Fast Color-Independent Ball Detection for Mobile Robots

Sara Mitri, Kai Pervölz, Hartmut Surmann, Andreas Nüchter
Fraunhofer Institute for Autonomous Intelligent Systems (AIS)
Schloss Birlinghoven
D-53754 Sankt Augustin, Germany
{firstname.surname}@ais.fraunhofer.de

***Abstract*— This paper presents a novel scheme for fast color invariant ball detection in the RoboCup context. Edge filtered camera images serve as an input for an Ada Boost learning procedure that constructs a cascade of classification and regression trees (CARTs). Our system is capable to detect different soccer balls in the RoboCup and other environments. The resulting approach for object classification is real-time capable and reliable.**

I. Introduction

A fundamental problem in the design of autonomous mobile cognitive systems is to perceive the environment. A basic part of perception is to learn, detect and recognize objects, which must be done with respect to the limited resources of a mobile robot and the limited choice of available kinds of sensors. The performance of a mobile robot crucially depends on the accuracy, duration and reliability of its perceptions and the involved interpretation process. This paper describes a real time capable, color and scale invariant object learning and detection scheme. The arrangement of Haar-like features of objects is learned. To calculate the features, the computationally efficient representation of integral images is applied. The Gentle Ada Boost learning technique is used to learn a selection of Classification and Regression Trees (CARTs) with four splits. Several selections are combined to a cascade of classifiers. To ensure the color-invariance of the input images, they are first preprocessed by applying an edge detection Sobel filter. By passing the filtered images through a threshold, all color information is omitted.

The motivation of our research was triggered by our interest in the Robot World Cup Soccer Games and Conferences (RoboCup) [5], which was created as a standard scenario where technologies could be integrated and developed, thereby encouraging innovative work in the fields of robotics and computer vision and promoting the public understanding of science.

Our experiments were carried out by the autonomous mobile robot Kurt3D, originally constructed to digitalize environments in 3D [13], [15], [14]. Kurt3D also has other applications, such as educational robotics, or in our case, soccer robotics [5].

The most common techniques for object detection, i.e., ball detection, in the RoboCup context rely on color information. In the last few years, fast color segmentation algorithms have been developed to detect and track objects in this scenario [1], [7]. The community agreed that in the near future, visual cues like color will be removed to come to a more realistic setup with robots playing with a "normal" soccer ball [16].

Some research groups have already started to develop algorithms for color invariant ball detection. One is described by Coath and Musumeci, who presented an edge-based ball detection system [2]. They developed an adaptive arc identification and location system that processes image data containing edge information.

General Object detection and classification, i.e., not within the RoboCup context, has intensely been researched in computer vision [10], [11], [17]. Common approaches use neural networks or support vector machines (SVM), for example, to detect and classify objects. Rowley et al. detect faces using a small set of simple features and neural networks [11] and Papageorgiou et al. recognize pedestrians with simple vertical, horizontal and diagonal features and SVMs [10]. Recently, Viola and Jones have proposed a boosted cascade of simple classifiers for fast face detection [17].

Independent from our work Treptow et al. used Viola and Jones' algorithm to track objects without color information. In contrast to their object detection system we preprocess the images to enhance the simple vertical and horizontal features. In addition to this, diagonal features and rotated integral images are used. To recognize different balls we learned Classification and Regression Trees.

The rest of the paper is structured as follows: First we describe the robot platform Kurt3D. Section III presents the learning algorithm. Results are given in section IV and section V concludes the paper.

