Learning to close loops from range data

Karl Granström[*]
Division of Automatic Control
Department of Electrical Engineering
Linköping University
Linköping, Sweden
`karl@isy.liu.se`

Thomas B. Schön
Division of Automatic Control
Department of Electrical Engineering
Linköping University
Linköping, Sweden
`schon@isy.liu.se`

Juan I. Nieto
Australian Centre for Field Robotics
University of Sydney
Sydney, Australia
`j.nieto@acfr.usyd.edu.au`

Fabio T. Ramos
School of Information Technologies
Australian Centre for Field Robotics
University of Sydney
Sydney, Australia
`f.ramos@acfr.usyd.edu.au`

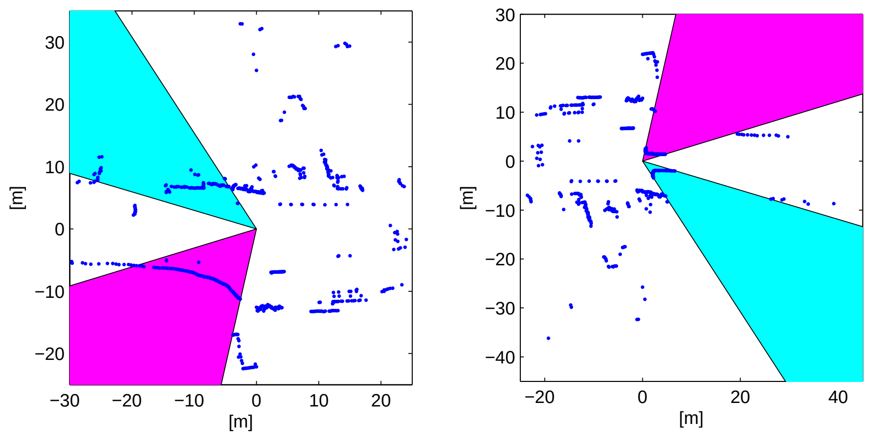
Abstract

This paper addresses the loop closure detection problem in SLAM, and presents a method for solving the problem using pairwise comparison of point clouds in both 2D and 3D. The point clouds are mathematically described using features that capture important geometric and statistical properties. The features are used as input to the machine learning algorithm AdaBoost, which is used to build a non-linear classifier capable of detecting loop closure from pairs of point clouds. Vantage point dependency in the detection process is eliminated by only using rotation invariant features, thus loop closure can be detected from arbitrary direction. The classifier is evaluated using publicly available data, and is shown to generalise well between environments. Detection rates of 66%, 63% and 53% for 0% false alarm rate are achieved for 2D outdoor data, 3D outdoor data and 3D indoor data, respectively. In both 2D and 3D, experiments are performed using publicly available data, showing that the proposed algorithm compares favourably to related work.

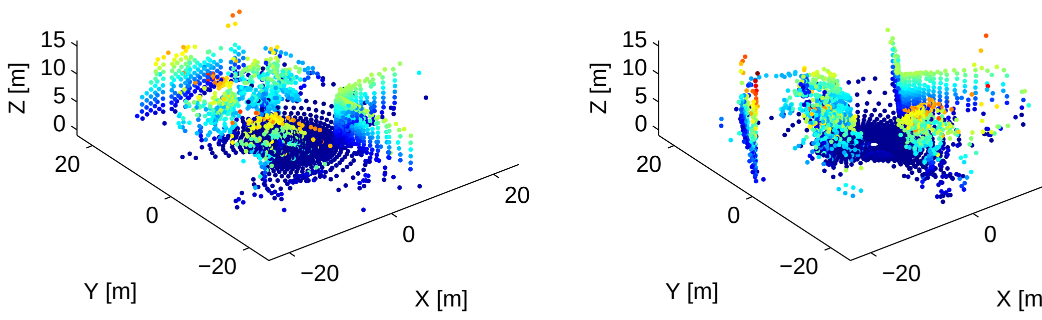
1 Introduction

Loop closure detection is defined as the problem of detecting when the robot has returned to a previously visited location. Being an integral part of the Simultaneous Localisation and Mapping (SLAM) problem, loop closure detection has received considerable attention in recent years. In particular, methods using vision sensors have broadly been presented, see e.g. (Goedeme et al., 2006; Tapus and Siegwart, 2006; Fraundorfer et al., 2007; Ho and Newman, 2007; Cummins and Newman, 2008; Angeli et al., 2008; Callmer et al., 2008; Eade and Drummond, 2008; Milford and Wyeth, 2008; Cummins and Newman, 2009; Konolige et al., 2010; Paul and Newman, 2010). Range sensors on the other hand, have not been so widely considered for loop detection, in particular with 2D sensors. This paper addresses the problem of loop closure detection using range sensor measurements and, similarly to many vision solutions, the problem is solved via pairwise data comparison. The proposed method applies equally well for both 2D and 3D data (see Figure 1 for examples of the problem in 2D and 3D).

[*]Author personal homepage `http://www.control.isy.liu.se/~karl`



(a) 2D example.



(b) 3D example.

Figure 1: Examples of the loop closure detection problem. In (a) two planar point clouds from the same location are shown. Parts of the scene are occluded by vehicles parked along the side of the road in the right point cloud, these parts are highlighted in colour. In (b) two 3D point clouds are shown, colour is used to accentuate the height. These point clouds are from the same location, which can be identified by the building corner and the wall opposite the corner.

In this paper the loop closure detection problem is cast as a classification task, in which a data pair is classified as either being from the same location, or not. The range sensor measurements, represented as point clouds, are mathematically described using features which capture important statistical and geometrical properties. The features provide an efficient means for dimensionality reduction, and also facillitate easy comparison of the point clouds. Furthermore, the features are fully invariant to rotation, thus enabling loop closure detection from arbitrary direction. Following the feature extraction process, a machine learning algorithm called AdaBoost is used to train a classifier. AdaBoost builds a classifier by combining simple binary classifiers, resulting in a decision boundary which is non-linear. AdaBoost renders a classifer with good generalisation properties which is able to robustly detect loop closure.

Similar point cloud features have been used in (Mozos et al., 2005; Arras et al., 2007; Brunskill et al., 2007). In this work, the set is extended with novel features to better address the problem of loop closure detection. The major contribution of this paper is the formulation of the loop closure detection algorithm with extensive experimental evaluation in urban environments, comparisons to related work using publicly available data sets, and a detailed implementation description.

Early versions of this work have been presented previously (Granström et al., 2009; Granström and Schön, 2010). This paper presents extensions of the previously presented results. In particular, a more detailed and thorough evaluation is presented by using different publicly available datasets. The paper is organised as follows: the next section contains an overview of related work. Section 3 presents the suggested loop closure detection method. A general framework for SLAM using the loop closure methodology is presented in Section 4. Extensive experimental results and comparisons are presented in Section 5, and Conclusions and Future work are presented in Section 6.

2 Related work

In this section we give an overview of related work on large-scale SLAM and loop closure detection, using 2D and 3D range sensors, as well as cameras. A detailed comparison between the work in this paper and the related work using similar sensor setups is given in Section 5.

SLAM algorithms based on raw laser scans have been shown to present a more general solution than classic feature-based (Gutmann and Konolige, 1999). For example, in (Hähnel et al., 2003; Bosse and Zlot, 2008; Newman et al., 2006), raw laser scans were used for relative pose estimation. The mapping approach presented in (Gutmann and Konolige, 1999) joins sequences of laser scans to form local maps. The local maps are then correlated with a global laser map to detect loop closures. Laser range scans are used in conjunction with EKF-SLAM in (Nieto et al., 2007). The authors introduced an algorithm where landmarks are defined by templates composed of raw sensed data. The main advantage claimed is that the algorithm does not need to rely on geometric landmarks as traditional EKF-SLAM. When a landmark is re-observed, the raw template could be augmented with new sensor measurements, thus improving the landmark's representation. The authors also introduced a shape validation measure as a mechanism to enhance data association. In summary, the main advantage in all these works is the ability of the algorithms to work in different environments thanks to the general representation obtained from raw sensor data.

Mapping algorithms based on laser scans and vision have shown to be robust. The work presented in (Ho and Newman, 2005) performs loop closure detection using visual cues and laser data. Shape descriptors such as angle histograms and entropy are used to describe and match the laser scans. A loop closure is only accepted if both visual and spatial appearance comparisons credited the match. In (Ramos et al., 2007b), laser range scans are fused with images to form descriptors of the objects used as landmarks. The laser scans are used to detect regions of interest in the images through polynomial fitting of laser scan segments while the landmarks are represented using visual features.

An approach to loop closure detection in 2D which is similar to the one taken in this paper is presented in (Brunskill et al., 2007). The authors construct submaps from multiple consecutive point clouds. Before initialising a new submap, it is checked whether the current point cloud is similar to any of the previous submaps. Each submap is described using a set of rotation invariant features, several of which are similar to the features used in this work. Next, AdaBoost is used to train one classifier for each submap, where each classifier test whether or not a point cloud belongs to the corresponding submap. The learning is unsupervised and performed online during SLAM, in contrast to learning in this work which is supervised and performed offline, prior to SLAM. The authors present results from two small scale indoor data sets, and show detection rates of 91% and 83% at precision rates of 92% and 84% for the two data sets, respectively. Comparing to the work presented in this paper, the main difference is that (Brunskill et al., 2007) learn representations for each submap, and require one classifier for each submap, while a general similarity metric between two arbitrary point clouds is learnt in this paper. Thus the loop closure detection problem can be solved using just one classifier.

Another example of loop closure detection for 2D point clouds is the work in (Bosse and Zlot, 2008). They use consecutive laser scans to build submaps, typically a submap contains laser scans from tens of meters of travel. The submaps 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 (Bosse and Zlot, 2009a), thus making it possible to avoid pairwise comparison of all local submaps which can prove to be very time consuming for large data sets.

Work on object recognition and shape matching in 2D using point based descriptions includes the work on shape context (Belongie et al., 2002). The shape context is a global descriptor of each point which allows the point correspondence problem to be solved as an optimal assignment problem. For loop closing in 2D, a method which relies on the extraction of linear landmarks is proposed in (Rencken et al., 1999). Loops are detected by matching landmarks from partial maps in structured indoor environments. For global robot localisation using 2D laser in unstructured environments, the gestalt features have also been proposed (Walthelm, 2002).

For the similar problem of object recognition using 3D points, regional shape descriptors have been used (Johnson and Hebert, 1999; Frome et al., 2004). Object recognition must handle occlusion from other objects, similarly to how loop closure detection must handle occlusion from moving objects. However object recognition often relies on an existing database of object models. Regional shape descriptors have also been used for place recognition for 3D point clouds (Bosse and Zlot, 2009b). Here, place recognition is defined as the problem of detecting the return to the same place and finding the corresponding relative pose (Bosse and Zlot, 2009a; Bosse and Zlot, 2009b), i.e. it includes both relative pose estimation, and what we here define as loop closure detection.

In (Magnusson et al., 2009) results for loop closure detection are presented for outdoor, indoor and underground mine data. The method presented is based on the Normal Distribution Transform (NDT) (Biber and Strasser, 2003), which acts as a local descriptor of the point cloud. After discretising space into bins, or cubes, the points in each bin are described as either linear, planar or spherical by comparing the size of the covariance matrix eigenvalues. The NDT is exploited to create feature histograms based on surface orientation and smoothness. Invariance to rotation is achieved after scans have been aligned according to the dominant planar surface orientation. The authors show detection rates of 47.0%, 69.6% and 28.6% for 0% false alarm, for outdoor, indoor and mine data, respectively.

In more recent work another method for loop detection for 3D point clouds was presented (Steder et al., 2010). The point cloud is transformed into a range image, from which features are extracted by computing the second derivative of the image gradient. The extracted features are compared to features from previous scans using the Euclidean distance. Using feature correspondence, a relative rotation and translation can be computed, and the operation is evaluated by computing a score for how well the two scans are aligned. Rotation invariance is achieved by orienting image patches along the world z-axis. According to the authors this does not restrict the performance of the method as long as the robot moves on a flat surface. This assumption is however not valid in all environments, e.g. underground mines (Magnusson et al., 2009).

Work on vision based loop closure detection have been presented in (Cummins and Newman, 2008; Cummins and Newman, 2009), with detection rates of up to 37% and 48% at 0% false alarm for the City Centre and the New College datasets (Smith et al., 2009), respectively. The authors show results for very large data sets (1000km), and also present interesting methods to handle occlusion, a problem that is often present in dynamic environments. The work is extended via inclusion of a laser range sensor in (Paul and Newman, 2010), and the detection rate for the New College data set is increased to 74%. Another vision based loop closure detection method is suggested in (Callmer et al., 2008). SURF features are extracted from images, and classified as words using Tree-of-Words. A spatial constraint is imposed by checking nearest neighbours for each word in the images. A similar approach using visual words for monocular SLAM is taken in (Eade and Drummond, 2008), however the vocabulary is built online in contrast to offline as in (Cummins and Newman, 2008; Cummins and Newman, 2009; Callmer et al., 2008). In a Graph-SLAM system, loops are closed when new edges are created. A SLAM system inspired by rodents is presented in (Milford and Wyeth, 2008). The authors use a monocular camera to collect data over a 66km trajectory with multiple nested loops. More than 51 loops are closed, with no false loops, however there is no explicit loop closure detection. A topological mapping method where loop closure is detected via strong geometrical constraints for stereo images is presented in (Konolige et al., 2010). Another topological method using vision is the work by (Tapus and Siegwart, 2006). It should be noted that it is difficult to compare results from different types of sensors.

A classification approach based on point cloud features and AdaBoost has been used for people detection in cluttered office environments (Arras et al., 2007) and indoor place recognition (Mozos et al., 2005). For people detection the point clouds were segmented and each segment classified as either belonging to a pair of legs or not. Detection rates of over 90% were achieved. For place recognition multiple classes (> 2) are generally used. For this reason the results do not easily compare to the present loop closure detection problem, which has two classes (either same place or not).

3 Loop closure detection

Loop closure detection can be seen as a place recognition problem - it consists of detecting that the robot has previously visited the current location. The problem is central to SLAM, as it allows the estimated map and robot location to be refined. This section presents the suggested loop closure detection algorithm. Here, we pose the loop closure problem as being the problem of determining whether or not two point clouds are from the same location or not. A mobile robot equipped with a 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

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

where N is the number of points in the cloud and D is the dimensionality of the data, here $\mathsf{D} = 2$ or $\mathsf{D} = 3$. The points are given in Cartesian coordinates

$${}^{\mathsf{c}}p_i^k = \begin{cases} \begin{bmatrix} \mathsf{x}_i^k & \mathsf{y}_i^k \end{bmatrix}^{\mathrm{T}}, & \text{if } \mathsf{D} = 2 \\ \begin{bmatrix} \mathsf{x}_i^k & \mathsf{y}_i^k & \mathsf{z}_i^k \end{bmatrix}^{\mathrm{T}}, & \text{if } \mathsf{D} = 3 \end{cases} \tag{2}$$

but can of course be converted into polar/spherical coordinates

$${}^{\mathsf{p}}p_i^k = \begin{cases} \begin{bmatrix} r_i^k & \varphi_i^k \end{bmatrix}^{\mathrm{T}}, & \text{if } \mathsf{D} = 2 \\ \begin{bmatrix} r_i^k & \varphi_i^k & \psi_i^k \end{bmatrix}^{\mathrm{T}}, & \text{if } \mathsf{D} = 3 \end{cases} \tag{3}$$

using the appropriate Cartesian to polar/spherical transformation. Here r, φ and ψ is range, horizontal angle and vertical angle, respectively. For simplicity, time index k and the differentiation between coordinate systems, i.e. c and p in Equations (2) and (3), is dropped in the remainder of the paper. In Appendix A, where the features are defined, it will be clear from context which coordinate system that is intended.

