Experimental comparison of AdaBoost algorithms applied on Leg Detection with Different Range Sensor Setups

Srećko Jurić-Kavelj and Ivan Petrović
Department of Control and Computer Engineering
University of Zagreb, Faculty of Electrical Engineering and Computing
Zagreb, Croatia
{srecko.juric-kavelj, ivan.petrovic}@fer.hr

***Abstract*—When tracking people or other moving objects with a mobile robot, detection is the first and most critical step. At first most researchers focused on the tracking algorithms, but recently AdaBoost (supervised machine learning technique) was used for people legs detection in 2D range data. The results are promising, but it is unclear if the obtained classifier could be used on the data from another sensor. As it would be a huge inconvenience having to train a classifier for every sensor (setup), we set out to find if, and when is a classifier trained on one sensor setup transferable to another sensor setup. We tested two sensors in five different setups. In total, we acquired 2455 range scans. Experiments showed that the classifier trained on *noisier* sensor data performed better at classification of data coming from other sensor setups. Classifiers trained on less noisy data were shown to be overconfident, and performed poorly on noisy data. Furthermore, experiments showed that classifiers learned on ten times smaller datasets performed as good as classifiers trained on larger datasets. Since AdaBoost is a supervised learning technique, obtaining same classifier efficiency with significantly smaller dataset means less hand labeling of the data for the same results.**

***Index Terms*—AdaBoost, 2D range, legs, SICK, Hokuyo**

I. Introduction

Application of mobile robots as supporting or autonomous agents in working environments alongside people is becoming increasingly popular. In order to perform well, in addition to be able to localize themselves, robots must have the capacity to detect and track other moving objects. Given their precision and high density of raw useful data, laser range scanners quickly became sensors of choice for such problems among researchers. Due to that popularity today we have a fair choice of different laser scanners. As it would be a huge inconvenience having to train a classifier to detect legs for every sensor (setup), we set out to find if, and when is a classifier trained on one sensor setup transferable to another sensor setup.

This paper is organised as follows. In section II we discuss related work. Boosting and measured scan segment features are described in section III. Experimental setup and results are presented in section IV. Final thoughts and considerations of future work are given in section V.

II. Related work

Previously, in context of tracking people, or other moving objects, in range sensor data the focus was on the tracking algorithms. Little attention was given to the problem of extracting useful observations from range scan data. Most often, geometric or motion features with simple rules of thumb were used to detect segments corresponding to moving objects. Same is true for our previous work [1].

Arras et al. [2] discuss previous approaches in grater detail and they propose construction of a strong scan segment classifier using AdaBoost on simple features. They used AdaBoost variant similar to RealAdaBoost from [3] and simple decision *stumps*[1] as weak classifiers. They gave number of iterations as a rule of thumb, based on few experiments they conducted with different number of iterations. They compared their classifiers on data acquired in different environments.

In this paper we experimentally compare three AdaBoost algorithms paired with three different weak learners on five different laser scanning configurations. Firstly, we proposed a criterion function in order to chose the best variant and number of iterations used to train a classifier. Further, the selected variant was used to analyze if it is possible to use a classifier learned on data from one sensor setup and apply it to classify data from other sensor setup. The obtained results are discussed and some recommendations are given regarding the segment features used for weak learner training and sensor data used for classifier boosting.

III. Boosting weak classifiers

Boosting is a method for obtaining highly accurate classifier by combining many *weak* classifiers. Each *weak* classifier is required only to be *moderately* accurate, i.e. better than random guessing.

In this work we will use variants of the AdaBoost algorithm that was first introduced in [4] and implemented in [3]. Generalized version of AdaBoost algorithm is given in Algorithm 1. More about AdaBoost algorithm variants used in this work is discussed in section IV.

[1]same as 1-node decision tree

Input data for the algorithm variants used to solve binary classification problems is set of tuples (e_i, l_i) where e_i is an example, e.g. scan segment in our case, and $l_i \in \{\pm 1\}$ is example label, where value of 1 indicates positive, and value of -1 indicates a negative example. Training is done in number of iterations $t = 1, \dots, T$. In each iteration algorithm selects a weak classifier $h_t(e)$, that we expect to be better than random guessing for the training examples weighted by distribution D_t. At the end of each iteration distribution for the next iteration is updated in such a way that more weight is given to incorrectly classified examples. The final classifier is weighted majority vote of T weak classifiers selected during training.