II. The Autonomous Mobile Robot Kurt3D

A. The Kurt Robot Platform

Kurt3D (Fig. 1) is a mobile robot platform with a size of 45 cm (length) × 33 cm (width) × 26 cm (height) and a weight of 15.6 kg, for which both indoor as well as outdoor models exist. Equipped with the 3D laser range finder, the height increases to 47 cm and the weight increases to 22.6 kg.[1] Kurt3D's maximum velocity is 5.2 m/s (autonomously controlled: 4.0 m/s). Two 90W motors are used to power the 4 wheels. Kurt3D operates for about 4 hours with one battery (28 NiMH cells, capacity: 4500 mAh) charge. The core of the robot is an Inter-Centrino-1400 MHz with 768 MB RAM and a Linux operating system. An embedded 16-Bit CMOS microcontroller is used to control the motor.

[1] Videos of the exploration with the autonomous mobile robot can be found at: http://www.ais.fraunhofer.de/ARC/kurt3D/index.html

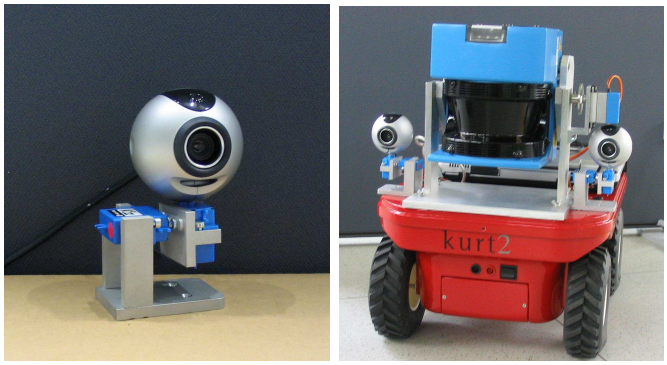

Fig. 1. Left: The pan-tilt camera system. Right: The autonomous mobile robot Kurt3D (outdoor version) equipped with the AIS 3D laser range finder and two pan-tilt cameras.

B. The Camera System

The camera system (Fig. 1, left) consists of two Logitech Quickcam 4000 Pro USB webcams. They are equipped with a manual focus lens and the resolution is limited to 640 × 480 pixels with 7 fps as the maximum frame rate. To cover the whole area in front of the robot the webcam is mounted on a pan-tilt unit which is based on 2 servo drives (Volz Micro-Maxx), one for the horizontal axis and the other for the vertical axis. Each axis can be rotated by $\pm 45°$. Due to the high-grade servo drives, an excellent repeat accuracy in positioning is guaranteed.

The webcams are powered over the USB interface and the servo drives are fed by the same batteries as the 3D laser range finder servo.

III. Learning a Sphere Classifier

Recently, Viola and Jones have proposed a boosted cascade of simple classifiers for fast face detection [17]. Inspired by these ideas, we detect objects, e.g., balls, in camera images.

A. Color Invariance using Linear Image Filters

The problem with recognizing general shapes, such as balls, as in our particular case, is the number of possibilities in the visual appearance of a ball. A ball can take on any color, size and may have any pattern on its surface. In order to generalize the concept of a ball, the initial goal was the elimination of any color information in the data images representing the balls.

In this paper, we have achieved this using linear image filters to detect the edges in the image, followed by a threshold to eliminate noise data, which would then be given as input to the classifier, which in turn handles differences in size, pattern, lighting, etc.

The most common edge detection techniques used are gradient and Laplacian operators. For this paper, we have experimented with multiple gradient filters, as well as a Laplacian filter, which we implemented ourselves, according to the algorithm described in [3]. The technique of the gradient operator is defined as follows:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots \\ a_{21} & a_{22} & a_{23} & \dots \\ a_{31} & a_{32} & a_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots \\ b_{21} & b_{22} & b_{23} & \dots \\ b_{31} & b_{32} & b_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$$\mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{pmatrix} \quad \mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}$$

$$\begin{aligned} v_{i,j} &= \sum_{x=i-1}^{i+1} \sum_{y=j-1}^{j+1} v\_factor * a_{x,y} * v_{x,y} \\ h_{i,j} &= \sum_{x=i-1}^{i+1} \sum_{y=j-1}^{j+1} h\_factor * a_{x,y} * h_{x,y} \\ b_{i,j} &= \sqrt{(v_{i,j}^2 + h_{i,j}^2) * edge\_intensity} \end{aligned} \tag{1}$$

