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

Martin Persson [a,*,1] Tom Duckett [b] Achim Lilienthal [a]

[a] *Center for Applied Autonomous Sensor Systems, Department of Technology, Örebro University, Örebro, Sweden*

[b] *Department of Computing and Informatics, University of Lincoln, Lincoln, 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 grey level images. The features are based on edge orientation, configurations of these edges, and on grey 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.

Key words: Human-robot communication, human concepts, virtual sensor, automatic building detection, AdaBoost

* Corresponding author.
Email addresses: `martin.persson@tech.oru.se` (Martin Persson), `tduckett@lincoln.ac.uk` (Tom Duckett), `achim@lilienthals.de` (Achim Lilienthal).

1 Supported by the Swedish Defence Material Administration

4 December 2006

1 Introduction

The use of human concepts is very important in robot-human communication. Skubic *et al.* [1] discuss the benefits of human spatial concepts (which they call linguistic spatial descriptions) for different types of robot control, and point out that these descriptions are especially important for novice robot users. To enable human operators to interact with mobile robots in, e.g., task planning, or to allow the system to use data from external sources, e.g. GIS, it is necessary for the robot to be able to relate its sensor readings to human spatial concepts. We believe that virtual sensors can facilitate robot-human communication. Here, a virtual sensor is understood as one or several physical sensors with a dedicated signal processing unit for recognition of real world concepts. As an example of a virtual sensor, this paper describes a virtual sensor for building detection using methods for classification of views as buildings or nature based on vision. The purpose is to detect one very distinctive type of object that is often used by humans, for example, in 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 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 based on edge orientation, edge colors, and edge positions [3]. For more references on ground-level building detection, see [2].

This paper presents a learned virtual sensor for building detection. The proposed method combines different types of features such as edge orientation, grey 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 is expected to be extendable to other classes, such as windows and doors. We use AdaBoost for learning a classifier that classifies close range monocular grey scale images into ‘buildings’ and ‘nature’. AdaBoost has the ability to select the best so-called weak classifiers and produces a strong classifier as a linear combination of the weak classifiers. 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 weigh the importance of each feature.

The paper is organized as follows. Section 2 describes the set of features from which the weak classifiers are calculated. The classifiers that are compared in this paper are described in Section 3 (AdaBoost) and Section 4 (Bayes classifier). Section 5 presents the image sets used to evaluate the classifiers and an analysis of the most important weak classifiers as found in the training phase. Section 6 shows the results from the performance evaluation and Section 7 describes an experiment using the virtual sensor for building detection on a mobile robot. Finally, conclusions are given in Section 8.

2 Feature Extraction

We select a large number of image features that are supposed to capture the properties of man-made structures. These features can be divided into three groups. The first type of features is derived from edge orientation and uses the property that man-made structures, especially buildings, often have a high content of vertical and horizontal edges. The second type of features combines the edges into more complex structures such as corners. Based on the observation that buildings often contain surfaces with constant grey level, the third type of features uses grey level clusters. The particular set of features in this work was selected with regard to a virtual sensor for building detection. In general, i.e., as a base for different virtual sensors, a more generic set of features has to be used. However, the concept of the virtual sensor presented in this paper does not need to be modified if the feature set is extended.

The features used for building detection are described in Sections 2.1-2.3. They are numbered 1 to 24 and all features except 9 and 13 are normalized in order to avoid scaling problems. The parameters that have been used in the computation of the features are found in Table 1.

2.1 Edge Orientation

For edge detection we use Canny's edge detector [4]. It includes a Gaussian filter and is less sensitive to noise than other edge detectors. For line extraction in the edge image an algorithm implemented by Peter Kovesi[2] was used. This algorithm includes a few parameters that have been optimized empirically ($\mu_1 - \mu_3$, see Table 1). The absolute values of the lines' orientation are used to calculate the following features:

(1) 3-bin histogram of absolute edge orientation values.

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

(2) 8-bin histogram of absolute edge orientation values.
(3) Fraction of vertical lines (bin no. 1 of feature 1) out of the total number.
(4) Fraction of horizontal lines (bin no. 3 of feature 1).
(5) Fraction of non-horizontal and non-vertical lines (bin no. 2 of feature 1).
(6) As 1) but based on edges longer than $\Theta_1\%$ of the longest edge.
(7) As 1) but based on edges longer than $\Theta_2\%$ of the shortest image side.
(8) As 1) but weighted with the lengths of the edges.

