Virtual Sensors for Human Concepts - Building Detection by an Outdoor Mobile Robot

Martin Persson[1], Tom Duckett[2] and Achim Lilienthal[1]
[1]*Center for Applied Autonomous Sensor Systems*
Dept. of Technology, Örebro University, Sweden
{martin.persson,achim.lilienthal}@tech.oru.se
[2]*Department of Computing and Informatics*
University of Lincoln, Lincoln, UK
tduckett@lincoln.ac.uk

***Abstract*— In human-robot communication it is often important to relate robot sensor readings to concepts used by humans. We suggest to use a virtual sensor (one or several physical sensors with a dedicated signal processing unit for recognition of real world concepts) and a method with which the virtual sensor can be learned from a set of generic features. The virtual sensor robustly establishes the link between sensor data and a particular human concept. In this work, we present a virtual sensor for building detection that uses vision and machine learning to classify image content in a particular direction as buildings or non-buildings. The virtual sensor is trained on a diverse set of image data, using features extracted from gray level images. The features are based on edge orientation, configurations of these edges, and on gray level clustering. To combine these features, the AdaBoost algorithm is applied. Our experiments with an outdoor mobile robot show that the method is able to separate buildings from nature with a high classification rate, and extrapolate well to images collected under different conditions. Finally, the virtual sensor is applied on the mobile robot, combining classifications of sub-images from a panoramic view with spatial information (location and orientation of the robot) in order to communicate the likely locations of buildings to a remote human operator.**

***Index Terms*— Automatic building detection, virtual sensor, vision, AdaBoost, Bayes classifier**

I. Introduction

The use of human spatial concepts is very important in, e.g., robot-human communication. Skubic *et al.* [1] discussed the benefits of linguistic spatial descriptions for different types of robot control, and pointed out that this is especially important when there are novice robot users. In those situations it is necessary for the robot to be able to relate its sensor readings to human spatial concepts. To enable human operators to interact with mobile robots in, e.g., task planning, or to allow the system to combine data from external sources, semantic information is of high value. We believe that virtual sensors can facilitate robot-human communication. We define a virtual sensor as one or several physical sensors with a dedicated signal processing unit for recognition of real world concepts. As an example, this paper describes a virtual sensor for building detection using methods for classification of views as buildings or nature based on vision. The purpose with this is to detect one type of very distinct objects that often is used in, e.g., textual description of route directions. The suggested method to obtain a virtual sensor for building detection is based on learning a mapping from a set of possibly generic features to a particular concept. It is therefore expected that the same method can be extended to virtual sensors for representation of other human concepts.

Many systems for building detection, both for aerial and ground-level images, use line and edge related features. Building detection from ground-level images often uses the fact that, in many cases, buildings show mainly horizontal and vertical edges. In nature, on the other hand, edges tend to have more randomly distributed orientations. Inspection of histograms based on edge orientation confirms this observation. Histograms of edge direction in different scales can be classified by, e.g., support vector machines [2]. Another method, developed for building detection in content-based image retrieval uses consistent line clusters with different properties [3]. This clustering is based on edge orientation, edge colors, and edge positions. For more references on ground-level building detection, see [2].

This paper presents a virtual sensor for building detection. We use AdaBoost for learning a classifier for classification of close range monocular gray scale images into ‘buildings’ and ‘nature’. AdaBoost has an ability to select the best so-called weak classifiers out of many features. The selected weak classifiers are linearly combined to produce one strong classifier. Bayes Optimal Classifier, BOC, is used as an alternative classifier for comparison. BOC uses the variance and covariance of the features in the training data to weight the importance of each feature. The proposed method combines different types of features such as edge orientation, gray level clustering, and corners into a system with high classification rate. The method is applied on a mobile robot as a virtual sensor for building detection in an outdoor environment and can be extended to other classes, such as windows and doors.

The paper is organized as follows. Section II describes the feature extraction. AdaBoost is presented in Section III and Bayes classifier is presented in Section IV. In Section V the used image sets, the description of the training phase, and some properties of the weak classifiers are presented. Section VI shows the results from the performance evaluation and Section VII describes the virtual sensor for building detection. Finally, conclusions are given in Section VIII.