where A represents the matrix of pixels in the input image, B the output image and V and H the 3×3 vertical and horizontal masks that are moved over the image pixels starting at the top left corner, through to the bottom right. A number of parameters need to be adjusted for each filter instance, such as the $edge\_intensity$, which defines how "thick" the output edges should be, and h_factor and v_factor to emphasize the differentiation in each direction (horizontal or vertical, respectively).

In our approach, we used a Sobel filter, with V and H defined below, $edge\_intensity = 4.0$ and both v_factor and h_factor equal to 1.

$$\mathbf{V} = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad \mathbf{H} = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Two techniques were attempted in terms of filtering and thresholding:

- Converting the image into gray scale mode and then applying the filtering algorithm, followed by the threshold.
- Applying the filter to the colored image and then using a threshold to include any pixel in any of the 3 color channels that crossed the threshold value in the output image.

The difference in results is shown in Fig. 2. A typical output of the Sobel filter is shown in Fig. 3.

This edge detection and thresholding technique is applied to all images used as input to the training of the Haar classifier. The training process itself is illustrated in the following subsections.

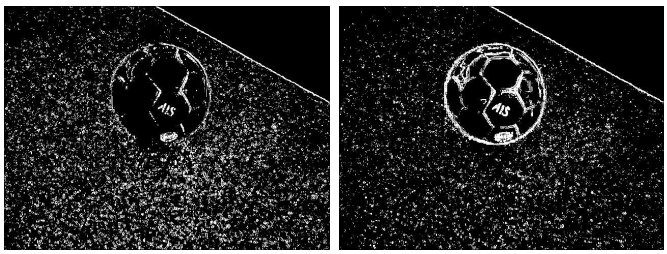

Fig. 2. Left: A Sobel filter applied to a gray scale image. Right: Sobel filter applied to a colored image and then thresholded.

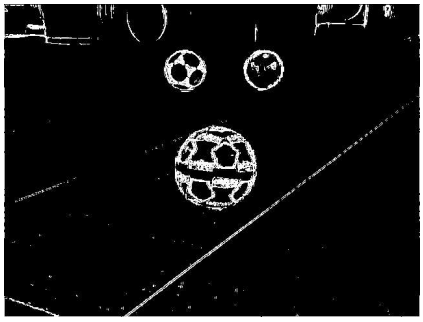

Fig. 3. Typical output of the Sobel filter

B. Feature Detection using Integral Images

There are many motivations for using features rather than pixels directly. For mobile robots, a critical motivation is that feature based systems operate much faster than pixel based systems [17]. The features used here have the same structure as the Haar basis functions, i.e. step functions introduced by Alfred Haar to define wavelets [6]. They are also used in [8], [9], [10], [17]. Fig. 4 (left) shows the eleven basis features, i.e. edge, line, diagonal and center surround features. The base resolution of the object detector is 30×30 pixels, thus, the set of possible features in this area is very large (642592 features, see [9] for calculation details). A single feature is effectively computed on input images using integral images [17], also known as summed area tables [8], [9]. An integral image I is an intermediate representation for the image and contains the sum of gray scale pixel values of image N with height y and width x, i.e.,

$$I(x,y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} N(x',y').$$

The integral image is computed recursively, by the formulas: $I(x,y) = I(x,y-1) + I(x-1,y) + N(x,y) - I(x-1,y-1)$ with $I(-1,y) = I(x,-1) = I(-1,-1) = 0$, therefore requiring only one scan over the input data. This intermediate representation $I(x,y)$ allows the computation of a rectangle feature value at (x,y) with height and width (h,w) using four references (see Figure 4 (top right)):

$$\begin{aligned} F(x,y,h,w) &= I(x,y) + I(x+w,y+h) - \\ &\quad I(x,y+h) - I(x+w,y). \end{aligned}$$