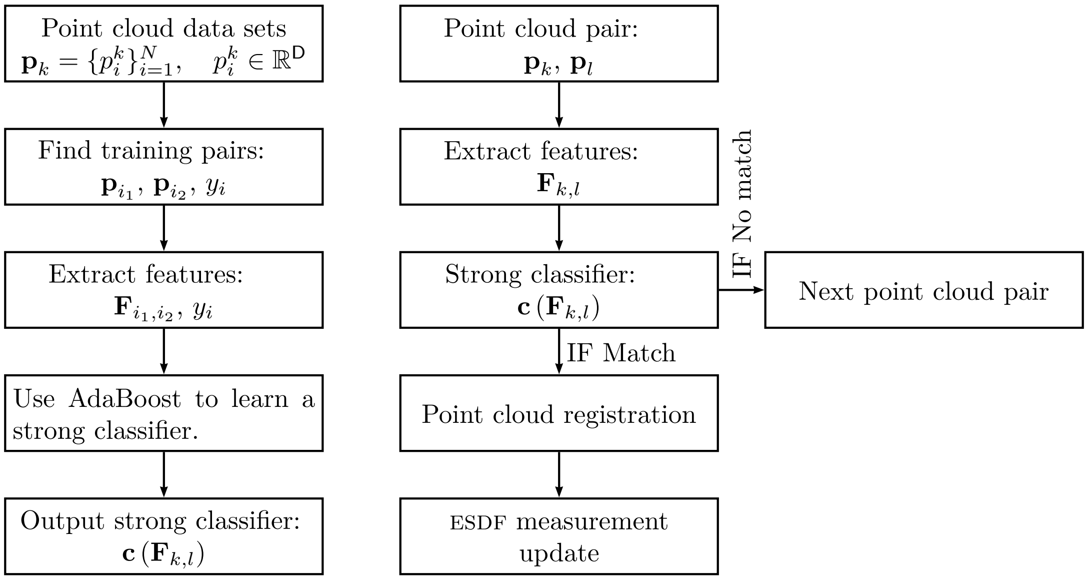
After moving in a loop the robot arrives at a previously visited location, and the two point clouds, acquired at different times, should resemble each other. A comparison of the point clouds is performed in order to determine if a loop closure has occurred or not. To facilitate this comparison, two types 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 the experiments.

3.1 Algorithm overview

Our loop detection algorithm uses the same principle as in other loop detection approaches, i.e. pairwise comparison of data, see e.g. (Bosse and Zlot, 2008; Callmer et al., 2008; Cummins and Newman, 2008; Magnusson et al., 2009; Steder et al., 2010). Each point cloud is described using a large set of rotation invariant features. These features are combined in a non-linear manner using a boosting classifier which outputs the likelihood of the two point clouds being from the same location.

There are two main parts to the algorithm, the first is the learning phase where a classifier is learned from training data. The second part is the classification phase, where the learned classifier is used to classify pairs of point clouds in SLAM experiments. A diagram of the algorithm is given in Figure 2. In the learning phase (left), pairs of point clouds with corresponding class labels y are found in point cloud data sets. From each point cloud features are computed. Examples of the features employed are mean range, area (in 2D) or volume (in 3D), distance etc. A detailed description of the features is presented in subsection 3.2 and Appendix A. The difference between the features from point clouds k and l is called the set of extracted features, and is denoted $\mathbf{F}_{k,l}$. The set of extracted features with corresponding class labels are input to AdaBoost (Freund and Schapire, 1995), a machine learning procedure which greedily builds a strong classifier $\mathbf{c}(\mathbf{F}_{k,l})$ by a linear combination of simple, so called weak, classifiers. When the weak classifiers are combined into a strong classifier, the resulting decision boundary is non-linear. The same strategy has been employed for face detection in (Viola and Jones, 2004).

In the classification phase of our algorithm, Figure 2 (right), the learned classifier is used to detect loop closure in SLAM experiments. The SLAM framework used here is trajectory based, meaning that the state vector contains a history of previous poses. The particular SLAM framework is described in Section 4.



Point cloud data sets $\mathbf{p}_k = \{p_i^k\}_{i=1}^N, \quad p_i^k \in \mathbb{R}^{\mathsf{D}}$
Find training pairs: $\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, y_i$
Extract features: $\mathbf{F}_{i_1,i_2}, y_i$
Use AdaBoost to learn a strong classifier.
Output strong classifier: $\mathbf{c}(\mathbf{F}_{k,l})$
Point cloud pair: $\mathbf{p}_k, \mathbf{p}_l$
Extract features: $\mathbf{F}_{k,l}$
Strong classifier: $\mathbf{c}(\mathbf{F}_{k,l})$
IF No match
Next point cloud pair
IF Match
Point cloud registration
ESDF measurement update

Figure 2: Algorithm overview. Left diagram, learning phase. Right diagram, classification phase. In the learning phase, one or more point cloud data sets are used to learn a strong classifier. The learning phase is run first in our algorithm. In the classification phase the strong classifier is used to detect loop closure in SLAM experiments. In our SLAM implementation, the classification phase is run each time a new pose has been added to the state vector.

3.2 Features

The main reason for working with features is the ability to compress the information in point clouds by defining meaningful statistics describing shape and other properties – working with n_f features is easier (e.g. requires less memory and is less computationally expensive) than working with the full point clouds since $n_f \ll N$. In this work, two types of features f_j are used. The first type is a function that takes a point cloud as input and returns a real number. Typically, features that represent geometrical or statistical 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 $\mathbf{f}_k \in \mathbb{R}^{n_f^1}$, where k again refers to the time t_k when the point cloud was acquired. Here, n_f^1 is the number of features of the first type. The second type of feature used is a range histogram with bin size b_j. In total, n_f^2 histograms are computed, giving a total of $n_f^1 + n_f^2 = n_f$ features.

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

$$\mathbf{F}^1_{k,l} = |\mathbf{f}_k - \mathbf{f}_l| \,. \tag{4}$$

The underlying idea here is that point clouds acquired at the same location will have similar feature values $\mathbf{f}_k$ and $\mathbf{f}_l$, and hence each element of $\mathbf{F}_{k,l}$ 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 should 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

$$\mathbf{F}_{k,l} = \left[\mathbf{F}^1_{k,l}, \mathbf{F}^2_{k,l}\right] . \tag{5}$$

$\mathbf{F}_{k,l}$ is referred to as the set of extracted features for two point clouds indexed k and l.

In (Granström et al., 2009) 20 features are used in 2D, in (Granström and Schön, 2010) those features are extended to 3D and augmented with new features. Some of the features used here are the same regardless of dimension, e.g. mean range, while other features are generalised, e.g. from area in 2D to volume in 3D. Similar 2D features can be found in (Mozos et al., 2005; Arras et al., 2007; Brunskill et al., 2007). In total, $n_f = 44$ features are used in 2D and $n_f = 41$ features are used in 3D. For formal definitions, see Appendix A.

3.3 Classification using AdaBoost

Boosting is a machine learning method for finding combinations of simple base classifiers in order to produce a form of committee whose performance can be significantly better than any one of the base classifiers used alone. The simple base classifiers need to be just slightly better than a random guess, thus they are called weak classifiers, see e.g. (Bishop, 2006). The resulting combination is better than the best individual weak classifier, and analogously the resulting classifier is thus called strong. Each weak classifier is learned using a weighted form of the data set, where the weighting of each data point depends on the performance of the previous weak classifiers.

A widely used form of boosting is AdaBoost, which constructs a strong classifier by a linear combination of weak classifiers (Freund and Schapire, 1995). When the weak classifiers are combined into a strong classifier, the resulting decision boundary is non-linear. As more weak classifiers are added, the classification error on the training data converges towards zero, and eventually becomes zero. Although this might be interpreted as overfitting, AdaBoost has been shown to generalise well on testing data (Freund and Schapire, 1995).

Although later generalised to multiple classes, AdaBoost was originally designed for problems with two classes. Here, the two classes are called positive and negative, or p and n, respectively. The positive class consists of point cloud pairs from the same location, the negative class consists of point cloud pairs from different locations. As input to the AdaBoost learning algorithm, n hand-labeled training data pairs are provided,

$$\left(\mathbf{F}_{1_1,1_2}, y_1\right), \ldots, \left(\mathbf{F}_{i_1,i_2}, y_i\right), \ldots, \left(\mathbf{F}_{n_1,n_2}, y_n\right), \tag{6}$$

where each data point $\mathbf{F}_{i_1,i_2}$ has a corresponding class label y_i. Let $\mathbf{F}_i$ be a compact way of writing $\mathbf{F}_{i_1,i_2}$. To learn a classifier using AdaBoost, data points from each class are needed. Let N_p and N_n be the number of training data points belonging to p and n, respectively, i.e. $n = N_\text{n} + N_\text{p}$. The data labels in the two class problem are defined as

$$y_i = \begin{cases} 1 & \text{if } \mathbf{F}_i \in \text{p} \\ 0 & \text{if } \mathbf{F}_i \in \text{n} \end{cases} \tag{7}$$

In the AdaBoost algorithm, each data pair $(\mathbf{F}_i, y_i)$ is given a weight W^i_t, where t denotes the specific iteration of the algorithm. The weights are initialised as $W^i_1 = \frac{1}{2N_\text{n}}$ if $y_i = 0$, or $W^i_1 = \frac{1}{2N_\text{p}}$ if $y_i = 1$. This initialisation ensures that each class is given half the weight of the data, and all data pairs within a class are given an equal weight.

After initialisation, AdaBoost iteratively adds weak classifiers to a set of previously added weak classifiers. The weak classifiers used here are decision stumps, i.e. one node decision trees, defined as

$$c(\mathbf{F}_i, \theta) = \begin{cases} 1 & \text{if } p\mathbf{F}_i^{(f)} < p\lambda \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

with parameters $\theta = \{f, p, \lambda\}$, where $\mathbf{F}_i^{(f)}$ is the selected component f of $\mathbf{F}_i$, p is the polarity ($p = \pm 1$), and $\lambda \in \mathbb{R}$ is a threshold. The result of a weak classifier (8) is that the input space is partitioned into two half spaces, separated by an affine decision boundary which is parallel to one of the input axes.

In each iteration, the weak classifier that minimises the weigthed classification error with respect to θ is chosen. Given the parameters of the best weak classifier, the training data is classified and the weights of the mis-classified data are increased (or, conversely, the weights of the correctly classified data are decreased). Further, using the classification error a weight α_t is computed for the best weak classifier. Details on how the weights are computed are given below.

This procedure is repeated until T weak classifiers have been computed. Weak classifiers can be added several times in each dimension of $\mathbb{R}^{n_f}$, each time with a new polarity and threshold, i.e. same f and new p and λ. The weighted combination of T weak classifier together create the strong classifier. A detailed presentation of AdaBoost is given in Algorithm 1.

In this work, to find the best weak classifier, we employ a similar technique as is used in (Viola and Jones, 2004). The search for the best weak classifier is summarised in Algortithm 2. In our implementation, we search over all features each time we find the best weak classifier. With just over 40 features, doing so does not pose a significant complexity issue. However, if the number of features were in the order of thousands, as in (Viola and Jones, 2004), searching over all features could prove to be very time consuming.

4 Simultaneous localisation and mapping

In this section we briefly outline the SLAM framework used for testing the new method in detecting loop closure. The algorithm is well known and not part of our main contribution, hence we only provide the specific design choices made and refer to the relevant references for the exact implementation details.

4.1 Exactly Sparse Delayed-state Filters

The Exactly Sparse Delayed-state Filter (ESDF), a delayed state extended information filter, maintains a delayed state vector containing the poses where point clouds were acquired. The state vector is augmented with a new pose when a new point cloud is acquired. The state information matrix is sparse without approximation, which results in an estimation comparable to the full covariance matrix solution. Using sparse solutions, like SEIF (Thrun et al., 2004), has been shown to be inconsistent (Eustice et al., 2005). Using the ESDF, prediction and update can be performed in constant time regardless of the information matrix size. Refer to (Eustice et al., 2006) for details on the implementation.

4.2 Robot pose, process and measurement models

In this paper we use the coordinate frame notation introduced by Smith, Self and Cheeseman (Smith et al., 1990) to handle the robot pose, process model and measurement model. Let $\mathbf{x}_{i,j}$ denote the location of coordinate frame j with respect to coordinate frame i. In 2D, $\mathbf{x}_{i,j}$ is a 3-DOF pose consisting of $(\mathsf{x}_{i,j}, \mathsf{y}_{i,j})$-position and heading angle $\psi_{i,j}$. In 3D, $\mathbf{x}_{i,j}$ is a 6-DOF pose consisting of $(\mathsf{x}_{i,j}, \mathsf{y}_{i,j}, \mathsf{z}_{i,j})$-position and Euler angles $(\phi_{i,j}, \theta_{i,j}, \psi_{i,j})$ representing roll, pitch and heading angles. Here, the roll, pitch and heading definitions from (Eustice, 2005) are used. This Euler representation is singular at pitch $\theta = \pm 90°$, however ground robots

```
Algorithm 1 AdaBoost
Input: $(\mathbf{F}_1, y_1), \ldots, (\mathbf{F}_n, y_n)$
Initialize weights: $W_1^i = \frac{1}{2N_\mathrm{n}}$ if $y_i = 0$, $W_1^i = \frac{1}{2N_\mathrm{p}}$ if $y_i = 1$
1: for $t = 1, \ldots, T$ do
2:     Normalise the weights:
           $$\widetilde{W}_t^i = \frac{W_t^i}{\sum_{j=1}^{N_\mathrm{n}+N_\mathrm{p}} W_t^j}, \quad i = 1, \ldots, N_\mathrm{n} + N_\mathrm{p} \qquad (9)$$
3:     Select the best weak classifier, i.e. the one that minimizes the weighted error,
           $$\theta_t = \arg\min_\theta \sum_{i=1}^{n} \widetilde{W}_t^i \left| c(\mathbf{F}_i, \theta) - y \right| \qquad (10)$$
       where $\theta = \{f, p, \lambda\}$.
4:     Define $c_t(\mathbf{F}_i) = c(\mathbf{F}_i, \theta_t)$, and let $\varepsilon_t$ be the corresponding weighted error.
5:     Update the weights:
           $$W_{t+1}^i = \widetilde{W}_t^i \beta_t^{1-e_i}, \qquad (11)$$
       where $e_i = 0$ if $\mathbf{F}_i$ is classified correctly and $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
6: end for
The strong classifier is:
           $$\mathbf{c}(\mathbf{F}_{k,l}) = \frac{\sum_{t=1}^{T} \alpha_t c_t(\mathbf{F}_{k,l})}{\sum_{t=1}^{T} \alpha_t} \in [0, 1] \qquad (12)$$
where $\alpha_t = \log \frac{1}{\beta_t}$. The closer $\mathbf{c}(\mathbf{F}_{k,l})$ is to one, the higher the loop likelihood is. To obtain a binary decision, i.e. loop or no loop, the classification likelihood can be thresholded using a threshold
           $$K \in [0, 1]. \qquad (13)$$
Output: $\mathbf{c}(\mathbf{F}_{k,l})$
```

rarely operate at such configurations and the singularity has not been any problem in our SLAM experiments. It can be noted that an alternative angle representation that does not suffer from singularities could have been used, e.g. axis angle or quaternions. Using $\mathbf{x}_{k,l}$ and $\mathbf{x}_{l,m}$, the location of coordinate frame m with respect to coordinate frame k can be expressed using the compounding operator $\oplus$ introduced in (Smith et al., 1990) as

$$\mathbf{x}_{k,m} = \mathbf{x}_{k,l} \oplus \mathbf{x}_{l,m}. \tag{18}$$

Using the inverse compounding operator $\ominus$ from (Smith et al., 1990), the location of coordinate frame k with respect to coordinate frame l is expressed as

$$\mathbf{x}_{l,k} = \ominus \mathbf{x}_{k,l}. \tag{19}$$

Formal mathematical definitions of the compounding operators $\oplus$ and $\ominus$ in 2D and 3D can be found in Appendix B. Subsequently, if the locations of coordinate frames l and m with respect to coordinate frame k are known, the location of m with respect to l is expressed as

$$\mathbf{x}_{l,m} = \ominus \mathbf{x}_{k,l} \oplus \mathbf{x}_{k,m}. \tag{20}$$

