Received: 30 January 2017 | Revised: 16 August 2017 | Accepted: 13 October 2017

DOI: 10.1002/rob.21768

REGULAR ARTICLE

WILEY

Learning to detect misaligned point clouds

Håkan Almqvist | Martin Magnusson | Tomasz P. Kucner | Achim J. Lilienthal

Örebro University, Sweden

Correspondence
Håkan Almqvist, Centre for Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden.
Email: hakan.almqvist@oru.se

Funding information
The european union's Horizon 2020 research and innovation programme, Grant/Award Number: 732737 (ILIAD); Stiftelsen för Kunskaps- och Kompetensutveckling, Grant/Award Number: 20110214 (ALLO)

Abstract

Matching and merging overlapping point clouds is a common procedure in many applications, including mobile robotics, three-dimensional mapping, and object visualization. However, fully automatic point-cloud matching, without manual verification, is still not possible because no matching algorithms exist today that can provide any certain methods for detecting misaligned point clouds. In this article, we make a comparative evaluation of geometric consistency methods for classifying aligned and nonaligned point-cloud pairs. We also propose a method that combines the results of the evaluated methods to further improve the classification of the point clouds. We compare a range of methods on two data sets from different environments related to mobile robotics and mapping. The results show that methods based on a Normal Distributions Transform representation of the point clouds perform best under the circumstances presented herein.

KEYWORDS
perception, mapping, position estimation

1 | INTRODUCTION

Point-cloud registration is a central aspect of robot perception. It is used in numerous applications of mobile robotics, including mapping, object detection, manipulation, etc. Point-cloud registration can be formulated as the problem of finding the relative transformation (i.e., a translation and rotation) between two three-dimensional (3D) point clouds that best aligns them. However, no existing point-cloud registration algorithm will always perfectly align a pair of point clouds. Therefore, methods for automatic detection of misaligned point clouds are important, but this is a problem that has not been thoroughly studied as of yet.

The problem is visualized in Figures 1 and 2: two point clouds that are aligned are shown in Figure 1 and two point clouds that are misaligned are shown in Figure 2.

In this article we will present an investigation of methods that can be used for automatic detection of misaligned point clouds.

In related work, some authors have previously proposed measuring global map quality by comparing the output of a mapping algorithm to a ground-truth reference. References 1 and 2 proposed a formal definition of map brokenness, and an algorithm for measuring how many such inconsistencies are present in a map. However, being dependent on a given reference map is a severe limitation in autonomous mapping and localization applications. In the present paper, we will only investigate methods for determining whether two point clouds are aligned or not, without requiring a comparison to a ground-truth estimate.

The typical work cycle when doing metric mapping with, for example, a mobile robot is to acquire consecutive, overlapping 3D point clouds and to merge them together using a point cloud registration algorithm. Such algorithms try to find the best relative transformation between two point clouds, starting from an initial alignment estimate.

All currently available local and global registration algorithms can fail to find the correct alignment. For challenging data sets, both local and global state-of-the-art registration algorithms can have relatively low success rates.[3] The predominant causes of failure are bad initial alignment or small overlap of the point clouds. Typically, the registration algorithm gets trapped in a local minimum (with respect to its objective function) in the failure cases.

In any scenario where the process of aligning point clouds is intended to be automated, a reliable method for verifying the point cloud alignment is required.

There are a number of popular methods available in the literature that approach this problem (see Section 2), however there is currently no consensus regarding which method is most reliable. To the best of our knowledge, no methodical evaluation of these measures has yet been carried out. We are going to evaluate some common methods for point cloud misalignment detection, as well as some less established ones, by assessing how well they can be used to discriminate between well-aligned and misaligned scans on two labeled large-scale data sets.

The methods presented in this article can be applied to any situation where an automated method for determining whether two point clouds are aligned or not is required, although the datasets

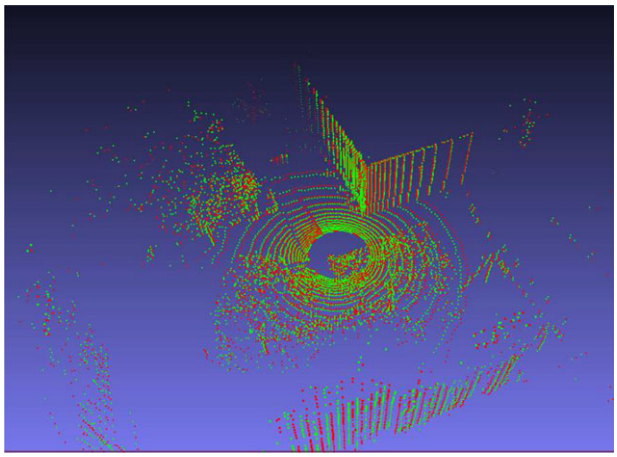

FIGURE 1 Two point clouds, one in red and one in green, that are well aligned

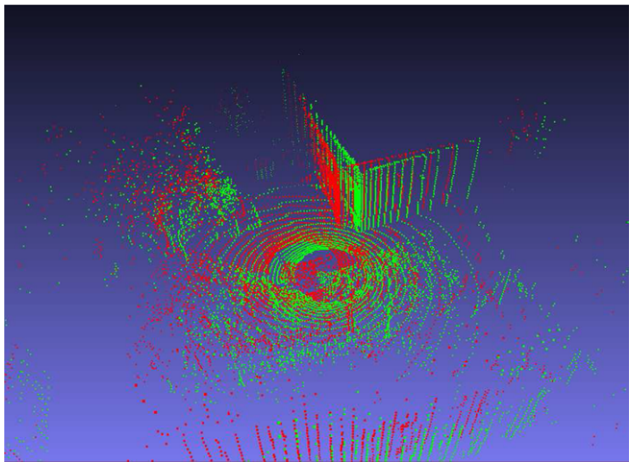

FIGURE 2 Two point clouds, one in red and one in green, that are not well aligned

chosen for the work herein are connected to localization in mobile robotics.

We will refer to a method capable of determining whether two point clouds are aligned or not as a "classifier," for the simple reason that what it does is to "classify" the point clouds as aligned or not aligned.

This article provides two main contributions:

- Evaluations of several existing methods for detecting whether two point clouds are aligned or not.
- The use of boosting to learn a strong classifier by combining the weaker individual classifiers.

2 | ALIGNMENT QUALITY MEASURES

This section briefly lists the methods that we have studied. More details of each method can be found in Section 3. Alignment quality of point clouds can be investigated at different levels, from maps consisting of many point clouds to point cloud pairs.

Alignment quality measures for point cloud pairs can roughly be divided into two classes: those that can be used directly upon any pair of point clouds, and those that rely on measures taken both before and after a point cloud registration, thus measuring how the alignment of the point clouds has changed as a consequence of the registration. The second class is based on the assumption that a point cloud pair where an alignment quality measure generates a better result after registration than before indicates that the pairs' alignment has been improved. Since we are interested in methods that can classify whether two point clouds are aligned or not in all contexts, without requiring registration, we will not consider the second class of methods here. It is also precarious to evaluate point cloud alignment before and after registrations using the same measures as used by the registration algorithm. We will discuss this more thoroughly in Section 5.1.3.

A common property of all point cloud alignment measures is that the result is dependent on the scene being evaluated. Therefore, no method provides a universal solution that is capable of correctly describing the alignment of any point cloud pair.

Previous work in the area of evaluating the performance of different point cloud alignment measures is scarce. Notable, however, is Ref. 4, in which evaluations were recently done on four measures and also a combined measure was performed using a multiclass support vector machine. The evaluations show that the combined measure achieves better results than the single measures at the cost of requiring a larger training set.

RMS

A popular method[5] is to use the root-mean-squared (RMS) point-to-point distance between the point clouds, which also is the function that is minimized by iterative closest point (ICP) registration.[6] ICP registration will be further introduced in Section 5.1.3. The RMS distance is a measure of the average distance between the nearest-neighbor point pairs in two point clouds. Point clouds that are well aligned should receive a small positive RMS value.

However, as demonstrated by, for example, Ref. 7, the mean-squared error is not always a good measure of alignment quality—especially not for detecting small errors, since it does not reveal any information concerning the distribution of the errors.

When computing the RMS distance, care has to be taken not to include outlier points, or points from nonoverlapping regions, because such points can influence the computed value drastically, even for otherwise well-aligned point clouds. Therefore, points that do not have a neighbor in the other point cloud within a threshold distance are considered outliers and removed.

NDT Score

NDT, the normal distributions transform,[8–10] can be used to acquire a measure of point cloud alignment. In the NDT representation, one[11] or both[12] point clouds are represented by a combination of Gaussians, describing the probability of finding part of the surface at any point in space. The level of alignment can then be measured by evaluating the NDT representation for the point clouds as described in detail in Section 3.2.

NDT Hessian

The inverse Hessian matrix of the NDT score function can be used as an estimate of the covariance matrix of the pose parameters, and as such gives an indication of the certainty by which each pose parameter can be determined. Although the NDT is often used for point cloud registration,[3,8] the score and Hessian, given a pair of point clouds and a relative transformation, can be computed without actually performing a registration.

Well-aligned point clouds should have an inverse NDT Hessian where all eigenvalues are small.

Plane extraction

References 13 and 14 train a conditional random field to detect "suspicious" and "plausible" areas of a map. The suspicious areas, in this case, are those that contain intersecting plane patches or parallel planes in close proximity. This work is one of few where the accuracy of the proposed method is evaluated. In their evaluations on 3D-laser data, the percentage of correct classifications reaches approximately 80%. We have made an attempt to adapt the method to work on point cloud pairs in Section 3.4.

