Combining Adaboost Learning and Evolutionary Search to select Features for Real-Time Object Detection

André Treptow and Andreas Zell
University of Tuebingen
Department of Computer Science WSI-RA
Sand 1
D-72076 Tuebingen, Germany
Email: treptow@informatik.uni-tuebingen.de

Abstract— **Recently Viola et al. [13] presented a method for real-time object detection in images using a boosted cascade of simple features. In this paper we show how an Evolutionary Algorithm can be used within the Adaboost framework to find new features providing better classifiers. The Evolutionary Algorithm replaces the exhaustive search over all features so that even very large feature sets can be searched in reasonable time. Experiments on two different sets of images prove that by the use of evolutionary search we are able to find object detectors that are faster and have higher detection rates.**

I. Introduction

Detecting objects in images in real-time is a challenging problem. Due to the time constraint (40ms to process one frame) one has to find a good trade-off between high detection rates and runtime. Discriminative and fast to compute features have to be found that allow to build robust classifiers. Viola at. al. [13] were the first who developed a real-time frontal face detector that achieves comparable detection and false positive rates to actual state-of-the art systems [5][16][4][15]. According to Viola et al. the detector is about 15 times faster than the method described by Rowley et al. [5], which is considered to be one of the fastest systems.

Due to the robustness and real-time capability of the approach, different publications emerged that enhanced the idea of boosting simple weak classifiers. Li et al. [17] describe a variant of Adaboost called Floatboost for learning better classifiers. Lilienhart et al. [14] showed that extending the basic feature set yields detectors with lower error rates. However, extending the feature set leads to much higher training times because exhaustive search over all possible features is done. To reduce training time McCane et al. [10] proposed a simple heuristic search known as local search to find suboptimal features. However, they only use the base feature set with 4 different types of features and are only able to find classifiers that have slightly worse detection rates than those produced with exhaustive search. Bartlett et al. [12] also use a heuristic to find promising features: After selecting 5% of all possible features randomly, they refine their selection by shifting, scaling and reflecting the best found feature in small steps.

Recapitulating the research that is based on the publication of Viola et al. one can see that there are mainly two problems to deal with: Extending the feature set and being able to search over these very large sets in reasonable time. Another problem is, that one does not know the best feature base in advance. To overcome these problems, we use an Evolutionary Algorithm in combination with Adaboost to search over a large number of possible features. Our goal is to find faster classifiers that use fewer and more significant features and that achieve comparable or even better classification results compared to the classifiers that are trained with Adaboost in combination with exhaustive search over a small set of features.

The use of Evolutionary Algorithms in the field of image processing, especially automatic learning of features for object detection, is a field of research which receives growing interest. Howard et al. [3] apply Genetic Programming (GP) to build a classifier that detects ships in satellite images. Krawiec [8] extends standard GP by a local search operation for visual learning. Lin et al. [9] propose a co-evolutionary GP to learn composite features based on primitive features that are designed by human experts. Bala et al. [6] combine a Genetic Algorithm (GA) with decision tree learning: The GA selects a good subset of features from a fixed set and a decision tree is learned to build the detector structure. Guarda et al. [1] combine a GA to select different convolution masks (features) with GP to evolve the final detector based logical combinations of pixel convolutions in subwindows.

Our approach pays special attention to the real-time capability of the resulting object detector that is meant to be applied in real-time vision tasks on our robots to detect and track objects in gray value images. Therefore, the EA operates on a large set of fast to calculate features. Instead of evolving a detector structure by GP, the Adaboost algorithm learns a weighted linear combination of the best features that were selected by evolutionary search. The final detector can be evaluated very quickly because of this simple linear structure and the fast wavelet-like features.

0-7803-8515-2/04/$20.00 ©2004 IEEE

The paper is organized as follows: The Adaboost learning procedure proposed in [13] is introduced in section II. Section III describes our idea of integrating an evolutionary search into the Adaboost framework. To demonstrate the advantages of this combination, we evaluate different object detectors, which are trained with our algorithm and compare detection rates and runtimes on two different image sets. The results of those experiments are shown in section IV. Section V summarizes our work and points out perspectives for future research.

II. Adaboost Learning of Object Detectors

Recently, Viola and Jones developed a reliable method to detect faces in pictures in real-time. An object that has to be detected is described by a combination of a set of simple Haar-wavelet like features shown in figure 1.

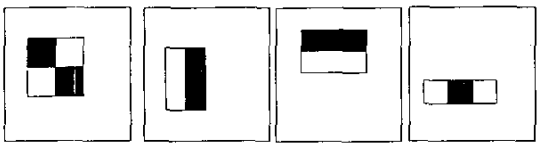

Fig. 1. Four different types of rectangle features within their bounding box. The sum of pixels in the white boxes are subtracted from the sum of pixels in the black areas.

