Learning Loop Clusures by Boosting

Jeffrey M. Walls*, Ryan W. Wolcott†
Department of Mechanical Engineering*
Department of Computer Science and Engineering†
University of Michigan, Ann Arbor, Michigan 48109
Email: {jmwalls,rwolcott}@umich.edu

Abstract—**This paper details a loop closure detection method that cheaply classifies planar laser scans as originating from the same location or not with low error rate. Two methods are presented built around the popular supervised learning AdaBoost framework and its variant Gentle AdaBoost. A real world data set collected throughout the University of Michigan North Campus is used to both train and test this classification method. For benchmarking purposes, error rates are compared to k-Nearest Neighbor and Support Vector Machine classifiers.**

I. Introduction

Mobile robots operating in a priori unknown environments employ a class of algorithms known as simultaneous localization and mapping (SLAM) to build a model of their environment while concurrently localizing themselves within that environment. Although SLAM has been a well researched area, loop closure (detecting when the robot has returned to a previously observed area of the environment, essentially place recognition) remains a difficult task for many reasons including dynamic environments and varying viewpoints.

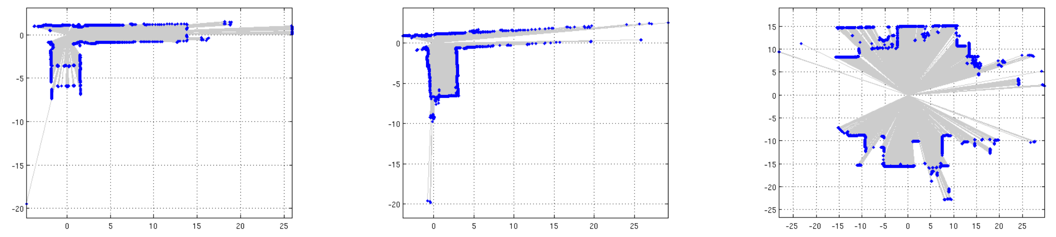
As a robot traverses an environment without access to absolute position references, such as GPS, it builds a pose estimate based on odometric observations. This method, known as dead-reckoning, produces an estimate whose error grows unbounded in time. The crux of SLAM algorithms is that by observing previously visited locations, the robot can bound its error growth. Identifying these loop-closures is a challenge, though, as the robot may be very uncertain of its position and, thus, unable to suggest likely candidate locations or at best suggest an extremely large number of potential locations. Comparing perceptual data is also, in general, an expensive task especially over large sets of high dimensional data.

This paper presents a supervised learning method for computing a binary classifier for planar laser range scans that enables effective loop closure detection invariant to differing viewpoints. A planar laser scan (i.e., Fig. 1) is characterized by a set of range points effectively describing a cross-section of the environment. Loop-closure detection entails determining whether or not the current scan is taken from the same part of the environment as a previous scan. Comparing scan pairs can be an arduous task over large data sets containing thousands of scans. In this work, we boost the performance of several simple weak classifiers learned via training over a small number of features extracted from scan data to collectively classify a scan pair as a possible loop-closure or not.

This paper is organized as follows: Section II describes previous work in the areas of laser scan-alignment and boosting algorithms. Section III formalizes the mathematical framework for this supervised learning problem and describes our novel contribution to this problem. Section IV elaborates on our data collection methodology as well as boosting implementation. Section V evaluates our algorithm performance. Finally, Section VI concludes. Section VII annotates group member contributions.

II. Previous Work

So called scan-matching algorithms, [1], [2], or more brute-force methods [3] seek to determine the rigid body transformation between two laser scans. These moderately expensive methods generally require large scan overlap and are sensitive to initial alignment. While these methods work quite well for inferring a rigid body transformation between sequential laser scans, the computation becomes



(a) Intersection (b) Intersection (c) Atrium

Fig. 1. Planar lidar scan examples. The first two images, Fig. 1(a) and Fig. 1(b), show scans from the same hallway intersection. The third scan example, Fig. 1(c), is from a different location. A loop closure detections algorithm should recognize the first two scans as being from the same place, but not the final scan.

too exhaustive to iteratively compare all previous observations to determine whether a pair is a valid loop closure.

As an alternative to scan-matching, several laser scan based algorithms isolate keypoints from the laser scans [4], [5]. These keypoints can then be integrated into a feature-based SLAM framework. However, these approaches do not inherently solve the challenge of detecting loop closures and other algorithms must still be used to detect loop closure candidates.