Algorithm 1: A generalized AdaBoost algorithm

```
**Data**: Set of tuples $(e_1, l_1), \dots, (e_N, l_N)$ where $e_i \in \mathcal{E}$,
      $l_i \in \{\pm 1\}$
**Result**: Strong classifier, cascade of *weak* classifiers
      $F(e) = \sum_{t=1}^{T} \alpha_t h_t(e)$
$D_1(n) \leftarrow \frac{1}{N}$;
**for** $t \leftarrow 1$ **to** $T$ **do**
    Train weak learner using $D_t$.;
    Get weak hypothesis $h_t : \mathcal{E} \to \mathbb{R}$. ;
    Chose $\alpha_t \in \mathbb{R}$. ;
    $D_{t+1}(i) \leftarrow \frac{D_t(i)\exp(-\alpha_t l_i h_t(e_i))}{Z_t}$ ;   /* where $Z_t$
    is normalization factor */
```

A. Scan segment features

Here we describe which set of features of laser range scan segments that were available to weak learners. Also, we present our segmentation method. But firstly, let us introduce mathematical model of a laser range scanner. Scan data delivered by the laser range scanner is set $Z = \{b_1, \dots, b_L\}$ of beams. Each beam b_i is a tuple (ϕ_i, r_i), where ϕ_i is the angle of the beam relative to the laser coordinate frame, and r_i is the length of the beam. Beam is assumed to spread freely in space until it hits an obstacle, i.e. beam length is the distance to the nearest obstacle in the beam's direction. Laser coordinate frame is usually set such that the origin is in the sensor's center, x axis points in front of the laser range sensor and xy-plane corresponds to scanning plane.

Scan segmentation was performed as in our previous work [1]. Two consecutive range scan points are considered as part of the same segment if

$$\begin{aligned} &d(b_j, b_{j+1}) \leq \\ &C_0 + C_1 \min(r_j, r_{j+1}) \approx \\ &C_0 + \frac{3}{2}\Delta\alpha \min(r_i, r_{i+1}) \ , \end{aligned} \tag{1}$$

where $d(b_j, b_{j+1}) = \sqrt{r_j^2 + r_{j+1}^2 - 2r_j r_{j+1}\cos\Delta\alpha}$ is the distance between two consecutive scan points, $\Delta\alpha = \phi_{j+1} - \phi_j$ is the angular resolution and C_0 is a constant that compensates for noise in range scan data. Constant $C_1 = \sqrt{\frac{9}{2}(1-\cos\Delta\alpha)} \approx \frac{3}{2}\Delta\alpha$ compensated for consecutive beams increasing inter distance, proportionally to the beam's length.

The output of the segmentation function is an ordered sequence $\mathcal{P} = \{S_1, \dots, S_M\}$ of segments, such that $\cup S_i \subset Z^2$. Each segment consists of beams that correspond to it $S = \{b_s, \dots, b_e\}$. Indexes s and e indicate starting and ending beams of a segment, respectively. When performing calculations, we will use beam representation in Cartesian coordinates $\mathbf{x} = (x, y)$, where $x = r\cos(\phi)$ and $y = r\sin(\phi)$. In context of AdaBoost input data, segment S is a first element of example tuple.

Feature f is defined as a function $f : \mathcal{S} \to \mathbb{R}$, where $\mathcal{S}$ represents a set of all possible segments. Function f takes a segment S as an argument and returns a real value, value of a measured feature. We use most of the features introduced in [2], with the exception of a *mean speed* feature, and introduce a simple extra feature, *distance of the segment mean*. For completeness, all features are listed below:

1) Number of points: $n = |S_i|$.
2) Standard deviation: This feature is given by
$$\sigma = \sqrt{\frac{1}{n-1}\sum_j \|\mathbf{x}_j - \overline{\mathbf{x}}\|^2} \ ,$$
where $\overline{\mathbf{x}}$ denotes the segment mean (center of gravity).
3) Mean average deviation from median: This feature is designed to measure the segment compactness more robustly than the standard deviation. The median of a distribution $f(x)$ is the value where the cumulative distribution function $F(x) = \frac{1}{2}$. Given an ordered set of K scalar random samples x_i the median $\tilde{x}$ is defined as
$$\tilde{x} = \begin{cases} x_{\frac{K+1}{2}} & \text{if } K \text{ is odd} \\ \frac{1}{2}(x_{\frac{K}{2}} + x_{\frac{K}{2}+1}) & \text{if } K \text{ is even} \end{cases}$$
Opposed to the mean, median is less sensitive to outliers. In our two dimensional case, we calculate $\tilde{\mathbf{x}}$ using the vector of medians approach, i.e. $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y})$. The average deviation from the median is then
$$\varsigma = \frac{1}{n}\sum_j \|\mathbf{x}_j - \tilde{\mathbf{x}}\| \ .$$
For a more in depth discussion about median definition in multi-dimensional case see [5].
4) Segment mean distance: The Euclidean distance between laser range scanner and the segment mean, i.e. $\|\overline{\mathbf{x}}\|$.
5) Jump distance from preceding segment: This feature corresponds to the Euclidean distance between the first point of S_i and the last point of S_{i-1}. For S_1 this feature is set to the length of the first beam in the segment.
6) Jump distance to succeeding segment: The Euclidean distance between the last point of S_i and the first point of S_{i+1}. For the last segment S_M this feature is set to the length of the last beam in the segment.

[2]we observed this to be strict subset in all our scans

7) Width: This feature measures the Euclidean distance between the first and last point of a segment.
8) Linearity: This feature measures the straightness of the segment and corresponds to the residual sum of squares to a line fitted into the segment in the least squares sense. Given the segment points in polar coordinates $b_j = (\phi_j, r_j)$, fitting a line in the Hessian (α, ρ) representation that minimizes perpendicular errors from the points onto the line has a closed form solution. We use the (unweighed) expressions from [6]. Once the line parameters (α, ρ) are found, the residual sum of squares is calculated as

$$s_l = \sum_j (x_j \cos(\alpha) + y_j \sin(\alpha) - \rho)^2 \ ,$$

where $x_j = r_j \cos(\phi_j)$ and $y_j = r_j \sin(\phi_j)$.
9) Circularity: This feature measures the circularity of a segment. Like for the previous feature, we sum up the squared residuals to a fitted circle. Given a set of points in Cartesian coordinates, an elegant and fast way to find the best circle in the least squares sense is to parametrize the problem by the vector of unknowns as $x = [x_c \ y_c \ x_c^2 + y_c^2 - r_c]^T$ where x_c, y_c and r_c denote the circle center and radius. With this, the over-determined equation system $Ax = b$ can be established,

$$A = \begin{bmatrix} -2x_1 & -2y_1 & 1 \\ -2x_2 & -2y_2 & 1 \\ \vdots & \vdots & \vdots \\ -2x_n & -2y_n & 1 \end{bmatrix} \quad b = \begin{bmatrix} -x_1^2 - y_1^2 \\ -x_2^2 - y_2^2 \\ \vdots \\ -x_n^2 - y_n^2 \end{bmatrix}$$

and can be solved e.g. using pseudo-inverse $x = (A^T A)^{-1} A^T b$. The residual sum of squares is then

$$s_c = \sum_j \left(r_c - \sqrt{(x_c - x_j)^2 + (y_c - y_j)^2} \right)^2 \ .$$

This parametrization of the least squares problem has better geometric properties than the approach used by Song et al. [7]. When geometry plays a role in fitting (opposed, e.g., to regression in statistics), care has to be taken what errors are minimized. Otherwise algebraically correct but geometrical useless least squares fits can be the result.
10) Radius: This feature is the radius r_c of the circle fitted to the segment. It corresponds to an alternative measure of the size of a segment S_i.
11) Boundary length: This feature measures the length