Note, that since each $\mathbf{x}$ consists of a position and a heading, the compounding operator is just a short hand representation for combinations of rigid body transformations. In our SLAM experiments the pose

$$\mathbf{x}_{0,k} = \begin{cases} [\mathsf{x}_{0,k} \ \mathsf{y}_{0,k} \ \psi_{0,k} \ ]^T & \text{in 2D} \\ [\mathsf{x}_{0,k} \ \mathsf{y}_{0,k} \ \mathsf{z}_{0,k} \ \phi_{0,k} \ \theta_{0,k} \ \psi_{0,k} \ ]^T & \text{in 3D} \end{cases} \tag{21}$$

```
Algorithm 2 Find the best weak classifier
Input: $(\mathbf{F}_1, y_1), \ldots, (\mathbf{F}_n, y_n)$ with corresponding weights $W_t^1, \ldots, W_t^n$.
Initialise: $T_- = \sum_{i:y_i=0} W_t^i$ and $T_+ = \sum_{i:y_i=1} W_t^i$.
1: for $d = 1, \ldots, n_f$ do
2:     Sort the data in the current feature dimension $d$ in ascending order, and let $i^1, \ldots, i^n$ be the corresponding index, i.e. $\mathbf{F}_{i^1}^{(d)} \leq \mathbf{F}_{i^2}^{(d)} \leq \ldots \leq \mathbf{F}_{i^n}^{(d)}$.
3:     Compute the cummulative sum of weights for each class,
           $S_-^j = \sum_{k=1}^{j} (1 - y_{i^k}) W_t^{i^k} \quad j = 1, \ldots, n$    (14a)
           $S_+^j = \sum_{k=1}^{j} y_{i^k} W_t^{i^k} \quad j = 1, \ldots, n$    (14b)
4:     Compute errors $\varepsilon_1^j = S_+^j + T_- - S_-^j$ and $\varepsilon_2^j = S_-^j + T_+ - S_+^j$.
5:     Find the minimum error, $\varepsilon_d = \min_{j,k} \varepsilon_k^j$. Let $\ell, \gamma$ be the minimising arguments, i.e. $\varepsilon_d = \varepsilon_\gamma^\ell$.
6:     Compute the threshold,
           $\lambda_d = \frac{\mathbf{F}_{i^\ell}^{(d)} + \mathbf{F}_{i^{\ell+1}}^{(d)}}{2}$    (15)
7:     Compute polarity,
           $p_d = \begin{cases} -1 & \text{if } \gamma = 1 \\ 1 & \text{if } \gamma = 2 \end{cases}$    (16)
8: end for
Find the feature dimension with lowest error,
           $f = \arg\min_d \varepsilon_d$    (17)
and set $p = p_f$ and $\lambda = \lambda_f$. The optimal parameters, and corresponding error, are $\theta_t = \{f, p, \lambda\}$ and $\varepsilon_t = \varepsilon_f$.
Output: $\theta_t$ and $\varepsilon_t$
```

is the location of point cloud k's local coordinate frame in the global coordinate frame 0. Both process and measurement model are defined as coordinate frame operations using the compounding operator. The process, or motion, model is

$$\mathbf{x}_{0,k+1} = f\left(\mathbf{x}_{0,k}, \mathbf{x}_{k,k+1}\right) + \mathbf{w}_{k+1} = \mathbf{x}_{0,k} \oplus \mathbf{x}_{k,k+1} + \mathbf{w}_{k+1}, \tag{22}$$

where $\mathbf{x}_{k,k+1}$ is computed using point cloud registration, Section 4.3, and $\mathbf{w}_{k+1}$ is a white gaussian process noise. After a loop closure has been detected between point clouds m and n, the corresponding relative pose $\mathbf{x}_{m,n}$ is computed using the measurement model, defined as

$$\mathbf{x}_{m,n} = h\left(\mathbf{x}_{0,m}, \mathbf{x}_{0,n}\right) + \mathbf{e}_{m,n} = \ominus\mathbf{x}_{0,m} \oplus \mathbf{x}_{0,n} + \mathbf{e}_{m,n} = \mathbf{x}_{m,0} \oplus \mathbf{x}_{0,n} + \mathbf{e}_{m,n}, \tag{23}$$

where $\mathbf{e}_{m,n}$ is white Gaussian measurement noise.

4.3 Point cloud registration

Point cloud registration, also referred to as scan matching, is the process of finding a rigid body transformation (rotation and translation) that aligns two point clouds to each other. Typically, this is performed by minimising a cost function, e.g. the sum of distances to nearest neighbour points. There are a number of different methods proposed in the literature, in this work we have used four different methods; the well known Iterative Closest Point (ICP) (Besl and McKay, 1992; Chen and Medioni, 1992; Zhang, 1994), 3D NDT

(Magnusson et al., 2007), CRF-Match (Ramos et al., 2007a) and an implementation of the histogram based method by (Bosse and Zlot, 2008).

In 2D, we use ICP to compute the vehicle motion, i.e. to compute $\mathbf{x}_{k,k+1}$ in (22). After loop closure has been detected, we use either the histogram method or CRF-Match to find an initial point cloud registration, which is then refined using ICP.

In 3D we use 3D NDT, initialised by odometry to compute vehicle motion. We have performed a SLAM experiment on a publicly available indoor data set, and for this data the consecutive relative poses are available together with the point clouds. After loop closure detection, we use ICP to compute the relative pose. Here, ICP is initialised with the relative pose estimate obtained from the ESDF state vector. While this method works well for the particular SLAM experiment presented here, in a general SLAM solution a point cloud registration method that does not rely on a good initial guess would be needed.

5 Experimental results

This section presents the results from the experiments performed. We examine the proposed method by evaluating the strong classifiers properties, and by doing SLAM experiments in both 2D and 3D. The classifier is evaluated in terms of detection rate (D), missed detection rate (MD) and false alarm rate (FA). The rates are defined as

$$\begin{aligned} D &= \frac{\#\text{ positive data pairs classified as positive}}{\#\text{ positive data pairs}}, \\ MD &= \frac{\#\text{ positive data pairs classified as negative}}{\#\text{ positive data pairs}}, \\ FA &= \frac{\#\text{ negative data pairs classified as positive}}{\#\text{ negative data pairs}}. \end{aligned}$$

These rates are important characteristics for any classification or detection problem, and typically it is difficult to achieve low MD and low FA simultaneously. Instead, a choice has to be made as to which error is more important to minimise. For the loop closing problem, we argue that the main concern is minimising FA, while keeping MD as low as possible. A relevant question is then how low FA should be, since lowering FA further comes at the price of higher MD.

In previous work, D has been reported at 1% FA (Bosse and Zlot, 2008), in other work D has been reported at 0% FA (or equivalently at 100% precision) (Magnusson et al., 2009; Cummins and Newman, 2008), yet others report D at 0.01% FA (Callmer et al., 2008). 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 (Bosse and Zlot, 2008), or using a cascade of several methods (Bosse and Zlot, 2009a). 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 typically sufficient to produce good results (Magnusson et al., 2009; Bosse and Zlot, 2008; Cummins and Newman, 2008; Cummins and Newman, 2009). Therefore, the detection rate at 0% false alarm is more robust. However, for completeness and ease of comparison, results at both 0% and 1% false alarm are presented.

The experiments in Section 5.2 were conducted using k-fold cross validation on the data sets. 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 different permutation of the data. The results presented are the mean of the cross validations. The data used in experiments is presented in Section 5.1. After evaluating the 2D and 3D classifiers, Section 5.2, the classifiers are tested in SLAM experiments which are presented in Section 5.3. The experiments are compared to the estimated SLAM trajectories with the dead reckoning sensors, and with GPS when it is available. The resulting SLAM maps are also shown, overlaid on aerial photographs in the outdoor cases. The results are summarised, and a comparison to related work is given, in Section 5.4.

5.1 Data

In this subsection we describe the data used in the 2D and 3D experiments. Three of the six data sets are publicly available, references to the data repositiories are provided. The datasets used for training are divided into two classes, positive and negative. Five of the data sets contain a large quantity of point clouds, thus making it possible to find tens of thousands of training pairs. However, to keep the computational cost tractable, the amount of training pairs were limited.

5.1.1 2D data

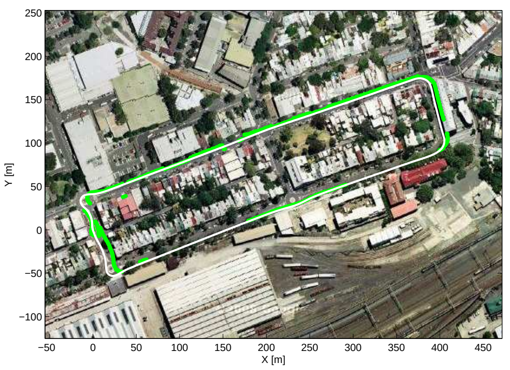
For the 2D experiments, four different datasets were used. The first data set, called kenmore_pradoroof (*ken*), is publicly available (Howard and Roy, 2003)[1]. It has maximum measureable range $r_{\max} = 50$m and horizontal angular resolution $\delta_\varphi = 1°$. The data set is approximately 18km long. The last three data sets all have maximum measureable range $r_{\max} = 50$m and horizontal angular resolution $\delta_\varphi = 0.5°$. Two of them, Sydney 1 (*syd1*) and Sydney 2 (*syd2*), were acquired in a residential and business area close to the University of Sydney, Sydney, Australia. These two datasets are approximately 0.65 and 2 km long. Using an initial set of 50 data pairs for each class, a classifier was learned and used for SLAM experiments using the *ken*, *syd1* and *syd2* data sets. The resulting trajectories are shown in Figure 3.

Using the estimated trajectories, positive and negative data pairs were extracted based on the translational distance between the poses at which the point clouds were acquired. For each of the three datasets, positive pairs were taken as all pairs where the translational distance was less than or equal to 1m, 2m and 3m. Negative pairs were obtained by taking a random subset of remaining data pairs, such that for each translational distance the number of positive N_{p} and negative N_{n} data pairs are equal. The number of data pairs for each dataset and each translational distance is shown in Table 1.

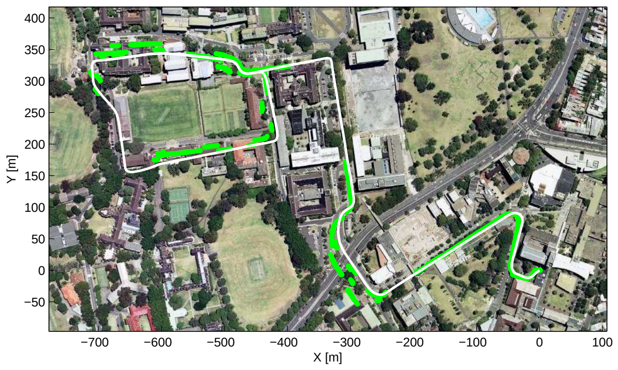
Careful visual examination of the *ken* trajectory in Figure 3c shows parts for which the trajectory estimation was of lower quality. The main reason for this was that our scan registration, an implementation of the histogram method from (Bosse and Zlot, 2008), failed to find the correct rotation and translation when true loops were detected from the opposite direction. Thus several detected loop closures from the opposite direction could not be included in the estimation. Since we want to use the SLAM results for finding training pairs at certain translational distances, we also want to be certain that the translational distance computed from the SLAM results is close to the true translational distance. For this reason, only the first half of the *ken* data set, for which we could estimate the trajectory with higher accuracy (only loops from the same direction), was used to find positive and negative training data. This trajectory is shown in Figure 3d. In addition to being used for finding training data, the *ken* data set is also used to evaluate the classifier's dependence to translation.

The fourth dataset, Sydney 3 (*syd3*), was also collected around the University of Sydney and is approximately 2 km long. This dataset contains point clouds with just a 180° field of view, and was therefore not used for learning the classifier. Instead it was integrated in a SLAM experiment, where GPS was used to collect ground truth data. All four data sets were collected using planar SICK laser range sensors. Placing two such sensors "back-to-back" gives a full 360 degree view of the environment. The sensors sweep from right to left, thus introducing an order for the range measurement. Some of the features defined in Appendix A use this ordering of the points when the feature value is computed.

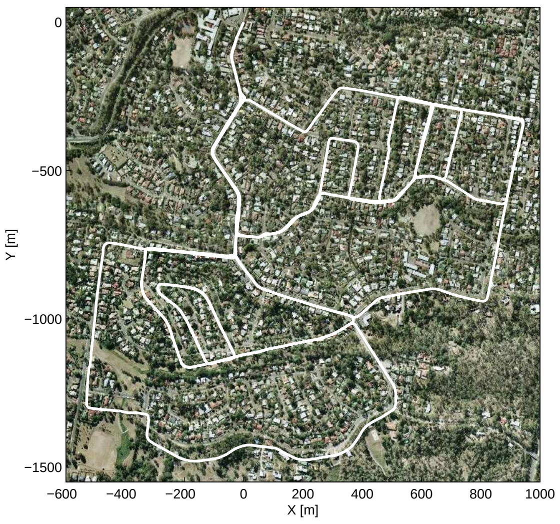
[1]Thanks to Michael Bosse, Autonomous Systems Laboratory, CSIRO ICT Centre, Australia, for providing the data set.



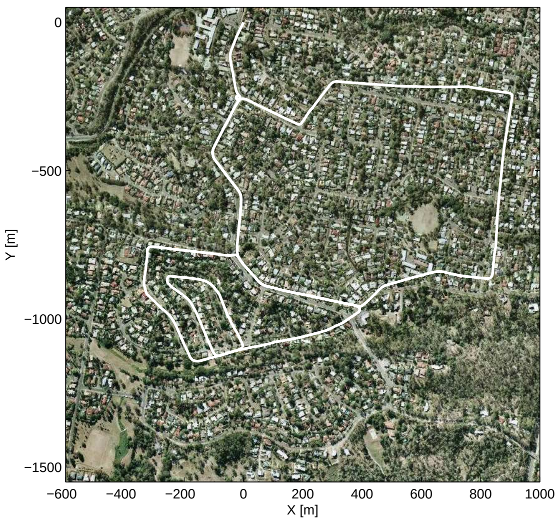
(a) SLAM results for *syd1*.



(b) SLAM results for *syd2*.



(c) SLAM results for *ken*.



(d) SLAM results for the first half *ken*.

Figure 3: Estimated SLAM trajectories (white) overlaid on Google maps images, and compared to GPS (green) when available. Note that the GPS signal becomes increasingly unreliable under trees and tall buildings. The SLAM trajectories were used to find point cloud pairs for training and evaluation of the classifier.

5.1.2 3D data

In the 3D experiments, two datasets were used, both are publicly available (Nüchter and Lingemann, 2009)[2]. 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 measurable range of 30m. From this dataset 3130 positive data pairs (point clouds from the same location) and 7190 negative data pairs (point clouds from different locations) were selected. The positive data pairs were chosen as the scan pairs taken less than 3m apart (Magnusson et al., 2009). The negative data were chosen as a random subset of the remaining data pairs, i.e. those more than 3m apart.

The second dataset, 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 measurable range of 15m. From this dataset 16 positive and 324 negative data pairs are taken. The positive data pairs are those taken less than 1m apart (Magnusson et al., 2009), the negative data pairs are a random subset of

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

Table 1: Number of 2D point cloud pairs, at various translational distances. The point cloud pairs were used for training and evaluation of the classifier. N_{p} and N_{n} are the number of positive and negative point cloud pairs, respectively.

Data set	Dist. [m]	$N_{\mathrm{p}},\ N_{\mathrm{n}}$
ken	1	1321
	2	4242
	3	7151
syd1	1	31
	2	568
	3	956
syd2	1	286
	2	2039
	3	3570

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 of one class, called unbalanced data, is discussed further in the paper.

Both 3D 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 can be ordered according to the order in which the points are measured by the sensor setup. Some of the features defined in Appendix A use this ordering of the points when the feature value is computed.

5.2 Classifier evaluation

In this section we evaluate the classifiers learned previously. An important aspect of any robotics application is computational complexity. If a method is to be implemented on a robot, it is important that it can be computed in real time in order to not significantly delay the robot's functionality. The computational times for different parts of the suggested method for loop closure detection are presented here. When learning a classifier, an initial important step is to determine an appropriate number of training rounds T (cf. Algorithm 1) for the classifier. Training should proceed as long as the validation error decreases, but not longer to avoid overfitting and to keep computational complexity low. Another important aspect is which features are the most beneficial to the final strong classifier. This is verified in two ways: 1) by considering which features are selected in early training iterations, and, 2) by removing features from the training data and checking how they affect the classifier's performance. The strong classifier's receiver operating characteristics (ROC) are evaluated, and a comparison is made between 2D and 3D performance by downsampling 3D data to 2D. The classifiers dependence to translation is also evaluated, as well as how it handles dynamic objects. Finally, the difficulty posed by repetitive structures in the environment is addressed.