The original AdaBoost algorithm as described by Freund and Schapire [6] is a solution to the binary classification problem by utilizing multiple weak classifiers to form a single strong classifier by majority vote. In their analysis, they proved that the bounded error of the strong classifier converges to zero as long as the weak hypotheses are slightly better than random guessing. AdaBoost is the classification algorithm famously used for face recogniction in [7]. AdaBoost and its many variants, including multi-class algorithms, has enjoyed a surge in popularity due to its simplicity and performance [11].

A few examples throughout the literature report AdaBoost based classifiers used with laser scan data. The general approach quantizes each laser scan observation into descriptive scalar features (such as scan area, perimeter, centroid, etc.). For example, [8] uses AdaBoost on a subset of features to detect human legs within a scan region. Additionally, [9] uses a similar method for place classification, in which simple features are fed into a sequential AdaBoost algorithm that allows for a multi-class classifier.

Previous work by [10] tackles a supervised learning problem for building a laser scan classifier used for loop closure. This work most closely resembles our own, though our novel contribution is in using a smaller scan-derived feature vector as well as applying the AdaBoost variant, Gentle AdaBoost, to the loop closure classification problem. Furthermore, we present a comparison to two popular classifcation algorithms, k-nearest neighbor (k-NN) and support vector machine (SVM).

III. Boosting Place Recognition

A single planar lidar scan consists of a set of range return points that sweep out a cross section of the surrounding environment. Formally, the i^{th} scan is the set of M range return points relative to the laser reference frame,

$$S^i = \{(x_1^i, y_1^i), \ldots, (x_M^i, y_M^i)\} \in \mathbb{R}^{2\times M},$$

where the pair (x, y) is the location of a single laser point in cartesian coordinates. From each scan we extract a set of R features ($R \ll M$) based on both raw scan points and a polygonal approximation of the scan area, similar to [9], [10], such that for the j^{th} feature,

$$f_j(S^i) : \mathbb{R}^{2\times M} \rightarrow \mathbb{R}.$$

These features are described in detail in Appendix A. We define the feature vector for the k^{th} scan as

$$\mathbf{f}_k = [f_1(S^k) \ldots f_R(S^k)]^\top.$$

A scan pair input is characterized by the difference of the features between the two scans,

$$\mathrm{F}_i = \left|\mathbf{f}_{i_1} - \mathbf{f}_{i_2}\right|.$$

Given a range pair input, F_i, our goal is to classify the pair as a loop-closure or not. We use labeled data $\{(\mathrm{F}_1, y_1), \ldots, (\mathrm{F}_N, y_N)\}$, where

$$y_i = \begin{cases} 1 & \text{if the } i^{th} \text{ pairs match} \\ -1 & \text{otherwise} \end{cases}.$$

Each set of scan data contains N_p positive (matching) and N_n negative (nonmatching) scan pairs.

We use AdaBoost, referred to as Discrete AdaBoost, for scan pair classification because of its simplicity and scalability to large training sets. After a strong classifier is learned via AdaBoost, classifying new test inputs can be done in extremely rapidly, unlike K-Nearest Neighbors, for example, which requires computing the minimum distance to training inputs. AdaBoost also builds a parametric model, so that test classification complexity does not increase with the size of the training set, unlike k-NN. AdaBoost is also capable of modeling complex, nonlinear decision boundaries, while other methods, such as standard SVMs, determine a linear decision boundary. Weak classifiers computed via boosting also offer some interpretibility as to the importance of each feature element, whereas SVM parameters are less understandable.

A. AdaBoost

Discrete AdaBoost learns an additive model that can approximate a complex decision boundary. In particular, AdaBoost classifies test inputs by weighted majority vote over so-called ‘weak’ classifiers, or classifiers that can guess the class of the input roughly half of the time. The committee of these weak classifiers is a final strong classifier that performs remarkably well.

We design a weak classifier as a decision stump for each element or feature of the input. Therefore, the weak classifier for the j^{th} feature element, F^j, is

$$h_j(\mathrm{F}) = \begin{cases} 1 & \text{if } p_j \mathrm{F}^j < p_j \theta_j \\ -1 & \text{otherwise} \end{cases}$$

where θ_j is a threshold and p_j sets the direction of the inequality. The parameters (θ_j, p_j) are chosen to minimize the number of misclassified scans

$$(\theta_j, p_j) = \operatorname*{arg\,min}_{(\theta_j, p_j)} \sum_i w^i \mathbf{1}_{h_j(\mathrm{F}_i) \neq y_i},$$

where w^i is the weight given to each training data point.