$$l = \sum_j d(b_j, b_{j+1})$$

of the poly-line corresponding to the segment.
12) Boundary regularity: Here we calculate the standard deviation of the distances $d(b_j, b_{j+1})$ of adjacent points in a segment.
13) Mean curvature: The average curvature $\overline{k} = \sum_j \hat{k}_j$ over the segment S_i is calculated using the following curvature approximation. Given a succession of three points $\mathbf{x}_A$, $\mathbf{x}_B$ and $\mathbf{x}_C$, let A denote the area of the triangle $\mathbf{x}_A\mathbf{x}_B\mathbf{x}_C$ and let d_A, d_B, d_C denote the three distances between the points. Then, an approximation of the discrete curvature of the boundary at $\mathbf{x}_B$ is given by

$$\hat{k} = \frac{4A}{d_A d_B d_C} \ .$$

This is an alternative measurement of r_c as curvature and radius are inverse proportional.
14) Mean angular difference: This feature traverses the segment boundary and calculates the average of the angles β_j between the vectors $\overline{\mathbf{x}_{j-1}\mathbf{x}_j}$ and $\overline{\mathbf{x}_j\mathbf{x}_{j+1}}$, i.e.

$$\beta_j = \angle(\overline{\mathbf{x}_{j-1}\mathbf{x}_j}, \overline{\mathbf{x}_j\mathbf{x}_{j+1}}) \ .$$

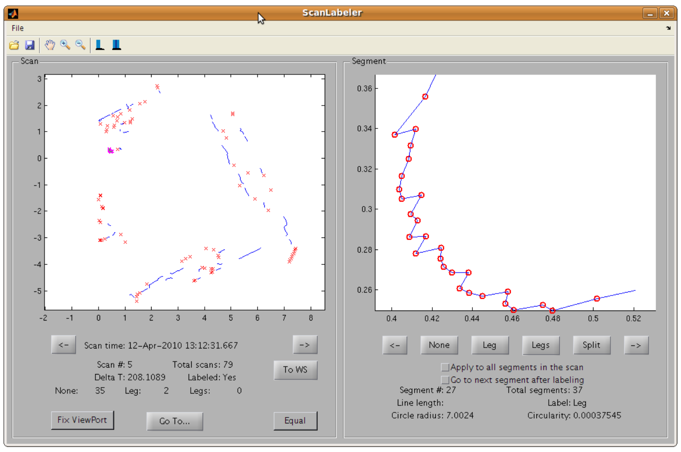
Care has to be taken that the angle differences are properly unwrapped. This feature is a measure of convexity/concavity of a segment.

IV. Experiments

Two laser range scanners were available for experiments, SICK and Hokuyo. The experiments were designed to test if a strong classifier trained on a dataset of one sensor, can be transfered to another, i.e. can it be used to classify data coming from another sensor. SICK was connected in high speed mode, over RS422 serial connection, so we were able to achieve SICK's maximal data rate of 75 Hz. Since SICK can be configured to take scans at different field of view, different angular resolution and speed, we decided to take four representative setups and test them all individually. Maximal field of view of 180° was taken in all four setups. They are differentiated with resolution and connection speed parameters. We chose 1° and 0.5° angular resolutions[3], 500 kbps and 38.4 kbps connection speeds and thus defined our four SICK setups. Scans were taken in a classroom, where tables and chairs were temporarily placed around the classroom by the walls, in order to achieve clear line of sight to the target. Both sensors were mounted[4] on a Pioneer 3-DX mobile robot, which was placed in a corner of the classroom. In each scan sequence one person walked from the corner where the mobile robot was, entering laser's field of view on the left, to the opposite corner of a classroom and walked out the door (and lasers maximum range), and back. Grand total of 2455 scans were taken over 88.375 seconds time span. Total number of segments extracted was 65888. The segments with at least 3 points were considered for labeling. To aid the daunting task of labeling all those segments, a GUIDE application in Matlab® was developed. Application GUI is shown in Fig. 1. ScanLabeler features segment labeling with a mouse click and global shortcut keys in order to maximally reduce labeling time. Later, segments consisting of exactly 3 points were discarded, for comparison of trained classifiers. The resulting data set consisted of 50320 segments.

