The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Learning to Close the Loop from 3D Point Clouds

Karl Granström and Thomas B. Schön

***Abstract*— This paper presents a new solution to the loop closing problem for 3D point clouds. Loop closing is the problem of detecting the return to a previously visited location, and constitutes an important part of the solution to the Simultaneous Localisation and Mapping (SLAM) problem. It is important to achieve a low level of false alarms, since closing a false loop can have disastrous effects in a SLAM algorithm. In this work, the point clouds are described using features, which efficiently reduces the dimension of the data by a factor of 300 or more. The machine learning algorithm AdaBoost is used to learn a classifier from the features. All features are invariant to rotation, resulting in a classifier that is invariant to rotation. The presented method does neither rely on the discretisation of 3D space, nor on the extraction of lines, corners or planes. The classifier is extensively evaluated on publicly available outdoor and indoor data, and is shown to be able to robustly and accurately determine whether a pair of point clouds is from the same location or not. Experiments show detection rates of 63% for outdoor and 53% for indoor data at a false alarm rate of 0%. Furthermore, the classifier is shown to generalise well when trained on outdoor data and tested on indoor data in a SLAM experiment.**

I. Introduction

Over the past two decades, the Simultaneous Localisation and Mapping (SLAM) problem has received considerable attention [1, 2]. A central and highly important part of SLAM is loop closing, i.e. detecting that the robot has returned to a previously visited location. In this paper we consider robots equipped with laser range sensors, and define the problem of loop closure detection as determining whether or not the laser point clouds are from the same location. See Figure 1 for an illustration of the problem.

In previous work we showed that the problem of detecting loop closure from 2D horizontal laser point clouds could be cast as a two class (either same place or not) classification task [3]. By introducing 20 features, we were able to learn a classifier for real-time loop closure detection. The classification technique used is based on the machine learning algorithm AdaBoost [4], which builds a classifier by concatenating decision stumps (one level decision trees). The result is a powerful nonlinear classifier which has good generalisation properties [5, 6].

The main contribution of the paper is the extension of previous work on 2D horizontal point clouds [3] to full 3D point clouds. 41 features are defined and used to create decision stumps. The stumps are combined into a classifier using AdaBoost. We evaluate our approach for loop closing on publicly available data and compare our results to previously published results. The loop closure classifier is used in a SLAM framework using an Exactly Sparse Delayed-state Filter (ESDF) [7], and is shown to generalise well between environments.

This work was supported by CADICS, a Linnaeus research environment for Control, Autonomy, and Decision-making In Complex Systems, funded by the Swedish Research Council.

K. Granström and T. B. Schön is with the Division of Automatic Control, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden `{karl, schon}@isy.liu.se`

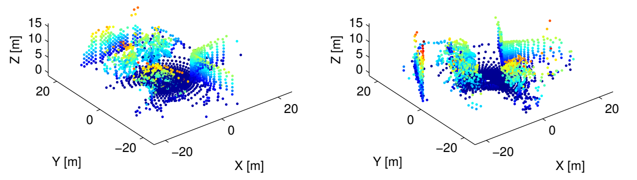


Fig. 1: Illustration of the loop closure detection problem. Are the laser point clouds from the same location? Color is used to accentuate height.

II. Related Work

This section summarizes previous work on loop closure detection using range sensors, in both 2D and 3D, as well as cameras. The detection results are summarised in Table I, where we compare our results to the results reported in related work. Neither one of the methods presented here uses prior knowledge of the relative pose for the data pair that is compared, however tests were performed in slightly different manner, making a direct comparison of the results difficult.

TABLE I: Comparison of previous results and our results. Detection rate (D) for given levels of false alarm rate (FA).

Source	FA [%]	D [%]	Comment
[8]	0	37/48	Images, City Centre/New College
[9]	1	51	2D point clouds
[3]	1	85	2D point clouds
[10]	0	47	3D point clouds
Our results	0	63	3D point clouds

Previously we presented loop closure detection by compressing point clouds to feature vectors which were then compared using an AdaBoost learned classifier [3]. Detection rates of 85% were achieved at 1% false alarm rate. The point clouds were described using 20 rotation invariant features describing different geometric properties of the point clouds.

A similar classification approach based on point cloud features and AdaBoost has been used for people detection [11] and place recognition [12]. For people detection the point clouds were segmented and each segment classified as belonging to a pair of legs or not, detection rates of over 90% were achieved. For place recognition, three classes were used (corridor, room and doorway) [12], hence the results do not easily compare to the two class loop closure detection results presented here.

An example of loop closure detection for 2D point clouds is the work by Bosse *et al* [9]. They use consecutive point clouds to build submaps, which are then compressed using orientation and projection histograms as a compact description of submap characteristics. Entropy metrics and quality metrics are used to compare point clouds to each other. A 51% detection rate for 1% false alarm rate is reported for suburban data. Extending the work on 2D data, keypoints are designed which provide a global description of the point clouds [13], thus making it possible to avoid pairwise comparison of all local submaps which can prove to be very time consuming for large data sets.

For the similar problem of object recognition using 3D points, regional shape descriptors have been used [14, 15]. Object recognition must handle occlusion from other objects, similarly to how loop closure detection must handle occlusion from moving objects, however object recognition often rely on an existing database of object models. Regional shape descriptors have also been used for place recognition for 3D point clouds [16]. Here, place recognition is defined as the problem of detecting the return to the same place and finding the corresponding relative pose [13, 16], i.e. it includes both relative pose estimation, and what we here define as loop closure detection.