The 3-bin histogram has limits of $[0\ ,\ 0.2\ ,\ \pi/2 - 0.2\ ,\ \pi/2]$ and the 8-bin histogram $[0\ ,\ \pi/16\ ,\ \dots\ ,\ 7\pi/16\ ,\ \pi/2]$ 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 of short lines.

2.2 Edge Combinations

The lines extracted from the image (as described above) are then combined to form corners and rectangles. The features based on these combinations are:

(9) Number of right-angled corners as defined below.
(10) 9) divided by the number of edges.
(11) Fraction of right-angled corners with direction angles equal to $45^\circ + n \cdot 90^\circ \pm \beta_2, n \in 0, \dots, 3$.
(12) 11) divided by the number of edges.
(13) The number of rectangles, see text below.
(14) 13) divided by the number of corners.

We define a right-angled corner as two lines with close end points and $90^\circ \pm \beta_1$ angle in between. During the experiments $\beta_1 = 20^\circ$ was used. Features 9 and 10 are based on 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 structures such as windows are calculated. Corners can be used multiple times to form rectangles. The number of rectangles is used in features 13 and 14.

2.3 Grey Levels

Buildings are often characterized by large homogeneous areas in their facade, while nature images typically show larger variation. Other areas in images, however, can also be homogeneous as for example roads, lawns, water and the sky. Features 15 to 24 are based on grey levels. We use an equally spaced 25-bin grey level histogram, normalized by the image size and sum up the largest bins. This type of feature works globally in the image. To find local areas with homogeneous grey levels we search for the largest 4-connected areas with the same grey levels as used for the 25-bin grey level histogram. The features based on grey levels are:

(15) Largest value in grey level histogram.
(16) Sum of the 2 largest values in grey level histogram.
(17) Sum of the 3 largest values in grey level histogram.
(18) Sum of the 4 largest values in grey level histogram.
(19) Sum of the 5 largest values in grey 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.

Parameter	Value	Description
Θ_1	20%	Threshold for long lines in feature 6
Θ_2	10%	Threshold for long lines in feature 7
β_1	20°	Max. angle deviation for forming right-angled corners
β_2	15°	Angle tolerance for corner direction
μ_1	2 pixels	Maximum deviation in pixels when calculating straight lines
μ_2	0.05 rad	Maximum angle difference between lines that should be merged
μ_3	2 pixels	Maximum gaps between lines that should be merged

Table 1
Description of used parameters.

3 AdaBoost

AdaBoost is the abbreviation for adaptive boosting. It was developed by Freund and Schapire [5] and has been used in diverse applications, e.g., as classifiers for image retrieval [6], for ball tracking with soccer-robots [7], and to classify laser scans for learning of places in indoor environments [8,9]. The latter work provides a nice demonstration of the use of machine learning and a set of generic features to transform sensor readings into human 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 D_t is recalculated. The boosting process uses D_t to increase the weights of 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 D_t does not change, otherwise the loop runs through a manually determined number of iterations T. 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.

To be able to handle feature arrays from the histogram data, we use a minimum distance classifier, MDC. We use D_t to bias the hard training examples by including it in the calculation of a weighted mean value for the MDC prototype vector:

$$\vec{m}_{l,k,t} = \frac{\sum_{\{n=1...N|y_n=k\}} \vec{f}(n,l) D_t(n)}{\sum_{\{n=1...N|y_n=k\}} D_t(n)}$$

where $\vec{m}_{l,k,t}$ 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 $\vec{f}(n,l)$ 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

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

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

4 Bayes Classifier

It is instructive to compare the result from AdaBoost with another classifier. For this purpose 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

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

where $P(w_k)$ is the prior probability (here set to 0.5), $\vec{m}_k$ is the mean vector of class k, and $\vec{C}_k$ is the covariance matrix of class k calculated on the training set, and $\vec{x}$ 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 5.3) and adding the second best feature etc., while observing the condition value of the covariance matrices.

5 Experiment Set-Up

5.1 Image Sets

We have used three different sources for the collection of nature and building images used in the experiments:

- Set 1 is taken by an ordinary consumer digital camera. The images were taken over a period of several months in our intended outdoor environment.
- Set 2 consists of stored images from manually controlled runs with a mobile robot, performed on two different occasions.
- Set 3 contains images downloaded from the Internet using Google's Image Search. For buildings the search term *building* was used. For nature images, the search terms *nature* (15 images), *vegetation* (20 images), and *tree* (15 images), were used.