5.2.1 Computational complexity

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 is not optimized for speed. The timing results are presented in Table 2. The times to compute the features are averages over all point clouds in the data sets *syd2*, *hann2* and *AASS*. As expected the time increases with the number of points in each cloud. Computing the features only needs to be performed once per point cloud in a SLAM experiment. Since comparing the features and computing $\mathbf{c}\left(\mathbf{F}_{k,l}\right)$ are the same operations in both 2D and 3D, the presented times are averages over the training pairs for all 2D and 3D data sets. Comparing features

Table 2: Execution time of loop closure detection classifier, all times in ms. The times to compute the features are shown separately for 2D and 3D data, respectively. Comparing the features is a procedure that is equal in 2D and 3D, thus the time to compare the features is the same in 2D and 3D.

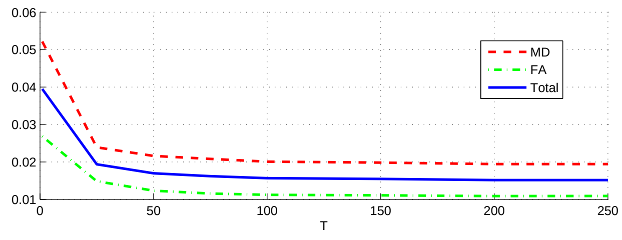
Compute features	*syd2*	*hann2*	*AASS*
Type 1	14.35	15.96	206.11
Type 2	0.22	3.38	18.99
Total	14.57	19.34	225.10
Total per point	20.2×10^{-3}	1.17×10^{-3}	2.00×10^{-3}

Compare features	Time
Type 1	6.58×10^{-3}
Type 2	0.824
Total	0.831
Compute $\mathbf{c}(\mathbf{F}_{k,l})$	0.78

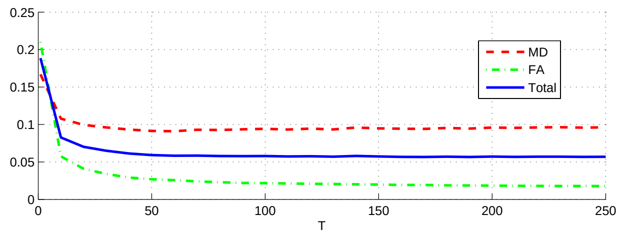
and classifying a set of extracted features takes about 1.6ms when $T = 50$ weak classifiers are used. Training a strong classifier for $T = 50$ iterations takes 15s when about 10000 data pairs are used.

5.2.2 Number of training rounds T

Strong classifiers were trained for different values of T, the resulting error rates are shown in Figure 4. The total error rate is the ratio between the number of misclassified data pairs and the total number of data pairs. As can be seen in Figure 4, the validation error levels decrease as the learning algorithm iterates up until about 50 training iterations, when the validation error levels stop decreasing. Hence, $T = 50$ was chosen for all subsequent experiments in both 2D and 3D.



(a) 2D results.



(b) 3D results.

Figure 4: Error rates during AdaBoost training plotted against the number of training rounds T. For both 2D and 3D data the validation error levels stop decreasing after about 50 training rounds, hence $T = 50$ is an appropriate choice for the subsequent experiments.

5.2.3 Most informative features

An interesting aspect of the suggested solution to the loop closure detection problem is which features are the most informative for classification. In each training iteration of the learning phase, the weak classifier that best improves performance is selected. Each feature can be chosen multiple times, each time with a new polarity and threshold. The features that are chosen in early training iterations will have a larger weight than features chosen in later training iterations. Therefore, we considered the features chosen in the first five training rounds in this analysis. To further examine the importance of the features, strong classifiers were learned from the training data after removing the features individually. The best features are those that negatively affected the validation error rate the most when removed. Results for the 2D data are presented in Table 3, 3D results are presented in Table 4. In the tables, Test 1 shows which features were chosen in the first five training rounds, Test 2 indicates the features whose removal resulted in the highest validation error rates.

Results for the 2D data are in Table 3. Both tests suggest that for *ken*, features 4 (average of all ranges), 44 (range histogram with bin size 3m) and 22 (range kurtosis for all ranges) are most informative. For *syd1*, feature 23 (mean relative range) appears in both tests as an important feature, and for *syd2* feature 34 (mean group size) appears in both tests as important for loop closure detection. For both Sydney datasets, Test 1 suggests that feature 38 is best for loop closure detection. This feature corresponds to range histograms with bin size 0.5m. For *ken* and *syd1*, Test 2 suggests that feature 21 (range kurtosis excluding maximum ranges) is most informative. The difference in total error is quite small for the five best features (Table 3, Test 2), however the results do suggest that the best five features are different for the three data sets. As all three data sets were acquired in rather similar suburban environments, see Figure 3, this raises the important question of whether the presented loop closure detection method generalises between data sets. This question is addressed further in Section 5.3.2.

Table 3: Most informative features for loop closure detection using 2D data. The feature numbers correspond to the list numbers in Appendix A. Test 1 shows which features were added in the first five training rounds. Test 2 shows the resulting validation error when the features were removed from the training data before learning the classifier.

Test 1					
Training Round	1	2	3	4	5
Added Feature, *ken*	4	44	22	12	19
Added Feature, *syd1*	38	23	17	40	4
Added Feature, *syd2*	38	22	34	7	35
Test 2					
Feature removed, *ken*	21	22	4	44	35
Total error [%]	2.98	2.94	2.92	2.83	2.82
Feature removed, *syd1*	21	10	17	23	27
Total error [%]	0.36	0.31	0.30	0.30	0.30
Feature removed, *syd2*	34	6	43	42	29
Total error [%]	0.38	0.36	0.36	0.35	0.34

Results for the 3D data are presented in Table 4. As can be seen, 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. As with the 2D data, the results from Test 2 suggest that two different sets of five features are best at detecting loop closure. Comparing to the 2D results in Table 3, the difference in total error is larger for the 3D data.

Furthermore, considering all results in Tables 3 and 4 together shows that the most important features for loop closure detection are not the same for either the 2D data sets or the 3D data sets. As mentioned above, an important and immediate question raised by this is whether or not the method is able to generalise between environments (i.e. between data sets). In e.g. 3D, *hann2* is an outdoor data set and *AASS* is an indoor data set, suggesting that the learned classifier might not generalise from outdoor to indoor. Again, this issue is adressed further in Section 5.3.2, where it is shown that the classifer does in fact generalise well between different environments and different sensor setups.

5.2.4 Classifier performance

In this section we present the performance of the classifiers in terms of D and FA, as defined above. Figure 5 shows ROC curves for the classifier in 2D and 3D. Good levels of detection are achieved for all levels of false alarm for the data used here. Table 5 shows a summary of the results achieved compared to related work.

Table 4: Most informative features for loop closure detection using 3D data. The feature numbers correspond to the list numbers in Appendix A. Test 1 shows which features were added in the first five training rounds. Test 2 shows the resulting validation error when the features were removed from the training data before learning the classifier.

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					
Feature removed, *hann2*	21	8	10	28	35
Total error [%]	1.29	1.15	1.14	1.13	1.13
Feature removed, *AASS*	41	22	33	32	40
Total error [%]	2.27	2.24	2.16	2.08	2.04

For the 3D results and the 2D *ken* results, the same data sets were used in the experiments. The two Sydney data sets have not been used in any previous work.

For the 2D classifier, the performance for different translational distances can be compared for all three data sets. In general, performance degrades with increasing translational distance, however even at 3m distance detection rates are sufficient for reliable SLAM.

As seen in Table 5 and Figure 5, for *syd1* the classifier characteristics are worse for 1m distance than for 2m distance. Further, 0% was the lowest D for 0% FA for *AASS*. This occurred in 5 out of 10000 cross validations. Furthermore, the mean D is lower than (Magnusson et al., 2009) and the standard deviation of D is higher than for *hann2*. For these two data sets, the number of training data pairs is small and unbalanced (31+31 and 16+324, respectively), 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 data 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 from the subset of data. Out of 1000 such random subsets, 30 resulted in 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 small 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 the green curve in Figure 5d. If compared to the curve for the full *hann2* data set, it shows a clear negative effect.

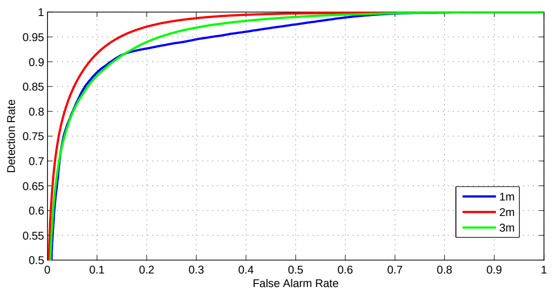
5.2.5 Comparison of 2D and 3D performance

This section presents a quantitative comparison of the performance of the classifier in 2D and 3D. Intuitively, performance in the 3D case should be considerably better than in 2D, since the added dimension and larger quantity of points significantly increases the information content of the point cloud. To obtain 2D data which is comparable to 3D data, 2D point clouds were extracted from the 3D data set *hann2* by taking all points which were located 1m $\pm$ 15cm above the ground plane. The z-components were removed, i.e. the points were projected into the 1m-above-ground plane. The process is illustrated in Figure 6a and Figure 6b.

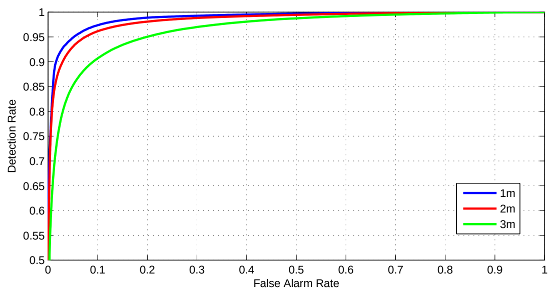
After extracting the 2D point clouds, a classifier was learned and evaluated using the same data pairs as when the 3D data was used. The results are shown in Figure 6c and Table 6. As expected the performance is worsened due to the lower information content in the point clouds. It is difficult to elaborate why the

Table 5: Classification performance, all numbers in %. Results for the presented algorithm are shown in the middle column, results for related work are shown in the right column. The 3D results, and the 2D *ken* results, are for the same data sets as were used in related work. However, the 2D *syd1* and *syd2* data sets have not been used in any related work. Note that (Bosse and Zlot, 2008) detected loops between submaps consisting of laser range scans from tens of meters of travel, while our results are for loop detection between individual laser range scans.

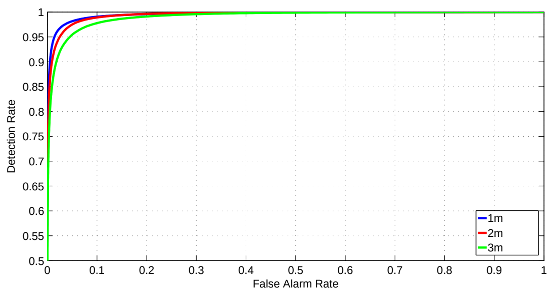
Data set	FA	D	Min/Max	D	Source
hann2	0	63 ± 6	28/76	47	(Magnusson et al., 2009)
				85	(Steder et al., 2010)
	1	99 ± 0.1	98/99	81	(Magnusson et al., 2009)
AASS	0	53 ± 14	0/88	70	”
	1	78 ± 6	56/88	63	”
ken, 1m	0	59.33 ± 11.22	18.48/84.47	N/A	(Bosse and Zlot, 2008)
	1	93.07 ± 0.95	89.77/95.91	51	”
ken, 2m	0	45.11 ± 12.96	3.21/72.00	N/A	”
	1	89.54 ± 0.87	86.42/92.15	51	”
ken, 3m	0	29.79 ± 9.85	3.29/60.22	N/A	”
	1	84.43 ± 0.71	80.25/86.88	51	”
syd1, 1m	0	56.79 ± 22.75	0/93.33		N/A
	1	56.79 ± 22.75	0/93.33		”
syd1, 2m	0	32.61 ± 9.45	4.11/61.61		”
	1	62.47 ± 3.73	45.18/74.82		”
syd1, 3m	0	19.38 ± 7.40	1.76/35.10		”
	1	63.39 ± 2.12	60.67/68.29		”
syd2, 1m	0	66.03 ± 9.03	26.79/84.29		”
	1	81.36 ± 5.16	60.71/90.71		”
syd2, 2m	0	34.75 ± 9.30	7.73/56.55		”
	1	80.94 ± 1.65	75.27/84.88		”
syd2, 3m	0	18.57 ± 7.56	2.35/37.65		”
	1	63.13 ± 2.27	56.33/68.35		”



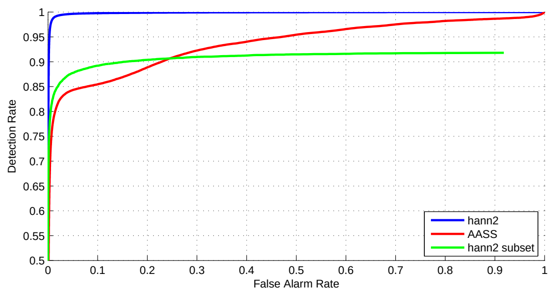
(a) *syd1*



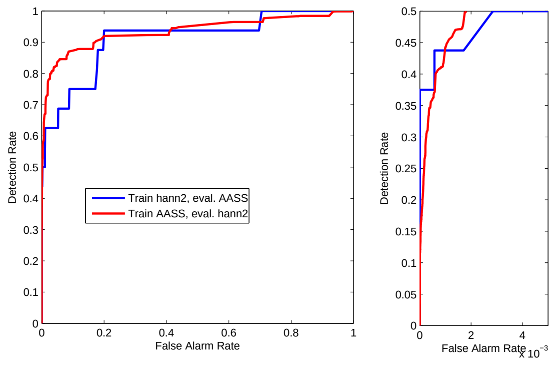
(b) *syd2*



(c) *ken*



(d) *hann2* and *AASS*.

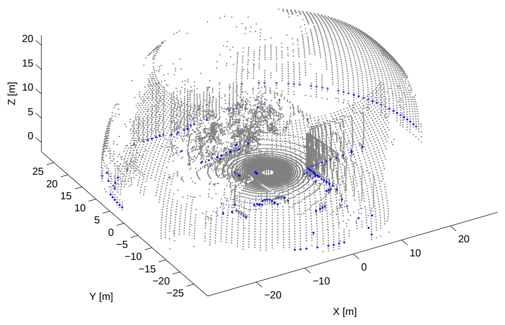


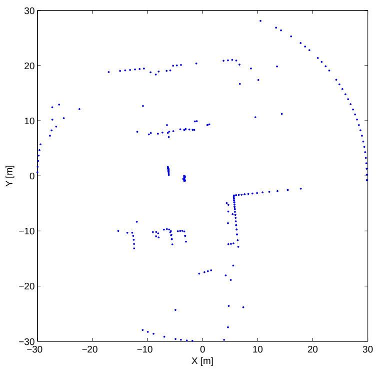
(e) 3D SLAM results.

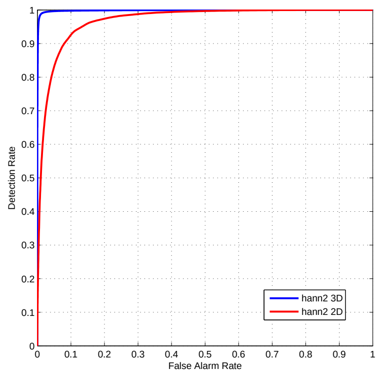
Figure 5: Receiver operating characteristic (ROC) curves. For each level of false alarm, the higher the detection is, the better the classifier is.

performance is so much worse, in comparison to the 2D results presented in Table 5 and Figure 5. A possible explanation is that for the *hann2* 2D point clouds the horizontal angular resolution is $\delta_\varphi = 1.0° \pm 0.33°$ (mean $\pm$ one standard deviation), compared to the *syd1* and *syd2* data sets which have a horizontal angular resolution of $\delta_\varphi = 0.5°$. Thus, the *hann2* 2D data contains less information, about 400 ± 163 points (mean $\pm$ one standard deviation), compared to *syd1* and *syd2* which both contain 722 points per cloud.

To test the hypothesis that the lower angular resolution is detrimental to classifier performance, the 3m data pairs from *syd2* were converted to $\delta_\varphi = 1°$ data by taking every second range measurement. Cross validation experiments similar to the ones presented in Table 5 were performed, and the results presented in Table 6 show that decreasing the angular resolution has a negative effect on performance. This experiment does not rule out other explanations, e.g. different degrees of structure or self-similarity in the two environments, however it does support the hypothesis that the lower angular resolution contributes to some degree to the poor performance. Similar experiments were performed evaluating the different maximum measureable ranges, $r_{\max} = 30$m for *hann2* and $r_{\max} = 50$m for *syd2*, however the results showed no statistically significant difference in performance.