Magnusson *et al* have presented results for loop closure detection for outdoor, indoor and underground mine data [10]. Their method is based on the Normal Distribution Transform (NDT) [17], which is a local descriptor of the point cloud. The point cloud is discretised using bins and the points in each bin are described as either linear, planar or spherical. The NDT is exploited to create feature histograms based on surface orientation and smoothness. For the outdoor data 47% detection is achieved with no false alarms.

Cummins and Newman have presented results on loop closure detection using images [8], with detection rates of up to 37% and 48% at 0% false alarm for the City Centre and the New College datasets [18], respectively. However, it should be noted that it is difficult to compare results from different types of sensors. Cummins and Newman also present interesting methods to handle occlusion, a problem that is often present in dynamic environments. In more recent work, they show results for a very large 1000km data set [19]. A 3% detection rate is achieved at 0% false alarm rate, which is still enough detection to produce good mapping results.

III. Loop Closure Detection

This section presents the loop closure detection algorithm. A mobile robot equipped with a laser range sensor moves through unknown territory and acquires point clouds $\mathbf{p}_k$ at times t_k along the trajectory. A point cloud $\mathbf{p}_k$ is defined as

$$\mathbf{p}_k = \{p_i^k\}_{i=1}^N, \quad p_i^k \in \mathbb{R}^3, \tag{1}$$

where N is the number of points in the cloud. For simplicity, index k is dropped from p_i^k in the remainder of the paper. After moving in a loop the robot comes to a previously visited location, and the two point clouds, acquired at different times, should resemble each other. A comparison of the point clouds is performed in order to determine if a loop closure has occurred or not. To facilitate this comparison, two types of features are first introduced. From the features a classifier is then learned using AdaBoost. The learned classifier is used to detect loop closure in experiments. A block diagram overview of the loop closure detection is shown in Figure 2.

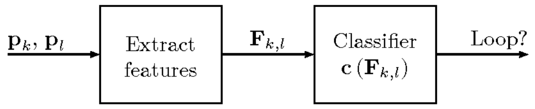

$\mathbf{p}_k, \mathbf{p}_l$
Extract features
$\mathbf{F}_{k,l}$
Classifier $\mathbf{c}(\mathbf{F}_{k,l})$
Loop?

Fig. 2: Overview of loop closure detection algorithm. The input is a pair of point clouds and the output is a loop closure classification decision.

A. Data

For experiments, two data sets have been used, both are publicly available [20][1]. The first one, Hannover 2 (*hann2*), contains 924 outdoor 3D scans from a campus area, covering a trajectory of approximately 1.24 km. Each 3D point cloud contains approximately 16600 points with a maximum measureable range of 30m. From this data set 3130 positive data pairs (point clouds from the same location) and 7190 negative data pairs (point clouds from different locations) are taken. With 924 point clouds, several more pairs could have been found in the data, however the number of training data was limited to the 10320 pairs to keep computational times during experiments tractable. The positive data pairs were chosen as the scan pairs taken less than 3m apart [10]. The negative data were chosen as a random subset of the remaining data pairs, i.e. those more than 3m apart.

The second data set, AASS-loop (*AASS*), contains 60 indoor 3D scans from an office environment, covering a trajectory of 111 m. Each 3D point cloud contains approximately 112000 points with a maximum measureable range of 15m. From this data set 16 positive and 324 negative data pairs are taken. The positive data pairs are those taken less than 1m apart [10], the negative data pairs are a random subset of the remaining data pairs. Due to the limited number of positive data pairs, we chose to not use all negative data. The impact of few data pairs is adressed further in Section IV.

Both data sets were acquired using 2D planar laser range finders, and 3D point clouds were obtained using pan/tilt units. Each 3D point cloud thus consists of a collection of 2D planar range scans. The points in each 3D point cloud are be ordered according to the order in which the points are measured by the sensor setup. Some of the features defined in Section III-B use this ordering of the points when the feature value is computed.

B. Features

The main reason for working with features is dimensionality reduction - working with n_f features is easier than working with the full point clouds since $n_f \ll N$. In this work, two types of features are used. The first type, f_j^1, is a function that take a point cloud as input and returns a real number. Typically, features that represent geometric properties of the point cloud are used, e.g. volume of point cloud or average range. The features of the first type are collected in a vector $\mathbf{f}_k^1 \in \mathbb{R}^{n_{f^1}}$, where k again refers to the time t_k when the point cloud was acquired. The second type of feature used, f_j^2, are range histograms with various bin sizes b_j. In total $n_f = 41$ features are used, $n_{f^1} = 32$ of type 1 and $n_{f^2} = 9$ of type 2.

[1]Thanks to Oliver Wulf, Leibniz University, Germany and Martin Magnusson, AASS, Örebro University, Sweden for providing the data sets.