Table 2 presents the different sets of images and the number of images in each set. Set 1, 2 and 3 are disjunctive in the sense that they do not contain images of the same buildings or nature. The digital camera used in Set 1 was used in order to collect images from a larger area than was practical when using a mobile robot as in Set 2. Set 3 was collected in order to verify the system performance with an independent set of images. The first images found by Google with a minimum resolution of 240×180 pixels containing a dominant building or nature scenes were downloaded. 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.

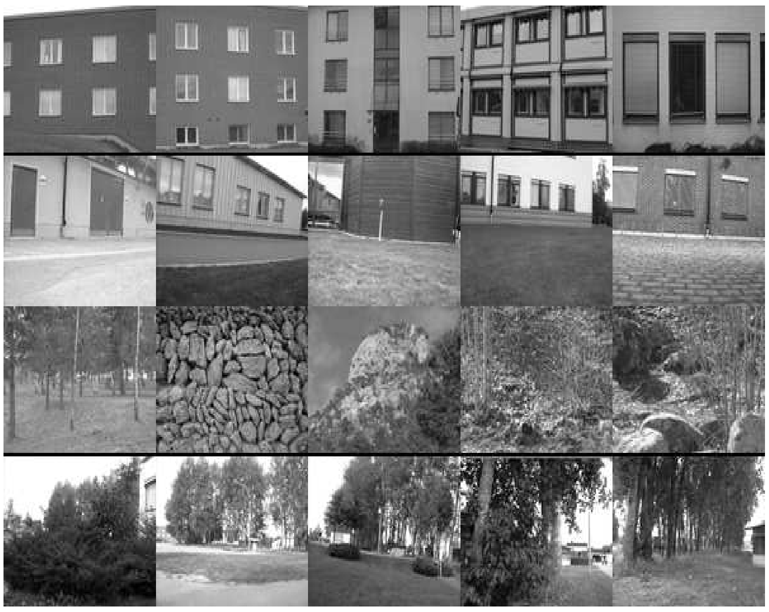

Fig. 1. Example of images used for training. The uppermost row shows buildings in Set 1. The following rows show buildings in Set 2, nature in Set 1, and nature in Set 2, respectively.

All images were converted to grey scale and stored in two different resolutions (maximum side length 120 pixels and 240 pixels, referred to as size 120 and 240 respectively), enabling a comparison of the classification rate and robustness at different resolutions. The comparatively low resolution challenges the system with a rather difficult classification task and allows for fast computation of the classifier. We further decided to investigate the classifier performance with low resolution images so that the algorithm could later be applied to sub-windows in a full image, e.g., for application on a mobile robot in order to communicate the likely locations of buildings to a remote human operator. 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 2
Number of collected images. The digital camera is a 5 Mpixel Sony (DSC-P92) and the mobile camera is a Sony XC-999 camera module mounted on an iRobot ATRV-Jr.

5.2 Test Description

Four tests were defined for evaluation of our system. *Test 1* shows whether it is possible to manually collect training data with a consumer camera and use this to train a classifier for the intended platform, the mobile robot. In *Test 2*, the classifier is learned and tested with disjunctive subsets of images from the same environment using both local image sets 1 and 2 together. *Test 3* tests 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. It is the same as Test 2 but all three data sets are used instead of only set 1 and set 2. Table 3 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 remaining images were used for the evaluation of the classifiers. This procedure 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 3
Summary of the tests described in the text (N_{run} is the number of runs).

5.3 Analysis of the Training Results

The distribution D_t that is updated by AdaBoost controls the influence of the training examples in the calculation of the weak classifiers. Therefore, a multitude of different weak classifiers can be computed from the same features. 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 classify 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 $\vec{D}_t$ is updated. Because 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 6, the result obtained