II. Feature Extraction

We select a large number of image features, divided into three groups, that we expect can capture the properties of man-made structures. The obvious indication of man-made structures, especially buildings, is that they have a high content of vertical and horizontal edges. The first type of features use this property. The second type of features combines the edges into more complex structures such as corners. The third type of features is based on the assumption that buildings often contain surfaces with constant gray level. The features that we calculate for each image are numbered 1 to 24. All features except 9 and 13 are normalized in order to avoid scaling problems. Here, the features were selected with regard to a particular virtual sensor, but one could also use a generic set of features for different virtual sensors.

A. Edge Orientation

For edge detection we use Canny's edge detector [4]. It includes a Gaussian filter and is less likely than others to be fooled by noise. A drawback is that the Gaussian filter can distort straight lines. For line extraction in the edge image an algorithm implemented by Peter Kovesi [5] was used. This algorithm includes a few parameters that have been optimized empirically. The absolute values of the line's orientation are calculated and used in different histograms. The features based on edge orientation are:

1) 3-bin histogram of absolute edge orientation values.
2) 8-bin histogram of absolute edge orientation values.
3) Fraction of vertical lines out of the total number.
4) Fraction of horizontal lines.
5) Fraction of non-horizontal and non-vertical lines.
6) As 1) but based on edges longer than 20% of the longest edge.
7) As 1) but based on edges longer than 10% of the shortest image side.
8) As 1) but weighted with the lengths of the edges.

The 3-bin histogram has limits of $[0\ 0.2\ 1.37\ 1.57]$ and the 8-bin histogram $[0\ 0.2\ \ldots\ 1.4\ 1.6]$ radians. Values for the vertical (3), horizontal (4) and intermediate orientation lines (5) are taken from the 3-bin histogram and normalized with the total number of lines. Features 6, 7, and 8 try to eliminate the influence from short lines.

B. Edge Combinations

The lines defined above can be combined to form corners and rectangles. The features based on these combinations are:

9) Number of right-angled corners.
10) 9) divided by the number of edges.
11) Share of right-angled corners with direction angles close to $45^\circ + n \cdot 90^\circ, n \in 0, \ldots, 3$.
12) 11) divided by the number of edges.
13) The number of rectangles.
14) 13) divided by the number of corners.

We define a right-angled corner as two lines with close end points and $90^\circ \pm \beta_{dev}$ angle in between. During the experiments $\beta_{dev} = 20^\circ$ was used. Features 9 and 10 are the number of corners. For buildings with vertical and horizontal lines from doors and windows, the corners most often have a direction of 45°, 135°, 225° and 315°, where the direction is defined as the 'mean' value of the orientation angle for the respective lines. This is captured in features 11 and 12. From the lines and corners defined above rectangles representing, e.g., windows are detected. We allow corners to be used multiple times to form rectangles with different corners. The number of rectangles is stored in features 13 and 14.

C. Gray Levels

Buildings are often characterized by large homogeneous areas in the facades, while nature images often show larger variation. Other areas in images that can also be homogeneous are, e.g., roads, lawns, water and the sky. Features 15 to 24 are based on gray levels. We use a 25-bin gray level histogram, normalized with the image size and sum up the largest bins. This type of feature works globally in the image. To find local areas with homogeneous gray levels we search for the largest connected areas within the same gray level. Based on the gray level histogram, we calculate the largest regions of interest that are 4-connected. The features based on gray levels are:

15) Largest value in gray level histogram.
16) Sum of the 2 largest values in gray level histogram.
17) Sum of the 3 largest values in gray level histogram.
18) Sum of the 4 largest values in gray level histogram.
19) Sum of the 5 largest values in gray level histogram.
20) Largest 4-connected area.
21) Sum of the 2 largest 4-connected areas.
22) Sum of the 3 largest 4-connected areas.
23) Sum of the 4 largest 4-connected areas.
24) Sum of the 5 largest 4-connected areas.