In order to facilitate comparison of two point clouds from times t_k and t_l, the features of both types are compared. For the first type, elementwise absolute value of the feature vector difference is computed,

$$\mathbf{F}^1_{k,l} = \left|\mathbf{f}^1_k - \mathbf{f}^1_l\right|. \tag{2}$$

The underlying idea here is that point clouds acquired at the same location will have similar feature values $\mathbf{f}^1_k$ and $\mathbf{f}^1_l$, and hence each element of $\mathbf{F}^1_{k,l}$ should be small. For the second type of feature, for each bin size b_j the correlation coefficient for the two corresponding range histograms is computed. Here, the underlying idea is that point clouds acquired at the same location will have similar range histograms, and thus the correlation coefficient should be close to 1. The correlation coefficients are collected in a vector $\mathbf{F}^2_{k,l}$, and the comparisons of both types of features are concatenated in a vector as $\mathbf{F}_{k,l} = \left[\mathbf{F}^1_{k,l}, \mathbf{F}^2_{k,l}\right]$. $\mathbf{F}_{k,l}$ will henceforth be referred to as the set of extracted features for two point clouds indexed k and l.

In 2D 20 features were used [3], some of these features have been generalised to 3D (e.g. area to volume) while others have been kept as they were inherently in 2D (e.g. average range). Similar 2D features have been used for people detection and place recognition [11, 12]. A few of the utilised features are defined using the range from sensor to point, thus introducing a depedency on the sensor position from which the point cloud was acquired. An interesting implication of this is that the method could possibly be limited to situations where the robot is following a defined roadway, e.g. a street or an office hallway, and may not succeed in a more open area, e.g. a surface mine. In this work it is shown that the method can detect loop closure from point clouds with up to 3m relative translation, see Section IV for experimental results. It remains within future work to fully evaluate how the method scales against translations $> 3\text{m}$, i.e. how the method handles point clouds with partial overlap.

Given a point cloud $\mathbf{p}_k$, 14 constants need to be specified for computing the features. The first one, denoted $r_{\max}$, is the maximum measurable range, which is determined by the sensor that was used for data acquisition. For the *hann2* and *AASS* data sets we set $r_{\max} = 30\text{m}$ and $r_{\max} = 15\text{m}$ respectively. Remaining thresholds need to be specified manually. For both data sets, the parameters were set to: $g_{\text{dist}} = 2.5\text{m}$, $g_{r_1} = r_{\max}$, $g_{r_2} = 0.75 r_{\max}$ and $g_{r_3} = 0.5 r_{\max}$. Bins of size 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 2.5 and 3 metres were used for the range histograms. For each point p_i, the range r_i is computed as the distance from the origin (sensor location) to the point. Any point with $r_i > r_{\max}$ is translated towards the origin so that $r_i = r_{\max}$ before the features are computed. The following features are used:

1) - 2) ***Volume:*** Measures the volume of the point cloud by adding the volumes of the individual laser measurements. Each point is seen as the centre point of the base of a pyramid with its peak in the origin. Let α and β be the laser range sensor's vertical and horisontal angular resolution, and let $l_i = 2r_i \tan\left(\frac{\alpha}{2}\right)$ and $w_i = 2r_i \tan\left(\frac{\beta}{2}\right)$ be length and width of the pyramid base, and $h_i = r_i$ the height at point i. The volume of the pyramid is $v_i = \frac{l_i w_i h_i}{3}$. The volume is computed as

$$v_{\max} = \frac{4}{3}\tan\left(\frac{\alpha}{2}\right)\tan\left(\frac{\beta}{2}\right) r^3_{\max} \tag{3a}$$

$$f^1_1 = \frac{1}{N v_{\max}} \sum_{i=1}^{N} v_i = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{r_i}{r_{\max}}\right)^3 \tag{3b}$$

The volume is normalised by dividing by the maximum measurable volume $N v_{\max}$, i.e. the volume when all ranges equal $r_{\max}$. Notice that the explicit values of α and β do not matter. f^1_2 is the volume computed using points with $r_i < r_{\max}$.

3) - 4) ***Average Range:*** Let the normalised range be $r^{\text{n}}_i = r_i / r_{\max}$. f^1_3 is the average r^{n}_i for ranges $r_i < r_{\max}$ and f^1_4 is the average r^{n}_i for all ranges.

5) - 6) ***Standard Deviation of Range:*** f^1_5 is the standard deviation of r^{n}_i for ranges $r_i < r_{\max}$ and f^1_6 is the standard deviation of r^{n}_i for all ranges.

7) - 9) ***Sphere:*** A sphere is fitted to all points in the cloud in a least squares sense, which returns the centre of the fitted sphere p_c and the radius of the fitted sphere r_c. f^1_7 is $r_c / r_{\max}$, f^1_8 is the residual sum of squares divided by $N r_c$,

$$f^1_8 = \frac{1}{N r_c}\sum_{i=1}^{N}\left(r_c - \|p_c - p_i\|\right)^2, \tag{4}$$

where $\|\cdot\|$ is the Euclidean norm. f^1_9 is $\frac{\|p_c\|}{r_{\max}}$.

10) - 12) ***Centroid:*** Let $\bar{p}$ be the mean position of the point cloud, computed for all points $r_i < r_{\max}$. $f^1_{10} = \|\bar{p}\|$, f^1_{11} is the mean distance from $\bar{p}$ for points $r_i < r_{\max}$ and f^1_{12} is the standard deviation of the distances from $\bar{p}$ for points $r_i < r_{\max}$.

13) - 14) ***Maximum Range:*** f^1_{13} is the number of ranges $r_i = r_{\max}$ and f^1_{14} is the number of ranges $r_i < r_{\max}$.