Partitioned mean normals

The partitioned mean normals measure, used by Ref. 15, uses consistency between normals as the alignment measure. Normals are calculated for each point in a point cloud by using a number of nearest-neighbor points within a specified radius. The space containing each point cloud is voxelized and the mean of the normals value is calculated for each voxel. By comparing the mean normals of the corresponding voxels in two point clouds, a consistency value can be computed. Well-aligned point clouds should have a small mean difference between corresponding voxel normals.

Surface interpenetration measure

Reference 7 proposes a surface interpenetration measure, based on the observation that well-aligned point clouds should present a "splotchy" surface, where coinciding surfaces from the point clouds cross over each other repeatedly. The alignment measure can be acquired by identifying and counting "interpenetrating" points between two point clouds, i.e., points where surfaces are intersecting. Well-aligned point clouds should have a high number of interpenetrating points.

3 | POINT CLOUD ALIGNMENT CLASSIFICATION

Point cloud alignment classification is the method of classifying the alignment of point clouds into two or more "classes." The aim of the classification is to detect whether the point clouds are correctly aligned (within some tolerance margin) or not. The scope of our evaluation of point clouds is to make a binary classification of each point cloud pair as "aligned" or "not aligned." It would also be possible to treat the problem as a continuous one where a value could be used to describe the degree of misalignment, however since we evaluate several different classifiers with different objective functions it could be difficult to find a common measure that applies to all classifiers. To do binary classification, we wish to find a quantitative measure of "alignedness" that we can threshold. Each classifier has been designed to provide such a measure. By using AdaBoost we find a suitable threshold for each classifier on a subset of the available data. The threshold is then used to evaluate the remaining part of the dataset. It is in general not possible to find a single threshold suitable for all classifications for any classifier, but by training on relevant datasets we can find thresholds that provide a suitable tradeoff between classification precision and recall.

In this section we will describe, in detail, all the alignment measures that we will be evaluating, and the parameters we select for each classifier.

3.1 | RMS distance classifier

The neighbor threshold distance, explained in Section 2, has a major impact on the behavior of the RMS-distance classifier, and also affects the magnitude of the errors that it can classify. A too small threshold might remove misaligned but overlapping points (which should not be considered outliers) leading to an overconfident error measure. A too large threshold might include those points that are truly outliers, i.e., points in nonoverlapping parts of the point clouds, and consequently classify well-aligned point cloud pairs as not aligned. We have included a set of six thresholds, chosen to range from fine to coarse errors (in the context of our datasets), and we have also included an RMS classifier with a statistical threshold suggested by Ref. 16. The statistical threshold is defined as 2.5 standard deviations from the mean RMS-distance of all points.

The RMS-classifiers and their respective thresholds are specified in Table 1.

3.2 | NDT score classifier

Essentially, a 3D-NDT representation is constructed by creating a voxel grid over a point cloud, and for each voxel we compute the mean and covariance of the points in it (that is, the parameters of a Gaussian function).

Consider two point clouds $\mathcal{A}$ and $\mathcal{B}$, i.e., two unordered sets of 3D points. We compute an NDT grid $\mathcal{M}$ for the points in $\mathcal{A}$, and keep the point cloud representation of $\mathcal{B} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

Using the coordinate frame of $\mathcal{A}$, we express the likelihood of $\mathcal{B}$ being aligned with $\mathcal{A}$ with the NDT score function

$$s(\mathcal{M}, \mathcal{B}, \mathbf{p}) = -\frac{\sum_{k=1}^{n} \tilde{p}(T(\mathbf{p}, \mathbf{x}_i))}{n}, \tag{1}$$

where $\tilde{p}$ is the Gaussian of the closest voxel of $\mathbf{x}_i$ in $\mathcal{M}$, $\mathbf{p}$ is the pose of $\mathcal{B}$ in the coordinate frame of $\mathcal{A}$, T is a function that transforms point $\mathbf{x}_i$ according to $\mathbf{p}$, and n is the number of points in the point clouds.

As described by Ref. 9, $\tilde{p}$ is a Gaussian that approximates the logarithm of a linear combination of two probability functions: the normal distribution of the points of $\mathcal{A}$ that are contained in the voxel, and a constant distribution that is used for modeling outliers.

The sum $s(\mathcal{M}, \mathcal{B}, \mathbf{p})$ corresponds to the negative log-likelihood that point cloud $\mathcal{B}$ lies on the surface of point cloud $\mathcal{A}$, (Ref. 11, Chap. 6), and is used directly as the alignment quality measure.

TABLE 1 A summary of all classifiers and their abbreviations used in the evaluations

Name	Description
RMS1	Root-mean-square classifier with nearest-neighbor threshold 4 m.
RMS2	Root-mean-square classifier with nearest-neighbor threshold 2 m.
RMS3	Root-mean-square classifier with nearest-neighbor threshold 0.5 m.
RMS4	Root-mean-square classifier with nearest-neighbor threshold 0.25 m.
RMS5	Root-mean-square classifier with nearest-neighbor threshold 0.15 m.
RMS6	Root-mean-square classifier with nearest-neighbor threshold 0.05 m.
RMS7	Root-mean-square classifier with statistical nearest-neighbor threshold.
NDT1	NDT Score classifier with standard implementation.
NDT2	NDT Score classifier with removed ground floor.
NDT3	NDT Score classifier with overlap evaluation.
NDT4	NDT Score classifier with both removed ground floor and overlap evaluation.
HEST	NDT Hessian translation parameters classifier.
HESR	NDT Hessian rotation parameters classifier.
PLEX	Plane extraction classifier.
NORM	Partitioned mean normals classifier.
SIM1	Surface interpenetration measure classifier with standard implementation.
SIM2	Surface interpenetration measure classifier with overlap scaling.
ADA	AdaBoost classifier.

A good alignment should attain a large negative value. The closer to zero the NDT score is, the worse is the alignment.

A characteristic weakness of the described implementation of this classifier is that the evaluation score will be better for point clouds with a large overlap and worse for point clouds with a small overlap. We are using four different variations of the NDT Score classifier to account for such effects: the standard NDT implementation as described above and three different methods of point-cloud segmentation described below. The voxel size for the normal distribution calculation has been set to 0.5 m for all classifiers. In our experience, this voxel size provides a good tradeoff between the number of points in each voxel and the NDT representation accuracy.

3.2.1 | Remove ground floor

A common scenario is the acquisition of point clouds by a movable sensor in urban or indoor environments where the ground floor is flat. The errors that appear when trying to fit point clouds in similar environments are typically translations parallel to the ground floor and rotations around the ground floor normal since those are the major movement directions. Translations along the normal of the ground floor and rotations around any axis in parallel with the ground floor are often orders of magnitude smaller. Even with a large pose error between the two scans, the ground floor is still often well aligned (for ground vehicles). The ground points then risk outweighing the other structures in the scene that better show the pose error. Reference 17 proposed using a Gaussian process model to remove ground points before registration. For the data sets evaluated here, the ground is smooth enough so that it is reliably removed by filtering out all points with a vertical normal vector. The normal is computed using the 10 nearest neighbors. All points that have a normal in close proximity to the vertical axis, i.e., where the dot product of the normal and a vector of the same length along the vertical axis exceeds 0.9, are removed from the point cloud.

3.2.2 | Overlap evaluation

Another subsampling strategy is to only evaluate those voxels where the point clouds overlap. By taking Eq. (1) and replacing the denominator n with m, where m is the number of points from B that are in the occupied voxels of M, we get a measure that is less sensitive to the amount of overlap.

This method does have an increased risk, in comparison with the standard implementation, of classifying point cloud pairs with bad alignment as well-aligned as long as a small portion of the point clouds are aligned.

3.2.3 | Ground floor removal and overlap evaluation

Finally, we also evaluate a classifier that removes the ground floor and only evaluates the overlapping parts of the point clouds. The risk here might be that excessive subsampling leaves not enough data to make a reliable classification.

3.3 | NDT Hessian classifier

The NDT Hessian is also suitable as a binary classifier. We have divided it into two classifiers:

1. The Hessian translation classifier.
2. The Hessian rotation classifier.

The reason for this is because there is a distinct difference in magnitude between the eigenvalues related to the translation parameters and the eigenvalues related to the rotation parameters. This means that a single threshold value would be severely biased toward either the translation or rotation parameters. Both classifiers do work, however, according to the same principle, where the alignment measure is the largest of the three diagonal elements in the inverse Hessian matrix related to the translational and rotational parameters, respectively.

The Hessian is calculated as follows:

$$H_{ij} = \frac{\delta^2 s}{\delta p_i \delta p_j}, \tag{2}$$

where p_i and p_j are elements from the pose vector $\mathbf{p}$ in Eq. (1) and s is the score function in Eq. (1). The voxel size for the Hessian classifiers is the same as for the NDT score classifiers.

3.4 | Plane extraction classifier

Furthermore, we have investigated a method that is a modification of the plane extraction method of Refs. 14 and 13. Similarly to the

ALGORITHM 1 Patch clustering

Data: List of existing patches P of size l
Result: Each patch has an assigned label
Variables: L_c-label counter

```
for l_i ← 1 to l do
    if label of P[l_i] is unassigned then
        increment L_c
        set the label of P[L_i] as L_c
    end
    foreach adjacent patch P[a_i] to P[l_i] do
        if P[a_i] is on the same plane as P[l_i] then
            check which of the patch labels is smaller
            replace the higher label by the lower one
            decrement L_c
        end
    end
end
```

work of Chandran-Ramesh and Newman, this method approximates the point cloud as a set of plane patches. However, in our implementation, instead of classifying the relations between patches within a single point cloud map, we perform pairwise evaluation between two scans. Knowing which points belong to which scan makes the conditional random field used by Chandran-Ramesh and Newman superfluous, and thus we focus on scoring plane patch alignment. The approach can be split into two steps: plane extraction and scoring.