III. AdaBoost

AdaBoost is the abbreviation for adaptive boosting. It was developed by Freund and Schapire [6] and has been used in diverse applications, e.g., as classifiers for image retrieval [7]. In mobile robotics, AdaBoost has, e.g., been used in ball tracking for soccer-robots [8] and to classify laser scans for learning of places in indoor environments [9]. This work is a nice demonstration of using machine learning and a set of generic features to transform sensor readings to human spatial concepts.

The main purpose of AdaBoost is to produce a strong classifier by a linear combination of weak classifiers, where *weak* means that the classification rate has to be only slightly better than 0.5 (better than guessing). The principle of AdaBoost is as follows (see [10] for a formal algorithm). The input to the algorithm is a number, N, of positive (buildings) and

negative (nature) examples. The training phase is a loop. For each iteration t, the best weak classifier h_t is calculated and a distribution $\boldsymbol{D}_t$ is recalculated. The boosting process uses $\boldsymbol{D}_t$ to increase the weights of the hard training examples in order to focus the weak learners on the hard examples.

The general AdaBoost algorithm does not include rules on how to choose the number of iterations T of the training loop. The training process can be aborted if the distribution $\boldsymbol{D}_t$ does not change, otherwise the loop runs through the manually determined T iterations. Boosting is known to be not particularly prone to the problem of overfitting [10]. We used $T = 30$ for training and did not see any indications of overfitting when evaluating the performance of the classifier on an independent test set.

The weak classifiers in AdaBoost use single value features. To be able also to handle feature arrays from the histogram data, we have chosen to use a minimum distance classifier, MDC, to calculate a scalar weak classifier. We use $\boldsymbol{D}_t$ to bias the hard training examples by including it in the calculation of a weighted mean value for the MDC prototype vector:

$$\boldsymbol{m}_{l,k,t} = \frac{\sum_{\{n=1\ldots N|y_n=k\}} \boldsymbol{f}(n,l)\boldsymbol{D}_t(n)}{\sum_{\{n=1\ldots N|y_n=k\}} \boldsymbol{D}_t(n)}$$

where $\boldsymbol{m}_{l,k,t}$ is the mean value array for iteration t, class k, and feature l and y_n is the class of the nth image. The features for each image are stored in $\boldsymbol{f}(n,l)$ where n is the image number. For evaluation of the MDC at iteration t, a distance value $d_{k,l}(n)$ for each class k (building and nature) is calculated as

$$d_{k,l}(n) = \|\boldsymbol{f}(n,l) - \boldsymbol{m}_{l,k,t}\|$$

and the shortest distance for each feature l indicates the winning class for that feature.

IV. Bayes Classifier

It is instructive to compare the result from AdaBoost with another classifier. For this we have used Bayes Classifier, see e.g. [11] for a derivation. Bayes Classifier, or Bayes Optimal Classifier, BOC, as it is sometimes called, classifies normally distributed data with a minimum misclassification rate. The decision function is

$$d_k(\boldsymbol{x}) = \ln P(w_k) - \frac{1}{2}\ln|\boldsymbol{C}_k| - \frac{1}{2}[(\boldsymbol{x}-\boldsymbol{m}_k)^T\boldsymbol{C}_k^{-1}(\boldsymbol{x}-\boldsymbol{m}_k)]$$

where $P(w_k)$ is the prior probability (set to 0.5), $\boldsymbol{m}_k$ is the mean vector of class k, and $\boldsymbol{C}_k$ is the covariance matrix of class k calculated on the training set, and $\boldsymbol{x}$ is the feature value to be classified.

Not all of the defined features can be used in BOC. Linear dependencies between features give numerical problems in the calculation of the decision function. Therefore normalized histograms can not be used, hence features 1, 2, 6, 7, and 8 were not considered. The set of features used in BOC was 3, 4, 9-15, 17, 20, 23. This set was constructed by starting with the best individual feature (see Figure 3, Section V-C) and adding the second best feature etc., while observing the condition value of the covariance matrices.