[3]SICK does not have 0.5° angular resolution, it is achieved by taking two scans of 1° resolution with 0.5° offset

[4]Hokuyo was mounted facing backwards, so the robot was rotated before acquiring data with Hokuyo



Scan time: 12-Apr-2010 13:12:31.667

Scan #: 5 Total scans: 79

Delta T: 208.1089 Labeled: Yes

None: 35 Leg: 2 Legs: 0

Apply to all segments in the scan

Go to next segment after labeling

Segment #: 27 Total segments: 37

Line length: Label: Leg

Circle radius: 7.0024 Circularity: 0.00037545

Fig. 1. ScanLabeler GUI

GML AdaBoost toolbox features three variants of AdaBoost algorithm and decision tree as a weak learner. The implemented algorithms are described in [8]–[11].

A. *Classifiers on each sensor setup*

To get an idea how those algorithms perform, we applied each on all datasets with segments that contained a minimum of 4 points. Furthermore, we used 3 different weak learners, i.e. decision trees with 1, 2, and 3 nodes. Hundred training iterations were conducted with each algorithm, and at each iteration error rate was recorded. Training data was selected by taking every odd scan, and control data was selected by taking every even scan. Results are shown in Figs. 2, 3 and 4. Colors are assigned to algorithm variants. Blue depicts results of Real, green depicts results of Gentle and red depicts results of Modest AdaBoost. Solid line represent 1-node, dashed line represents 2-node and dotted line represent 3-node decision tree.

General remark for the result of this experiment is that Modest AdaBoost variant rarely shows characteristics of a boosting algorithm. On most datasets it gets stuck at the first iteration, and in some the error rate grows with the number of iterations.

Results shown in Fig. 2 are from datasets that were acquired by SICK laser range scanner operating in high speed mode, i.e. 75 Hz for $\Delta\alpha = 1^\circ$ and 37.5 Hz for $\Delta\alpha = 0.5^\circ$. Since these datasets contain largest amount of scans and segments, small declining trend of error rate is observed even when number of iterations approaches 100.

Results in Figs. 3 and 4 are from datasets acquired at considerably lower rates, namely 9.1 Hz for SICK at $\Delta\alpha = 1^\circ$ angular resolution, 4.6 Hz for $\Delta\alpha = 0.5^\circ$ angular resolution of the same sensor and 10 Hz for Hokuyo. Small size of those data sets is evident in error rate increase that even one incorrectly labeled control example contributes.

B. *Applying classifier to different sensor setup*

In order to select the best algorithm and weak learner from the previous experiment we propose the following criterion