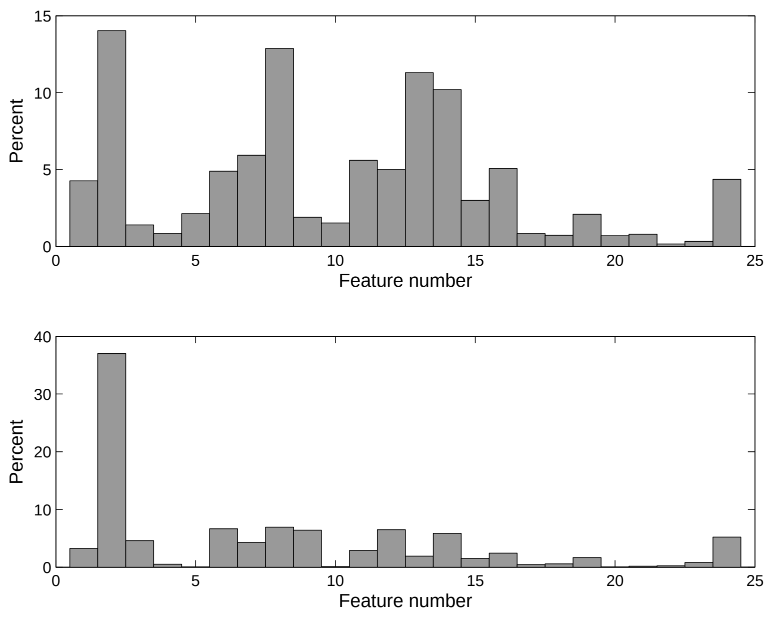

Fig. 2. Histogram describing the feature usage by AdaBoost in Test 2 as an average of 100 runs, using image size 120 (upper) and 240 (lower).

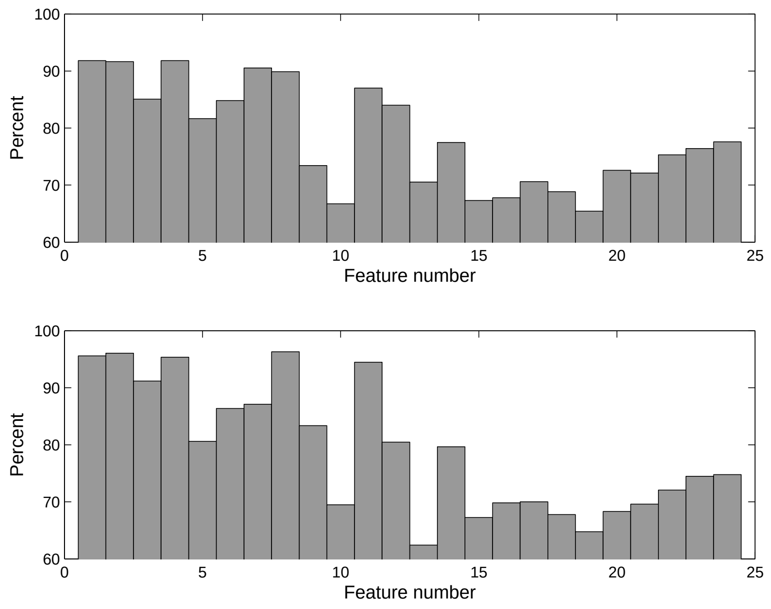

Fig. 3. Histogram of classification rate of individual features in Test 2 as an average of 100 runs, image size 120 (upper) and 240 (lower).

on the training data using combinations of image sets is also presented in Table 4.

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 4
Results on the training image sets in Table 2.

6 Results

Training and evaluation were performed for the tests specified in Table 3 with features extracted from images of both size 120 and 240. The results are presented in Tables 5 and 6 respectively. The tables show the mean value of the total classification rate, the standard deviation for tests 2 and 4 where the training and test sets were selected randomly, and the mean value of the classification rates for building images and nature images separately. The results from AdaBoost and BOC were obtained with the same training and testing data.

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 5
Results for Test 1-4 using images with size 120.

A classification rate of over 92% was found in Test 1 for image size 240. This shows that it is possible to build a classifier based on digital camera images that achieves very good results with images from a different camera on our mobile robot, even though the corresponding image sets (Set 1 and 2) have

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 6
Results for Test 1-4 using images with size 240.