(a) Original 3D point cloud in gray, with the extracted 2D point cloud in blue.


(b) Extracted 2D point cloud.



(c) Resulting receiver operating characteristic.

Figure 6: Illustration of the extraction of 2D point clouds from 3D point clouds.

Table 6: Comparison of 2D and 3D performance, all numbers in % unless otherwise stated.

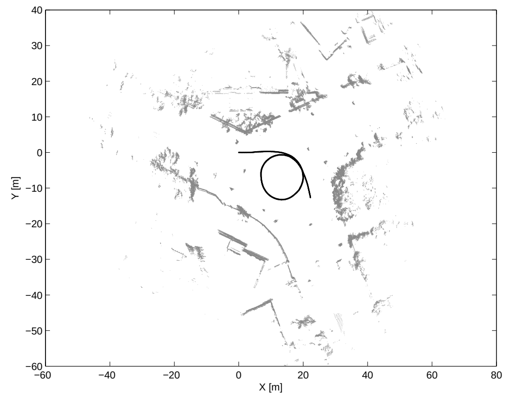
Data set	FA	D	Min/Max
hann2, 2D	0	2.12 ± 1.40	0/8.34
	1	43.96 ± 2.81	36.36/55.08
hann2, 3D	0	63 ± 6	28/76
	1	99 ± 0.1	98/99

Data: *syd2*, 3m	$\delta_\varphi = 1°$	$\delta_\varphi = 0.5°$
D at 0% FA	12.33 ± 6.21	18.57 ± 7.56

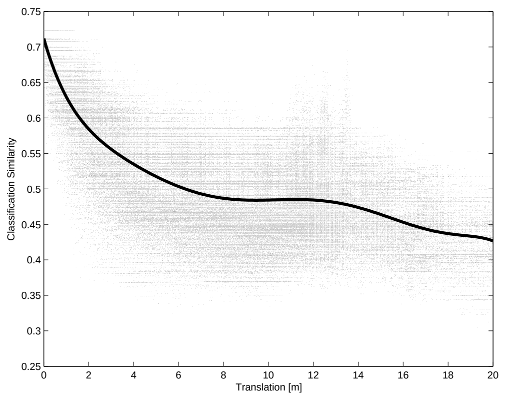
5.2.6 Dependence to translation

This section presents results from experiments testing how the learned classifier handles translation between the point clouds. While invariance to rotation is explicitly built into the features, and thus also into the learned classifier, there is no translational counterpart to the rotation invariance. For this experiment, data from the densely sampled *ken* data set was used. From the data set, a 78m trajectory which traverses a roundabout was chosen. Along the trajectory 1000 point clouds were acquired, with a mean translation of just 8cm between consecutive point clouds. The resulting trajectory and area map are shown in Figure 7a. Each of the 1000 point clouds was compared to the remaining 999 point clouds, and the resulting classification similarity is plotted against translation in Figure 7b. The figure also features a polynomial fitted to the results in the least squares sense. As can be seen, there is a rather rapid decay in classification similarity as the translation increases, suggesting that the classifier is highly sensitive to translation. The explanation is, however, not so simple. As shown in Figure 8 the point clouds change significantly in appearance after a small translation, thus making it difficult to determine that the point clouds are acquired in close vicinity of each other. Rather than showing the learned classifier's dependence to translation, this experiment shows the inherent difficulty of sensing a 3D environment with a 2D sensor.

However, further results which show the dependence to translation are found in Table 5 and Figure 5. While the detection rate decreases with increasing translational distance, it is still possible to achieve good loop closure detection results for up to 3m distance, in both 2D and 3D. The decreasing detection rates do imply an inversly proportional dependence between detection and translational distance. This suggests that the presented method is more suitable for environments where the vehicle is expected to travel along defined paths, e.g. office hallways, urban or rural roads, etc. In an environment where the vehicle is expected to be less restrained to defined paths, the presented method would possibly perform worse. To summarise, the presented method can handle translation, however considerable overlap of sensor field of view appears to be needed.



(a) Map of the area for which translation dependence were experimented on. The poses are printed in black, the point clouds are printed in gray.



(b) Plot of translation against classification similarity. The black line shows a seventh order polynomial fitted to the points in the least squares sense.

Figure 7: Classifier dependence to translation.

5.2.7 Dynamic objects

A challenge in loop closure detection is the presence of dynamic objects in the environment. Dynamic objects change the appearance of a scene, making it more difficult to recognise that two point clouds are indeed from the same location. An example of the challenge that dynamic objects present was given in Figure 1a, where the robot returns to a place where two vehicles have been parked along the side of the street.

In this section, we present results from experiments where the classifiers sensitivity to dynamic objects are tested. From the 2D *syd2* data set we were able to obtain 287 pairs of point clouds from the same location where dynamic objects have changed the appearance of the scene. In order to isolate the challenge of dynamic objects from other factors which may also affect the loop closure classification, the pairs of point clouds that are tested in this experiment are acquired at very low translational distance. Thus, the differences between the point clouds can be said to fully be an effect of the dynamic objects.

In order to assess the point cloud difference, which can be compared to the classification likelihood, we have computed the root mean square of the normalised extracted features for each pair of point clouds,

$$\sqrt{\frac{1}{n_f^1 + n_f^2} \sum \left(\frac{\mathbf{F}_{k,l}}{\mathbf{F}_\mu} \right)^2}. \tag{24}$$

The extracted features $\mathbf{F}_{k,l}$ were normalised with the average extracted feature $\mathbf{F}_\mu$, since the extracted features are quite different in magnitude. The normalisation was thus performed to give each component of the extracted feature vector an approximately equal weight. To compute $\mathbf{F}_\mu$, the positive class data pairs from the same data set were used. We chose to use the pairs from the same data set to assess the average similarity for the particular environment. A simple relation that can be used to better understand (24) is that if $\mathbf{F}_{k,l} = k\mathbf{F}_\mu$, the point cloud difference is k.

Results from the experiment are shown in Figure 9. The plot of classification likelihood against feature difference, Figure 9a, does not show any clear trend, in contrast to, e.g., Figure 7b which shows a downward trend in classification likelihood as translational distance increases. While the likelihood for some point cloud pairs is rather low, around 0.5, at the highest feature difference computed, around 5.25, the likelihood of loop closure is high for several of the point cloud pairs. It appears that the classifier can handle dynamic objects in many cases, which is in accordance with empirical impressions from using the classifier in SLAM experiments. If the positive training data includes point cloud pairs with dynamic objects, e.g. cars and humans, then the learned classifier can handle dynamic objects in the test data to some extent.

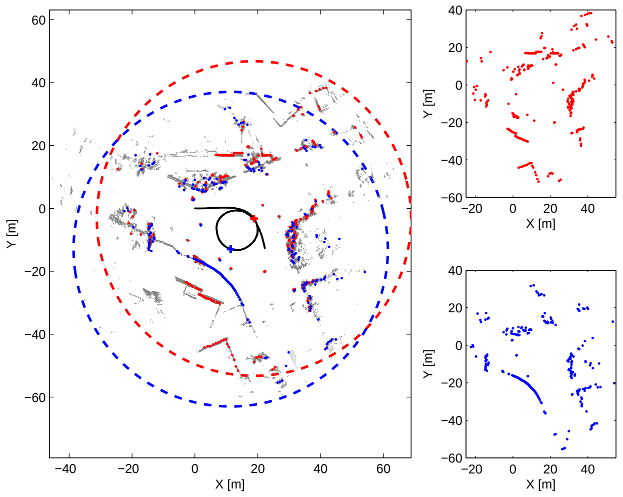


Figure 8: Illustration of the translation problem, for 2D data in outdoor environments. The plot on the left shows a trajectory, black line, along which 1000 point clouds were aquired, the constructed area map is shown in gray. Two poses are marked by crosses in blue and red, respectively, and the corresponding sensor surveilance boundary is shown by the dashed circles. Despite the fact that the relative pose translation, 12m, is well below the sensor's maximum range, 50m, the point clouds corresponding to the poses are significantly different.

5.2.8 Repetitive structures

Another difficulty faced by loop closure detection methods is the presence of repetitive structures in the environment. In e.g. an office environment, many hallways look similar, and many doorways also look similar. A high degree of repetitiveness in the environment is thus difficult, since the appearance of many places will be similar and consequently the computed feature values will be similar. While performing the experiments presented previously, repetitive structures in the data sets did not appear to pose a major difficulty to the presented loop closure detection method. To test our empirical observation that repetitive structures were not an issue in the data sets used, we considered the sample mean and standard deviation of the extracted features. If a feature is repetitive, many point clouds will measure the same feature value, and the feature difference will thus be similar for both point cloud pairs that are from the same location, and for point cloud pairs that are not from the same location.

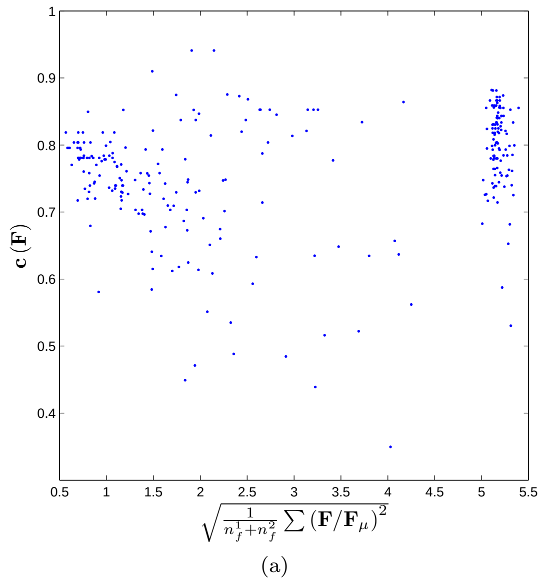
Let $\mathbf{F}_i^{(k)}$ be component k of the extracted features for data pair i, $\mathbf{F}_{i_1,i_2}$. Further, let μ_k^{p} and μ_k^{n} denote the mean and let σ_k^{p} and σ_k^{n} denote the standard deviation of $\mathbf{F}_i^k$ for the positive and negative data pairs, respectively. For features of the first type, the mean should be small for the positive class, and larger for the negative class. Features of the second type should be close to one for the positive class, and smaller for the negative class. If a feature is repetitive, μ_k^{p} and μ_k^{n} will be of similar size. Thus, the ratio

$$\begin{cases} \mu_k^{\text{p}}/\mu_k^{\text{n}} & \text{for type 1 features} \\ \mu_k^{\text{n}}/\mu_k^{\text{p}} & \text{for type 2 features} \end{cases} \tag{25}$$

can be used as a measure of repetitiveness. A value closer to one means that the environment is repetitive with respect to that feature.

For each of the positive data pairs, $\mathbf{F}_i^{(k)}$ should be low for the first feature type, or close to one for the second feature type, and σ_k^{p} should thus be small. The negative pairs are random samples of point clouds, thus σ_k^{n} will be small if the feature is repetitive, and otherwise larger. Thus, the ratio

$$\frac{\sigma_k^{\text{p}}}{\sigma_k^{\text{n}}} \tag{26}$$



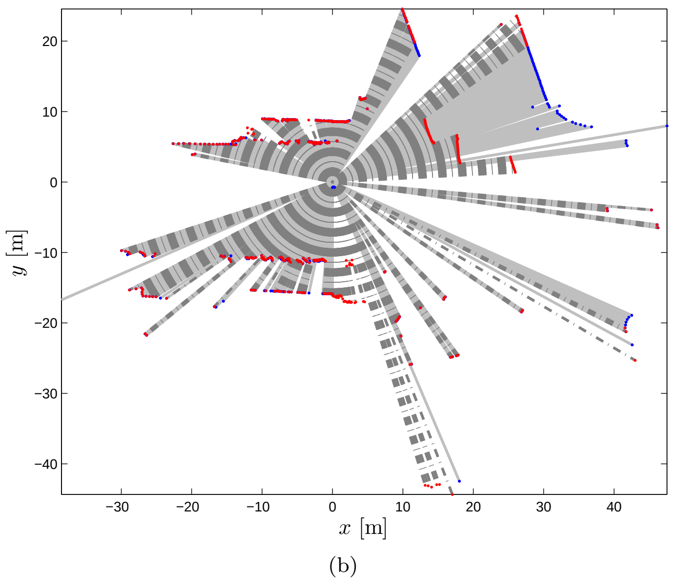


Figure 9: Results from experiment with dynamic objects. (a): Comparison of classifier likelihood and feature difference, computed as (24). Despite large feature differences, resonable classification likelihood can still be achieved in many cases. (b): Example of point cloud pair, where the appearance of the scene is changed by dynamic objects. The first point cloud is shown as blue dots, with the measurement rays shown in light gray. The other point cloud is shown as red dots, with the measurement rays shown in dash-dotted dark gray. The feature difference is 5.31, the classification likelihood, 0.53 is quite low. Note that range measurements at maximum range are not plotted for increased clarity.r

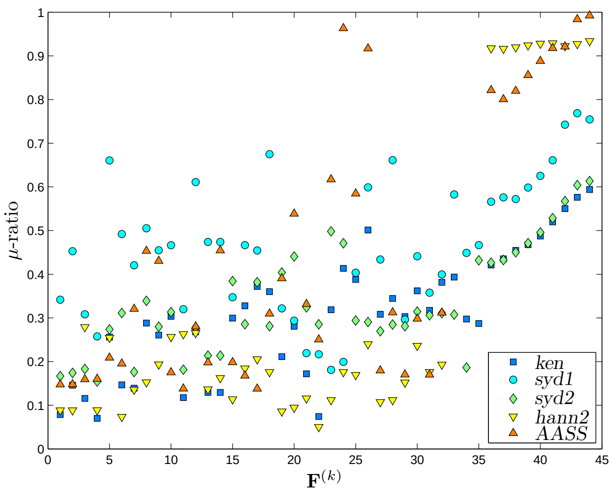
can also be used as a measure of repetitiveness. Analogously to (25), a value closer to one means that the environment is repetitive with respect to that feature. Using the 1m 2D data pairs and both sets 3D data pairs, the ratios (25) and (26) were computed. The results are shown in Figure 10.

For *AASS*, features 24 and 26 both have μ and σ ratios that suggest a high degree of repetitiveness. Considering the small size of this data set, it is difficult to draw any definite conclusions though. Regarding remaining data sets, for the mean ratio (25), in general none of the features appear to suffer from repetitiveness. The range histograms the 3D data are quite close though. For the standard deviation ratio (26), the features corresponding to range histograms with smaller bins appear to be somewhat sensitive to repetitiveness in 2D. Both the μ and σ ratios are below 0.5 in about 80% of the cases. To summarise, the results from the experiment largely support our observation that repetitive structures in the environment is not a major problem in the data sets used in this paper.

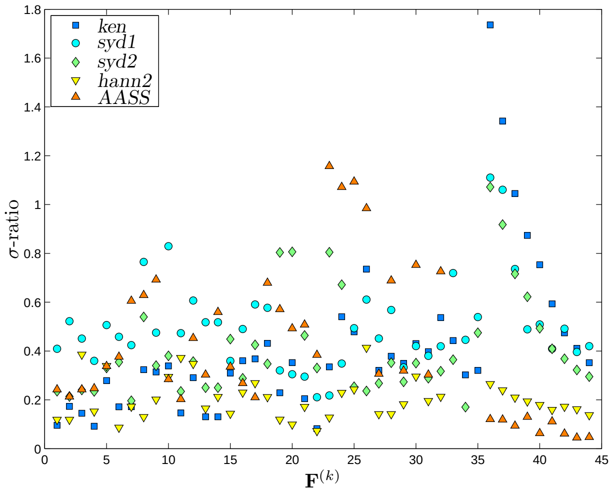
However, in a very large scale data set from a highly repetitive environment, repetitiveness could possibly become a problem. One way to handle such a difficulty is to modify the main SLAM filter to handle multiple hypotheses, similarly to the multiple hypothesis filter for target tracking, see e.g. (Bar-Shalom and Rong Li, 1995). This way, ambiguous loop detections could be kept as separate hypotheses until one or more hypotheses could be rejected.