Algorithm 1 Discrete AdaBoost Classifier

```
Require: training data $\{(\mathrm{F}_1, y_1), \ldots, (\mathrm{F}_N, y_N)\}$
    where F represents a scan pair and $y \in \{-1, 1\}$
    for loop closure or not
1: initialize weights $w_1^i = \frac{1}{2N_p}, \frac{1}{2N_n}$
2: for $t = 1, \ldots, T$ do
3:     normalize weights $w_t^i = \frac{w_t^i}{\sum_j w_t^j}$
4:     compute a weak classifier, $h_j$, that minimizes
       the weighted training error, $\epsilon_t$.
         $(\theta_j, p_j) = \arg\min_{(\theta_j, p_j)} \epsilon_t$
                    $= \arg\min_{(\theta_j, p_j)} \sum_i w_t^i \mathbf{1}_{h_j(\mathrm{F}_i) \neq y_i}$
5:     set $(h_t, \epsilon_t) = (h_j, \epsilon_j)$ for the classifier $j$ with
       the lowest error $\epsilon_j$
6:     update weights $w_{t+1}^i = w_t^i \beta_t^{1-e_i}$, where $e_i =$
       $0$ if $x_i$ is classified correctly, or $1$ otherwise,
       and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
7: end for
8: return final strong classifier
         $h(\mathrm{F}) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(\mathrm{F}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$
   where $\alpha_t = \log \frac{1}{\beta_t}$.
```

The process for learning a classifier under AdaBoost is described in Algorithm 1. A classifier is learned at each iteration over the weighted set of training data to minimize the classification error. Each weight is associated with a data point. The weights on each input point are adjusted so as to correctly classify previously misclassified training inputs. The weights are initialized to be uniform over the set of training data, i.e., no preference is given a priori to certain training data. The number of iterations, T, is chosen judiciously to minimize the training error while attempting to not overfit.

Fig. 2 illustrates the first two iterations of AdaBoost over training data. The first two iterations each select a weak classifier based on a single feature element. Each feature element is plotted, and the threshold value, θ, for each weak classifier. The final strong classifier after these two iterations is a weighted sum of each weak classifier.

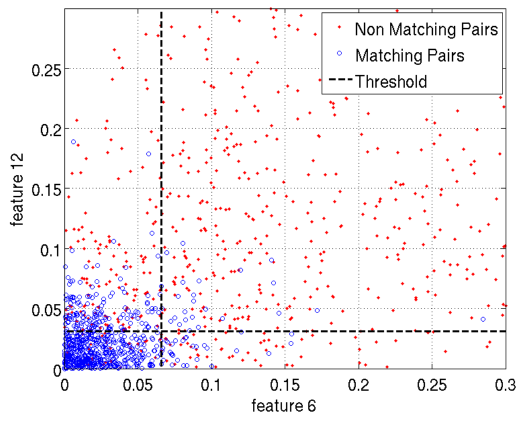


Fig. 2. AdaBoost classification example. Training data is plotted over two feature values. AdaBoost uses a weighted sum of the two weak classifiers to classify test inputs.

B. *Gentle AdaBoost*

We also implemented a variation of AdaBoost, Gentle AdaBoost [11], as empirical evidence throughout the literature suggests that Gentle AdaBoost outperforms AdaBoost in terms of stability, misclassification rate, and robustness to outliers. The Gentle AdaBoost algorithm is outlined in Algorithm 2.