For the computation of the rotated features, Lienhart et. al. introduced rotated summed area tables in 2002 that contain

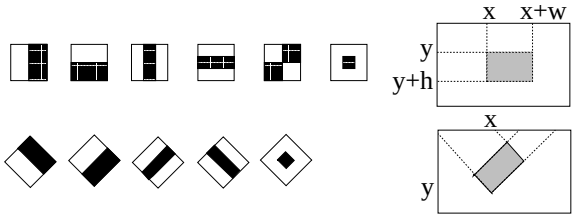

Fig. 4. Left: Edge, line, diagonal and center surround features are used for classification. Right: Computation of feature values in the shaded region is based on the four upper rectangles.

the sum of the pixels of the rectangle rotated by 45° with the bottom-most corner at (x,y) and extending till the boundaries of the image (see Figure 4 (bottom right)) [9]:

$$I_r(x,y) = \sum_{x'=0}^{x} \sum_{y'=0}^{x-|x'-y|} N(x',y').$$

The rotated integral image I_r is computed recursively, i.e., $I_r(x,y) = I_r(x-1,y-1) + I_r(x+1,y-1) + -I_r(x,y-1) + N(x,y) + N(x,y-1)$ using the start values $I_r(-1,y) = I_r(x,-1) = I_r(x,-2) = I_r(-1,-1) = I_r(-1,-2) = 0$. Four table lookups are required to compute the pixel sum of any rotated rectangle with the formula:

$$\begin{aligned} F_r(x,y,h,w) &= I_r(x+w-h,y+w+h-1) + \\ &\quad I_r(x,y-1) - I_r(x-h,y+h-1) - \\ &\quad I_r(x+w,y+w-1). \end{aligned}$$

Since the features are compositions of rectangles, they are computed with several lookups and subtractions weighted with the area of the black and white rectangles.

To detect a feature, a threshold is required. This threshold is automatically determined during a fitting process, such that a minimum number of examples are misclassified. Furthermore, the return values (α, β) of the feature are determined, such that the error on the examples is minimized. The examples are given in a set of images that are classified as positive or negative samples. The set is also used in the learning phase that is briefly described next.

C. Learning Classification Functions

1) Classification and Regression Trees: For all 642592 possible features a Classification and Regression Tree (CART) is created. CART analysis is a form of binary recursive partitioning. Each node is split into two child nodes, in which case the original node is called a parent node. The term recursive refers to the fact that the binary partitioning process is applied over and over to reach a given number of splits (4 in this case). In order to find the best possible split features, we compute all possible splits, as well as all possible return values to be used in a split node. The program seeks to maximize the average "purity" of the two child nodes using the misclassification error measure [12]. Figure 5 (left) shows a simple feature classifier and a simple CART (right).

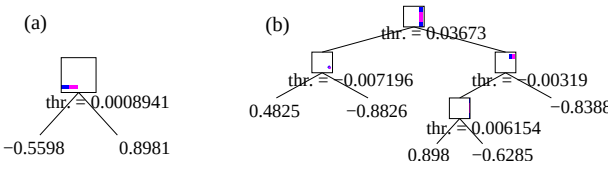

Fig. 5. Left: A simple feature classifier. Right: A Classification and Regression Tree with 4 splits. According to the specific filter applied to the image input section x, the output of the tree, $h(x)$ is calculated, depending on the threshold values.

2) Gentle Ada Boost for CARTs: The Gentle Ada Boost Algorithm is a variant of the powerful boosting learning technique [4]. It is used to select a set of simple CARTs to achieve a given detection and error rate. In the following, a detection is referred to as a hit and an error as a false alarm. The various Ada Boost algorithms differ in the update scheme of the weights. According to Lienhart et al. the Gentle Ada Boost Algorithm is the most successful learning procedure tested for face detection applications [9], [8].