15) - 17) ***Distance:*** Let the distance between consecutive points be $\delta p_i = \|p_i - p_{i+1}\|$. f^1_{15} is the sum of δp_i for all points. f^1_{16} is the sum of δp_i, for consecutive points with $r_i, r_{i+1} < r_{\max}$. f^1_{17} is the sum of all $\delta p_i < g_{\text{dist}}$, for consecutive points with $r_i, r_{i+1} < r_{\max}$.

18) ***Regularity:*** f^1_{18} is the standard deviation of δp_i, for consecutive points with $r_i, r_{i+1} < r_{\max}$.

19) - 20) ***Curvature:*** Let A be the area covered by the triangle with corners in p_{i-1}, p_i and p_{i+1}, and let $d_{i-1,i}$, $d_{i,i+1}$ and $d_{i-1,i+1}$ be the pairwise point to point distances. The curvature at p_i is computed as $k_i = \frac{4A}{d_{i-1,i} d_{i,i+1} d_{i-1,i+1}}$. Curvature is computed for $p_i \in \mathbf{I}$, where $\mathbf{I} = \{p_i : r_{i-1}, r_i, r_{i+1} < r_{\max},\ d_{i-1,i}, d_{i,i+1}, d_{i-1,i+1} < g_{\text{dist}}\}$. f^1_{19} is the mean curvature and f^1_{20} is the standard deviation of the curvatures.

21) - 22) ***Range Kurtosis:*** Range kurtosis is a measure of the peakedness of the histogram of ranges. Sample kurtosis is computed for all points $r_i < r_{\max}$ as follows

$$m_k = \frac{1}{N_{r_i < r_{\max}}} \sum_{i\,:\,r_i < r_{\max}} \left(r_i - \bar{r}\right)^k, \tag{5a}$$

$$f^1_{21} = \frac{m_4}{\left(m_2\right)^2} - 3, \tag{5b}$$

where $\bar{r}$ is mean range, and $N_{r_i < r_{\max}}$ is the number of ranges $r_i < r_{\max}$. f^1_{22} is range kurtosis computed for all points in the cloud.

23) - 26) ***Relative Range:*** Let the relative range be $r_i^{\mathrm{r}} = r_i / r_{i+1}$. f_{23}^1 is the mean of r_i^{r} and f_{24}^1 is the standard deviation of r_i^{r} for all ranges. f_{25}^1 and f_{26}^1 are the mean and the standard deviation of r_i^{r}, respectively, computed for $r_i, r_{i+1} < r_{\max}$.

27) - 32) ***Range Difference 1-6:*** Mean and standard deviation of range difference $r_i^{\mathrm{d}} = |r_i - r_{i+1}|$. The features are calculated for all ranges less than or equal to a varying range gate g_r. g_{r_1} gives f_{27}^1 (mean) and f_{28}^1 (standard deviation), and g_{r_2} and g_{r_3} gives f_{29}^1 to f_{32}^1. The features are normalised by division by the respective g_{r_i}.

33) - 41) ***Range Histogram:*** f_{33}^2 to f_{41}^2 are range histograms. Bins of sizes b_j are used to tabulate the ranges.

C. Classification and Boosting

AdaBoost is a machine learning algorithm that is used to learn a classifier from the 41 features. As input to the learning algorithm, n hand-labeled training data pairs $\{(\mathbf{F}_{i_1,i_2}, y_i)\}_{i=1}^n$ are provided. y_i is a binary variable, $y_i = \{0, 1\}$ for negative and positive laser pairs, respectively. $\mathbf{F}_{i_1,i_2}$ is the set of extracted features from the i:th training data pair.

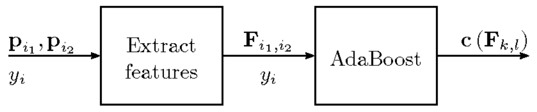
The learning phase of AdaBoost is an iterative procedure that consecutively adds weak classifiers to a set of previously added weak classifiers to find a good combination that together constitutes a strong classifier. The weak classifiers adopted are decision stumps defined as:

$$c(\mathbf{F}_{k,l}, \theta) = \begin{cases} 1 & \text{if } pf < p\lambda \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

with parameters $\theta = \{f, p, \lambda\}$, where p is the polarity ($p = \pm 1$), f is the particular feature selected and λ is a threshold. Learning proceeds for T iterations, details and motivation for how T is chosen is presented in Section IV-A. The output from AdaBoost is a strong classifier

$$\mathbf{c}(\mathbf{F}_{k,l}) = \begin{cases} 1 & \text{if } \sum_{t=1}^T \alpha_t c(\mathbf{F}_{k,l}, \theta_t) \geq K \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where α_t are weights for the weak classifiers $c(\mathbf{F}_{k,l}, \theta_t)$ added during learning, and $K \in [0, 1]$ is a user specified threshold. The strong classifier (7) is used in SLAM experiments to detect loop closure. A detailed description of AdaBoost goes beyond the scope of this paper, the reader is refered to the references [3,4]. An overview of the classifier learning is given in Figure 3.