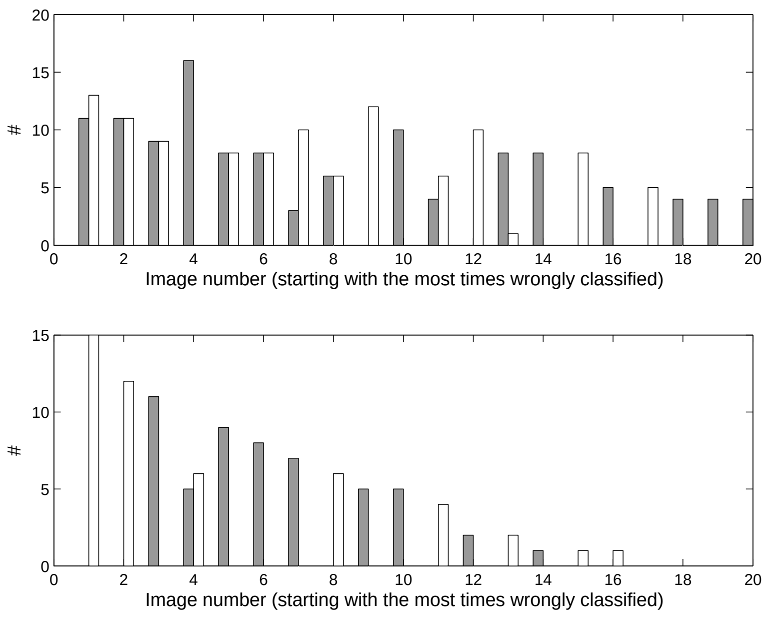

Fig. 4. Distribution of the 20 most frequently misclassified images from AdaBoost (grey) and BOC (white), using image size 120 (upper) and 240 (lower).

structural differences, see Section 5.1.

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. Both train on one set of images and then validate on a different set. 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 substantially from the other image sets. The buildings in Set 3 are often 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.

Test 4 is the same type of test as Test 2 but with a larger and more diverse set of images. Accordingly the classification rate is lower for Test 4, especially for image size 120. On closer inspection, it was found that disproportionately many of the misclassified images were from Set 3 (Internet). For both image sizes 60% of the misclassified images came from Set 3.

To investigate the scale invariance we trained two classifiers as in Test 2 with images of size 120 and evaluated them with images of size 240 and vice versa. The result is presented in Table 7 and should be compared to Test 2 in Tables 5 and 6. The conclusion from this test is that the learned classifiers have scale invariant properties over a certain range and that AdaBoost performs substantially better than BOC in this respect, which again demonstrates AdaBoost's better extrapolation capability.

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 7
Results for Test 2 using training with images sized 120 and testing with images sized 240 and vice versa.

7 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 ($\pm 120°$ in relation to its heading) at a number of points along its track. The images are classified into buildings and 'nature' (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

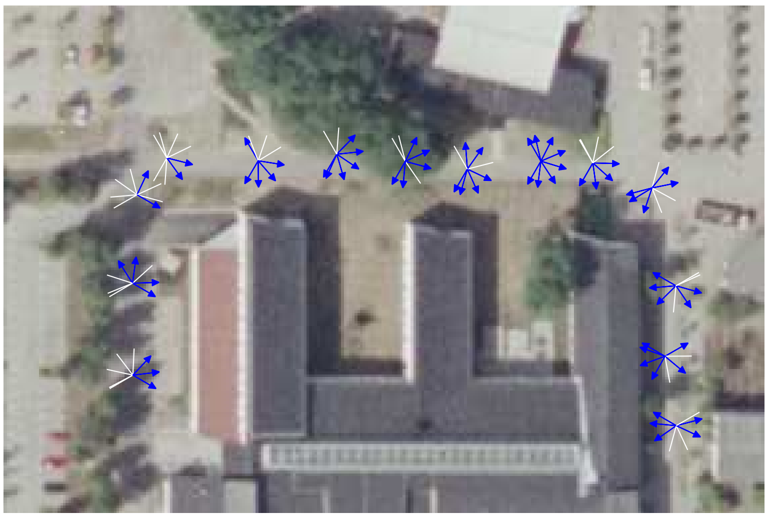

Fig. 5. Classification of images used as a virtual sensor pointing at the two classes (blue arrows indicate buildings and white lines non-buildings).

towards buildings and the white 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 lower left sweep point in Figure 5). This experiment was conducted with yet another camera[3] and during winter, and the result was qualitatively found to be convincing. Note that the good generalization performance of AdaBoost is further demonstrated by the fact that the classifier was trained on images taken in a different environment and season.

8 Conclusions

We have shown how a virtual sensor for pointing out buildings along a mobile robot's track can be designed using image classification. Virtual sensors relate robot sensor readings to a human concept. They are applicable, for example, when semantic information is necessary for communication between robots and humans or for building a semantic map. For these cases a limited number of virtual sensors are needed. 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 doors, cars and trees. To handle these new concepts, features that capture their characteristic properties should be added to the present feature set. Apart from that, the method suggested in this paper does not have to be modified to add new human concepts.