Algorithm 2 Gentle AdaBoost Classifier

```
Require: training data $\{(\mathrm{F}_1, y_1), \dots, (\mathrm{F}_N, y_N)\}$ where F represents a scan pair and $y \in \{-1, 1\}$ for loop closure or not
1: initialize weights $w_1^i = \frac{1}{2N_p}, \frac{1}{2N_n}$
2: for $t = 1, \dots, T$ do
3:     normalize weights $w_t^i = \frac{w_t^i}{\sum_j w_t^j}$
4:     compute a weak classifier, $h_j$, as a regression function fitted by weighted least squares of $y$ to $\mathrm{F}^j$ that minimizes the weighted training error, $\epsilon_t$.
5:     set $(h_t, \epsilon_t) = (h_j, \epsilon_j)$ for the classifier $j$ with the lowest error $\epsilon_j$
6:     update weights $w_{t+1}^i = w_t^i \exp(-y_i h_t(\mathrm{F}_i))$.
7: end for
8: return final strong classifier
```

$$h(\mathrm{F}) = \begin{cases} 1 & \sum_{t=1}^{T} h_t(\mathrm{F}) \geq 0 \\ -1 & \text{otherwise} \end{cases}.$$

Gentle AdaBoost most noticably differs from AdaBoost in the form of the weak classifier. Where AdaBoost computes a simple decision stump at each iteration, Gentle AdaBoost fits a weighted least squares model to the training data. At each iteration, Gentle AdaBoost, as in Discrete AdaBoost, chooses the weighted least squares fit that minimizes the weighted error.

One important difference between Gentle AdaBoost and Discrete AdaBoost is that Gentle AdaBoost's final, strong classifier does not depend on the sum of log ratios. Therefore, the resulting prediction is more numerically stable because each weak classifier provides large updates that span pure, continuous regions of the sample space. This methodology will be less susceptible to outliers and should be able to reject a majority of them. Because our data is inherently noisy, Gentle AdaBoost is an ideal classifier for our problem.

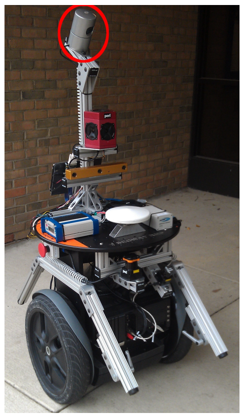
IV. Experiments

Data was accumulated using a Velodyne HDL-32E mounted on a Segway RMP 200 mobile robot. Using a single horizontal beam from this 3D scanning laser, we were able to extract a 360° cross section of the surrounding environment. Each scan contains on the order of 3000 points and was logged every 0.5m along the path of the Segway. We collected laser scans throughout various buildings on the North Campus of the University of Michigan including: NAME, FXB, EECS, Dow, CSE, the Duderstadt Center, and Pierpont Commons. Fig. 3 shows our test robot and accumulated laser scans for a portion of the covered area.

We manually labelled pairs of laser scans as matching ($y_i = 1$) by comparing pose estimates obtained through highly-accurate inertial sensors and wheel encoders. Nonmatching pairs ($y_i = -1$) were generated by randomly sampling scans that were taken from some threshold distance apart (50m to guarantee that there would be no scan overlap). Our final dataset consisted of 2436 scan pairs, with an even distribution of 1218 matching and 1218 non-matching scan pairs.

A. *Results*

10-fold cross validation was used to evaluate precision, recall, and overall error rates of Discrete AdaBoost and Gentle AdaBoost. The resulting rates


(a) Segway Robot

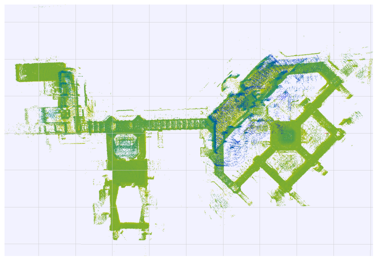

(b) Portion of University of Michigan North Campus

Fig. 3. Fig. 3(a) shows the Segway robot used for data collection, the Velodyne laser scanner is circled in red. Fig. 3(b) illustrates a set of laser scans collected for a portion of the University of Michigan North Campus.

over each fold were averaged and tabulated for varying iteration values, T. The results can be seen in Fig. 4.

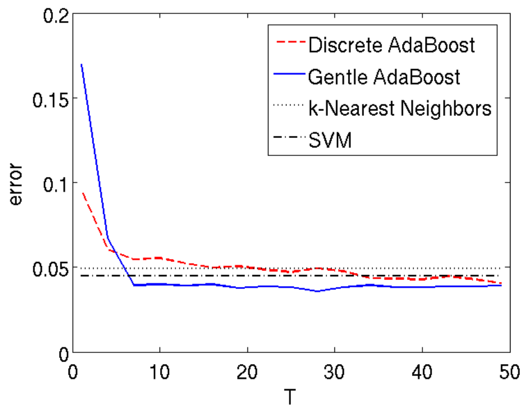


Fig. 4. Test error for number of boosting iteration steps. Both Discrete AdaBoost and Gentle AdaBoost demonstrate convergence in test error with number of iterations (weak classifiers) used. The convergence values of both boosting algorithms outperforms a k-NN classifier using $k = 5$ as well as an SVM, which both have test error rates of 4.9% and 4.4%, respectively.