5.3 SLAM experiments

In this section we present SLAM experiments in both 2D and 3D, using the framework presented in Section 4. These experiments were conducted for two reasons, the first is to verify how the classifier would perform in a SLAM setting, the other is to verify how the classifier performs when it is trained on data from one environment and then tested on data from another. Thus, in each experiment, the classifier was trained on



(a)



(b)

Figure 10: Results from repetitive structures experiment. The plots show the ratio for the positive and negative data pairs of mean, in (a), and standard deviation, in (b), of the extracted features. Ratios close to, or larger than one, suggest a high degree of repetitive structures.

one dataset and then used to detect loop closure on another dataset. For the 2D results, both training and testing were performed with outdoor data. For the 3D experiments, both outdoor and indoor data were used, and thus we are able to demonstrate how the classifier generalises from one environment to another.

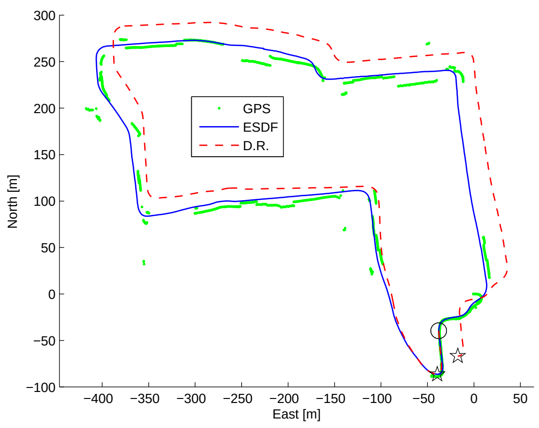
5.3.1 2D SLAM

The data pairs from *syd2* were used to train a classifier, which was then used to detect loop closure in experiments with the datasets *syd1* and *syd3*. Figure 11 shows the estimated ESDF trajectories compared to dead reckoning and GPS, and also the resulting point cloud maps overlaid onto aerial photographs. The results show a clear improvement in trajectory estimation when the suggested loop closure detection classifier was used.

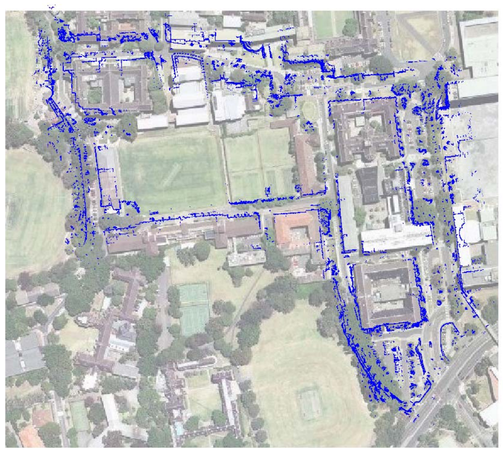
5.3.2 3D SLAM

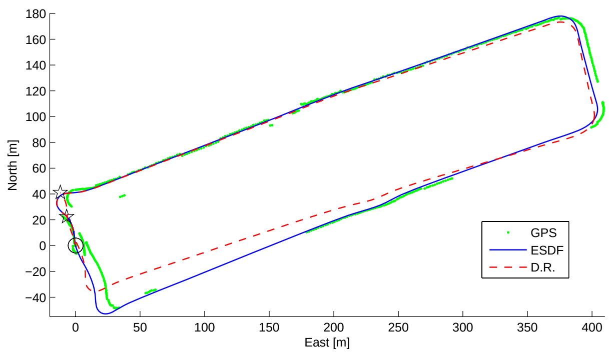
In the first experiment, the positive and negative data pairs from *hann2* were used to train a classifier. The classifier was then used to classify data pairs from the *AASS* dataset. Each point cloud $\mathbf{p}_k$ was compared to all previous point clouds, $\{\mathbf{p}_i\}_{i=1}^{k-1}$. The result from the experiment is shown as a classification matrix in Figure 12a. The k, l:th element of the classification matrix is the classification likelihood $\mathbf{c}(\mathbf{F}_{k,l})$, (12) in Algorithm 1. For completeness, the classification matrix in Figure 12a contains the classification of the (k, k) point cloud pairs. In a SLAM experiment however, such tests are obviously redundant. Figure 12b shows the classification matrix after thresholding each element, (13) in Algorithm 1. Black squares correspond to pairs of point clouds classified as being from the same location. Figure 12c shows the corresponding ground truth: black squares correspond to pairs of point clouds acquired less than 1m apart (Magnusson et al., 2009).

There is a high similarity between Figures 12b and 12c, showing the generalisation properties of the features and the classifier. The classifier used in the experiment was trained on outdoor data containing 17000 points per cloud, $r_{\max} = 30$, and then tested on indoor data containing 112000 points per cloud, $r_{\max} = 15$. Figure 12e 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 12d. For this part of the experiment, a minimum loop size of five poses was introduced, explaining why the detected loop closure between poses



(a) Estimated trajectory compared to dead reckoning (D.R.) and GPS.


(b) Resulting SLAM map overlaid onto an aerial photograph.



(c) Estimated trajectory compared to dead reckoning (D.R.) and GPS.

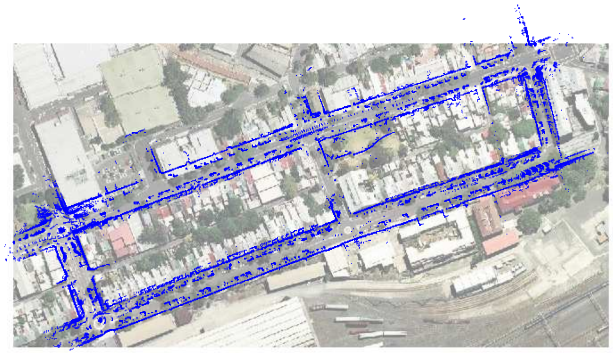

(d) Resulting SLAM map overlaid onto an aerial photograph.

Figure 11: 2D SLAM results. In (a) and (c), the rings mark the starting points and the stars mark the respective end points of the estimated trajectory and dead reckoning.

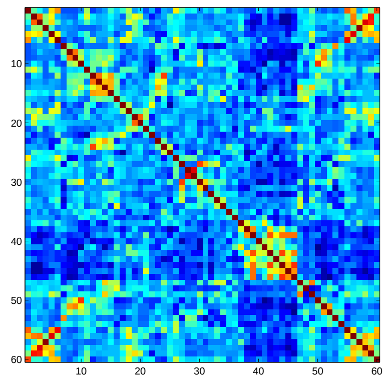
28 and 29 in Figure 12b is not present in Figure 12e.

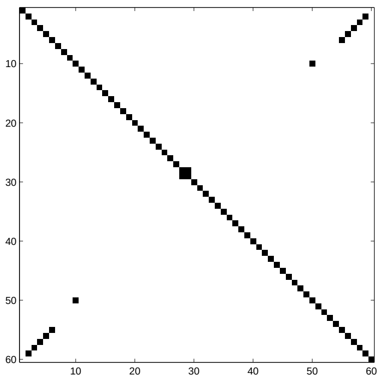
In the second experiment, the *AASS* data was used to train a classifier, which was then used to classify the *hann2* data. The classification results from this experiment are presented in Figure 13. For this experiment the detection rate is just 3% for 0% false alarm, an intuitive explanation for the poor performance is again the small number of training data, $N_{\mathrm{p}} = 16$ and $N_{\mathrm{n}} = 324$. It could be noted though that even such low detection rates can be enough to produce good SLAM maps (Cummins and Newman, 2009). The ROC-curves for both the 3D SLAM experiments are shown in Figure 5e. Both ROC-curves show good detection rates for false alarm rates $\geq 1\%$. The SLAM experiment where the *AASS* data was used for training does not handle very low levels of false alarm though, which the right plot in Figure 5e shows.

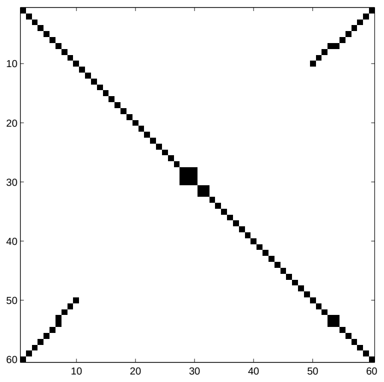
5.4 Summary and comparison

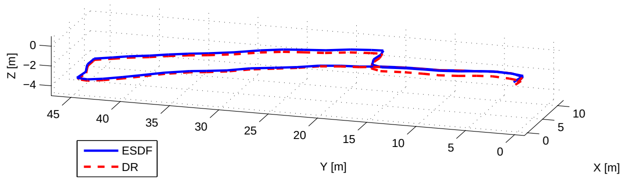
In this last subsection, we summarise the results from the experiments, and discuss how the presented loop closure detection method compares to related work. In experiments, we have presented results that

1. show the classifiers's execution time,

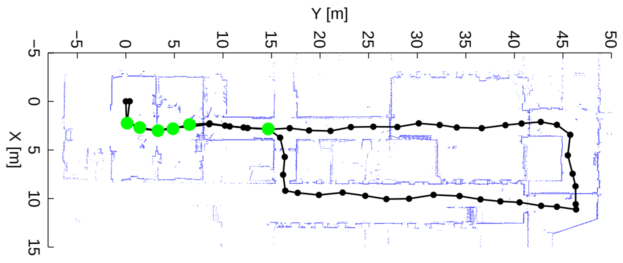

(a) Classification matrix, warmer color means higher similarity.


(b) Thresholded classification matrix, $K = 0.717$ (cf. (13) in Algorithm 1).


(c) Ground truth distance matrix, showing data pairs less than 1m apart (Magnusson et al., 2009).

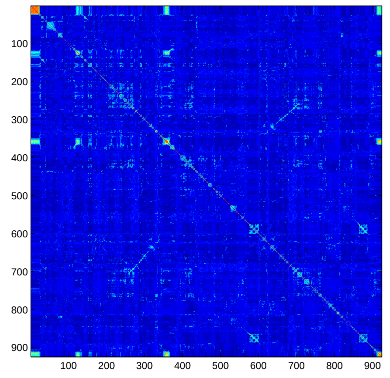


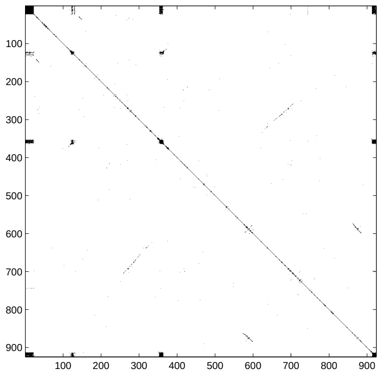
(d) Estimated trajectory (ESDF) vs. dead reckoning (DR).



(e) Resulting map with robot trajectory overlaid. Detected loop closures are marked with thick green circles.

Figure 12: Results from SLAM experiment on *AASS* data.


(a) Classification matrix, warmer color means higher similarity.


(b) Thresholded classification matrix, $K = 0.35$ (cf. (13) in Algorithm 1).

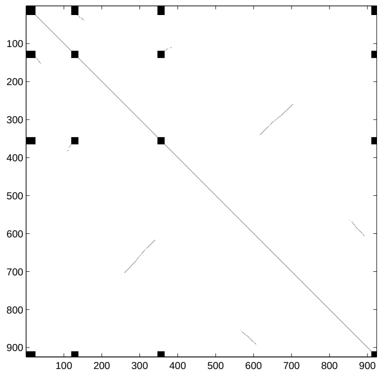

(c) Point cloud pairs from same location used for training.

Figure 13: Results from SLAM experiment on *hann2* data.

2. show the number of weak classifiers needed to construct a strong classifier,
3. test which features are most informative,
4. evaluate the classifier's receiver operating characteristic,
5. compares performance in 2D and 3D,
6. evaluate the dependence to translation,

7. show how the classifier handles dynamic objects,

8. test how the classifier handles repetitive structures in the environment, and

9. show that the classifier can be used to detect loops in SLAM experiments, both in 2D and 3D.

The presented loop detection method was evaluated on the publicly available *ken* data set, used in previous work on the same problem (Bosse and Zlot, 2008). It should be noted however that the work by (Bosse and Zlot, 2008) is for submaps containing laser scans from tens of meters of travel, while our results are for single laser scans. Further, as mentioned in Section 5.1, our results are for the first half of the data set. Thus, while the presented results are for the same data set, a *ceteris paribus* comparison is not possible. Even so, 84% detection at 1% false alarm is significantly higher than 51% detection at 1% false alarm, thus the presented work can be claimed to outperform related work on 2D data by (Bosse and Zlot, 2008) in terms of achieving high detection rates at low false alarm rates. In the work by (Brunskill et al., 2007), detection rates are not reported at low false alarm rates (or conversely, at high precision rates), and the data sets used are smaller in scale. A thorough comparison of quantitative results is thus unfortunately not possible.

For *hann2* 3D data, detection rates are higher than the NDT work by (Magnusson et al., 2009), while the work by (Steder et al., 2010) outperforms the presented method. It could be noted here that the method presented by (Steder et al., 2010) includes registering the two point clouds, which is not included in the detection here. Registering the two point clouds allows the robot to evaluate the registration result, and weed out possible false alarms as point clouds that are poorly aligned. It is possible that the presented method's detection rates could be improved further if the method was coupled with a registration process.

The presented method is faster than (Steder et al., 2010) however and, which is noted above and in related work, detecting only a subset of loops is typically sufficient to produce good SLAM results. Low execution times are of high importance, especially for larger data sets, since each point cloud must be compared to all previous point clouds. Furthermore, the presented method is fully invariant to rotation, while the work by (Steder et al., 2010) relies on the assumption that the robot is travelling over a flat surface, and the work by (Magnusson et al., 2009) relies on finding dominant planar surfaces.

The experiments that showed the most informative features showed that the results differed between different data sets, suggesting that the classifer might not generalise well between different data sets or environments. It was shown in Section 5.3.2 however, that the classifier does in fact generalise between different environments and sensor setups. This fact is important – since the classifier relies of being learned from manually labeled data, it must generalise well in order to function in an environment which is different from that on which it was learned. Experiments with dynamic objects showed that the type of dynamic objects that typically appear in suburban environments can be handled in most cases. Repetitive structures in the environment was shown to not pose a considerable challenge in the data sets used here.

The presented method is, compared to related work in both 2D and 3D, at a disadvantage in terms of the ability to handle translation. When the environment contains well defined pathways, such as office hallways or urban or rural roads, and the data is sampled without much translation inbetween point clouds, the sensitivity to translation is not a problem, which is shown by the SLAM experiments in 2D and 3D. Obtaining densely sampled data in 2D is easy using standard sensors, i.e. the SICK LMS200-sensors. In 3D, densely sampled data can be obtained using state-of-the-art sensors, i.e. the Velodyne HDL-64E. Thus, the need for densely sampled data does not pose a significant limitation. For data which is from environments without well defined pathways the dependence to translation could possibly prove to be problematic.

To summarise, the presented method performs well in environments with defined pathways, the execution times are favourable and detections rates at low false alarm rates compare well to related work and are sufficient to produce good SLAM results.

6 Conclusions and future work

This paper presented a method for loop closure detection, using pairwise comparison of point clouds. The presented method uses rotation invariant features, which provide a way to compress the sensed information into meaningful statistics. This reduces the dimension of the data by up to a factor of 2000 (point clouds with $>$ 100k points), thus the features also present a way to store the data in an efficient manner. The features are input to AdaBoost, which builds a classifier with good generalisation properties. Inheriting the rotation invariance from the features, the learned classifier is fully invariant to rotation, without the need to discretise the metric space, assume that the robot is traveling over a flat surface, or be limited by predefined geometric primitives. Thus, it is possible to detect loop closure from arbitrary directions. Experiments in both 2D and 3D showed the algorithms ability to achieve levels of detection at 0% false alarm, at detection levels comparable to related work. The SLAM experiments presented showed that the method can perform relialble loalisation and mapping in very challenging environments. Experiments using both indoor and outdoor data showed the generalisation properties of the framework proposed. The method is shown to be suitable for realtime performance: computing the set of features takes at most 0.2s (for a point cloud with 112000 points), and comparing the set of features for two point clouds takes less than 2ms.

In the experiments, the dependence between number of training data and classifier performance was noted. In future work, we intend to investigate this dependence further, and also address how training data can be selected in order to achieve the best performance at the lowest computational cost. Experiments showed that in addition to being fully invariant to rotation, the classifier can also handle up to 3m translation when detection loop closure between pairs of point clouds. In future work, we wish to evaluate whether this distance can be extended, such that the classifier can handle loop closure detection with less partial overlap between the point clouds. It would also be interesting to test the classifier on a very large scale data set from a highly repetitive environment, to see how it would perform in such a situation. Further, the presented SLAM framework relies on pairwise comparison between the current point cloud and all previous point clouds, resulting in a time complexity which grows linearly with the robot trajectory. The computed set of features can possibly be used in an intial nearest neighbour search, candidates from which are then used as input to the classifier. A similar approach has previously been taken for 2D point cloud submaps using keypoints and $kd-$ and $Bkd-$trees (Zlot and Bosse, 2009).