The learning is based on N weighted training examples $(x_1, y_1), \ldots, (x_N, y_N)$, where x_i are the images and $y_i \in \{-1, 1\}, i \in \{1, \ldots, N\}$ the classified output. At the beginning of the learning phase the weights w_i are initialized with $w_i = 1/N$. The following three steps are repeated to select simple CARTs until a given detection rate d is reached:

1) Every simple classifier, i.e., a CART, is fit to the data. Hereby the error e is calculated with respect to the weights w_i.
2) The best CART h_t is chosen for the classification function. The counter t is incremented.
3) The weights are updated with $w_i := w_i \cdot e^{-y_i h_t(x_i)}$ and renormalized.

The final output of the classifier is $\text{sign}(\sum_{t=1}^{T} h_t(x)) > 0$, with $h(x)$ the weighted return value of the CART. Next, a cascade based on these classifiers is built.

D. The Cascade of Classifiers

The performance of a single classifier is not suitable for object classification, since it produces a high hit rate, e.g., 0.999, but also a high error rate, e.g., 0.5. Nevertheless, the hit rate is much higher than the error rate. To construct an overall good classifier, several classifiers are arranged in a cascade, i.e., a degenerated decision tree. In every stage of the cascade, a decision is made whether the image contains the object or not. This computation reduces both rates. Since the hit rate is close to one, their multiplication results also in a value close to one, while the multiplication of the smaller error rates approaches zero. Furthermore, this speeds up the whole classification process. Figure 6 shows an example cascade of classifiers for detecting balls in 2D images, whose results are given in Table I.

An overall effective cascade is learned by a simple iterative method. For every stage the classification function $h(x)$ is learned, until the required hit rate is reached. The process continues with the next stage using the correct classified positive and the currently misclassified negative examples. The number of CARTs used in each classifier may increase with additional stages.

Fig. 7. Top left: RoboCup football arena. Top right: Uniform background. Bottom left: RoboCup Rescue arena. Bottom right: Complex images.

IV. Ball Detection and Results

Although the process of generating the cascade of classifiers is relatively time-consuming, it produces quite promising results. The first three stages of a learned cascade are shown in Fig. 6. The cascade was tested systematically using four categories of input data (see Fig. 7):

1) RoboCup football arena
2) Uniform background
3) Rescue robotics arena
4) Complex scenes

The idea behind this categorization was to be able to determine exactly where the strengths and weaknesses of the technique lie. The algorithm is to be applied to football-playing robots participating in RoboCup to detect footballs, the innovation being the independence of color and surface pattern of the ball. The first test to be passed was therefore detecting the three different balls in the RoboCup arena. The algorithm not only works well in the arena, but with any images with a uniform background behind the footballs. This is shown by the next test set. To test our method to its limits we tested it both with images taken in the RoboCup Rescue arena, as well as other more complex images, with all sorts of noise and distracting objects. For each image category we ran the test with 25 pictures, including 15 of each ball, making a total of 100 images and a total of 180 balls (60 each). The detection speed averaged at 300msec per image. A sample of the test images in each category is shown in Fig. 7; the results of the experiments run on two different cascades are given in Tables I and II. The tables reveal how many red, white or yellow balls were correctly classified or not detected, as well as the number of false positives in each image category. The first cascade was learned with a total of 1000 images, where the second used a mere 570 input images.

