.77	4.38	0.20	0.43	1.21	0.04	0.06	0.27
gazebo_winter	1.52	2.97	5.60	0.18	0.37	1.25	0.04	0.08	0.40
box_met	7.44	10.46	13.23	7.16	9.87	14.12	0.31	7.62	12.10
p2at_met	8.04	12.35	17.48	1.90	6.01	10.05	0.16	0.66	9.92
planetary_map	10.80	15.11	20.78	29.82	43.52	69.25	46.45	62.77	95.28
pioneer_slam	0.36	1.57	12.30	3.24	7.46	10.79	0.46	1.92	7.24
pioneer_slam3	0.13	0.19	0.43	0.13	0.23	0.64	0.14	0.21	0.90
long_office_household	0.48	1.69	18.07	1.47	6.91	12.22	0.66	1.67	15.04
urban05	2.93	3.89	5.53	1.53	2.13	3.44	2.77	4.07	6.40
Total	1.82	4.48	13.12	0.48	2.44	19.90	0.15	1.56	33.49

Table 13: FPFH final median of residual errors. Transformation estimated using TEASER.

	Median of Final Errors	Median of Initial Errors	Residual (%)
sequence			
plain	0.12	7.05	1.89
stairs	0.08	4.58	2.69
apartment	0.13	5.35	2.53
hauptgebaude	0.02	2.60	1.28
wood_autumn	0.05	3.09	1.65
wood_summer	0.04	2.99	1.44
gazebo_summer	0.04	2.76	1.19
gazebo_winter	0.04	3.54	1.14
box_met	0.31	7.61	4.40
p2at_met	0.16	9.98	2.04
planetary_map	46.45	10.80	448.93
pioneer_slam	0.46	13.22	5.02
pioneer_slam3	0.14	8.97	1.51
long_office_household	0.66	17.40	3.94
urban05	2.77	2.51	94.63
Total	0.15	4.98	2.58

Table 14: Results obtained by employing 3DSmoothNet features in conjunction with RANSAC, FastGlobal, and TEASER. Results are reported as the median, 0.75 quantile, and 0.95 quantile of the residual error

	RANSAC			FastGlobal			TEASER		
	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q	Median	0.75 Q	0.95 Q
sequence									
plain	0.04	0.09	10.33	1.40	2.22	3.18	0.03	1.30	2.83
stairs	0.01	0.03	5.16	1.08	2.55	4.95	0.01	0.03	4.54
apartment	0.01	0.01	0.06	0.59	1.67	3.63	0.01	0.02	0.04
hauptgebaude	0.79	0.85	2.35	0.76	1.34	2.00	0.80	1.58	2.37
wood_autumn	0.02	0.02	0.05	0.65	1.46	1.82	0.01	0.02	0.02
wood_summer	0.01	0.02	0.03	0.47	0.95	1.66	0.01	0.01	0.02
gazebo_summer	0.01	0.02	0.08	0.57	1.47	3.94	0.01	0.01	0.02
gazebo_winter	0.01	0.01	0.03	0.39	0.78	2.26	0.01	0.01	0.02
box_met	7.44	10.75	13.79	8.69	10.49	13.89	7.05	9.90	13.24
p2at_met	4.93	10.61	15.92	6.37	8.00	11.89	0.16	6.54	11.84
planetary_map	11.08	15.22	20.95	29.89	41.09	66.74	41.08	64.64	101.85
pioneer_slam	0.07	0.10	0.72	0.08	0.16	0.80	0.03	0.07	0.12
pioneer_slam3	0.03	0.05	0.09	0.26	0.66	3.14	0.01	0.02	0.03
long_office_household	0.08	0.14	0.27	0.06	0.13	0.35	0.04	0.08	0.20
urban05	2.51	3.62	5.29	1.66	2.07	3.10	1.32	1.93	3.37
Total	0.04	1.83	12.28	0.89	2.60	18.92	0.02	0.80	27.31

Table 15: 3DSmoothNet median of residual errors. Transformation estimated using TEASER

	Median of Final Errors	Median of Initial Errors	Residual (%)
sequence			
plain	0.03	7.05	0.52
stairs	0.01	4.58	0.32
apartment	0.01	5.35	0.23
hauptgebaude	0.80	2.60	25.66
wood_autumn	0.01	3.09	0.37
wood_summer	0.01	2.99	0.33
gazebo_summer	0.01	2.76	0.32
gazebo_winter	0.01	3.54	0.21
box_met	7.05	7.61	78.46
p2at_met	0.16	9.98	1.84
planetary_map	41.08	10.80	389.31
pioneer_slam	0.03	13.22	0.31
pioneer_slam3	0.01	8.97	0.15
long_office_household	0.04	17.40	0.26
urban05	1.32	2.51	49.65
Total	0.02	4.98	0.44