Plane extraction

This step is a modification of Ref. 18, which also is part of the pipeline of Refs. 14 and 13. The underlying idea is to split plane fitting into subproblems. Given a 3D point cloud, we split it into a regular voxel grid. Each voxel contains the points within its boundaries. Next, for each nonempty voxel we find the best fitting plane surface using least-squares. The last step is to define patch vertices by finding the points where the plane surface intersects with the voxel edges. In this way we obtain a set $\mathcal{P}$ of plane patches. Algorithm 1 shows how to cluster adjacent patches that belong to the same surface (having the same label) into sets $\Pi_{L_i} = \{P[L_i] | P \in \mathcal{P}\}$.

The next step is to perform plane reconstruction for each patch cluster Π_{L_i}. Reference 18 solves this problem by merging patches into convex polygons. This approach will cause windows, doors, etc. to disappear. To prevent this problem we have decided to replace this patch merging with an algorithm based on α-shapes.

The α-shapes were first introduced in Ref. 19. The idea behind α-shapes is to build a hull enclosing all the points in the data set. Reference 20 describes the intuition for this procedure as follows. We start with a hull enclosing all the points with some big margin. In each step we remove a chunk of hull of radius α that is not enclosing any of the points in the data set. In this method we can remove also part of the hull, even if such a chunk is isolated from the other removed ones.

First, for each Π_{L_i} we compute the best fitting plane, using the points from the voxels corresponding to the patches in Π_{L_i}. Next we fit a least-squares plane to those points. Afterwards we project the points in the affected voxels onto the obtained surface and employ α-shapes in 2D.[21]

Scoring alignment quality

After we extracted planar polygon segments in both point clouds, we measured the alignment and overlap of all segments with their nearest-neighboring segment in the other point cloud. The final score is computed based on three factors:

1. Distance—The difference in distance between the origin and center of gravity for both patches along the normal for each plane: $|\mathbf{n}_i \cdot \mathbf{cog}_i - \mathbf{n}_j \cdot \mathbf{cog}_j|$. The distance is normalized based on the size of the considered point cloud. As the furthest distance between the origin and the center of gravity of the furthest patch—D_f.
2. Orientation—The dot product of the normals of a plane pair. Since the dot product of two unit vectors is always between 0 and 1, this factor does not require normalization.
3. Size—The difference in size between two neighboring patches. This factor is normalized in respect to the biggest patch in the given data set A_f. The impact of this factor requires fine tuning. The difference in the size of the patches if addressed carelessly might lead to skewing the score. However, we should keep in mind that we are facing a real environment where dynamic changes can occur. In such circumstances we need to consider all the differences that will help to filter out information coming from such distortions.

To compute the final score, we use the same method as Refs. 14 and 13, and we employ the following equation:

$$S = \frac{1}{N} \sum_{i_1}^{N} \frac{\sqrt{(D_i^S)^2 + (A_i^S)^2} \cdot (1 + O_i^S)}{\sqrt{D_f^2 + A_f^2} \cdot 2}, \tag{3}$$

where D_i^S, A_i^S, and O_i^S are the distance, size, and orientation scores, while the normalization factors are D_f^2 and A_f^2. We normalize the score by the number of plane pairs to obtain a score between 0 and 1.

3.5 | Partitioned mean normals classifier

In our implementation of the partitioned mean normals classifier,[15] the normals are computed using the 20 nearest neighbors within a specified radius. In our experiments, we have set the neighborhood radius to 0.5 m, which is consistent with the general sample density and scale of structures in our data sets. The voxel grid used by this classifier must be static for all point clouds so that corresponding voxels covering the same space can be found between both point clouds in a point cloud pair. Point clouds that are well-aligned should receive a small absolute value from this classifier.

3.6 | Surface interpenetration measure classifier

Given two point clouds $\mathcal{A}$ and $\mathcal{B}$, the set of interpenetrating points, $C_{\mathcal{A},\mathcal{B}}$, is defined as all points $\mathbf{x}$ in $\mathcal{A}$ whose local neighborhood includes at least one pair of points that are separated by the local tangent plane computed at the closest neighbor of $\mathbf{x}$ in point cloud $\mathcal{B}$. The surface

interpenetration measure, *SIM*, is calculated as the fraction $C_{\mathcal{A},\mathcal{B}}$ of interpenetrating points in $\mathcal{A}$ according to Eq. (4).

$$\text{SIM} = \frac{|C_{\mathcal{A},\mathcal{B}}|}{\mathcal{A}} \tag{4}$$

Well-aligned point clouds should have a high rate of interpenetration points.

We have included two variants of the surface interpenetration measure classifier:

1. SIM1: This classifier is implemented from Ref. 7 without modifications.
2. SIM2: The measure in this classifier is scaled by the amount of points that have any neighbors within the specified threshold, i.e., the part of the point clouds that overlap, thus making it less sensitive to differences in the amount of overlap between different point cloud pairs.

3.7 | AdaBoost classifier

To achieve a stronger classification, we propose a combined classifier created using AdaBoost.[22] A complete explanation of this classifier will be given in Section 4.

In total we use 17 different classifiers in our evaluations. All classifiers are listed in Table 1 together with their respective abbreviation.

4 | ADAPTIVE BOOSTING OF QUALITY MEASURES

Adaptive boosting, or AdaBoost,[22] is a method for iteratively adding weak binary classifiers to build a stronger "expert" classifier. Each classifier emits a binary opinion, which, in this work, is whether two point clouds are aligned or not. A labeled training set $(x_1, y_1), \ldots, (x_m, y_m)$, where y_i is the label associated with feature x_i, is used as input to the algorithm. For each iteration we find the classifier h_t from the collection of weak classifiers H_t that improves the classification the most; that is, it maximizes the difference of the corresponding weighted error rate ϵ_t and 0.5 (which is the error rate of a random classifier). A distribution of weights, D_t, that indicates the importance of each feature is updated for each iteration round. The weights for the correctly classified features are decreased and the weights for the incorrectly classified features are increased, so the classifier focuses on the misclassified features. The procedure is repeated for T iterations, thus adding T classifiers. The set of classifiers with their respective weights α constitutes the strong classifier. The decision of the strong classifier is determined by the sign of the weighed sum of all classifiers. The algorithm is described in Algorithm 2.

Threshold selection

We use a straightforward method, adapted from Ref. 23, to train the threshold for each classifier in each evaluation round. The method is described in Algorithm 3. In short, we sort the results from the training set in ascending order and find the threshold where the number of incorrect classifications is minimized.

5 | EXPERIMENTS

In this section, we will present the procedures and the results of the evaluations.

5.1 | Environments

The applications of point clouds as spatial representations range from describing very small objects to vast landscapes. The selection of environments for this work, however, is made from the outdoor mobile robotic perspective. We have selected two environments: The first environment, the Kjula set, is situated at an asphalt manufacturing plant featuring a slightly hilly landscape with an abundance of gravel piles and some structures. The second environment, the Hannover set, is from a campus with buildings, streets, and other regular geometric features (i.e., objects with flat surfaces and/or well-defined edges). The environments have been chosen in order to validate how well the methods apply in very different environments while still being relevant in a field robotics context. Further, the sensor setup also differs where the Kjula data are acquired with a 180 by 40 degree field of view with high resolution while the Hannover data are using a spherical field of view with lower resolution. The Hannover set is heavily featured with flat surfaces and rectangular shapes while the Kjula environment features convex and concave structures with rugged surfaces. We have assumed that the environments are static (however there are some moving persons in one of the sets, but the vast majority of the environment is static).

5.1.1 | Kjula

The Kjula data, seen in Figure 3, were acquired at an asphalt manufacturing plant near a village called Kjula just outside the city of Eskilstuna in Sweden. The asphalt plant is a combined gravel pit and production unit for asphalt. The work at the plant is performed by construction equipment such as wheel loaders, dumpers, and excavators. The scans were acquired with a midsize Volvo wheel loader equipped with an actuated SICK LMS291 laser scanner. Figure 4 shows the wheel loader in the Kjula asphalt plant.

The horizontal resolution of the scanner was set to 1 degree and the vertical field of view was −40 degrees to +10 degrees, with 0 degrees being the horizontal line. The range of the scanner setup in the asphalt plant environment was approximately 30–50 m depending on the reflectivity of the surface. Each point cloud was acquired while standing still during the entire scanner sweep. Each point cloud contains between 60 000 and 120 000 readings.

We have used 65 scans from the Kjula plant to generate 132 point cloud pairs, where 64 are well aligned and 64 are not well aligned. These point cloud pairs have been used to generate a number of data sets, all containing the same 132 point cloud pairs, but each with a different set of error offsets.