V. Experiments

A. Image Sets

We have used three different sources for the collection of nature and building images used in the experiments. In Set 1, we used images taken by an ordinary consumer digital camera. These were taken over a period of several months in our intended outdoor environment. Our goal with the system is to classify images taken by a mobile robot. In Set 2, we therefore stored images from manually controlled runs with a mobile robot, performed on two different occasions. Set 1 and 2 are disjunctive in the sense that they do not contain images of the same buildings or nature.

In order to verify the system performance with an independent set of images, Set 3 contains images that have been downloaded from the Internet using Google's Image Search. For buildings the search term *building* was used. The first 50 images with a minimum resolution of 240×180 pixels containing a dominant building were downloaded. For nature images, the search terms *nature* (15 images), *vegetation* (20 images), and *tree* (15 images), were used. Only images that directly applied to the search term and were photos of reality (no arts or computer graphics) were used. Borders and text around some images were removed manually. Table I presents the different sets of images and the number of images in each set.

All images have been converted to gray scale and stored in two different resolutions (maximum side length 120 pixels and 240 pixels, referred to as size 120 and 240 respectively). By this way we can compare the performance at different resolutions, for instance the classification rate and scalability robustness. A benefit of using low resolution images is faster computations. Examples of images from Set 1 and 2 are shown in Figure 1.

Set	Origin	Area	Buildings	Nature
1	Digital camera	Urban	40	40
2	Mobile camera	Campus	66	24
3	Internet search	Worldwide	50	50
	Total number		156	114

TABLE I

Number of collected images. The digital camera is a 5 Mpixel Sony (DSC-P92) and the mobile camera is an analogue camera mounted on an iRobot ATRV-Jr.

B. Test Description

Four tests have been defined for evaluation of our system. *Test 1* shows whether it is possible to collect training data

Fig. 1. Example of images used for training. The uppermost row shows buildings in Set 1. The following rows show buildings in Set 2, nature in Set 1, and nature in Set 2, respectively.

with a consumer camera and use this for training of a classifier that is evaluated on the intended platform, the mobile robot. *Test 2* shows the performance of the classifier in the environment that it is designed for. *Test 3* shows how well the learned model, trained with local images, extrapolates to images taken around the world. *Test 4* evaluates the performance on the complete collection of images. Table II summarizes the test cases. These tests have been performed with AdaBoost and BOC separately for each of the two image sizes. For Test 2 and 4, a random function is used to select the training partition and the images not selected are used for the evaluation of the classifiers. This was repeated N_{run} times.

No.	N_{run}	Train Set	Test Set
1	1	1	2
2	100	90% of {1,2}	10% of {1,2}
3	1	{1,2}	3
4	100	90% of {1,2,3}	10% of {1,2,3}

TABLE II
DESCRIPTION OF DEFINED TESTS (N_{run} IS THE NUMBER OF RUNS).

C. Analysis of the Training Results

AdaBoost can compute multiple weak classifiers from the same features by means of a different threshold, for example. Figure 2 presents statistics on the usage of different features in Test 2. The feature most often used for image size 240 is the orientation histogram (2). For image size 120, features 2, 8, 13 and 14 dominate. Figure 3 shows how each individual feature manages to classifiy images in Test 2. Several of the histograms based on edge orientation are in themselves close to the result achieved for the classifiers presented in the next section. Comparing Figure 2 and Figure 3 one can note that several features with high classification rates are not used by AdaBoost to the expected extent, e.g., features 1, 3, 4, and 5. This can be caused by the way in which the distribution

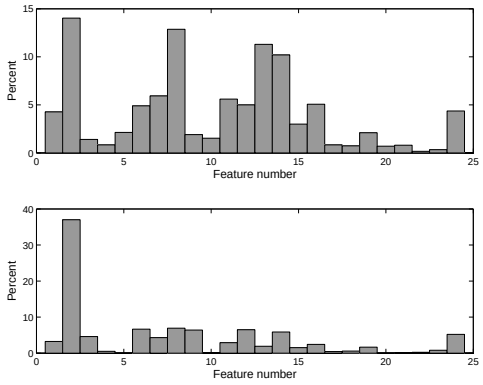

Fig. 2. Histogram describing the feature usage by AdaBoost in Test 2 as an average of 100 runs, using image size 120 (upper) and 240 (lower).