The advantage of using these simple features is that they can be calculated very quickly with the use of a so called "integral image". An integral image II over an image I is defined as follows:

$$II(x,y) = \sum_{x' \leq x, y' \leq y} I(x',y') \quad (1)$$

In [13] it is shown that every rectangular sum within an image can be computed with the use of an integral image by four array references. In our implementation the total integral image is calculated in less than 1 ms on a Athlon XP 1600.

To detect an object a classifier has to be trained consisting of several discriminating features within a subwindow. The possible positions and scales of the four different base types within a box sized 24x24 allow more than 160000 (far more than the number of pixels!) alternative features. Therefore, one has to select a small set of features that describe the object that has to be detected. Adaboost [18] is a mechanism so select a low number of good classification functions, so called "weak classifiers", to form a final "strong classifier", which is a linear combination of the weak classifiers. In the context of learning features, each weak classifier $h_j(x)$ consists of one feature f_j:

$$h_j(x) = \begin{cases} 1 & : \; if \; p_j f_j(x) < p_j \theta_j \\ 0 & : \; otherwise \end{cases} \quad (2)$$

where θ_j is a threshold and p_j a parity to indicate the direction of the inequality.

The algorithm to select a predefined number of features given a training set of positive an negative example images

1) Input: Training examples (x_i, y_i), $i = 1..N$ with positive ($y_i = 1$) and negative ($y_i = 0$) examples.

2) Initialization: weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ with m negative and l positive examples

3) For t=1,...,T:
   a) Normalize all weights
   b) For each feature j train classifier h_j with error $\epsilon_j = \sum_i w_{t,i} |h_j(x_i - y_i)|$
   c) Choose h_t with lowest error ϵ_t
   d) Update weights: $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$ with
   $e_i = \begin{cases} 0 & : \; x_i \text{ correctly classified} \\ 1 & : \; \text{otherwise} \end{cases}$
   and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

4) Final strong classifier:
$h(x) = \begin{cases} 1 & : \; \sum_{t=1}^{T} \alpha_t h_t(x) \geq 0.5 \sum_{t=1}^{T} \alpha_t \\ 0 & : \; \text{otherwise} \end{cases}$
with $\alpha_t = log(\frac{1}{\beta_t})$

Fig. 2. Adaboost learning algorithm as proposed in [13].

is shown in figure 2. The Adaboost algorithm iterates over a number of T rounds. In every iteration, the space of all possible features is searched exhaustively to train weak classifiers that consist of one feature. To train a single weak classifier, one has to find the threshold θ_j for the feature value to discriminate between positive and negative examples. Due to the fact that every weak classifier only has to be better than a guess, calculation of the threshold is very simple: We determine the mean value of the feature responses on the positive examples m_{pos} and the mean value of the feature results on the negative examples m_{neg}. The threshold is calculated as

$$\theta = \frac{m_{pos} + m_{neg}}{2} \quad (3)$$

After choosing the best weak classifier concerning the weighted classification error on the training set, all training examples are reweighted, to concentrate in the next round on those examples that were not correctly classified. At the end, the resulting strong classifier is a weighted linear combination of all T weak classifiers.

III. Evolutionary Algorithm for Feature Selection

Our goal is to replace the exhaustive search over all features (step 3b) in figure 2) by an evolutionary search (see figure 3) and to increase the number of possible features.

We enlarge the base feature set, so that we come up with 6 different base types that are shown in figure 4.

```
Step 3)b) EASearch()
begin
  t2:=0;
  initialize_feature_population(P(0));
  repeat
    P' := select(P(t2));
    crossover(P');
    mutate(P');
    train_classifiers(P');
    evaluate_classification_error(P');
    P(t2 + 1) := replace(P(t2), P');
    t2:=t2+1;
  until terminated;
end;
```

Fig. 3. Evolutionary Algorithm which replaces step 3)b) in figure 2

The weights w_i, $i = 1, ..., 4$ are integers between -4 and 4. Remember that the base features in [13] are special cases of our extended feature set. The first feature in figure 1 for example is described by feature type $t = 3$ with $w_1 = 1$, $w_2 = -1$, $w_3 = -1$, $w_4 = 1$.

We use these 6 base types to be comparable to [13]

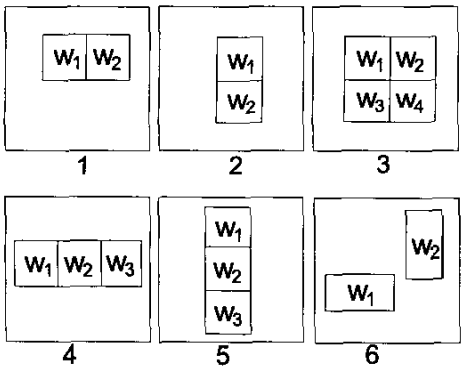

Fig. 4. New feature set: 6 different geometrical layouts

considering the runtime for calculating the feature response on an image. The most complex feature is the first one shown in figure 1 which requires 9 lookup operations in the integral image to calculate the result. None of the features in the extended set requires more than 9 lookup operations. The most general feature is feature number 6 in the new set (see figure 4). The two regions of interest can cover non symmetrical dependencies with larger spatial distance.