ALGORITHM 2 AdaBoost algorithm

Input: $(x_1, y_1), \ldots, (x_m, y_m)$
Output: A strong classifier $H(x)$
Initialize $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$
for $t = 1, ..., T$ **do**
1. find the weak classifier h_t that maximizes $|0.5 - \epsilon_t|$ with respect to D_t
2. choose $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
3. update $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$ where Z_t is the normalization factor: $\sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i))$

end
return *The final classifier* $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$

ALGORITHM 3 Parameter search algorithm

Input: Classifier results F_i for a classifier from m training iterations and corresponding labels y_i and weights $D_i, i = 1, \ldots, m$
Output: The optimal parameter value θ_t
for $i = 1, ..., m$ **do**
1. Sort F_i in ascending order
2. For each F_i, assume classifier outputs with lower F to be plausible and classifier outputs with higher F to be suspicious.
3. Compute the weighed error ϵ_{param} as the sum of weights for all labels mismatching the classification in 2
4. Find the optimal parameter value θ_t as the classifier output associated with the lowest ϵ_{param}.

end
return θ_t

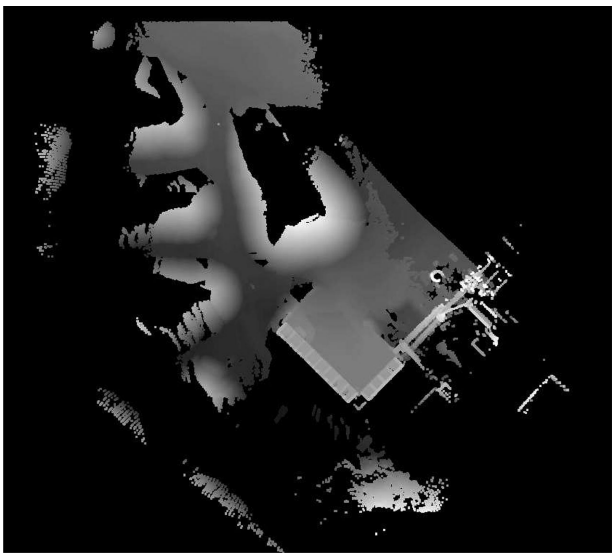

FIGURE 3 Overhead view of the Kjula data set. The set covers an area of approximately 150 by 150 m

5.1.2 | Hannover

The Hannover set is a publicly available set of point clouds* acquired at the university of Hannover (see Fig. 5). It is acquired with a moving platform equipped with two continuously rotating SICK scanners mounted back to back and odometry sensors. Each point cloud features one half revolution of the scanners (i.e., 360 degrees of readings) and the points are compensated for the platform's movement by using the odometry. The speed of the platform is low enough in relation to the accuracy of the odometry to not cause any significant movement errors. The density of these point clouds is much less than the Kjula set with only 4000 to 10 000 points in each point cloud. We have used 801 scans to create 1600 point cloud pairs where 800 are well aligned and 800 are not well aligned in the same manner as for the Kjula set.

5.1.3 | Ground truth datasets

One crucial aspect of evaluating point cloud alignment is how the aligned ground truth data are created. We used registration methods to align point clouds to each other, and afterward we made visual confirmations of the results. There are of course more sophisticated methods of acquiring ground truth data, such as using GPS or camera based motion capture (e.g., Vicon), however Vicon is not feasible in large environments since the volume must be in full view of the cameras. GPS might have been beneficial for the Kjula set, but it would probably be challenging to use in the Hannover set due to the possibility of multiple reflections when driving close to walls.

When verifying the registrations there can be variations in alignment that are not obvious upon visual inspection. Therefore, we have compared two common registration methods, NDT registration and ICP registration, since some of the classifiers explicitly use the objective function of NDT and some use the objective function of ICP (the RMS error). When a specific registration method is used to produce the

* http://kos.informatik.uni-osnabrueck.de/3Dscans/

FIGURE 4 The Volvo L120F wheel loader at the asphalt plant in Kjula. The laser scanner (see the inset) is mounted on top of the cabin underneath the gray protection shield

FIGURE 5 Overhead view of the Hannover data set.[24] The data set covers an area of approximately 180 by 200 m

ground truth data sets, it might be the case that this introduces a bias toward certain classifiers.

To quantify this bias, we have created two ground truth datasets each, for the Hannover and Kjula point clouds, respectively: one using NDT registration and one using ICP registration. We have made evaluations with both to see how the differences affect the performance of the classifiers. The failure rate (identified by visual inspection) of the registration methods was approximately 20% for the Kjula point cloud pairs and 1.4% for the Hannover pairs. In those cases we have used the registration of the nonfailing method as a replacement. Luckily both methods never failed for the same point cloud pair. The environment in Kjula is much less structured than the Hannover campus, which explains the higher number of failed registrations.

The magnitudes of the difference between the resulting transformations after applying the two registration algorithms have been calculated. The box plots in Figures 6, 7, and 8 show the differences. The resulting transformations for most of the point clouds differ by less than 0.05 m in translation, and the rotations differ by less than 0.032

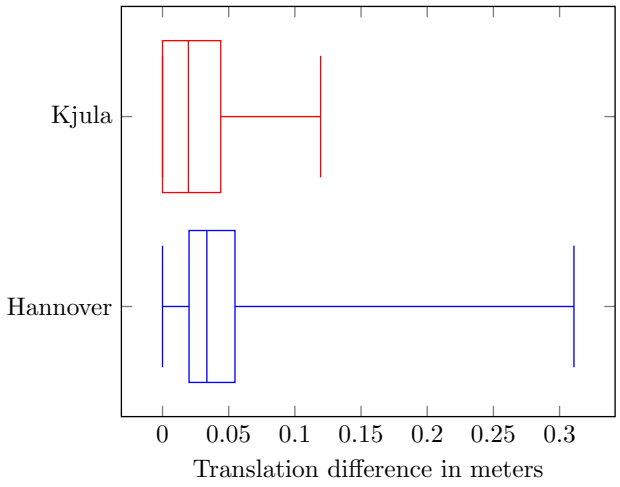

FIGURE 6 The box plots show the distribution of the translation differences, comparing the transformations given by NDT and ICP registration on the Hannover and Kjula data set, respectively

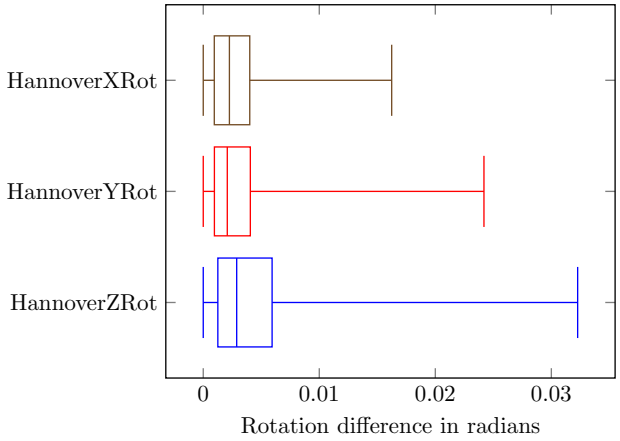

FIGURE 7 The box plots show the distribution of the rotation differences, comparing the transformations given by NDT and ICP registration on the Hannover data set

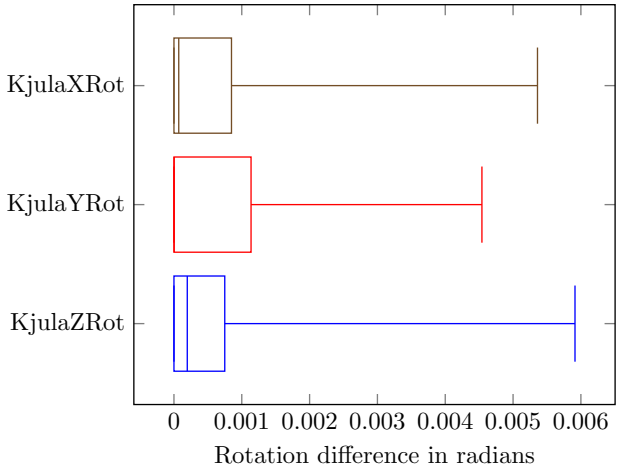

FIGURE 8 The box plots show the distribution of the rotation differences, comparing the transformations given by NDT and ICP registration on the Kjula data set

radians for all point clouds. However, 25% of the point cloud pairs in the Hannover set differ by 5–30 cm, which could introduce a bias toward some classifiers, but our evaluations in Section 5.3.2 will show that this is not the case. Errors of up to 30 cm would be clearly visible in most cases, but the ones encountered here are in such areas that the correct alignment is not obvious, for example in corridor-like areas.

5.1.4 | Induced errors

To generate datasets with both correctly aligned point cloud pairs and misaligned pairs, which is needed for training of the classifiers, we have manually added errors. To cover a wide set of scenarios, we have created datasets with several error types and magnitudes.

The added errors are made up of two components, a translational part and a rotational part. One of the point clouds in each pair has been translated and rotated to put it out of alignment with the other point cloud.

Further, we have defined three different error magnitudes with the aim of creating three distinct difficulties. The translational part is defined as a static distance with a random direction, meaning that the magnitude of the error will always be the same but the direction in which the error is applied will be random. The rotational part of the added errors is defined as a rotation with static magnitude but random direction (clockwise or counterclockwise). Each point cloud with an added error has first been translated followed by a rotation.

- Small errors: These errors are approximately twice the magnitude of the upper quartile of the difference we measured between ICP and NDT registration in Section 5.1.3. This magnitude is selected to be (just) distinguishable from noise occurring in point cloud registration methods. The error is only applied in three degrees of freedom to simulate how odometry errors often behave. The small translational error is 0.1 m in the X-Y plane. The small rotational error is 0.01 radians (0.57 degrees) around the vertical axis. These errors are most difficult to detect. Figure 9 shows a point cloud with a small error.
- Medium errors: These errors are chosen to be less of a challenge for the classifiers. They are also limited to three degrees of freedom but with a bigger magnitude. The medium translational error is 0.3 m in the X-Y plane. The medium rotational error is 0.03 radians (1.72 degrees) around the vertical axis.
- Large errors: These errors are chosen to be easy to detect for any reasonably good evaluation method. The errors are, in contrast to the small and medium errors, applied in six degrees of freedom and the magnitude is even larger. The large translational error is 0.5 m. The large rotational error is 0.05 radians (2.86 degrees).

By using manually verified datasets to create the datasets with errors, we also get a correct labeling, i.e., whether the point cloud pair is aligned or not, of all point cloud pairs. A correct labeling is a requirement for both training and evaluation of the classifiers.