Acknowledgments

Karl Granström and Thomas B. Schön are supported by the Strategic Research Center MOVIII, funded by the Swedish Foundation for Strategic Research, SSF and CADICS, a Linnaeus center funded by the Swedish Research Council.

Fabio T. Ramos is supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

Appendix A: feature definitions

In this appendix we define the features that were used to learn classifiers for loop closure detection. The first subsection defines the features used in 2D, the second subsection presents the features used in 3D.

Given a point cloud $\mathbf{p}_k$, 15 or 14 parameters need to be specified for computing the features in 2D or 3D, respectively. The parameters are given in Table 7. Except for $r_{\max}$, all parameters are set manually. In order to find appropriate values, we have used empirical results. For the range histograms, instead of choosing just one bin size, we use 9 different bin sizes and leave it to the algorithm to find which corresponding features are informative.

Table 7: Parameters used when the features are computed. All parameters except for $r_{\max}$ are set manually.

Parameter	Numerical value	Comment
$r_{\max}$	15m (*AASS*), 30m (*hann2*), 50m (all 2D data)	Maximum measureable range. Determined by sensor used.
g_{dist}	2.5m	
$g_{\text{min size}}$	3	Only used in 2D.
g_{r_1}, g_{r_2}, g_{r_3}	$r_{\max}$, $0.75r_{\max}$, $0.50r_{\max}$	
$b_1, \ldots, b_9$	0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 2.5 and 3 meters	Bin sizes for the range histograms.

2D features

The following features were used for loop closure detection in 2D. Features 1 to 35 are of type 1, features 36 to 44 are of type 2.

1. -2. **Area** Measures the area covered by a point cloud. Points whose range is greater than $r_{\max}$ have their range set to $r_{\max}$. Each point is seen as the centre point of the base of an isosceles triangle. The height of the triangle is $h_i = r_i$, and the width of the base is $w_i = 2r_i \tan\left(\frac{\delta_\varphi}{2}\right)$, where δ_φ is the horizontal angular resolution of the sensor. The area of the triangle is $a_i = \frac{h_i w_i}{2} = r_i^2 \tan\left(\frac{\delta_\varphi}{2}\right)$. The area feature is computed as

$$a_{\max} = r_{\max}^2 \tan\left(\frac{\delta_\varphi}{2}\right), \tag{27a}$$

$$f_1 = \frac{1}{N a_{\max}} \sum_{i=1}^{N} r_i^2 \tan\left(\frac{\delta_\varphi}{2}\right) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{r_i}{r_{\max}}\right)^2. \tag{27b}$$

The area is normalised by dividing by the maximum measurable area $N v_{\max}$. Note that the specific numerical value of δ_φ is not needed to compute the feature. f_2 is the area computed for all ranges $r_i < r_{\max}$.

3. - 4. **Average Range** Let the normalised range be $r_i^{\text{n}} = r_i / r_{\max}$. f_3 is the average r_i^{n} for ranges $r_i < r_{\max}$ and f_4 is the average r_i^{n} for all ranges.

5. - 6. **Standard Deviation of Range** f_5 is the standard deviation of r_i^{n} for ranges $r_i < r_{\max}$ and f_6 is the standard deviation of r_i^{n} for all ranges.

7. - 9. **Circularity** A circle is fitted to all points in the cloud in a least squares sense, which returns the centre of the fitted circle p_c and the radius of the fitted circle r_c. f_7 is $r_c / r_{\max}$, f_8 is the residual sum of squares divided by $N r_c$,

$$f_8 = \frac{1}{N r_c} \sum_{i=1}^{N} \left(r_c - \|p_c - p_i\|\right)^2, \tag{28}$$

where $\|\cdot\|$ is the Euclidean norm. f_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 $r_i < r_{\max}$. $f_{10} = \|\bar{p}\|$, f_{11} is the mean distance from $\bar{p}$ for points $r_i < r_{\max}$ and f_{12} is the standard deviation of the distances from $\bar{p}$ for points $r_i < r_{\max}$.

13. - 14. **Maximum Range** f_{13} is the number of ranges $r_i = r_{\max}$ and f_{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_{15} is the sum of δp_i for all points. f_{16} is the sum of δp_i, for consecutive points with $r_i, r_{i+1} < r_{\max}$. f_{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_{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}, d_i and d_{i+1} be the pairwise point to point distances. The curvature at p_i is computed as $k_i = \frac{4A}{d_{i-1}d_i d_{i+1}}$. Curvature is computed for $p_i \in \mathbf{I}$, where $\mathbf{I} = \{p_i \ : \ r_{i-1}, r_i, r_{i+1} < r_{\max},\ d_{i-1}, d_i, d_{i+1} < g_{\text{dist}}\}$. f_{19} is the mean curvature and f_{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

$$m_k = \frac{1}{N_{r_i < r_{\max}}} \sum_{i \ : \ r_i < r_{\max}} (r_i - \bar{r})^k, \tag{29a}$$

$$f_{21} = \frac{m_4}{(m_2)^2} - 3, \tag{29b}$$

where $\bar{r}$ is mean range, and $N_{r_i < r_{\max}}$ is the number of ranges $r_i < r_{\max}$. f_{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} is the mean of r_i^{r} and f_{24} is the standard deviation of r_i^{r} for all ranges. f_{25} and f_{26} 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** 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} (mean) and f_{28} (standard deviation), and g_{r_2} and g_{r_3} gives f_{29} to f_{32}. The features are normalised by division by the respective g_{r_i}.

33. - 34. **Group** A group is defined as a cluster of points in which the distance between consecutive points is less than a maximum distance gate g_{dist}. To be considered a group, the cluster has to contain more than a certain number of points specified by the minimum group size gate $g_{\text{min size}}$. f_{33} is the total number of groups found, f_{34} is the average number of points in each group.

35. **Mean Angular Difference** Measures the sum of the angles between consecutive point to point vectors. Given two consecutive points p_i and p_{i+1}, a vector that connects the points is given as $\bar{p}_{i,i+1} = [\mathsf{x}_{i+1} - \mathsf{x}_i \ , \ \mathsf{y}_{i+1} - \mathsf{y}_i]^{\mathrm{T}}$. The feature is calculated as

$$f_{15} = \sum_{i : r_{\{i,i+1,i+2\}} < r_{\max}} \arccos\left(\frac{\bar{p}_{i,i+1}^{\mathrm{T}} \bar{p}_{i+1,i+2}}{||\bar{p}_{i,i+1}|| \ ||\bar{p}_{i+1,i+2}||} \right). \tag{30}$$

36. - 44. **Range histogram** f_{33} to f_{41} are range histograms. Bins of sizes b_j, see Table 7, are used to tabulate the ranges.

3D features

The following features were used for loop closure detection in 3D. Features 1 to 32 are of type 1, features 33 to 41 are of type 2. Note that some of the 3D features are defined analogously to some of the 2D features, hence the definitions are not repeated.

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 δ_φ and δ_ψ be the laser range sensors horisontal and vertical angular resolution, and let

$l_i = 2r_i \tan\left(\frac{\delta_\varphi}{2}\right)$ and $w_i = 2r_i \tan\left(\frac{\delta_\psi}{2}\right)$ 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

$$v_{\max} = \frac{4}{3}\tan\left(\frac{\delta_\varphi}{2}\right)\tan\left(\frac{\delta_\psi}{2}\right) r_{\max}^3 \tag{31a}$$

$$f_1 = \frac{1}{N v_{\max}}\sum_{i=1}^{N} v_i = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{r_i}{r_{\max}}\right)^3 \tag{31b}$$

The volume is normalised by dividing by the maximum measurable volume $Nv_{\max}$, i.e. the volume when all ranges equal $r_{\max}$. Notice that the explicit values of δ_φ and δ_ψ do not matter. f_2 is the volume computed using points with $r_i < r_{\max}$.

3. - 6. Defined analogously to features 3 to 6 in 2D.

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_7 is $r_c/r_{\max}$, f_8 is the residual sum of squares divided by Nr_c,

$$f_8 = \frac{1}{Nr_c}\sum_{i=1}^{N}\left(r_c - \|p_c - p_i\|\right)^2, \tag{32}$$

where $\|\cdot\|$ is the Euclidean norm. f_9 is $\frac{\|p_c\|}{r_{\max}}$.

10. - 32. Defined analogously to features 10 to 32 in 2D.

33. - 41. **Range histogram** Defined analogously to features 36 to 44 in 2D.

Appendix B: compounding operations

This appendix contains definitions of the compounding operations $\oplus$ and $\ominus$ and their corresponding jacobians. Let $\mathbf{x}_{i,j}$ denote the location of coordinate fram j with respect to coordinate frame i. The defintions are taken from (Eustice, 2005; Smith et al., 1990).

Compounding in 2D

Let the 2D 3-DOF pose be given by

$$\mathbf{x}_{i,j} = \begin{bmatrix} \mathsf{x}_{i,j} & \mathsf{y}_{i,j} & \psi_{i,j} \end{bmatrix}^{\mathrm{T}}. \tag{33}$$

The compounding operation $\mathbf{x}_{i,k} = \mathbf{x}_{i,j} \oplus \mathbf{x}_{j,k}$ is defined as

$$\mathbf{x}_{i,k} = \mathbf{x}_{i,j} \oplus \mathbf{x}_{j,k} = \begin{bmatrix} \mathsf{x}_{i,j} + \mathsf{x}_{j,k}\cos(\psi_{i,j}) - \mathsf{y}_{j,k}\sin(\psi_{i,j}) \\ \mathsf{y}_{i,j} + \mathsf{x}_{j,k}\sin(\psi_{i,j}) + \mathsf{y}_{j,k}\cos(\psi_{i,j}) \\ \psi_{ij} + \psi_{jk} \end{bmatrix}. \tag{34}$$

The Jacobian of the compounding operator $\mathbf{J}_\oplus$ is given by

$$\begin{aligned} \mathbf{J}_\oplus &= \frac{d(\mathbf{x}_{i,j} \oplus \mathbf{x}_{j,k})}{d(\mathbf{x}_{i,j}, \mathbf{x}_{j,k})} = \frac{d(\mathbf{x}_{i,k})}{d(\mathbf{x}_{i,j}, \mathbf{x}_{j,k})} = \begin{bmatrix} \mathbf{J}_{1\oplus} & \mathbf{J}_{2\oplus} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & -\left(x_{j,k}\sin(\phi_{i,j}) + y_{j,k}\cos(\phi_{i,j})\right) & \cos(\phi_{i,j}) & -\sin(\phi_{i,j}) & 0 \\ 0 & 1 & x_{j,k}\cos(\phi_{i,j}) - y_{j,k}\sin(\phi_{i,j}) & \sin(\phi_{i,j}) & \cos(\phi_{i,j}) & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{35}$$

where $\mathbf{J}_{1\oplus}$ and $\mathbf{J}_{2\oplus}$ correspond to the left and right 3×3-matrix half partitioning of $\mathbf{J}_{\oplus}$. The inverse relationship $\ominus$, explaining $\mathbf{x}_{j,i}$ as a function of the coordinates in $\mathbf{x}_{i,j}$, is given by

$$\mathbf{x}_{j,i} = \ominus\mathbf{x}_{i,j} = \begin{bmatrix} -x_{i,j}\cos(\phi_{i,j}) - y_{i,j}\sin(\phi_{i,j}) \\ x_{i,j}\sin(\phi_{i,j}) - y_{i,j}\cos(\phi_{i,j}) \\ -\phi_{i,j} \end{bmatrix}, \tag{36}$$

with Jacobian $\mathbf{J}_{\ominus}$

$$\mathbf{J}_{\ominus} = \frac{d(\mathbf{x}_{j,i})}{d\mathbf{x}_{i,j}} = \frac{d(\ominus\mathbf{x}_{i,j})}{d\mathbf{x}_{i,j}} = \begin{bmatrix} -\cos(\phi_{i,j}) & -\sin(\phi_{i,j}) & x_{i,j}\sin(\phi_{i,j}) - y_{i,j}\cos(\phi_{i,j}) \\ \sin(\phi_{i,j}) & -\cos(\phi_{i,j}) & x_{i,j}\cos(\phi_{i,j}) + y_{i,j}\sin(\phi_{i,j}) \\ 0 & 0 & -1 \end{bmatrix}. \tag{37}$$

Compounding in 3D

Let the 3D 6-DOF pose be given by

$$\mathbf{x}_{i,j} = \begin{bmatrix} \mathsf{x}_{i,j} & \mathsf{y}_{i,j} & \mathsf{z}_{i,j} & \phi_{i,j} & \theta_{i,j} & \psi_{i,j} \end{bmatrix}^{\mathrm{T}}. \tag{38}$$

The compounding operation $\mathbf{x}_{i,k} = \mathbf{x}_{i,j} \oplus \mathbf{x}_{j,k}$ is defined as

$$\mathbf{x}_{i,k} = \mathbf{x}_{i,j} \oplus \mathbf{x}_{j,k} = \begin{bmatrix} \mathsf{R}_{i,j}\begin{bmatrix}\mathsf{x}_{j,k} & \mathsf{y}_{j,k} & \mathsf{z}_{j,k}\end{bmatrix}^{\mathrm{T}} + \begin{bmatrix}\mathsf{x}_{i,j} & \mathsf{y}_{i,j} & \mathsf{z}_{i,j}\end{bmatrix}^{\mathrm{T}} \\ \operatorname{atan2}\left(\mathsf{R}_{i,k}^{(1,3)}\sin\psi_{i,k} - \mathsf{R}_{i,k}^{(2,3)}\cos\psi_{i,k}, -\mathsf{R}_{i,k}^{(1,2)}\sin\psi_{i,k} + \mathsf{R}_{i,k}^{(2,2)}\cos\psi_{i,k}\right) \\ \operatorname{atan2}\left(-\mathsf{R}_{i,k}^{(3,1)}, \mathsf{R}_{i,k}^{(1,1)}\cos\psi_{i,k} + \mathsf{R}_{i,k}^{(2,1)}\sin\psi_{i,k}\right) \\ \operatorname{atan2}\left(\mathsf{R}_{i,k}^{(2,1)}, \mathsf{R}_{i,k}^{(1,1)}\right) \end{bmatrix}, \tag{39}$$

where the rotation matrix $\mathsf{R}_{i,j}$ is defined as

$$\mathsf{R}_{i,j} = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{40}$$

where the subscripts $_{i,j}$ are omitted for brevity. Further, $\mathsf{R}_{i,k} = \mathsf{R}_{i,j}\mathsf{R}_{j,k}$ and $\mathsf{R}_{i,k}^{(m,n)}$ is the (m,n):th element of the rotation matrix $\mathsf{R}_{i,k}$. The Jacobian of the compounding operator $\mathbf{J}_{\oplus}$ is given by

$$\begin{aligned} \mathbf{J}_{\oplus} = \frac{d(\mathbf{x}_{i,j} \oplus \mathbf{x}_{j,k})}{d(\mathbf{x}_{i,j}, \mathbf{x}_{j,k})} = \frac{d(\mathbf{x}_{i,k})}{d(\mathbf{x}_{i,j}, \mathbf{x}_{j,k})} &= \begin{bmatrix}\mathbf{J}_{1\oplus} & \mathbf{J}_{2\oplus}\end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{M} & \mathsf{R}_{i,j} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{K}_1 & \mathbf{0}_{3\times3} & \mathbf{K}_2 \end{bmatrix}. \end{aligned} \tag{41}$$

where $\mathbf{J}_{1\oplus}$ and $\mathbf{J}_{2\oplus}$ correspond to the left and right 6×6-matrix half partitioning of $\mathbf{J}_{\oplus}$, and

$$\begin{aligned} \mathbf{M} &= \begin{bmatrix} \mathsf{R}_{i,j}^{(1,3)}\mathsf{y}_{j,k} - \mathsf{R}_{i,j}^{(1,2)}\mathsf{z}_{j,k} & (\mathsf{z}_{i,k} - \mathsf{z}_{i,j})\cos\psi_{i,j} & -(\mathsf{y}_{i,k} - \mathsf{y}_{i,j}) \\ \mathsf{R}_{i,j}^{(2,3)}\mathsf{y}_{j,k} - \mathsf{R}_{i,j}^{(2,2)}\mathsf{z}_{j,k} & (\mathsf{z}_{i,k} - \mathsf{z}_{i,j})\sin\psi_{i,j} & (\mathsf{x}_{i,k} - \mathsf{x}_{i,j}) \\ \mathsf{R}_{i,j}^{(3,3)}\mathsf{y}_{j,k} - \mathsf{R}_{i,j}^{(3,2)}\mathsf{z}_{j,k} & -\mathsf{x}_{j,k}\cos\theta_{i,j} - (\mathsf{y}_{j,k}\sin\phi_{i,j} + \mathsf{z}_{j,k}\cos\phi_{i,j})\sin\theta_{i,j} & 0 \end{bmatrix} \\ \mathbf{K}_1 &= \begin{bmatrix} \cos\theta_{i,j}\cos(\psi_{i,k} - \psi_{i,j})\sec\theta_{i,k} & \sin(\psi_{i,k} - \psi_{i,j})\sec\theta_{i,k} & 0 \\ -\cos\theta_{i,j}\sin(\psi_{i,k} - \psi_{i,j}) & \cos(\psi_{i,k} - \psi_{i,j}) & 0 \\ \mathsf{R}_{j,k}^{(1,2)}\sin\phi_{i,k} + \mathsf{R}_{j,k}^{(1,3)}\cos\phi_{i,k}\sec\theta_{i,k} & \sin(\psi_{i,k} - \psi_{i,j})\tan\theta_{i,k} & 1 \end{bmatrix} \\ \mathbf{K}_2 &= \begin{bmatrix} 1 & \sin(\phi_{i,k} - \phi_{j,k})\tan\theta_{i,k} & \left(\mathsf{R}_{i,j}^{(1,3)}\cos\psi_{i,k} + \mathsf{R}_{i,j}^{(2,3)}\sin\psi_{i,k}\right)\sec\theta_{i,k} \\ 0 & \cos(\phi_{i,k} - \phi_{j,k}) & -\cos\theta_{j,k}\sin(\phi_{i,k} - \phi_{j,k}) \\ 0 & \sin(\phi_{i,k} - \phi_{j,k})\sec\theta_{i,k} & \cos\theta_{j,k}\cos(\phi_{i,k} - \phi_{j,k})\sec\theta_{i,k} \end{bmatrix} \end{aligned} \tag{42}$$