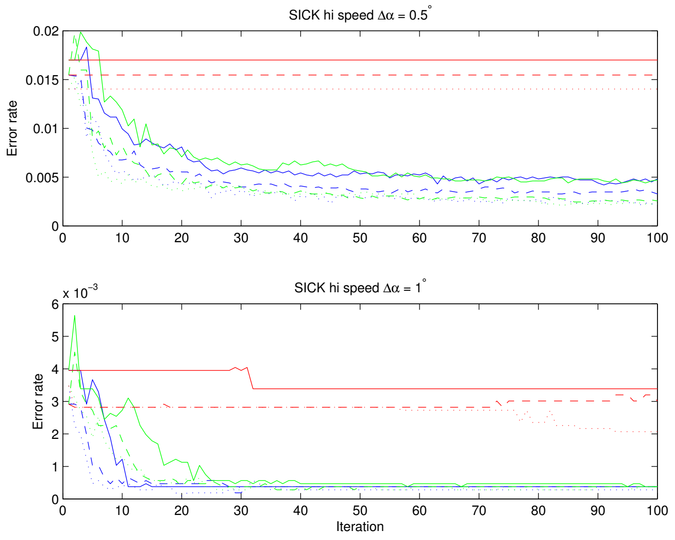


Fig. 2. SICK High Speed Datasets

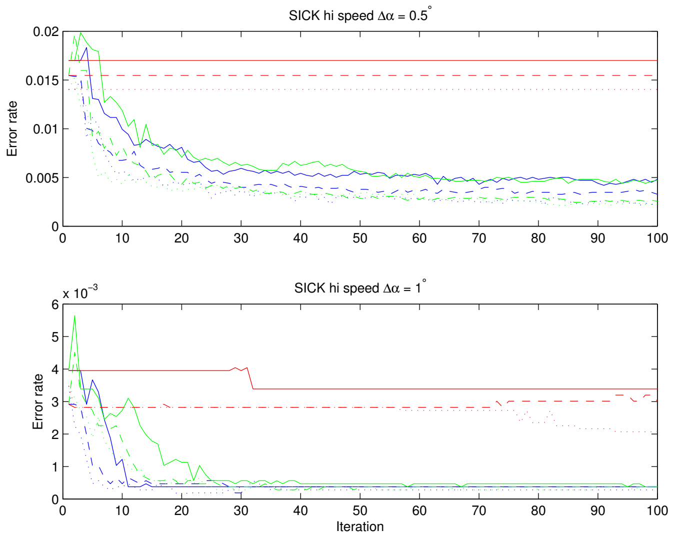


Fig. 3. SICK Low Speed Datasets

function:

$$J_{k,a,n} = \sum_{DatS} ER^2_{k,a,n} W(F_{k,a,n}) \ , \qquad (2)$$

where $DatS$ stands for datasets, ER is the error rate of the final classifier, $W(F_{k,a,n})$ is the weight of the obtained classifier. Indexes k, a and n stand for iteration, algorithm variant and the number of nodes in the decision tree respectively. Classifier weight can be defined in following way:

$$W(F_{k,a,n}) = \sum_{f \in F_{k,a,n}} w(f) \approx (1 - C_k + kC_k)(1 - C_n + nC_n) \ , \qquad (3)$$

where f represents a feature used in final classifier in any number of occasions, and $w(f)$ is the cost of a feature, e.g. time needed to calculate it for an average scan segment. We simplified the calculation of classifier weight by considering

TABLE I
CLASSIFIER APPLIED ON OTHER DATA SETS

		Data sets				
		SH1	SH0.5	SL1	SL0.5	Hok
Classifiers	SH1	6 7	6767 29	2 0	841 20	2800 0
		0.12%	34.79%	0.08%	35.42%	60.698%
	SH0.5	38 84	22 66	3 22	3 50	39 20
		0.57%	0.90%	1.01%	2.18%	1.28%
	SL1	14 32	5213 163	2 3	638 44	1845 0
		0.22%	27.52%	0.40%	28.05%	39.996%
	SL0.5	11 121	30 296	0 17	0 36	150 15
		0.62%	1.67%	0.69%	3.01%	3.58%
	Hok	1289 115	613 295	148 11	74 67	4 5
		6.60%	4.65%	6.44%	5.80%	0.39%