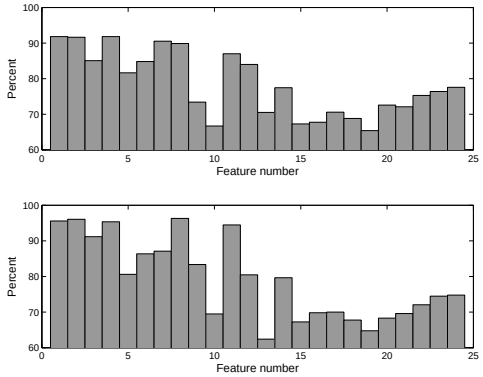

Fig. 3. Histogram of classification rate of individual features in Test 2 as an average of 100 runs with $T = 1$, image size 120 (upper) and 240 (lower).

$\boldsymbol{D}_t$ is updated. Becase the importance of correctly classified examples is decreased after a particular weak classifier is added to the strong classifier, similar weak classifiers might not be selected in subsequent iterations.

As a comparison to the test results presented in Section VI, the result obtained on the training data using combinations of image sets is also presented in Table III.

VI. RESULTS

Training and evaluation have been performed for the tests specified in Table II for features extracted both from images of size 120 and 240. The result is presented in Table IV and V respectively. The tables show the mean value of the total classification rate, its standard deviation, and the mean value of the classification rates for building images and nature images separately. Results from both AdaBoost and BOC using the same training and testing data are given.

Test 1 shows a classification rate of over 92% for image size 240. This shows that it is possible to build a classifier based on digital camera images and achieve very good results for classification of images from our mobile robot, even though Set 1 and 2 have structural differences, see Section V-A.

Sets	Size	Classifier	Build. [%]	Nat. [%]	Total [%]
1	120	AdaBoost	100.0	100.0	100.0
		BOC	100.0	100.0	100.0
1,2	120	AdaBoost	97.2	100.0	98.2
		BOC	95.3	93.8	94.7
1,2,3	120	AdaBoost	89.7	94.7	91.9
		BOC	86.5	94.7	90.0
1	240	AdaBoost	100.0	100.0	100.0
		BOC	100.0	100.0	100.0
1,2	240	AdaBoost	100.0	100.0	100.0
		BOC	98.1	100.0	98.8
1,2,3	240	AdaBoost	98.7	99.1	98.9
		BOC	95.5	98.2	96.7

TABLE III
RESULTS ON THE TRAINING IMAGE SETS IN TABLE I.

Test no.	Classifier	Build. [%]	Nat. [%]	Total [%]
1	AdaBoost	81.8	91.7	84.4
	BOC	93.9	58.3	84.4
2	AdaBoost	93.0	91.8	92.6 ± 5.8
	BOC	95.7	89.0	93.4 ± 5.5
3	AdaBoost	68.0	90.0	79.0
	BOC	72.0	74.0	73.0
4	AdaBoost	86.6	89.8	87.9 ± 6.2
	BOC	86.4	88.5	87.3 ± 6.0

TABLE IV
RESULTS FOR TEST 1-4 USING IMAGES WITH SIZE 120.

Test 2 is the most interesting test for us. This uses images that have been collected with the purpose of training and evaluating the system in the intended environment for the mobile robot. This test shows high (and highest) classification rates. For both AdaBoost and BOC they are around 97% using the image size 240.

Figure 4 shows the distribution of wrongly classified images for AdaBoost compared to BOC. It can be noted that for image size 120 several images give both classifiers problems, while for image size 240 different images cause problems.

Test 3 is the same type of test as Test 1. They both train on one set of images and then validate on a different set.