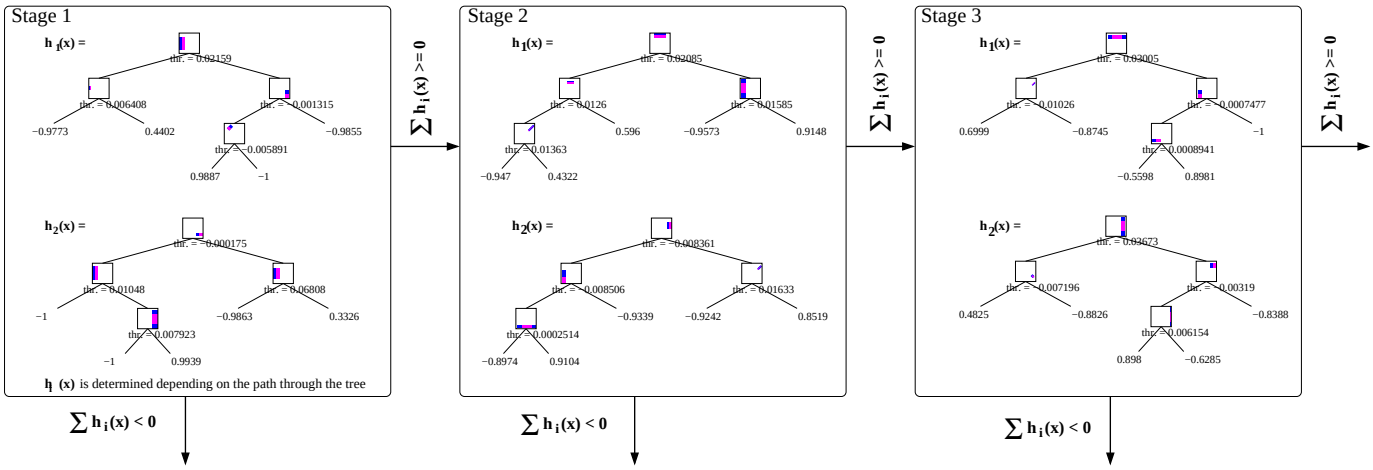

Fig. 6. The first three stages of a cascade of classifiers to detect a ball. Every stage contains several simple classifier trees that use Haar-like features with a threshold and return values of $\sum h(x)$.

		Correct		Not Detected		False Pos.
RoboCup	red	13		2		
	wht	5	26/45	10	19/45	4
	yel	8		7		
Uniform	red	13		2		
	wht	13	39/45	2	6/45	0
	yel	13		2		
Rescue	red	2		13		
	wht	2	5/45	13	40/45	10
	yel	1		14		
Complex	red	0		15		
	wht	0	0/45	15	45/45	9
	yel	0		15		
Total	red		28/60		32/60	
	wht		20/60		40/60	23
	yel		22/60		38/60	

TABLE I
SIMPLE TRAINING DATA

		Correct		Not Detected		False Pos.
RoboCup	red	14		1		
	wht	11	37/45	4	8/45	5
	yel	12		3		
Uniform	red	15		0		
	wht	14	42/45	1	3/45	4
	yel	13		2		
Rescue	red	5		10		
	wht	5	20/45	10	25/45	21
	yel	10		5		
Complex	red	5		10		
	wht	2	9/45	13	36/45	24
	yel	2		13		
Total	red		39/60		21/60	
	wht		32/60		28/60	54
	yel		37/60		23/60	

TABLE II
COMPLEX TRAINING DATA

Actual detected images are shown in Fig. 8 for all four categories, for both cascades, where the balls detected by the simple cascade only are marked by blue boxes, those detected by the complex cascade by green boxes and those detected by both by red boxes.

The two classifiers used in the experiments differ only in the data used for training. For the first classifier (Table I)