$\mathbf{p}_{i_1}, \mathbf{p}_{i_2}$
y_i
Extract features
$\mathbf{F}_{i_1,i_2}$
y_i
AdaBoost
$\mathbf{c}(\mathbf{F}_{k,l})$

Fig. 3: Overview of classifier learning. The input is hand labeled pairs of training data and the output is a learned classifier.

IV. Experimental Results

This section contains results from experiments[2]. The classifier is evaluated in terms of detection-rate D, missed detection-rate MD and false alarm-rate FA, defined as follows:

$$D = \frac{\#\text{ positive data pairs classified as positive}}{\#\text{ positive data pairs}}$$
$$MD = \frac{\#\text{ positive data pairs classified as negative}}{\#\text{ positive data pairs}}$$
$$FA = \frac{\#\text{ negative data pairs classified as positive}}{\#\text{ negative data pairs}}$$

For the loop closing problem, we are concerned with minimising the number of false alarms while keeping the detection rate as high as possible. In previous work, D has been reported at 1% FA [3,9], in other work D has been reported at 0% FA (or equivalently at 100% precision) [8,10]. While it is very important to keep FA low, it is possible to find and reject false alarms in subsequent stages, e.g. when the relative pose is found via point cloud registration [3,9], or using a cascade of several methods [13]. However, even if a combination of methods is used, the false alarms have to be rejected at some stage since closing a false loop could prove disastrous for the localisation and/or mapping process. Further, finding a subset of loop closures is most often sufficient to produce good results [8–10, 19]. Therefore, we find the detection rate at 0% false alarm to be most interesting. However, for completeness and ease of comparison, we present results at both 0% and 1% false alarm.

The experiments in Sections IV-A to IV-C were conducted using k-fold cross validation on the *hann2* and *AASS* data sets. It is important to note that in each experiment the validation portion of the data was fully disjoint from the training portion. The partitioning into folds was performed by randomly permuting the order of the data. Since different permutations give slightly different results, k-fold cross validation was performed multiple times, each time with a new data order permutation. The presented results are sample mean from the cross validations. While the validation and training data in these experiments are fully disjoint, they are from the same data set and thus from the same environment. Experiments where the validation and training data are from separate environments are presented in Section IV-D.

A. Number of training rounds T

The first experiment was conducted to determine an appropriate number of training rounds T for AdaBoost. Strong classifiers were trained on the *hann2* data for different values of T between 1 and 250, and the error rates from 200 10-fold cross validations were examined. Results from the experiment are shown in Figure 4. From this figure, $T = 50$ was determined to be an appropriate number of training rounds. $T = 50$ is used in all subsequent experiments.

B. Loop Closure Feature Analysis

During the learning phase, in each training round AdaBoost selects the feature that best improves classifier performance. The same feature can be added to the classifier multiple times, and features chosen in early training rounds will have a larger weight α_t than features chosen in later training rounds. Test 1 in Table II shows which features were chosen in the first five training rounds for both data sets, respectively.

To further examine which features are best for detecting loop closure, we started by training a strong classifier on all

[2]Matlab code is published online, contact the authors for a link.

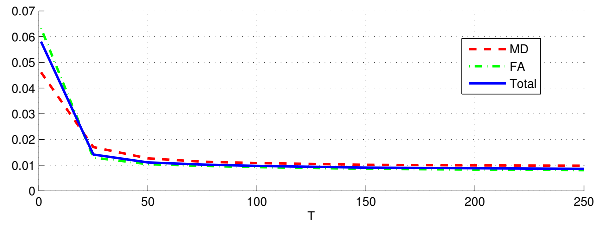


Fig. 4: Error rates for *hann2* data for different number of training rounds T in AdaBoost. Total error level (both MD and FA) stops declining after $T = 50$ training rounds.

TABLE II: Best features for loop closing

TEST 1					
Training round	1	2	3	4	5
Added feature, *hann2*	35	1	7	27	20
Added feature, *AASS*	33	40	32	36	41
TEST 2, *hann2*					
Feature removed	21	8	10	28	35
Total error [%]	1.29	1.15	1.14	1.13	1.13
TEST 2, *AASS*					
Feature removed	41	22	33	32	40
Total error [%]	2.27	2.24	2.16	2.08	2.04

features. For *hann2*, the resulting total test error rate was 1.10% and for *AASS* the total test error rate was 1.92%. We then proceeded to remove each feature one at a time and train classifiers on the remaining features. By examining the resulting test error rates, we could determine which features had the most negative effect on the error rates after being removed from the set of features. In Table II Test 2 summarises the results for the five most important features for both data sets.

As can be seen in Table II, for *AASS*, the features that are added in early training rounds also have the largest negative effect when removed. Those features, numbers 33, 40, 32 and 41, correspond to range histograms with bin sizes 0.1, 2.5 and 3 m, respectively, and standard deviation of range difference for ranges shorter than or equal to $g_{r_3} = 0.5r_{\max}$. For *hann2*, the results are less consistent, however feature 35, corresponding to range histogram with bin size 0.5 m, appears to be most effective at separating the two classes of data pairs.

Furthermore, Tests 1 and 2 show that the most important features for loop closure detection are not the same for the two data sets. Since *hann2* is an outdoor data set and *AASS* is an indoor data set, this could mean that the classifier does not generalise well when trained and tested on data from different environments. This issue is addressed further in Section IV-D, where it is shown that the classifier in fact does generalise well from outdoor to indoor data.