Test no.	Classifier	Build. [%]	Nat. [%]	Total [%]
1	AdaBoost	89.4	100.0	92.2
	BOC	95.5	87.5	93.3
2	AdaBoost	96.1	98.3	96.9 ± 4.3
	BOC	98.1	95.7	97.2 ± 4.0
3	AdaBoost	88.0	94.0	91.0
	BOC	90.0	82.0	86.0
4	AdaBoost	94.1	95.5	94.6 ± 3.8
	BOC	94.8	93.4	94.2 ± 4.7

TABLE V
RESULTS FOR TEST 1-4 USING IMAGES WITH SIZE 240.

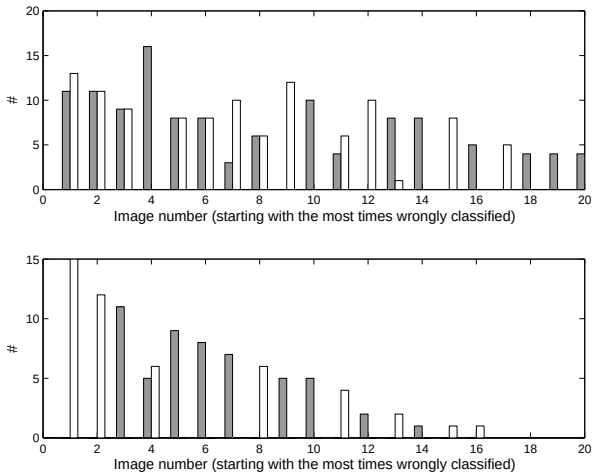

Fig. 4. Distribution of the 20 most frequently wrongly classified images from AdaBoost (gray) and BOC (white), using image size 120 (upper) and 240 (lower).

Test 3 shows lower classification rates than Test 1 with the best result for AdaBoost using image size 240. This is not surprising since the properties of the downloaded images differ from the other image sets. The main difference between the image sets is that the buildings in Set 3 often are larger and located at a greater distance from the camera. The same can be noted in the nature images, where Set 3 contains a number of landscape images that do not show close range objects. The conclusion from this test is that the classification still works very well and that AdaBoost generalizes better than BOC.

The result from Test 4 is compared with the result of Test 2. We can note that the classification rate is lower for Test 4, especially for image size 120. Investigation of the misclassified images in Test 4 shows that the share belonging to image Set 3 (Internet) is large. For both image sizes 60% of the misclassified images came from Set 3.

To show scale invariance we trained two classifiers on Test 2 with images of size 120 and evaluated them with images of size 240 and vice versa. The result is presented in Table VI and should be compared to Test 2 in Tables IV and V. The conclusion from this test is that the features we use have scale invariant properties over a certain range and that AdaBoost shows significantly better scale invariance than BOC, which again demonstrates AdaBoost's better extrapolation capability.

VII. VIRTUAL SENSOR FOR BUILDING DETECTION

We have used the learned building detection algorithm to construct a virtual sensor. This sensor indicates the presence of buildings in different directions related to a mobile robot. In our case we let the robot perform a sweep with its camera ($\pm 120^\circ$ in relation to its heading) at a number of points along its track. The images are classified into buildings and 'nature'

Train	Test	Classifier	B. [%]	N. [%]	Total [%]
120	240	AdaBoost	94.2	96.7	95.1 ± 4.2
		BOC	93.0	94.3	93.5 ± 5.3
240	120	AdaBoost	95.1	90.8	93.6 ± 6.0
		BOC	100.0	44.8	80.5 ± 6.7

TABLE VI

RESULTS FOR TEST 2 USING TRAINING WITH IMAGES SIZED 120 AND TESTING WITH IMAGES SIZED 240 AND VICE VERSA.

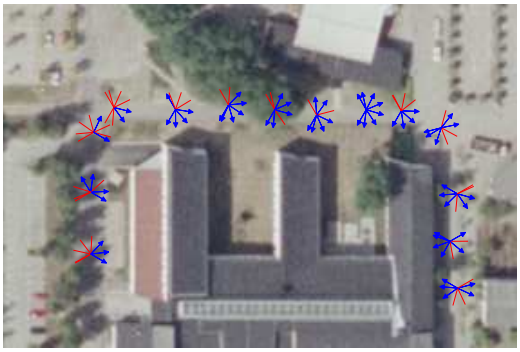

Fig. 5. Classification of images used as a virtual sensor pointing at the two classes (blue arrows indicate buildings and red lines non-buildings).