Two classifiers intended for use on a mobile robot using vision to discriminate buildings from nature were evaluated. The results from the evaluation show

[3] ImagingSource DFK 41F02 digital camera

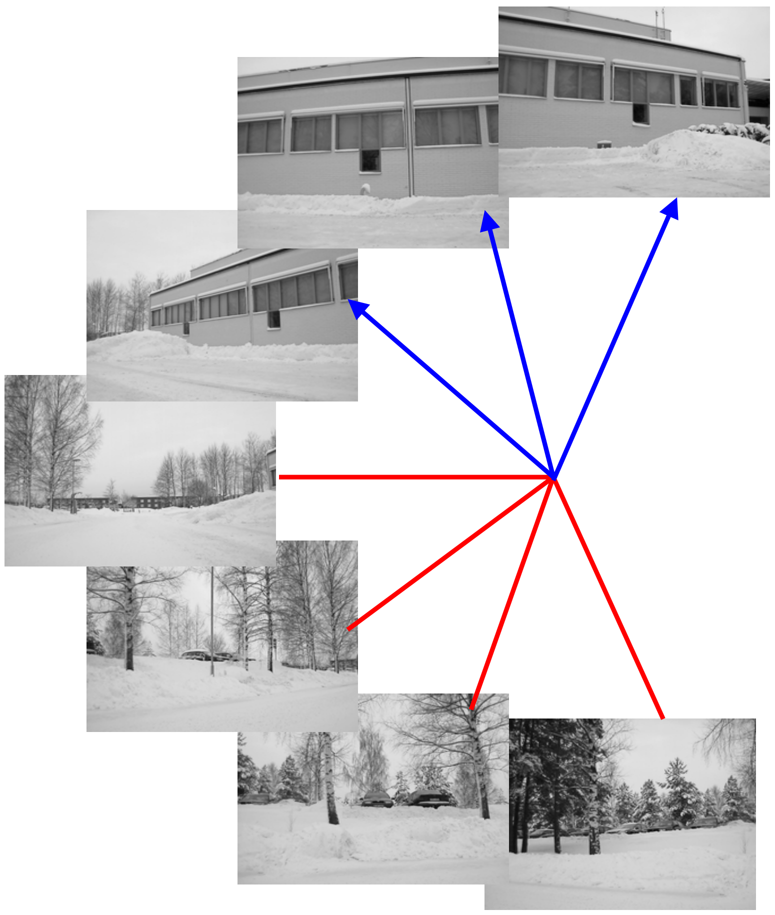

Fig. 6. Example of one sweep with the camera. The blue arrows point at images classified as buildings and the red lines point at non-buildings.

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 images with side length 240 pixels are used instead of those with 120 pixels. The learned classifiers showed scale invariant properties over a certain range, demonstrated by cross tests where we trained the classifier with one image resolution and tested on the other resolution. The benefits gained from AdaBoost include the highlighting of strong features and its improved generalization properties over the Bayes classifier.

References

[1] M. Skubic, P. Matsakis, G. Chronis, J. Keller, Generating multi-level linguistic spatial descriptions from range sensor readings using the histogram of forces, Autonomous Robots 14 (1) (2003) 51–69.

[2] J. Malobabic, H. L. Borgne, N. Murphy, N. O'Connor, Detecting the presence

of large buildings in natural images, in: Proc. 4th International Workshop on Content-Based Multimedia Indexing, June 21-23, 2005.

[3] Y. Li, L. G. Shapiro, Consistent line clusters for building recognition in CBIR, in: Proc. Int. Conf. on Pattern Recognition, Vol. 3, 2002, pp. 952–957.

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

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

[6] K. Tieu, P. Viola, Boosting image retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2000.

[7] A. Treptow, A. Masselli, A. Zell, Real-time object tracking for soccer-robots without color information, in: European Conf. on Mobile Robotics (ECMR 2003), 2003.

[8] O. Martínez Mozos, C. Stachniss, W. Burgard, Supervised learning of places from range data using AdaBoost, in: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2005, pp. 1742–1747.

[9] O. Martínez Mozos, Supervised learning of places from range data using AdaBoost, Master's thesis.

[10] R. 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, D. G. Stork, Pattern Classification, 2nd Edition, Wiley, New York, 2001.