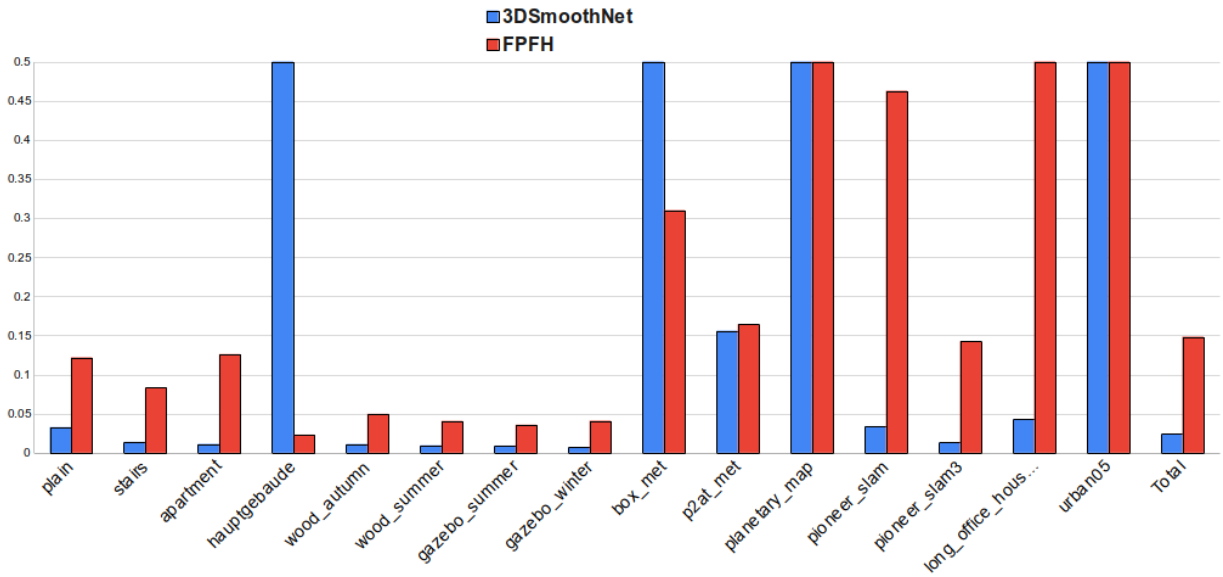

Figure 1: Comparison of results using 3DSmoothNet and FPFH features with TEASER

simpler problems and sometimes on single objects rather than on complex scenes. Therefore, it is not surprising that most of them performed very poorly on real complex datasets.

Nevertheless, we still consider very valuable a fair comparison on challenging datasets. 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 datasets, 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.

Of the approaches we tested, only 3DSmoothNet and FPFH (which is not based on a neural network) produced satisfactory results. In our opinion, the differences between the other algorithms are negligible, since none of them could achieve useful results. This seems obvious when looking at the residuals in tables 3, 5, 7, 10 and 11. As can be seen, the other approaches often actually worsened the misalignment, obtaining a residual of more than 100% of the initial error. 3DFeatNet achieved useful results only for the *stairs* and *apartment* sequences. FMR also produced useful results for some sequences, such as those from the TUM dataset (*pioneer_slam*, *pioneer_slam3*, and *long_office_household*) or the sequence *apartment*. While these results are not amazing, they can still be considered useful, considering that this is a global registration that is usually refined with a local registration approach. On the other hand, FCGF never achieved useful results.

On the other hand, 3DSmoothNet and FPFH both obtained very good results. These algorithms are actually feature extractor. Therefore, we need another algorithm to align the extracted features. We tested each approach with RANSAC, FastGlobal and TEASER. However, TEASER consistently obtained the best results, hence, for the rest of the discussion we will consider only results obtained with either 3DSmoothNet or FPFH and TEASER.

On the sequences from the ETH datasets, 3DSmoothNet obtained excellent results, often with a median error of 0.01, corresponding to an alignment that is essentially consistent with ground truth. Of these, the 75th percentile for *wood_summer*, *gazebo_winter*, and *gazebo_summer* is also 0.01 and the 0.95th percentile is only slightly worse, indicating that there is also very little variability in the quality of the results. The only exception is the *hauptgebaude* sequence, for which the result is still usable but not great. The *hauptgebaude*

sequence has a stronger geometric structure than the woods or gazebos, a property that should theoretically help feature-based algorithms. On the other hand, it is also repetitive, which could make global registration task hard. However, it should be noted that the same behavior is not observed with the other algorithms, which obtained comparable or even better results on the *hauptgebaude* sequence with respect to the others from the ETH dataset. It is possible that the results are due to a non-optimal choice of parameters.

It should be noted that comparing approaches without fine-tuning parameters specifically to each sequence is a goal of the Point Clouds Registration Benchmark and of this work. Indeed, in a real-world application, it is often impossible to fine-tune parameters with exactly the same data that will be used once the system is deployed. Therefore, we believe that this type of experimentation gives unrealistically good results that will not match the practical experience of the end user. Nevertheless, the results on the *hauptgebaude* sequence could still be considered useful as global registration, considering its median residual error of 25% of the initial value.