(or non-buildings) using AdaBoost trained on set 1. The experiments were performed using a Pioneer robot equipped with GPS and a camera on a PT-head. Figure 5 shows the result of a tour in the Campus area. The blue arrows show the direction towards buildings and the red lines point toward non-buildings. Figure 6 shows an example of the captured images and their classes from a sweep with the camera at the first sweep point (the lowest leftmost sweep point in Figure 5). This experiment was conducted with yet another camera and during winter, and the result was qualitatively found to be convincing. Note that the good generalization of AdaBoost is expressed by the fact that the classifier was trained on images taken in a different environment and season.

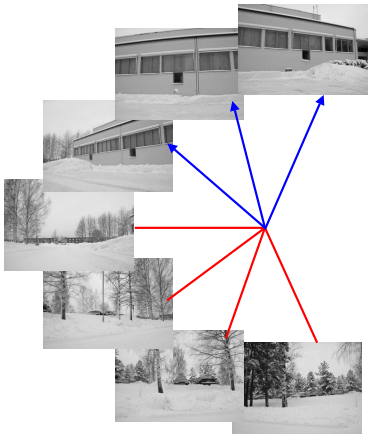

Fig. 6. Example of one sweep with the camera. The blue arrows point at images classified as buildings and the red lines point at non-buildings.

VIII. CONCLUSIONS

We have shown how a virtual sensor for pointing out buildings along a mobile robot's track can be designed using image classification. A virtual sensor relates the robot sensor readings to a human concept and is applicable, e.g., when semantic information is necessary for communication between robots and humans. The suggested method using machine learning and generic image features will make it possible to extend virtual sensors to a range of other important human concepts such as cars and doors. To handle these new concepts, features that capture their characteristic properties should be added to the present feature set, which is expected to be reused in the future work.

Two classifiers intended for use on a mobile robot to discriminate buildings from nature utilizing vision have been evaluated. The results from the evaluation show that high classification rates can be achieved, and that Bayes classifier and AdaBoost have similar classification results in the majority of the performed tests. The number of wrongly classified images is reduced by about 50% when the higher resolution images are used. The features that we use have scale invariant properties to a certain range, showed by the cross test where we trained the classifier with one image size and tested on another size. The benefits gained from Adaboost include the highlighting of strong features and its improved generalization properties over the Bayes classifier.

REFERENCES

[1] Marjorie Skubic, Pascal Matsakis, George Chronis, and James Keller. Generating multi-level linguistic spatial descriptions from range sensor readings using the histogram of forces. *Autonomous Robots*, 14(1):51–69, Jan 2003.

[2] J. Malobabic, H. Le Borgne, N. Murphy, and N. O'Connor. Detecting the presence of large buildings in natural images. In *Proc. 4th International Workshop on Content-Based Multimedia Indexing*, Riga, Latvia, June 21-23, 2005.

[3] Yi Li and Linda G. Shapiro. Consistent line clusters for building recognition in CBIR. In *Proc. Int. Conf. on Pattern Recognition*, volume 3, pages 952–957, Quebec City, Quebec, CA, Aug 2002.

[4] John Canny. A computational approach for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):279–98, Nov 1986.

[5] Peter Kovesi. http://www.csse.uwa.edu.au/∼pk/Research/MatlabFns/, University of Western Australia, Sep 2005.

[6] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[7] Kinh Tieu and Paul Viola. Boosting image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, South Carolina, June 2000.

[8] André Treptow, Andreas Masselli, and Andreas Zell. Real-time object tracking for soccer-robots without color information. In *European Conf. on Mobile Robotics (ECMR 2003)*, Radziejowice, Poland, 2003.

[9] O. Martínez Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1742–1747, Barcelona, Spain, April 2005.

[10] Robert E. Schapire. A brief introduction to boosting. In *Proc. of the Sixteenth Int. Joint Conf. on Artificial Intelligence*, 1999.

[11] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, second edition, 2001.