5.2 | Evaluation methods

We denote a point cloud pair that is classified as aligned as a *positive* result of the classifier, and a point cloud pair that is classified as

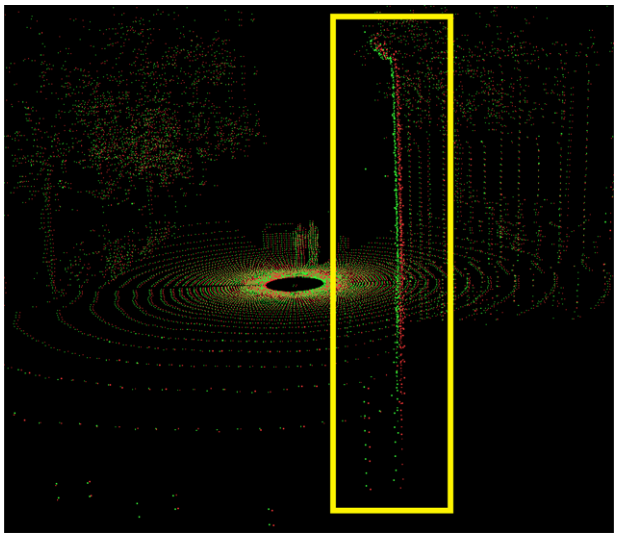

FIGURE 9 Two point clouds, one in red and one in green, from the Hannover data set that are misaligned with a small 10 cm error. The two vertical lines in the square marking in the figure are both forming the same wall but are clearly not aligned with each other

unaligned as a *negative* result. A true positive (*tp*) is a point cloud pair that is detected as aligned, and is indeed successfully aligned, according to our manually labeled ground truth. A false positive (*fp*) is a point cloud pair that is detected as aligned, but that is, in fact, not properly aligned. True and false negative (*tn*, *fn*) classifications are symmetrically defined.

The primary evaluation metrics used in this article is accuracy, formalized as follows:

$$A = \frac{tp + tn}{tp + fp + tn + fn}. \tag{5}$$

The accuracy measures the overall quality of a binary classifier, regarding both true positives and true negatives. The accuracy is also the measure that AdaBoost seeks to maximize. The range of the accuracy goes from 0 to 1, where 0 means that all evaluations are false and 1 means that all evaluations are true. Random classifications should end up with an accuracy of 0.5.

We also investigate the sensitivity to threshold selection for each classifier. We use receiver operator characteristic (ROC) plots to show how the performance of each classifier is affected by changes to the threshold. The step size for all single classifiers has been chosen by making a linear distribution of 100 steps between the maximum and minimum result from the classifier in the associated evaluation round. The boosted classifier is a special case where the returned value is a mixture of several classifiers that varies for different datasets. To produce the ROC-plot for the boosted classifier, we use Eq. (6), where α and h_t are the classifier weight and corresponding classifier, respectively, as introduced in Section 4, to normalize the output and then change the threshold between −1 and 1 with increments of 0.02, thus creating 100 steps,

$$\frac{\sum_{t=1}^{T} \alpha_t h_t(x)}{\sum_{t=1}^{T} \alpha_t}. \tag{6}$$

TABLE 2 The error types used in the evaluations

Error name	Error description
small	Small translational and rotational errors
medium	Medium translational and rotational errors
large	Large translational and rotational errors
varying	Evenly distributed small, medium, and large combined errors

5.3 | Evaluations

5.3.1 | Classifier performances

We have evaluated the performance of the classifiers using *k*-fold cross validation.

A requirement for all weak classifiers used in AdaBoost is that they perform better than random. Therefore, we remove from the boosting all classifiers whose performance is inside a 95% confidence interval of random accuracy. The outcome of a classifier is considered to be a random variable following a binomial distribution with *n* samples, where *n* is the number of point cloud pairs. Using the normal approximation, we can use the following equation:

$$I_p = p_{obs} \pm 1.96 * \sqrt{p_{obs}(1 - p_{obs})/n}. \tag{7}$$

With the observed probability $p_{obs} = 0.5$, the 95% confidence interval I_p is 0.4755–0.5245 for the Hannover set and 0.4147–0.5853 for the Kjula set. We use these numbers to disqualify weak classifiers from use by the AdaBoost classifier to avoid random classifiers.

To evaluate the performance of the classifiers, we have been running cross-validation for all classifiers on the ten datasets (five error distributions for each of the two environments). The error types are described in Table 2. On the Hannover-sets, consisting of 1600 point cloud pairs, we made an eightfold cross validation, i.e., training on 1400 pairs and evaluating on 200 pairs eight times. On the Kjula-sets, consisting of 132 point cloud pairs, we made a threefold cross validation, i.e., training on 88 pairs and evaluating on 44 pairs three times. In the case of single classifier evaluations, the training phase is used to find the classifier threshold as described in Section 4.

The performance of a classifier is determined by its accuracy. An accuracy of 1 means that the classifier can properly classify all point cloud pairs as aligned or not aligned. An accuracy of 0.5 means that the outcome of the classifier is random. We expect to see the best performance on the datasets with large errors and then declining performance as the errors become smaller.

Hannover results

Figures 10 and 11 shows the classifiers' accuracy on the Hannover datasets for all error types.

In Figure 10 we see that most classifiers obtain more than 90% accuracy for large errors, which can be considered as a good result. The NORM classifier manages well above a random result for large errors but fails to classify the point cloud pairs with medium and small errors, indicated by the accuracy of 50%, which implies a random classification result. When dealing with smaller error magnitudes, the

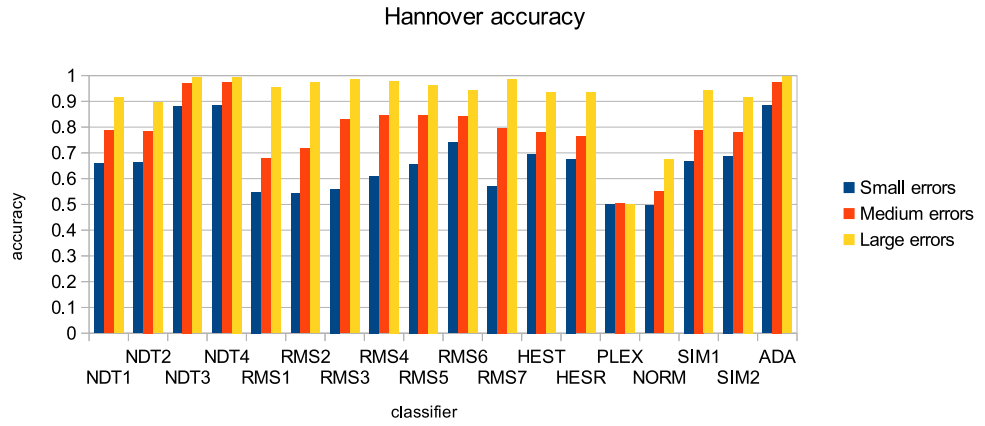

FIGURE 10 Classifier accuracy on the Hannover data set where the erroneous point cloud alignments consist of small, medium, and large errors, respectively

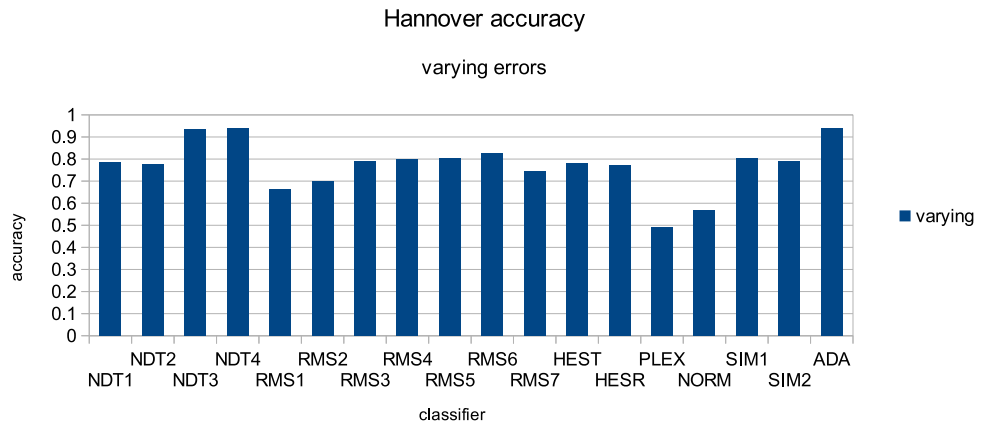

FIGURE 11 Classifier accuracy on the Hannover data set where the erroneous point cloud alignments consist of varying errors

difference between the mean normals is smaller and more sensitive to noise and irregularities in the point clouds from for example rough surfaces, which could explain the poor performance for the NORM classifier on small error magnitudes. The PLEX classifier fails to classify the point cloud pairs completely, which could imply that this is, in the implementation we have used, an unsuitable classifier for this type of problem. The NDT3 and NDT4 classifiers have the highest accuracy for all error magnitudes together with the ADA classifier. It is interesting how the RMS-classifiers perform differently, i.e., the best RMS-classifier for large errors is not the best RMS-classifier when the error magnitude changes. This indicates that it is important to select a suitable threshold for RMS-classifiers.

If we look at the dataset with errors of all three magnitudes, shown in Figure 11, we observe that the classifiers' performance is roughly the mean value of the accuracy obtained over all error magnitudes in Figure 10. This result suggests that the thresholds acquired during the training phase of the classification process are relatively stable for all classifiers even though error magnitudes vary.

Kjula results

Figures 12 and 13 show the classifiers' accuracy on the Kjula dataset for all error types.

The results from the Kjula set in Figure 12 show a more spread out result than the corresponding results in the Hannover set, indicating the higher difficulty of the less structured Kjula environment. There are still some classifiers that perform above 80% even on the small error magnitudes, but there are more classifiers in the Kjula set that perform closer to random than in the Hannover set.