The precision-recall curve is plotted in Fig. 5 for $T = 10$ for both Discrete AdaBoost and Gentle AdaBoost. These results were generated by adjusting the threshold value used to classify match or nonmatch, as each boosting algorithm returns a level of confidence in its classification.

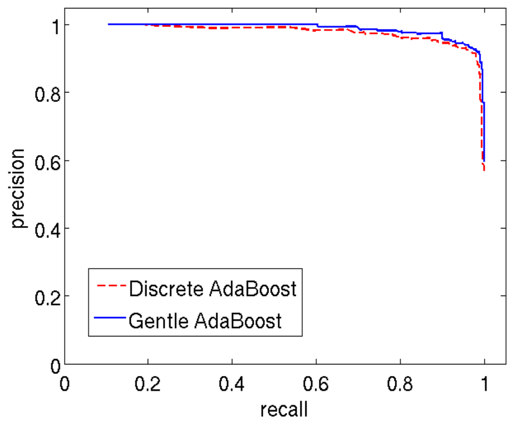


Fig. 5. Precision-recall curve for Discrete and Gentle AdaBoost.

V. Discussion

AdaBoost and its variant, Gentle AdaBoost, perform extremely well classifying pairs of planar laser scans as matches or nonmatches. Fig. 4 illustrates that by iteratively adding new weak classifiers to our final strong classifier, the overall classification error decreases until convergence. We can also see that with relatively few number of iterations or weak classifiers, both boosting algorithms obtain misclassification error rates less than 5%. Fig. 6 illustrates two misclassified scans resulting from our Discrete AdaBoost classifier.

At each boosting iteration, the algorithm chooses a classifier based on a single feature element of the input. In Discrete AdaBoost, this classifier is a simple threshold or decision stump. In Gentle AdaBooste, the classifier is a weighted least squares approximation. In both cases, we can consider that the most ‘informative’ feature is chosen at each step; the feature that produces the smallest weighted misclassification error is chosen. Table I lists the features selected by Discrete AdaBoost and Gentle AdaBoost for each of the first five iterations. After the addition of a new weak classifier during each iteration, we see that the error rate is reduced. Another interesting observation is that the algorithms do not always select a unique feature, as iterations two and four both select feature 12 for Gentle AdaBoost. Finally, note that each method chooses similar features, meaning that certain features separate the data more easily than others.

Since we have a level of confidence in each boost-

TABLE I
FEATURE SELECTION FOR DISCRETE AND GENTLE ADABOOST

AdaBoost Iteration:		1	2	3	4	5
Discrete	Feature Selected	6	12	14	4	3
	Error	9.8%	9.7%	6.9%	6.7%	6.2%
Gentle	Feature Selected	7	12	14	12	6
	Error	17.3%	11.9%	7.6%	6.5%	6.1%

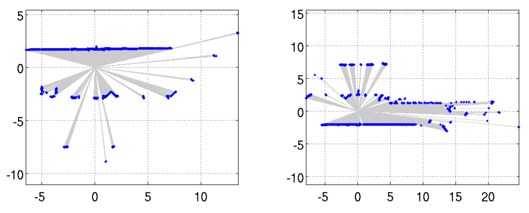
ing classification prediction, we can 'tune' the result of test input to obtain the desired precision-recall performance, Fig. 5. Precision-recall is incredibly important within a SLAM context, because although we want to maximize the number of true-positive pair matches, a single false-positive can cause estimation errors to diverge but false-negatives are not as harmful as long as they do not occur with high frequency.

We observe that Gentle AdaBoost outperforms Discrete AdaBoost in terms of iterations to classification error convergence, overall classification error and precision-recall characteristics. This is because Gentle AdaBoost can model a more complex decision boundary with each weak classifier than Discrete AdaBoost can with a simple decision stump. Additionally, Gentle AdaBoost is more robust to outliers as less weight is placed on them. Although training time is increased due to increased complexity with Gentle AdaBoost, testing each new point occurs in the same time as Discrete AdaBoost.