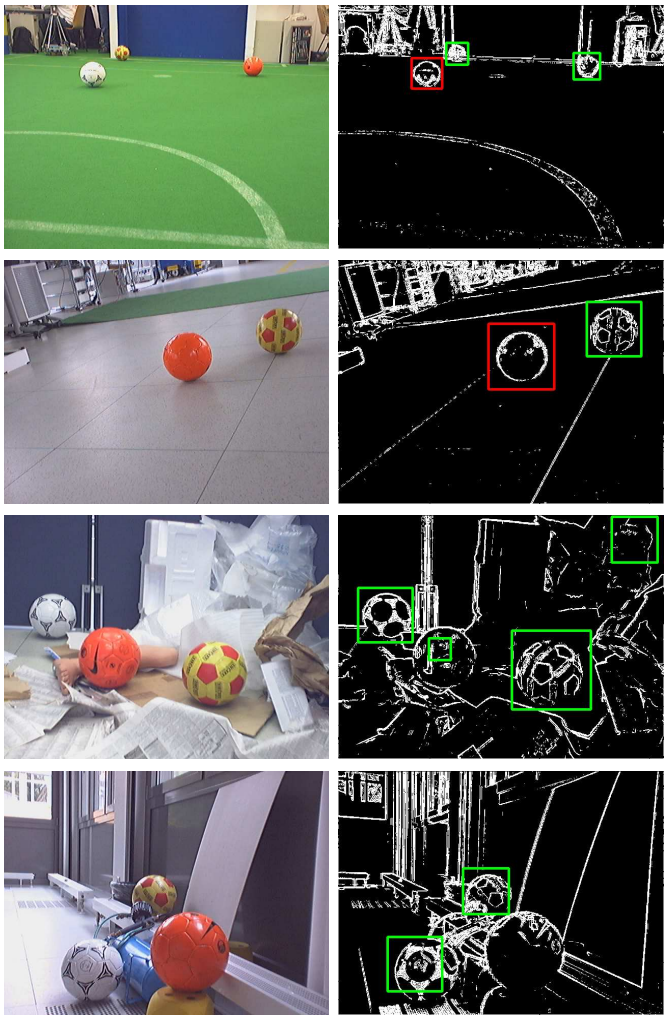

Fig. 8. Sample input images of each of the four categories (From top to bottom: RoboCup, uniform background, RoboCup Rescue and complex scenes) and the corresponding results of the two classifiers.

relatively simple images were used for training, where the images contained no background noise. The idea behind this approach was to make sure that the classifier would only use

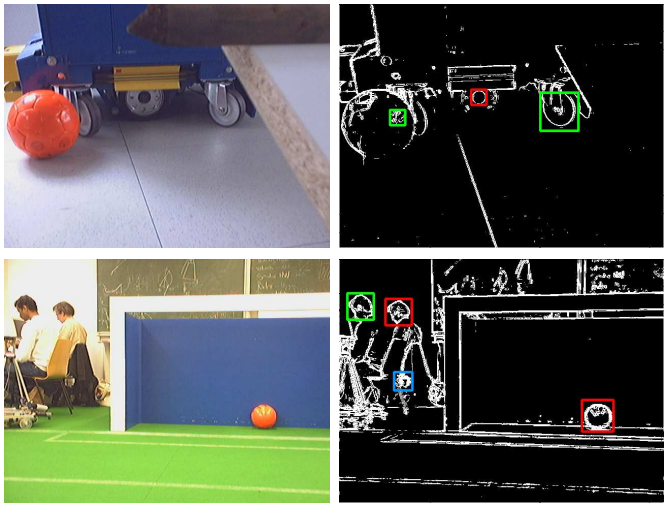

Fig. 9. Left: Input images including round objects. Right: False detections in filtered images.

information about the ball itself and that complex data would only be confusing and would eliminate all useful information about the ball shape. For the second classifier (Table II) a wider range of training data was used, including images with different lighting, in different surroundings (similar to the complex images used for testing), etc. The difference in performance is quite clear from the detected images (Fig. 8).

It can also be observed from the results that we will always face the difficulty of false positives when differentiating between footballs and any other round objects. Examples of such false positives are shown in Fig. 9.

V. Conclusions

The Gentle Ada Boost algorithm uses Classification and Regression trees (CARTs) with four splits to construct a cascade of classifiers to detect Haar-like features in integral images. To ensure the color-invariance of the input images, they are first preprocessed by applying an edge detection Sobel filter to each of the images and passing them through a threshold to rid them of all their color information. This has proven to be a relatively successful technique to be used by the autonomous mobile robot Kurt3D to detect footballs.