It is worth noting that among the large error evaluations, the boosted classifier (ADA) performed slightly worse than the NDT3 classifier. Our analysis suggests that this is due to overfitting. Even if AdaBoost is rather robust against overfitting, the Kjula data sets contain only 64 positive and 64 negative point cloud pairs. We have observed that the NDT3 and NDT4 classifiers perform equally well on the training data, but NDT3 performs better on the evaluation data. The boosted classifier, as a consequence, uses a suboptimal combination of the "weak" classifiers in this case. The effect of this on the AdaBoost classifier can be seen in both Figures 12 and 13, as it performs worse than NDT3 but better than or equal to NDT4. This result emphasizes the risks of small training sets and also shows that AdaBoost is not immune to overfitting.

We can also observe the same trend among the RMS-classifiers as in the Hannover evaluations. The best RMS classifier for large errors is not the best classifier for small errors, showing even further the importance of a suitable threshold for RMS-classifiers and that the RMS classification method is sensitive to varying error magnitudes.

We further notice the same behavior as in the Hannover evaluations when evaluating data with varying error magnitudes. The result

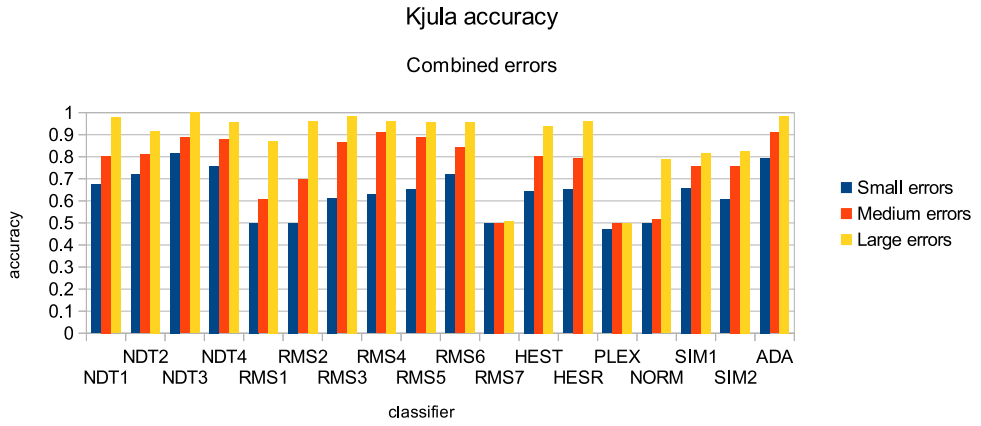

FIGURE 12 Classifier accuracy on the Kjula dataset where the erroneous point cloud alignments consist of small, medium, and large errors

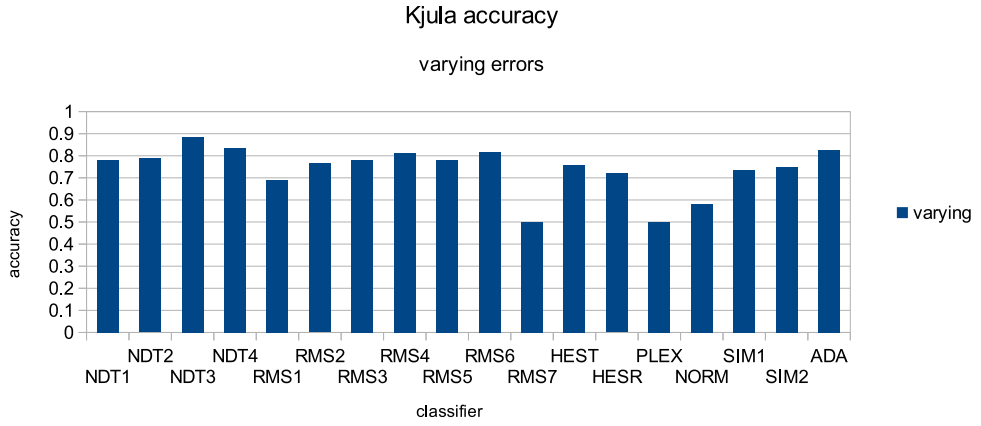

FIGURE 13 Classifier accuracy on the Kjula dataset where the erroneous point cloud alignments consist of varying errors

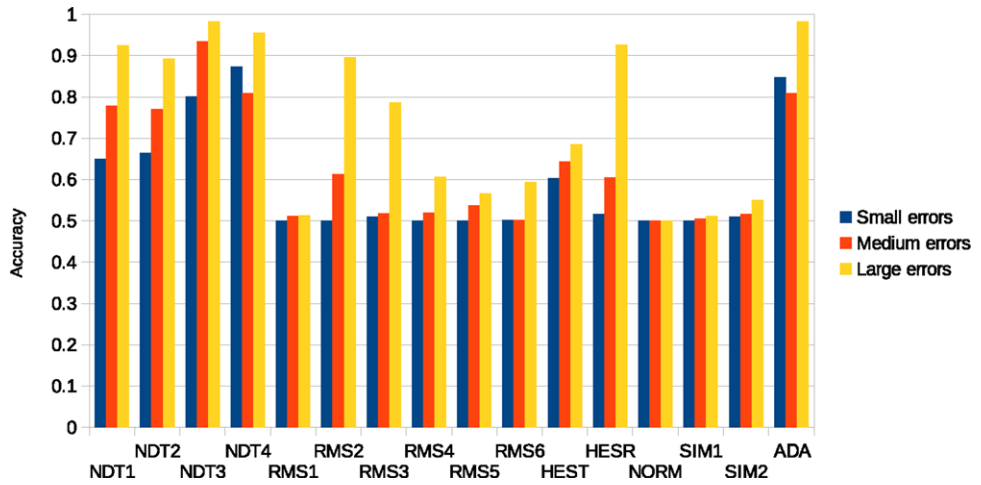

FIGURE 14 Accuracy of classifiers trained on datasets from the Kjula environment and evaluated on datasets from the Hannover environment

for each classifier is close to the mean value of the accuracy for small, medium, and large errors.

Cross-dataset results

In this section we have evaluated classifiers that have been trained on datasets from one of the two environments and evaluated on the other. These evaluations are performed to investigate the classifiers' sensitivity to the likeness between the training data and the evaluation data.

Figures 14 and 15 show the accuracy for small, medium, and large errors. In comparison to the evaluations with training and evaluation data from the same environment, we see that many of the classifiers produce random or close to random results. Some of the RMS classifiers perform well above random on large errors, but none perform

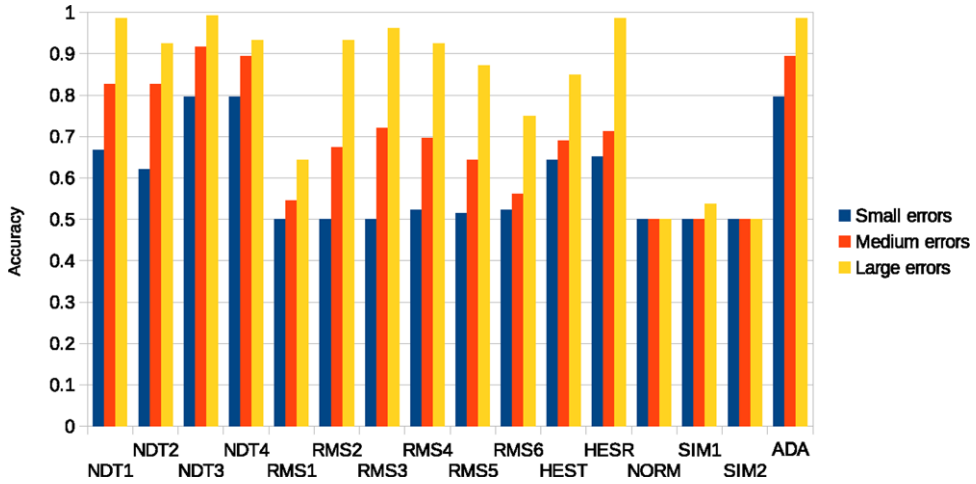

FIGURE 15 Accuracy of classifiers trained on datasets from the Hannover environment and evaluated on datasets from the Kjula environment

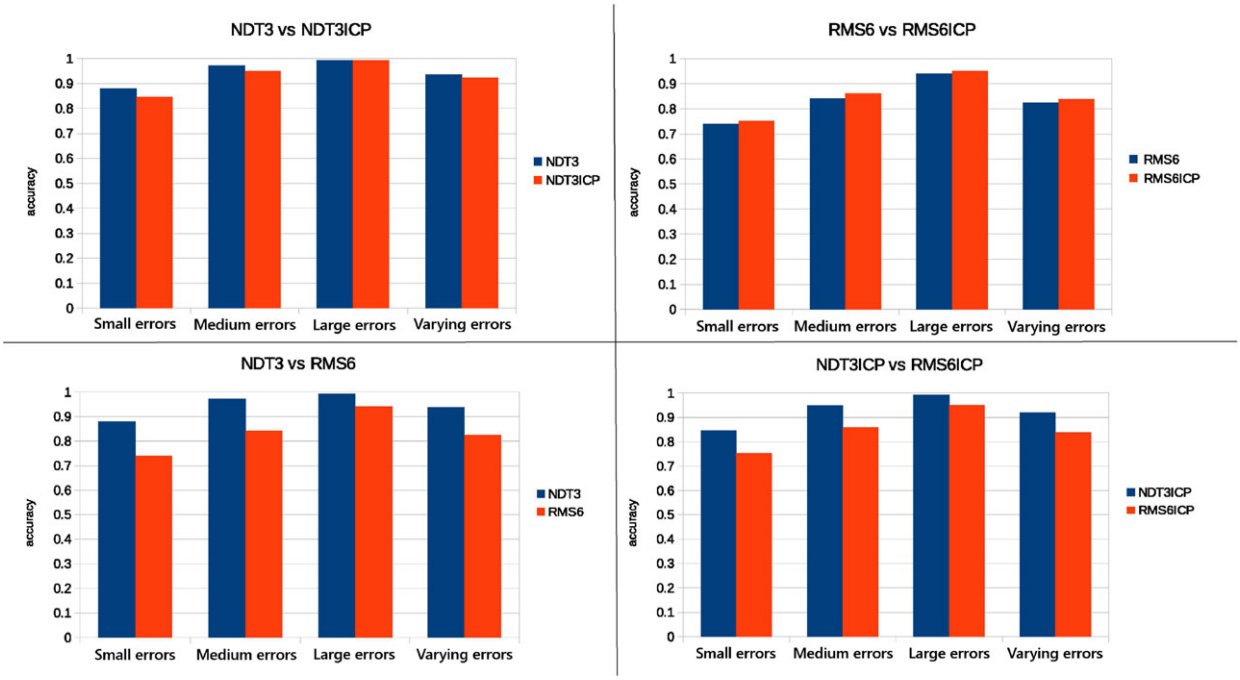

FIGURE 16 Diagrams explaining the effects on accuracy for classifiers when using ground truth created with different methods. The top diagram compares the change in accuracy for the classifiers NDT3 and RMS6 when evaluating data with NDT-based ground truth (blue/dark) and ICP-based ground truth (red/light). The bottom diagram shows the effect of the method used to create the ground truth when the classifiers are compared to each other with NDT-based ground truth to the left and ICP-based ground truth to the right