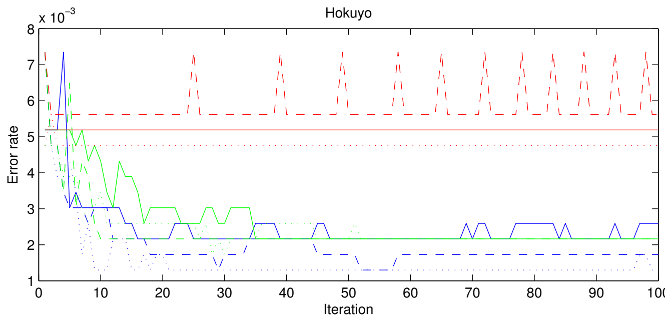


Fig. 4. Hokuyo Dataset

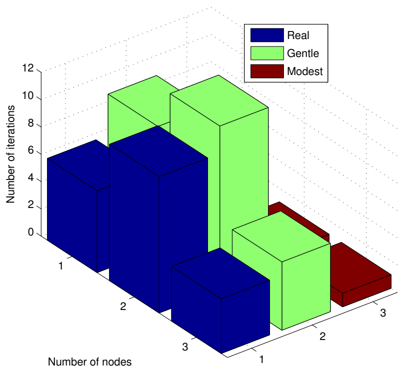


Fig. 5. Minimising number of iterations for a given variant

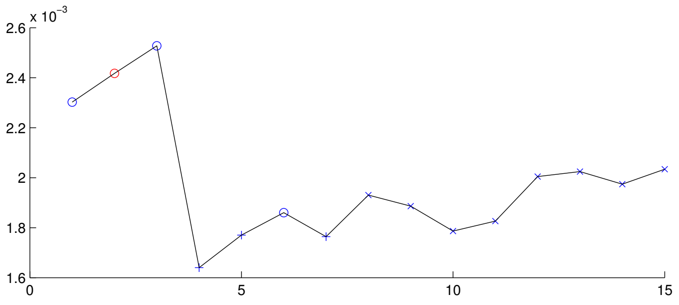


Fig. 6. Minimising error rate at given number of iterations

only number of iterations and number of nodes in decision tree. We can then check at which iteration we achieve minimum of $J_{k,a,n}$ for a particular choice of algorithm variant and number of nodes (a and n). Results of this check are depicted in Fig. 5. Alternatively, we can look for an iteration where minimum $J_{k,a,n}$ is achieved, regardless of the algorithm. In that way, we can find optimal number of iterations, algorithm variant and number of nodes in a weak learner with respect to (2). The results are shown in Fig. 6. Here again blue, green and red stand for Real, Gentle and Modest AdaBoost algorithm respectively. Circles stand for 1-node, 'x' for 2-node and '+' for 3-node decision tree. As can be seen, classifier boosted with Gentle AdaBoost was generally inferior to Real AdaBoost with respect to proposed criterion function (2) and only in the case of 2 iterations was Modest AdaBoost with 1-node decision tree better than any other variant.

Given the results in Fig. 6, we chose Real AdaBoost with 3 node decision trees for subsequent analysis. We will perform only 4 iterations to obtain the strong classifier. A strong classifier was trained for each data set. The resulting classifier was then tested on all other data sets. Results are shown in Table I. For each comparison three values are provided, number of false positives (left), number of false negatives (right) and error rate. SICK (S) data sets/classifiers were labeled according to speed (L - low speed, H - high speed) and angular resolution (0.5 - 0.5°, 1 - 1°). For example SL0.5 represents data set/classifier of SICK laser scanner configured in low speed mode (38.4 kbps) with $\Delta\alpha = 0.5°$ angular resolution. As you may notice, there are also results on the table diagonal. Those results were obtained by splitting the dataset in question in two parts, one of which was used for training and other for control. Training data consisted of the odd range scans, and control data consisted of even range scans.

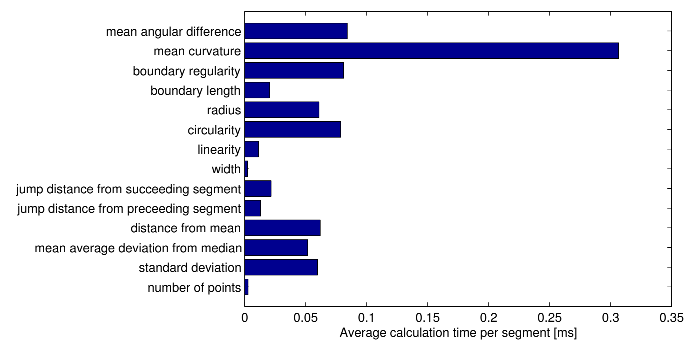


Fig. 7. Time needed to calculate segment features

C. *Discussion of the results*

As can be seen in Table I, classifiers *SH1* and *SL1* would be hardly recommended to classify data coming from the same sensor with $\Delta\alpha = 0.5°$ angular resolution, let alone Hokuyo.

On the other hand, it is obvious that the classifiers *SH0.5* and *SL0.5* can be used to classify data coming from other sensors/configurations very accurately. Much more interesting are the results of the classifier *SL0.5* That classifier has very acceptable error rate on all datasets. Therefore, one could label training data only on such data set, which has very low data rate, i.e. small number of laser ranges and segments to label and use that classifier on data at much higher rate.

Classifier *Hok* has moderate error rate on other datasets, but not as good as *SL0.5* has.