It has been found that there are quite a few parameters that need to be adjusted to get satisfactory results from the algorithms, such as the filtering algorithm used, its parameters, the number of splits in the CART, the number of training images used, and above all, the selection of training images. The aim of this paper was not to discuss the details of all these parameters, but the most important ones have been mentioned, and their influence on the results shown.

Although the results may not seem very positive, what we are concerned with is how it will perform in the RoboCup environment. In this case, the reliability of the algorithm seems to be sufficient. Even if the ball is not detected in one in every 5 pictures, for example, the robot will still be able to follow it quite confidently.

There still remains a lot of room for improvement, though, especially concerning the false detection of all round objects (see Fig. 9), as well as undetected footballs with lots of background noise, or partially visible footballs. This could be achieved by combining our approach with other techniques or by integrating 3D images to include depth information. Another idea is to use attention algorithms - assuming the color of the football is previously known - to define regions of interest in order in which to search for balls in order to eliminate false positives. It is planned to work on this in the near future.

References

[1] J. Bruce T. Balch and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the IEEE/RSL International Conference on Intelligent Robots and Systems*, volume 3, pages 2061–2066, 2000.

[2] G. Coath and P. Musumeci. Adaptive arc fitting for ball detection in robocup. In *APRS Workshop on Digital Image Analysing*, 2003.

[3] M. Das and J. Anand. Robust Edge detection in noisy images using and adaptive stochastic gradient technique. In *Proceedings of the 1995 International Conference on Image Processing (ICIP '95)*, Rochester, MI, USA, 1995.

[4] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the 13th International Conference*, pages 148 – 156, 1996.

[5] RoboCup. Robot World Cup Soccer Games and Conferences. "http://www.robocup.org".

[6] A. Haar. Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen*, (69):331 – 371, 1910.

[7] T. Bandlow M. Klupsch R. Haneka and T. Schmitt. Fast image segmentation, object recognition and localization in a robocup scenario. In *3. RoboCup Workshop, IJCAI'99*, 1999.

[8] R. Lienhart, L. Liang, and A. Kuranov. A Detector Tree of Boosted Classifiers for Real-time Object Detection and Tracking. In *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME '03)*, New York, USA, July 2003.

[9] R. Lienhart and J. Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. In *Proceedings of the IEEE Conference on Image Processing (ICIP '02)*, pages 155 – 162, New York, USA, Septmber 2002.

[10] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of the 6th International Conference on Computer Vision (ICCV '98)*, Bombay, India, January 1998.

[11] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23 – 38, January 1998.

[12] S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, Inc., Upper Sanddle River, NJ, USA, 1995.

[13] H. Surmann, A. Nüchter, and J. Hertzberg. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Journal Robotics and Autonomous Systems*, 45, December 2003.

[14] H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg. 6D SLAM - preliminary report on closing the loop in six dimensions. In *Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*, Lissabon, Portugal, June 2004.

[15] Kai Pervölz Andreas Nüchter Hartmut Surmann and Joachim Hertzberg. Automatic reconstruction of colored 3d models. In *Proceedings of Robotik 2004, VDI-Berichte 1841*, pages 215 – 222, 2004.

[16] Andr Treptow and Andreas Zell. Real-time object tracking for soccer-robots without color information. *Robotics and Autonomous Systems*, to appear.

[17] P. Viola and M. Jones. Robust Real-time Object Detection. In *Proceedings of the second International Workshop on Statistical and Computational Theories of Vision – Modeling, Learning, Computing and Sampling*, Vancouver, Canada, July 2001.

Acknowledgements

We would like to thank Matthias Hennig, Kai Lingemann and Joachim Hertzberg for supporting our work.