C. Classifier Characteristics

This experiment was conducted to evaluate the classifier characteristics, i.e. the classifier's ability to achieve good levels of D for low levels of FA. For *hann2*, 10-fold cross validation was performed for 750 different permutations of the data pairs. Due to the smaller number of positive data, for *AASS* 4-fold cross validation was performed for 10000 different permutations of the data pairs. By varying the threshold K in (7), different levels of D and FA are achieved when the validation data is classified. The results are presented in Table III, and in Figure 5 as Receiver Operating Characteristic (ROC) curves, where D is plotted against FA. In the table detection rates are given $\pm$ one standard deviation, and with the maximum and minimum values that were recorded. As a comparison, results from related work are included for both data sets [10]. It should be noted though, that while subsets of the data sets are used here, results at 1% FA are reported for the entire data sets in [10]. Thus, the results should be interpreted with care. In Figure 5, the area under the ROC-curve is approximately 0.999 for *hann2* and 0.936 for *AASS*.

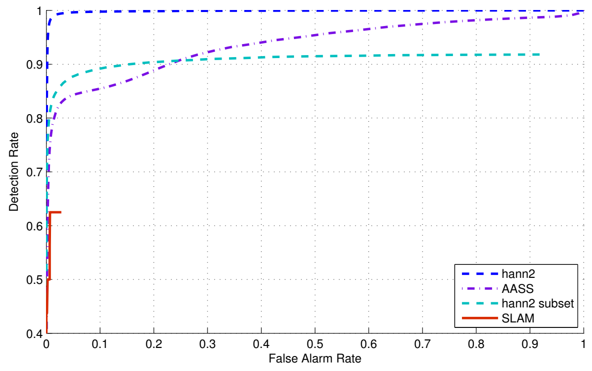


Fig. 5: ROC-curve showing D for different levels of FA.

TABLE III: Classification characteristics, all numbers in %

Data set	FA	D	Min/Max	D [10]
hann2	0	63 ± 6	28/76	47
	1	99 ± 0.1	98/99	81
AASS	0	53 ± 14	0/88	70
	1	78 ± 6	56/88	63

As is seen in Table III, 0% was the lowest D for 0% FA for *AASS*. This happened in 5 out of 10000 cross validations. Furthermore, the mean D is lower than related work [10], and the standard deviation of D is higher than for *hann2*. For this data set the number of positive data pairs is low, compared to the number of negative data pairs (16 vs. 324), which is an intuitive reason for the worse performance. The training data is crucial to the AdaBoost learning, and it is possible that there is not enough positive pairs to be able to achieve a high degree of class separation.

To test this hypothesis, 16 positive and 300 negative data pairs were randomly selected from the large set of *hann2* data pairs, and a classifier was learned and evaluated using 4-fold cross validation on the subset of data. Out of 1000 such random subsets, 30 gave classifiers with 0% D for 0% FA (mean D was $72\% \pm 19\%$ for 0% FA). While this result is not sufficient to conclude that the low number of positive data pairs is the sole reason for the worse results for *AASS* compared to related work and *hann2*, it does support the hypothesis that the relatively low number of positive training data has a strong negative effect on the learned classifiers ability to achieve a good degree of class separation. The ROC-curve corresponding to this test is labeled *hann2 subset* in Figure 5. Comparing to the curve for the full *hann2* data set shows a clear negative effect.

D. SLAM *Experiment*

This experiment was conducted for two reasons, one is to see how the classifier would perform in a SLAM setting, the other is to see how the classifier performs when it is trained on outdoor data and then tested on indoor data. The positive and negative data pairs from *hann2* were used to train a classifier. The classifier was then used to classify data pairs from the *AASS* data set.

The implemented SLAM framework is by now well known, hence only specific design choices are provided. The reader is refered to the references for exact implementation details. A delayed state extended information filter, called ESDF [7], is implemented. The state vector contains a history of 6-DOF poses, each with (x, y, z)-position and Euler angles (ϕ, θ, ψ) representing roll, pitch and heading as the angles are defined in [21]. Motion and measurement models are defined using the coordinate frame notation by Smith, Self and Cheeseman [22]. Robot motion is computed using 3D-NDT [23], initialised by odometry[3]. After loop closure is detected, ICP [24–26] initialised by the estimated relative pose from the ESDF is used to compute the relative pose. This is sufficient for the particular data set used here, however a general solution would require a method which is independent of the estimated relative pose.

In this experiment each point cloud $\mathbf{p}_k$ was compared to all previous point clouds $\{\mathbf{p}_i\}_{i=1}^{k-1}$. In each time step the pair with highest $\sum_{t=1}^{T} \alpha_t c_t (\mathbf{F}_{k,l})$, is considered a match if $\sum_{t=1}^{T} \alpha_t c_t (\mathbf{F}_{k,l}) \geq K \sum_{t=1}^{T} \alpha_t$, cf. (7). All other pairs are considered to not be matches. Since the time to compare two point clouds to each other is constant, comparing to all previous point clouds results in a linearly increasing time complexity as more point clouds are acquired. For the experiments presented here, this has not been a problem, however for very large data sets this could become problematic. An alternative to comparing to all previous data, is to only compare to the subset of data acquired at locations which are within the current uncertainty ellipsoid, thus reducing the amount of time needed to compare point clouds. Doing so is not without problems though, since inconsistencies in the estimation of trajectory mean and covariance, e.g. due to linearisation errors, may lead to true loop closure locations falling outside the uncertainty ellipsoid [27]. This is typically the case for larger data sets, where the accrued drift in trajectory estimation can lead to large estimation errors. Further, an important purpose of any loop closure detection method is to support the estimation in exactly such cases, when the estimation of trajectory mean and covariance is inconsistent. Thus, relying on mean and covariance to feed candidate pairs to the loop closure method is inadviceable. As a possible remedy to the linearly increase time demands of pairwise comparison to all previous data, global descriptors could be used to obtain a subset of the pairs [13, 19], for which pairwise comparison can be made.