We took a closer look at classifiers *SH1* and *SH0.5*. *SH1* uses following features in decision tree: 2, 3, 4, 6, 7, 8, 13, and *SH0.5* uses features: 2, 4, 5, 6, 7, 8, 10, 11, 14. Significant difference between *SH1* and *SH0.5* can be seen in the choice of feature to measure inscribed circle. While *SH1* uses feature 13, SH0.5 uses feature 10. Those features should be inversely proportional, but if we look at the data, we can quickly see why that statement does not hold. To acheive $\Delta\alpha = 0.5°$ angular resolution, SICK takes two scans with $\Delta\alpha = 1°$ angular resolution, offsetted for $0.5°$. A zig-zag pattern can be found in segments coming from moving people legs, as can be seen in Fig. 1. The pattern can be explained with the fact that the segment actually consists of two scans taken with small but significant temporal offset. When fitting a circle in such zig-zag pattern we usually get lower radius compared to fitting a circle in less noisy data. Furthermore, feature 13 (mean circularity) has highest average calculation time per segment, as shown in Fig. 7. When we dismissed features 10 and 13 from training, we got error rate of 21.77% when applied *SH1* classifier on *SH0.5* data. This is still pretty bad result, but we would definitely advise not to use the mean curvature feature (numbered 13), since it takes almost four times more processing time compared to other features.

As for Modest AdaBoost algorithm variant, we could not explain why it fails to boost on our data sets. Small boosting was observed on *SH1* and *SL1* datasets in second iteration with 2-node decision tree. Therefore, we used such settings to obtain classifier on *SH1* dataset. Resulting classifier was tested on *SH0.5* dataset. We observed 6129 false positives and 44 false negatives which amounts to 31.94% error rate. Those results are slightly better then four iterations of Real AdaBoost with 3-node decision trees on same datasets, but hardly enough to conclude that Modest variant generalizes better.

V. Conclusion and future work

In this paper we proposed a criterion function to select the best AdaBoost algorithm and weak learner combination, and number of iterations in order to train a classifier for detection of people legs in laser range scans. We recorded five datasets, one using Hokuyo laser range scanner, and other four with different configurations of SICK range scanner. Selected variant was used to construct a classifier for each dataset. Resulting classifiers were tested on a task of classification of segments from other datasets, and the results are listed in Table I.

It was experimentally shown that classifier trained on SICK laser scanner data taken with angular resolution of $\Delta\alpha = 0.5°$ performed best on all datasets. We concluded that the mode of operation when $\Delta\alpha = 0.5°$ introduced more noise in segment data, especially positive examples. In turn, that made weak classifiers chose more robust features (estimators) of certain geometric properties, since many of the 14 features introduced were redundant.

For future work we plan to tackle multi-class problems with AdaBoost, e.g. learn a classifier that distinguishes leg, legs and other mobile robots.

Acknowledgment

This work has been supported by the Ministry of Science, Education and Sports of the Republic of Croatia under grant No. 036–0363078–3018.

References

[1] S. Jurić-Kavelj, M. Seder, and I. Petrović, “Tracking Multiple Moving Objects Using Adaptive Sample-based Joint Probabilistic Data Association Filter,” in *Proceedings of 5th International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2008)*, pp. 99–104, 2008.

[2] K. Arras, O. Mozos, and W. Burgard, “Using boosted features for the detection of people in 2d range data,” *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, 2007.

[3] A. Vezhnevets, “GML AdaBoost Matlab Toolbox — Graphics and Media Lab.”

[4] Y. Freund and R. Schapire, “A desicion-theoretic generalization of on-line learning and an application to boosting,” *Computational Learning Theory*, 1995.

[5] G. Aloupis, *On computing geometric estimators of location*. PhD thesis, 2001.

[6] K. Arras, *Feature-based robot navigation in known and unknown environments*. PhD thesis, 2003.

[7] Z. Song, Y. Chen, L. Ma, and Y. Chung, “Some sensing and perception techniques for an omnidirectional ground vehicle with a laser scanner,” *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*, 2005.

[8] R. Schapire and Y. Singer, “Improved boosting algorithms using confidence-rated predictions,” *Machine learning*, 1999.

[9] P. Viola and M. Jones, “Robust real-time object detection,” *International Journal of Computer Vision*, 2002.

[10] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: A statistical view of boosting,” *Annals of statistics*, 2000.

[11] A. Vezhnevets and V. Vezhnevets, “Modest AdaBoost – teaching AdaBoost to generalize better,” *Graphicon 2005*, 2005.