better than random on small errors. The NDT classifiers (NDT1–4) all perform better than random. NDT3, using only overlapping points, and including the ground plane, is the best classifier on large and medium errors, and NDT4 is the best classifier on small errors. These results suggests that NDT-based classifiers are the least sensitive classifiers when the environment changes, and that the other classifiers are much more sensitive to differences between the training data and the evaluation data.

These results show that the above-mentioned classifiers generalize rather well between two datasets that have different geometric characteristics and different point-cloud resolutions. However, the results might be different for denser point clouds (e.g., from land survey equipment such as Faro Focus) or ones with a restricted field of view (e.g., from RGB-D sensors).

5.3.2 | ICP vs NDT evaluations

In Section 5.1.3 we discussed the creation of ground truth datasets and the possible problems with using a certain registration algorithm to create such datasets. In this section, we compare evaluations made where the ground truth datasets are created using either NDT registration or ICP registration.

Figure 16 shows the difference in accuracy for the best performing NDT and RMS classifiers when evaluated on the Hannover set with

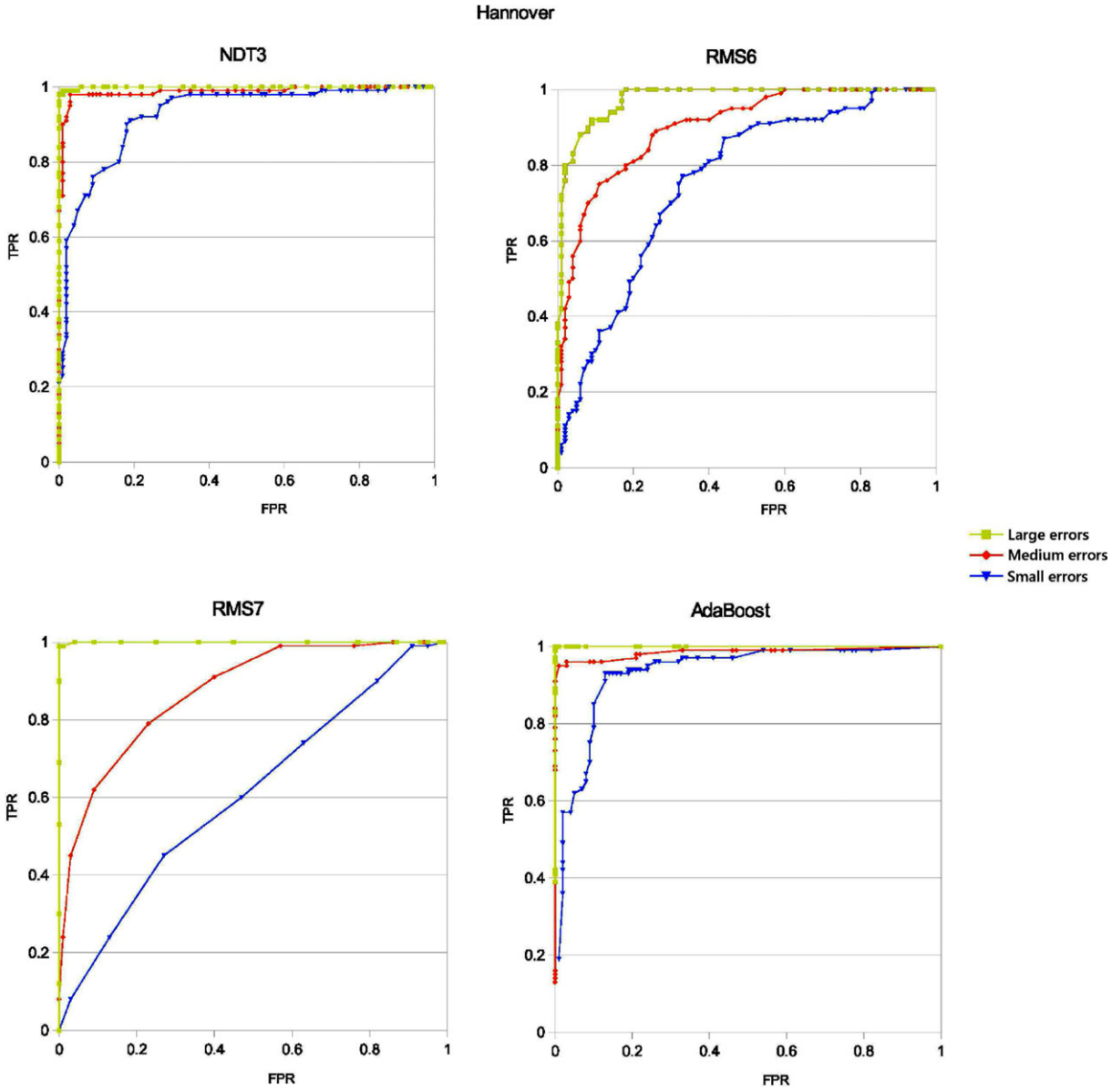

FIGURE 17 ROC-plots of small, medium, and large errors from the Hannover dataset for the NDT3, RMS6, RMS7, and AdaBoost classifiers. The concentration of points along the curves shows the sensitivity for threshold selection for each classifier. A high concentration of points close to the point 0,1 in each plot shows that the classifier is not sensitive to threshold selection, while a sparse concentration of points at 0,1 shows that the classifier is sensitive to threshold selection

ground truth data created using NDT registration and ICP registration. The upper left diagram, NDT3 vs NDT3ICP, shows the accuracy for the NDT3 classifier on the Hannover set with NDT-based ground truth (blue/dark) and with ICP-based ground truth (red/light). The accuracy is, as expected, highest for the evaluations made with NDT-based ground truth. The upper right diagram shows in the same way the accuracy for the RMS6 classifier on the same Hannover sets with NDT-based ground truth and ICP-based ground truth. In this case, the accuracy is better on the dataset with ICP-based ground truth. The differences explained in the two upper diagrams show that the method used to create the ground truth data does have some effect on the outcome of the evaluations. However, the effect of the method used to create the ground truth is not significant enough in our experiments to alter the outcome of the evaluations. This is shown in the lower two diagrams in Figure 16 where the two classifiers NDT3 (blue/dark) and RMS6 (red/light) are compared to each other with NDT-based ground truth to the left and ICP-based ground truth to the right. The diagrams are nearly identical since the differences in accuracy for the classifiers are much larger than the effect of the method used to create the ground truth. It might be important, however, to take this effect into account in a scenario where the difference in accuracy between classifiers is very small.

5.3.3 | Classifier threshold sensitivity

The threshold sensitivity evaluation is performed by inspecting ROC plots of the classifiers performance when varying the threshold as described in Section 5.3.1. It is preferable to have a low sensitivity to threshold selection, as this increases the robustness of the classifier. A

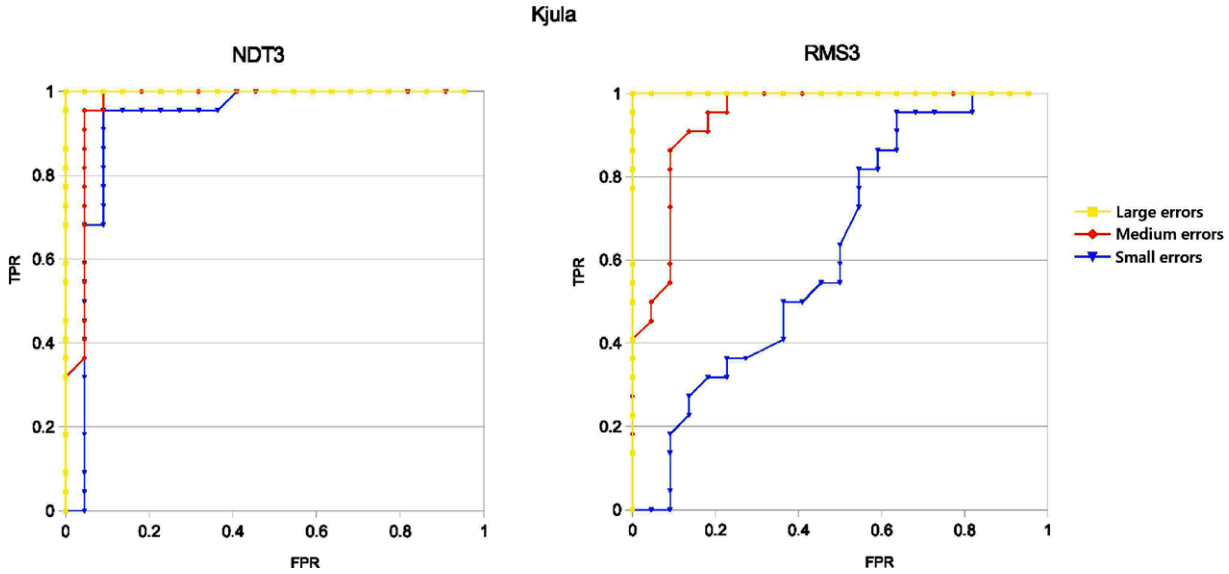

FIGURE 18 Sample ROC-plots of small, medium, and large errors from the Kjula dataset. Both classifiers show an even spread of samples along the curves

low sensitivity is displayed as a high concentration of samples (dots) in the upper left corner of the ROC plot, where the true positive rate is high and the false positive rate is low.