The result from the experiment is shown as a classification matrix in Figure 6a. The $(k, l)^{\text{th}}$ element of the classification matrix is $\frac{\sum_{t=1}^{T} \alpha_t c_t (\mathbf{F}_{k,l})}{\sum_{t=1}^{T} \alpha_t}$. The corresponding ROC-curve is labeled SLAM in Figure 5, with 44% D for 0% FA. There is a high similarity between Figures 6b and 6c, showing that the generalisation properties of the features and the classifier are good. The classifier used in the experiment was trained on outdoor data containing 16600 points per cloud, $r_{\text{max}} = 30$, and then tested on indoor data containing 112000 points per cloud, $r_{\text{max}} = 15$. Figure 6e shows a 2D projection of the resulting map from the SLAM experiment, with the robot trajectory overlaid. The robot trajectory is compared to dead reckoning in Figure 6d. For this part of the experiment, a minimum loop size of 5 poses was introduced, explaining why the detected loop closure between poses 28 and 29 in Figure 6b is not present in Figure 6e.

[3]These transformations are available together with the point clouds.

E. *Time complexity*

This experiment was conducted to determine the time complexity of the proposed method. The code used in this work was implemented in Matlab and run on a 2.83GHz Intel Core2 Quad CPU with 3.48 GB of RAM running Windows. It should be noted that the implementation has not been optimized for speed.

The time to compute the features are 19.34ms and 225.10ms for *hann2* and *AASS* respectively. Comparing the features takes 0.845ms, and computing $\mathbf{c}(\mathbf{F}_{k,l})$ takes 0.78ms. The times are averages from computing and comparing features for the data pairs in each data set. As expected the time to compute the features is longer for *AASS*, which contains on average 112000 points per cloud, than for *hann2*, which contains on average 16600 points per cloud. Computing the 41 features only needs to be performed once per point cloud. Comparing the features from two point clouds takes just under 1ms, and classifying a set of extracted features takes just under 1ms when $T = 50$ weak classifiers are used in the strong classifier. Training a strong classifier for $T = 50$ iterations takes 15s when the *hann2* data pairs are used.

V. Conclusions and future work

This paper presented a new machine learning approach to the loop closure detection problem using 3D laser range data. 41 features were defined and combined into a classifier using AdaBoost. The classifier shows promising and competitive results for an outdoor data set, as well as reasonable results for an indoor data set. Furthermore, the classifier was shown to generalise well, since it can be trained on data from one environment and still perform well using data from another environment.

The tests with subsets of the *hann2* data pairs show that the number of training data is important for the resulting classifier properties. In future work we plan to investigate further the dependence on the number of training data for good class separation. An evaluation of how the method scales with translation is also needed, especially to address how the presented method handles partial point cloud overlap. Future work also include an investigation of how the linearly increasing time complexity, due to comparison to all previous data, can be overcome.

References

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): part I," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99 – 110, June 2006.

[2] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 108 – 117, Sept. 2006.

[3] K. Granström, J. Callmer, F. Ramos, and J. Nieto, "Learning to Detect Loop Closure from Range Data," in *Proceedings of 2009 IEEE International Conference on Robotics and Automation (ICRA)*, A. Bicchi, Ed., Kobe, Japan, May 2009, pp. 15–22.