The inverse relationship $\ominus$, explaining $\mathbf{x}_{j,i}$ as a function of the coordinates in $\mathbf{x}_{i,j}$, is given by

$$\mathbf{x}_{j,i} = \ominus\mathbf{x}_{i,j} = \begin{bmatrix} -\mathsf{R}_{i,j}^{\mathrm{T}} \begin{bmatrix} \mathsf{x}_{i,j} & \mathsf{y}_{i,j} & \mathsf{z}_{i,j} \end{bmatrix}^{\mathrm{T}} \\ \operatorname{atan2}\left(\mathsf{R}_{i,j}^{(3,1)} \sin\psi_{j,i} - \mathsf{R}_{i,j}^{(3,2)} \cos\psi_{j,i}, -\mathsf{R}_{i,j}^{(2,1)} \sin\psi_{j,i} + \mathsf{R}_{i,j}^{(2,2)} \cos\psi_{j,i} \right) \\ \operatorname{atan2}\left(-\mathsf{R}_{i,j}^{(1,3)}, \mathsf{R}_{i,j}^{(1,1)} \cos\psi_{j,i} + \mathsf{R}_{i,j}^{(1,2)} \sin\psi_{j,i} \right) \\ \operatorname{atan2}\left(\mathsf{R}_{i,j}^{(1,2)}, \mathsf{R}_{i,j}^{(1,1)} \right) \end{bmatrix} \tag{43}$$

with Jacobian $\mathbf{J}_{\ominus}$

$$\mathbf{J}_{\ominus} = \frac{d(\mathbf{x}_{j,i})}{d\mathbf{x}_{i,j}} = \frac{d(\ominus\mathbf{x}_{i,j})}{d\mathbf{x}_{i,j}} = \begin{bmatrix} -\mathsf{R}_{i,j}^{\mathrm{T}} & \mathbf{N} \\ \mathbf{0}_{3\times 3} & \mathbf{Q} \end{bmatrix} \tag{44}$$

where

$$\mathbf{N} = \begin{bmatrix} 0 & -\mathsf{R}_{i,j}^{(3,1)} \left(\mathsf{x}_{i,j} \cos\psi_{i,j} + \mathsf{y}_{i,j} \sin\psi_{i,j} \right) + \mathsf{z}_{i,j} \cos\theta_{i,j} & \mathsf{R}_{i,j}^{(2,1)} \mathsf{x}_{i,j} - \mathsf{R}_{i,j}^{(1,1)} \mathsf{y}_{i,j} \\ \mathsf{z}_{j,i} & -\mathsf{R}_{i,j}^{(3,2)} \left(\mathsf{x}_{i,j} \cos\psi_{i,j} + \mathsf{y}_{i,j} \sin\psi_{i,j} \right) + \mathsf{z}_{i,j} \sin\theta_{i,j} \sin\phi_{i,j} & \mathsf{R}_{i,j}^{(2,2)} \mathsf{x}_{i,j} - \mathsf{R}_{i,j}^{(1,2)} \mathsf{y}_{i,j} \\ -\mathsf{y}_{j,i} & -\mathsf{R}_{i,j}^{(3,3)} \left(\mathsf{x}_{i,j} \cos\psi_{i,j} + \mathsf{y}_{i,j} \sin\psi_{i,j} \right) + \mathsf{z}_{i,j} \sin\theta_{i,j} \cos\phi_{i,j} & \mathsf{R}_{i,j}^{(2,3)} \mathsf{x}_{i,j} - \mathsf{R}_{i,j}^{(1,3)} \mathsf{y}_{i,j} \end{bmatrix}$$

$$\mathbf{Q} = \frac{1}{1 - \left(\mathsf{R}_{i,j}^{(1,3)} \right)^2} \begin{bmatrix} -\mathsf{R}_{i,j}^{(1,1)} & -\mathsf{R}_{i,j}^{(1,2)} \cos\phi_{i,j} & \mathsf{R}_{i,j}^{(1,3)} \mathsf{R}_{i,j}^{(3,3)} \\ \mathsf{R}_{i,j}^{(1,2)} \sqrt{1 - \left(\mathsf{R}_{i,j}^{(1,3)} \right)^2} & -\mathsf{R}_{i,j}^{(3,3)} \cos\phi_{i,j} \sqrt{1 - \left(\mathsf{R}_{i,j}^{(1,3)} \right)^2} & \mathsf{R}_{i,j}^{(2,3)} \sqrt{1 - \left(\mathsf{R}_{i,j}^{(1,3)} \right)^2} \\ \mathsf{R}_{i,j}^{(1,3)} \mathsf{R}_{i,j}^{(1,1)} & -\mathsf{R}_{i,j}^{(2,3)} \cos\psi_{i,j} & -\mathsf{R}_{i,j}^{(3,3)} \end{bmatrix} \tag{45}$$

Composite relationships

Using the two operations defined above, operations for more than two spatial relationships can be performed. The following composite relationships hold in both 2D and 3D:

$$\begin{aligned} \mathbf{x}_{i,l} &= \mathbf{x}_{i,j} \oplus \mathbf{x}_{j,l} = \mathbf{x}_{i,j} \oplus (\mathbf{x}_{j,k} \oplus \mathbf{x}_{k,l}) \\ &= \mathbf{x}_{i,k} \oplus \mathbf{x}_{k,l} = (\mathbf{x}_{i,j} \oplus \mathbf{x}_{j,k}) \oplus \mathbf{x}_{k,l} \qquad \text{(46a)} \\ \mathbf{x}_{i,j} \ominus \mathbf{x}_{k,j} &= \mathbf{x}_{i,j} \oplus (\ominus\mathbf{x}_{i,k}) \qquad \text{(46b)} \\ \mathbf{x}_{j,k} &= \ominus\mathbf{x}_{i,j} \oplus \mathbf{x}_{i,k} \qquad \text{(46c)} \end{aligned}$$

The Jacobian of (46c), ${}_{\ominus}\mathbf{J}_{\oplus}$, is given by

$${}_{\ominus}\mathbf{J}_{\oplus} = \frac{d\mathbf{x}_{j,k}}{d(\mathbf{x}_{i,j}, \mathbf{x}_{i,k})} = \frac{d\mathbf{x}_{j,k}}{d(\mathbf{x}_{j,i}, \mathbf{x}_{i,k})} \times \frac{d(\mathbf{x}_{j,i}, \mathbf{x}_{i,k})}{d(\mathbf{x}_{i,j}, \mathbf{x}_{i,k})} = \mathbf{J}_{\oplus} \times \begin{bmatrix} \mathbf{J}_{\ominus} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{1\oplus}\mathbf{J}_{\ominus} & \mathbf{J}_{2\oplus} \end{bmatrix}. \tag{47}$$

References

Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2008). Fast and incremental method for loop-closure detection using bags of visual words. *Robotics, IEEE Transactions on*, 24(5):1027 –1037.

Arras, K. O., Mozos, O. M., and Burgard, W. (2007). 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.

Bar-Shalom, Y. and Rong Li, X. (1995). *Multitarget-Multisensor Tracking: Principles and Techniques*. YBS.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522.

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.

Biber, P. and Strasser, W. (2003). 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)*, pages 2743–2748, Las Vegas, USA.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

Bosse, M. and Zlot, R. (2009a). Keypoint design and evaluation for place recognition in 2D lidar maps. *Robotics and Autonomous Systems*, 57(12):1211–1224.

Bosse, M. and Zlot, R. (2009b). Place recognition using regional point dsceriptors for 3D mapping. In *Proceedings of the International Conference on Field and Service Robotics (FSR)*, Cambridge, Massachusets, USA.

Bosse, M. C. and Zlot, R. (2008). Map Matching and Data Association for Large-Scale Two-dimensional Laser Scan-based SLAM. *International Journal of Robotics Research*, 27(6):667–691.

Brunskill, E., Kollar, T., and Roy, N. (2007). Topological mapping using spectral clustering and classification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3491–3496, San Diego, California, USA.

Callmer, J., Granström, K., Nieto, J., and Ramos, F. (2008). Tree of Words for Visual Loop Closure Detection in Urban SLAM. In *Proceedings of the Australasian Conference on Robotics & Automation (ACRA)*, Canberra, Australia.

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

Cummins, M. and Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, 27(6):647–665.

Cummins, M. and Newman, P. (2009). Highly scalable appearance-only SLAM FAB-MAP 2.0. In *Proceedings of Robotics Science and Systems (RSS)*, Seattle, USA.

Eade, E. and Drummond, T. (2008). Unified Loop Closing and Recovery for Real Time Monocular SLAM. In *Proceedings of the British Machine Vision Conference*, Leeds, United Kingdom.

Eustice, R., Walter, M., and Leonard, J. (2005). Sparse extended information filters: Insights into sparsification. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 641–648, Edmonton, Alberta, Canada.

Eustice, R. M. (2005). *Large-area visually augmented navigation for autonomous underwater vehicles*. PhD thesis, Massachusetts Institute of Technology / Woods Hole Oceanographic Institution Joint Program.

Eustice, R. M., Singh, H., and Leonard, J. J. (2006). Exactly sparse delayed-state filters for view-based SLAM. *IEEE Transactions on Robotics*, 22(6):1100–1114.

Fraundorfer, F., Engels, C., and Nister, D. (2007). Topological mapping, localization and navigation using image collections. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3872 –3877.

Freund, Y. and Schapire, R. E. (1995). 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.

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

Goedeme, T., Tuytelaars, T., and Van Gool, L. (2006). Visual topological map building in self-similar environments. In *Proceedings of International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Setubal, Portugal.

Granström, K., Callmer, J., Ramos, F., and Nieto, J. (2009). Learning to Detect Loop Closure from Range Data. In Bicchi, A., editor, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, Kobe, Japan.

Granström, K. and Schön, T. (2010). Learning to close the loop from 3D point clouds. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Gutmann, J.-S. and Konolige, K. (1999). Incremental mapping of large cyclic environments. In *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325, Monterey, California, USA.

Hähnel, D., Burgard, W., Fox, D., and Thrun, S. (2003). An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, Nevada, USA.

Ho, K. and Newman, P. (2005). Combining Visual and Spatial Appearance for Loop Closure Detection in SLAM. In *Proceedings of European Conference on Mobile Robots (ECMR)*, Ancona, Italy.

Ho, K. L. and Newman, P. (2007). Detecting loop closure with scene sequences. *International Journal of Computer Vision*, 74(3):261–286.

Howard, A. and Roy, N. (2003). The robotics data set repository (Radish). [Online; accessed 20-August-2010; http://radish.sourceforge.net/].

Johnson, A. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):433 – 449.

Konolige, K., Bowman, J., Chen, J., Mihelich, P., Calonder, M., Lepetit, V., and Fua, P. (2010). View-based maps. *The International Journal of Robotics Research*, 29(8):941–957.

Magnusson, M., Andreasson, H., Nüchter, A., and Lilienthal, A. J. (2009). Automatic appearance-based loop detection from 3D laser data using the normal distributions transform. *Journal of Field Robotics*, 26(11–12):892–914.

Magnusson, M., Duckett, T., and Lilienthal, A. J. (2007). Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10):803–827.

Milford, M. J. and Wyeth, G. F. (2008). Mapping a Suburb With a Single Camera Using a Biologically Inspired SLAM System. *IEEE Transactions on Robotics*, 24(5):1038–1053.

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

Newman, P., Cole, D., and Ho, K. (2006). Outdoor SLAM using visual appearance and laser ranging. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, Florida, USA.

Nieto, J., Bailey, T., and Nebot, E. (2007). Recursive Scan-Matching SLAM. *Journal of Robotics and Autonomous Systems*, 55(1):39–49.

Nüchter, A. and Lingemann, K. (2009). Robotic 3D scan repository. [Online; accessed 10-August-2009; http://kos.informatik.uni-osnabrueck.de/3Dscans/].

Paul, R. and Newman, P. (2010). FAB-MAP 3D: Topological mapping with spatial and visual appearance. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2649–2656, Anchorage, Alaska, USA.

Ramos, F. T., Fox, D., and Durrant-Whyte, H. F. (2007a). CRF-Matching: Conditional Random Fields for Feature-Based Scan Matching. In *Proceedings of Robotics: Science and Systems (RSS)*, Atlanta, Georgia, USA.

Ramos, F. T., Nieto, J., and Durrant-Whyte, H. F. (2007b). Recognising and Modelling Landmarks to Close Loops in Outdoor SLAM. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy.

Rencken, W. D., Feiten, W., and Zöllner, R. (1999). Relocalisation by Partial Map Matching. In *Sensor Based Intelligent Robots*, pages 21–35.

Smith, M., Baldwin, I., Churchill, W., Paul, R., and Newman, P. (2009). The New College Vision and Laser Data Set. *International Journal for Robotics Research*, 28(5):595–599.

Smith, R., Self, M., and Cheeseman, P. (1990). *Estimating Uncertain Spatial Relationships in Robotics*, pages 167–193. Springer-Verlag New York, Inc., New York, New York, USA.

Steder, B., Grisetti, G., and Burgard, W. (2010). Robust place recognition for 3D range data based on point features. In *Proceedings of 2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1400–1405, Anchorage, Alaska, USA.

Tapus, A. and Siegwart, R. (2006). A cognitive modeling of space using fingerprints of places for mobile robot navigation. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pages 1188–1193, Orlando, Florida, USA.

Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., and Durrant-Whyte, H. (2004). Simultaneous Localization and Mapping with Sparse Extended Information Filters. *International Journal of Robotics Research*, 23(7/8):693–716.

Viola, P. and Jones, M. (2004). Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154.

Walthelm, A. (2002). A new approach to global self-localization with laser range scans in unstructured environments. In *Proceedings of IEEE Intelligent Vehicle Symposium*, pages 202–208, Versailles, France.

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

Zlot, R. and Bosse, M. (2009). Place recognition using keypoint similarities in 2D lidar maps. In Khatib, O., Kumar, V., and Pappas, G., editors, *Experimental Robotics*, volume 54 of *Springer Tracts in Advanced Robotics*, pages 363–372. Springer Berlin / Heidelberg.