A. *Alternative Classification Methods*

We applied both k-NN and SVM in order to compare our methods to more common benchmark algorithms. In order to weight each feature equally within k-NN, we normalized the training data across each feature and used a simple Euclidean distance metric. Fig. 4 shows that both AdaBoost and Gentle AdaBoost compare favorably with both k-NN and SVM, which actually perform slightly worse for a given number of boosted weak classifiers, T. Computationally, classifiying a test point is cheaper with boosting algorithms because only a few simple weak classifiers must be evaluated, whereas with k-NN, distances to training points must be computed.

Furthermore, unlike k-NN, boosting returns a level of confidence in each prediction, allowing us to discount all but the most certain true positive classifications.


(a) False Negative

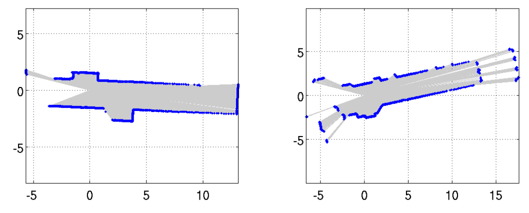

(b) False Positive

Fig. 6. Erroneous test classification examples. Each subfigure shows a misclassification by AdaBoost. We can see in each case how AdaBoost misses the correct class because either the scans look quite different although originating from similar places, Fig. 6(a), or the scans look similar despite being from different locations, Fig. 6(b)

VI. CONCLUSION

We have presented a supervised learning algorithm for the place recognition binary classification problem based on planar laser range data. Both Discrete AdaBoost and Gentle AdaBoost implementations have proven effective for classification. Each algorithm displayed convergence in error rate with the number of iterations through the training data. Furthermore, since the classifiers are learned offline, real-time place recognition is computationally cheap and runs in fixed time, which is an important property for field robotics. Additionally, since the two studied boosting algorithms return a notion of confidence, a threshold can be adjusted to improve the precision-recall properties of the classifier. This effectively allows for lower false positive occurences, which can lead to divergence in position estimation for robotic applications.

VII. GROUP MEMBER ACCOMPLISHMENTS

During this project, we jointly accomplished all tasks.

APPENDIX A
PLANAR LIDAR FEATURE DEFINITIONS

1) Centroid: distance from the origin to the mean scan position.
2) Close Area: polygonal approximation of scan area.
3) Close Distance: perimeter approximation.
4) Circular Radius: fits a circle to the scan in XY space.
5) Circular Residual: sum squared error of the circular fit to each scan.
6) Curvature Mean: mean curvature of the scan.
7) Curvature Standard Deviation: standard deviation of scan curvature.
8) Distance: perimeter of scan.
9) Number of Groups: number of groups, where a group is a cluster of consecutive scan points within a gate threshold.
10) Mean Group Size: mean number of points per group.
11) Mean Angular Difference: mean angular width between scan points.
12) Mean Deviation: sample standard deviatiton of scan.
13) Regularity: sample standard deviation of distance between consecutive scan points.
14) Size: number of points in scan.
15) Standard Deviation of Mean: pointwise distance standard deviation to the mean.
16) Standard Deviation of Ranges: sample standard deviation of the range of each scan point.

REFERENCES

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

[2] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[3] E. Olson, "Real-time correlative scan matching," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, June 2009, pp. 4387–4393.

[4] Y. Li and E. Olson, "Extracting general-purpose features from LIDAR data," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2010.

[5] R. Zlot and M. Bosse, "Place recognition using keypoint similarities in 2d lidar maps." in *ISER'08*, 2008, pp. 363–372.

[6] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[7] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

[8] K. O. Arras, O. M. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2D range data," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007, pp. 3402–3407. [Online]. Available: http://www.informatik.uni-freiburg.de/ omartine/publications/arras2007icra.pdf

[9] O. Mozos, C. Stachniss, and W. Burgard, "Supervised learning of places from range data using adaboost," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, april 2005, pp. 1730 – 1735.

[10] K. Granström, J. Callmer, F. Ramos, and J. Nieto, "Learning to detect loop closure from range data," in *Proceedings of 2009 IEEE International Conference on Robotics and Automation*, A. Bicchi, Ed., May 2009, pp. 15–22.

[11] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.