Hannover dataset

In Figure 17 we show an example of ROC plots for the classifiers that showed the best performance on the Hannover dataset. Each plot shows ROC curves for each of the three difficulties on the combined dataset. The best performance in terms of low sensitivity is achieved by the AdaBoost classifier in the bottom right corner where a large concentration of samples is found in the upper left corner of the diagram, suggesting that a wide range of thresholds will result in a high performance. The bottom left RMS7 classifier, on the other hand, shows only a few samples along the curves, and the highest concentration is found at the end points (close to 0,0 and 1,1). This suggests that the range of acceptable thresholds is small. The other two classifiers, NDT3 (top left) and RMS6 (top right), show an even spread of samples along the curves, showing that the sensitivity of these classifiers is between the RMS7 and AdaBoost classifiers.

Kjula dataset

In Figure 18 we see examples of ROC-plots for the Kjula set. The number of samples in the plots is lower because of the lower number of point clouds in the entire data set. Both classifiers (NDT3 to the left and RMS3 to the right) show an even distribution of samples along the curves, suggesting that the classifiers have a similar threshold sensitivity, which was also the case for the NDT3 and RMS6 classifiers in Figure 17.

5.4 | Conclusions

We can conclude from the data that NDT with overlap evaluation (and sometimes removing the ground floor), i.e., NDT3 and NDT4, are better than any of the other single classifiers (including several variants of the commonly used RMS measure as well as other proposed classifiers from the literature) for classifying data from both structured and unstructured outdoor environments.

In particular, the NDT score classifiers have been shown to be substantially more robust to changing environments than methods using RMS classifiers or SIM classifiers. Even when trained on a small sample from an unstructured outdoor work site and evaluated in an urban campus environment, the NDT classifiers alone achieve over 80% accuracy for the most difficult set of scan pairs. The NORM classifier seems to be more suitable for flat surfaces without irregularities and noise since its performance dropped significantly when the error magnitudes were smaller.

An AdaBoost classifier built by combining all these classifiers performs on par with the best single classifiers (i.e., the ones using NDT), suggesting that the classifiers do not complement each other in such a way that improved performance can be achieved. However, the ROC-plot evaluations suggest that AdaBoost might be less sensitive to threshold selection than any single classifier.

6 | SUMMARY AND FUTURE WORK

The present work is, to the best of our knowledge, the first comparative evaluation of geometric consistency methods for classifying aligned and nonaligned point cloud pairs. Such classification methods can be used, e.g., for detecting point cloud registration failures in applications of model reconstruction or localization and mapping. In particular, we have evaluated several variants of common classifiers in two environments pertinent to mobile robots: an urban campus environment and an unstructured work site.

We hope that the results presented here can also be used as a baseline for further research on automatically classifying point cloud alignedness. In particular, we have identified two outstanding issues that deserve further attention.

The first is to investigate existing classifiers more deeply in order to better isolate the specific cases in which they perform better or worse,

and to learn and evaluate optimal parameter selections for different environment types and also investigate other aspects not considered in this article, such as computational time of the different classifiers. Depending on the application, it might be an advantage to sacrifice some accuracy in favor of faster computational times. In this work, we have tried to make a broad survey covering many methods. In future work, we hope that more specific classifiers will be investigated in-depth to provide a good foundation for our second suggested direction.

The second direction is to perform more evaluations on other datasets, common for other applications, as well as to investigate further the impact of variations in the point clouds on classifiers to evaluate classifier robustness. One such variation, for example, could be different point cloud densities, where the results achieved on sparse point clouds, such as the ones evaluated in this article, might not be reproducible on dense point clouds, i.e., point clouds consisting of millions of points. A special category of errors that we have left out of this article is the output of scan registration failures (as opposed to induced error offsets with fixed magnitudes), which happens when a registration method converges to a local optimum instead of the best relative pose. To create such datasets and evaluate them would also be a useful contribution.

In this work, we used AdaBoost to create a strong classifier. The results are promising, but further research must be conducted in this area. We also know that there are other methods that might prove to be good candidates. We do encourage a deeper investigation into combinations of several classifiers, both using AdaBoost and with other methods.

ORCID

Håkan Almqvist http://orcid.org/0000-0001-5007-548X
Martin Magnusson http://orcid.org/0000-0001-8658-2985
Tomasz P. Kucner http://orcid.org/0000-0002-9503-0602
Achim J. Lilienthal http://orcid.org/0000-0003-0217-9326

REFERENCES

1. Birk A. A quantitative assessment of structural errors in grid maps. *Auton Robot*. 2010;28(2):187–196.
2. Schwertfeger S, Birk A. Evaluation of map quality by matching and scoring high-level, topological map structures. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, Germany: 2013:2221–2226.
3. Magnusson M, Vaskevicius N, Stoyanov T, et al. Beyond points: Evaluating recent 3D scan-matching algorithms. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2015.
4. Campbell D, Whitty M. Metric-based detection of robot kidnapping with an {SVM} classifier. *Robotic Auton Syst*. 2014;69:40–51.
5. Rusinkiewicz SM. Efficient variants of the ICP algorithm. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*. IEEE; 2001:145–152.
6. Besl PJ, McKay ND. A method for registration of 3-D shapes. *IEEE T Pattern Anal Mach Intell*. 1992;14(2):239–256.
7. Silva L, Bellon OR, Boyer KL. Precision range image registration using a robust surface interpenetration measure and enhanced genetic algorithms. *IEEE T Robotic*. 2005;27(5):762–776.
8. Biber P, Straßer W. The normal distributions transform: A new approach to laser scan matching. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas: 2003:2743–2748.
9. Biber P, Fleck S, Straßer W. A probabilistic framework for robust and accurate matching of point clouds. In *26th Pattern Recognition Symposium (DAGM 04)*. Springer-verlag; 2004.
10. Stoyanov T, Magnusson M, Lilienthal AJ. Comparative evaluation of the consistency of three-dimensional spatial representations used in autonomous robot navigation. *J Field Robotics*. 2013;30(2):216–236.
11. Magnusson M. *The Three-Dimensional Normal-Distributions Transform—An Efficient Representation for Registration, Surface Analysis, and Loop Detection*. Ph.D. thesis, Örebro University, Sweden: Örebro Studies in Technology; 2009:36.
12. Stoyanov T, Magnusson M, Lilienthal AJ. Fast and accurate scan registration through minimization of the distance between compact 3d ndt representations. *Int J Robotic Res*. 2012;31(12):1377–1393.
13. Chandran-Ramesh M, Newman P. Assessing map quality using conditional random fields. In *Proceedings of the International Conference on Field and Service Robotics*. Springer Tracts in Advanced Robotics: Springer, 2007b.
14. Chandran-Ramesh M, Newman P. Assessing map quality and error causations using conditional random fields. In *Proceedings of the 6th IFAC Symposium Intelligent Autonomous Vehicles (IAV)*. Toulouse, France: 2007a.
15. Makadia A, Patterson A, Daniilidis K. Fully automatic registration of 3D point clouds. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2006.
16. Masuda T, Sakaue K, Yokoya N. Registration and integration of multiple range images for 3-d model construction. In *Proccedings of the 13th International Conference on Pattern Recognition*. IEEE; 1996:879–883.
17. Das A, Servos J, Waslander SL. 3d scan registration using the normal distributions transform with ground segmentation and point cloud clustering. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*. 2013:2207–2212.
18. Weingarten J, Gruener G, Siegwart R. A fast and robust 3D feature extraction algorithm for structured environment reconstruction. In *Proceedings of the International Conference on Advanced Robotics*. IEEE; 2003.
19. Edelsbrunner H, Kirkpatrick DG, Seidel R. On the shape of a set. *T Inform Theory*. 1978;26(5):552–564.
20. Edelsbrunner H, Mücke EP. Three-dimensional alpha shapes. *ACM T Graphics (TOG)*. 1994;13(1):43–72.
21. Bernardini F, Bajaj CL. Sampling and reconstructing manifolds using alpha-shapes. 1997. http://docs.lib.purdue.edu/cstech/1350/
22. Freund Y, Schapire R. A decision-theoretic generalization of online learning and an application to boosting. In *Proceedings of the European Conference on Computational Learning Theory*. Springer-Verlag; 1995.
23. Granström K, Schön TB, Nieto JI, et al. Learning to close loops from range data. *J Robotic Res*. 2011;1728–1754.
24. Magnusson M, Andreasson H, Nüchter A, et al. Automatic appearance-based loop detection from 3D laser data using the normal distributions transform. *J Field Robotic*. 2009;26(11–12):892–914.

How to cite this article: Almqvist H, Magnusson M, Kucner TP, Lilienthal AJ. Learning to detect misaligned point clouds. *J Field Robotics*. 2018;35:662–677. https://doi.org/10.1002/rob.21768