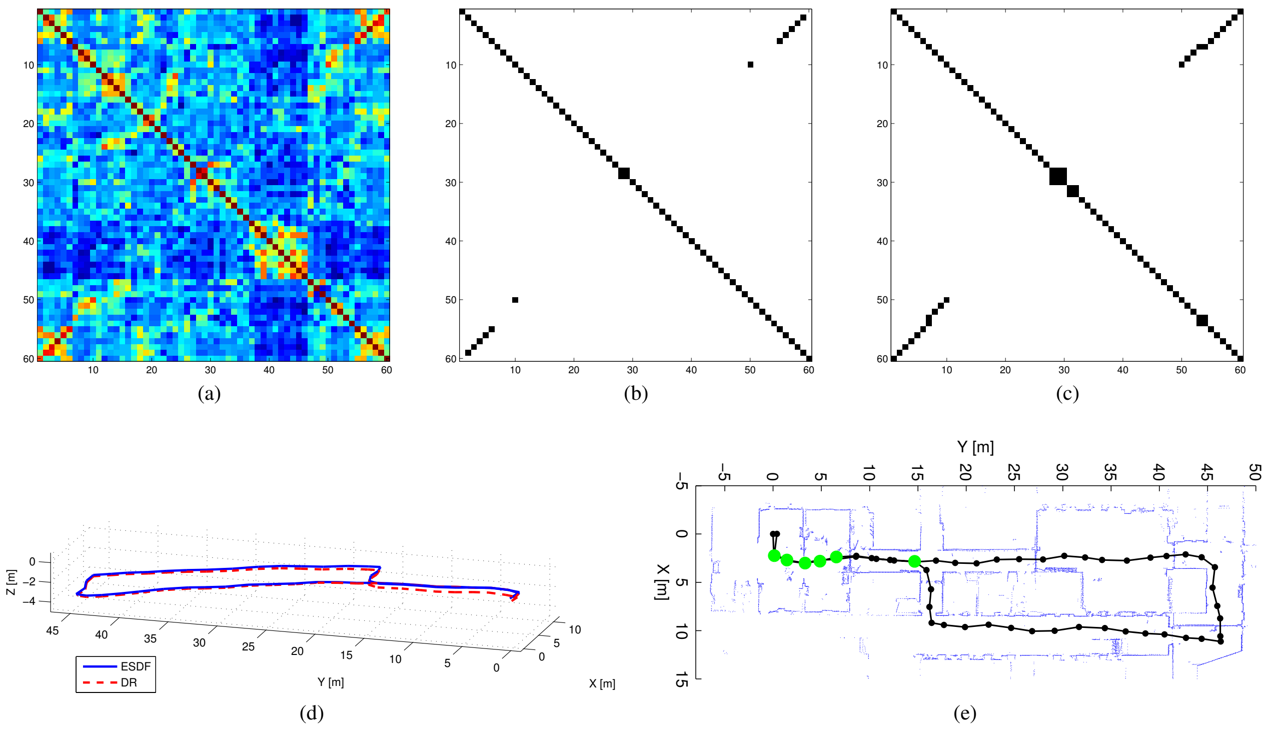


Fig. 6: Results from SLAM experiment on *AASS* data. (a) Classification matrix, warmer color means higher similarity. (b) Thresholded classification matrix, $K = 0.717$ (cf. (7)). (c) Ground truth distance matrix, showing data pairs less than 1m apart [10]. (d) Estimated trajectory (ESDF) vs. dead reckoning (DR). (e) Resulting map with robot trajectory overlaid. Detected loop closures are marked with thick green circles.

[4] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," in *Proceedings of the 1995 European Conference on Computational Learning Theory (EuroCOLT)*, Barcelona, Spain, March 1995.

[5] ——, "Experiments with a new boosting algorithm," in *In Proceedings of the Thirteenth International Conference on Machine Learning*, Bari, Italy, 1996, pp. 148–156.

[6] R. E. Schapire, "Theoretical views of boosting," in *Computational Learning Theory: 4^{th} European Conf., EuroCOLT'99*, Nordkirchen, Germany, 1999, pp. 1–10.

[7] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1100–1114, 2006.

[8] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[9] M. C. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based SLAM," *International Journal of Robotics Research*, vol. 27, no. 6, pp. 667–691, 2008.

[10] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, "Automatic appearance-based loop detection from 3D laser data using the normal distributions transform," *Journal of Field Robotics*, vol. 26, no. 11–12, pp. 892–914, Nov. 2009.

[11] K. O. Arras, O. M. Mozos, and W. Burgard, "Using Boosted Features for the Detection of People in 2D Range Data," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Roma, Italy, 2007.

[12] O. M. Mozos, C. Stachniss, and W. Burgard, "Supervised Learning of Places from Range Data using AdaBoost," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.

[13] M. Bosse and R. Zlot, "Keypoint design and evaluation for place recognition in 2d lidar maps," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1211–1224, 2009.

[14] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 1, pp. 433 – 449, May 1999.

[15] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Prague, Czech Republic, May 2004.

[16] M. Bosse and R. Zlot, "Place recognition using regional point dsecriptors for 3d mapping," in *Proceedings of the International Conference on Field and Service Robotics (FSR)*, Cambridge, Massachusets, USA, July 2009.

[17] P. Biber and W. Strasser, "The normal distribution transform: A new approach to laser scan matching," in *Proceedings of the IEEE RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, USA, October 2003, pp. 2743–2748.

[18] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The New College Vision and Laser Data Set," *International Journal for Robotics Research (IJRR)*, vol. 28, no. 5, pp. 595–599, May 2009.

[19] M. Cummins and P. Newman, "Highly scalable appearance-only slam fab-map 2.0," in *Robotics Science and Systems (RSS)*, Seattle, USA, June 2009.

[20] "Robotic 3d scan repository," Hannover, Germany, [Online; accessed 10-August-2009]. [Online]. Available: http://kos.informatik.uni-osnabrueck.de/3Dscans/

[21] R. M. Eustice, "Large-area visually augmented navigation for autonomous underwater vehicles," Ph.D. dissertation, Massachusetts Institute of Technology / Woods Hole Oceanographic Institution Joint Program, 2005.

[22] R. Smith, M. Self, and P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics," pp. 167–193, 1990.

[23] M. Magnusson, T. Duckett, and A. J. Lilienthal, "Scan registration for autonomous mining vehicles using 3D-NDT," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, Oct 24 2007.

[24] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[25] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image Vision Comput.*, vol. 10, no. 3, pp. 145–155, 1992.

[26] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[27] K. L. Ho and P. Newman, "Detecting loop closure with scene sequences," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 261–286, 2007.