The results on the planetary sequences are varied. On the *p2at_ met* sequence, the results are excellent (albeit with a high variability), while on the box_met sequence they are much worse. This is surprising given the similarities between the two sequences. The results on the *planetary_ map* sequence are not usable at all, but this is to be expected considering that this sequence is very difficult for global registration algorithms.

On the TUM sequences, the results are excellent and comparable to those of the ETH sequences. It should be noted that they use two different sensors: a laser scanner in the ETH sequences, an RGB-D camera in the TUM sequences. The ability to use the same model and parameters for two completely different sensors is remarkable.

On the urban05 sequence, the results are not as good as with other sequences, but still usable, with a median residual error that is 50% of the original, which is not sufficient for a final alignment, but could be a good starting point for a local refinement technique.

We also provide results using FPFH features, which although not based on a neural network, are a good starting point for comparison since they are the most commonly used feature for global registration of point clouds. From the results, we can see that they perform much better than most neural-based techniques. This is notable because they do not require any training phase, have fewer parameters, and require much less computational resources. The only exception is 3DSmoothNet, whose results are generally better and can be considered the only real improvement over the baseline represented by FPFH.

5 Conclusions

We compared well-known neural-based point cloud registration techniques on the Point Clouds Registration Benchmark. The results show that their performance is very poor when applied to real-complex data, in contrast to the much simpler experiments usually performed by the original authors. The only exception is 3DSmoothNet, which achieved excellent results on most sequences of the benchmark and outperformed a conventional state-of-the-art feature extractor. In addition, our tests show whether the results depend on fine-tuning the parameters for each sequence, which is often impossible in practice. Given the results, we believe that while neural-based techniques may theoretically have many advantages over conventional techniques, more research is needed to make them truly useful in practice. In addition, we encourage further research using unbiased and realistic testing procedures such as we have used. We believe this is a fundamental step toward a more sound literature.

References

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

Systems (IROS), pages 4092–4098. IEEE.

Aoki, Y., Goforth, H., Srivatsan, R. A., and 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., and Pomerleau, F. (2019). Analysis of robust functions for registration algorithms. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1451–1457.

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie.

Biber, P. and Straßer, W. (2003). The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2743–2748. IEEE.

Chen, Y. and 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*, pages 73–94.

Choy, C., Park, J., and Koltun, V. (2019). Fully convolutional geometric features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8958–8966.

Deng, H., Birdal, T., and Ilic, S. (2018). Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 195–205.

Fischler, M. A. and 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., and Sorrenti, D. G. (2021). A benchmark for point clouds registration algorithms. *Robotics and Autonomous Systems*, 140:103734.

Fu, K., Liu, S., Luo, X., and Wang, M. (2021). Robust point cloud registration framework based on deep graph matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8893–8902.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE.

Gojcic, Z., Zhou, C., Wegner, J. D., and Andreas, W. (2019). The perfect match: 3d point cloud matching with smoothed densities. In *International conference on computer vision and pattern recognition (CVPR)*.

Hertz, A., Hanocka, R., Giryes, R., and Cohen-Or, D. (2020). Pointgmm: A neural gmm network for point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12054–12063.

Huang, X., Mei, G., and Zhang, J. (2020). Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11366–11374.

Huang, X., Xu, Z., Mei, G., Li, S., Zhang, J., Zuo, Y., and Wang, Y. (2021). Genreg: Deep generative method for fast point cloud registration. *arXiv preprint arXiv:2111.11783*.

Jeong, J., Cho, Y., Shin, Y.-S., Roh, H., and 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., and 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., and 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., and 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)*, pages 12–21. IEEE.

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

Pomerleau, F., Liu, M., Colas, F., and 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., and Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE.

Rusu, R. B., Blodow, N., Marton, Z. C., and Beetz, M. (2008). Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ international conference on intelligent robots and systems*, pages 3384–3391. IEEE.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Sarode, V., Li, X., Goforth, H., Aoki, Y., Srivatsan, R. A., Lucey, S., and Choset, H. (2019). Pcrnet: Point cloud registration network using pointnet encoding. *arXiv preprint arXiv:1908.07906*.

Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA.

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

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE.

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

Wang, Y. and Solomon, J. M. (2019). Deep closest point: Learning representations for point cloud registration. In *The IEEE International Conference on Computer Vision (ICCV)*.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.

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

Yew, Z. J. and Lee, G. H. (2018). 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *ECCV*.

Yew, Z. J. and Lee, G. H. (2020). Rpm-net: Robust point matching using learned features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., and Funkhouser, T. (2017). 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*.

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., and Koltun, V. (2016). Fast global registration. In *European conference on computer vision*, pages 766–782. Springer.

Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*.