We encode every feature by a string of up to 13 integer variables (see figure 5):

- Base-type $k \in \{1, 2, ..., 6\}$ of the feature describing one of the six different geometrical layouts.
- Position (x_{tl}, y_{tl}) of the upper left corner within the detector subwindow
- Position (x_{br}, y_{br}) of the lower right corner within the detector subwindow
- Weights $w_i \in \{-4, ..., 4\}, i = 1, ..., 4$
- If $k = 6$: Upper left (x'_{tl}, y'_{tl}) and lower right (x'_{br}, y'_{br}) corner of second feature box

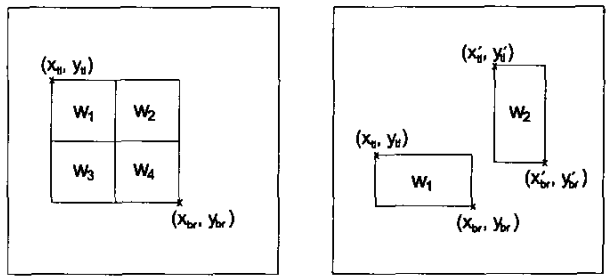

Fig. 5. Parameters for features ($k = 3$ and $k = 6$)

Therefore, the genotype of an individual is represented by the following 13-dimensional integer string: $(t, w_1, w_2, w_3, w_4, x_{tl}, y_{tl}, x_{br}, y_{br}, x'_{tl}, y'_{tl}, x'_{br}, y'_{br})$. With this representation, the problem of selecting features becomes a constrained non-linear integer programming problem. As a fitness function for evaluating individuals we use the error function $\epsilon_j = \sum_i w_{t,i} |h_j(x_i) - y_i)|$ which is the same as in the original Adaboost training procedure. Therefore, to calculate the fitness of an individual j, we first evaluate the feature on every training example and determine the threshold θ_j to build a single weak classifier h_j. Fitness is then calculated as the mean classification error of the weak classifier on the training set. Individuals that are not suitable for building a classifier with low classification error will be penalized with a low fitness value and features that are highly discriminative will receive a high fitness value.

The evolutionary search is driven by two main operators, crossover and mutation. We use is a standard uniform crossover. Given two parents A and B the resulting offspring C is calculated as follows:

$$C_i = \begin{cases} B_i & : \ r \leq 0.5 \\ A_i & : \ otherwise \end{cases}, \ i = 1...n \qquad (4)$$

where r is a uniform random number $\in [0, 1]$ and n describes the length of the individuals.

Mutation of an individual is done by the following scheme:

1) Choose new type $t \in \{1, ..., 6\}$ with probability p_{mt}
2) Choose new weight with probability p_{mw}
3) Mutate positions of feature corners by adding a random constant (x_{rm}, y_{rm}), $x_{rm}, y_{rm} \in \{-3, ..3\}$

We use a repair operator on individuals that are no longer feasible after applying mutation and crossover. Individuals for which the upper left and lower right feature corners are in wrong order are repaired by altering corner positions. As the feature value has to be average free, the weights are rescaled by the repair operator so that they sum up to zero. In case of feature type $k = 6$, the repair operator also adjusts the sizes

of the two regions of interest so that they are equal.
In our EA, parents are selected randomly and children replace the parent population with a standard $(\mu + \lambda)$ replacement.

IV. Experiments

In the following we compare the standard Adaboost learning with exhaustive search (ExBoost) against our combination of Adaboost and evolutionary search, which we will call EABoost. ExBoost searches over the initial limited feature set, which is shown in figure 1, while EABoost applies our extended set of features. For training and testing we use two different image sets: One set containing faces and another set containing soccer balls. The face image set is provided by P. Carbonetto [2] and contains 4916 images showing different faces and 7872 images which do not show a face. Figure 6 shows some of the face and non face images. The face images seem to be the same as in the original experiments described by Viola and Jones. The gray value images have the size of 24x24 pixels and are variance normalized. The face/non face sets are split randomly into a training and a test set containing 2423 positive examples (faces) and 3737 negative examples (non faces) each. We sorted out those images from the original image set that were duplicate or too similar to others, because we do not want to have images in the test set that are too similar to the training set.
The second image set which is used in our experiments contains images showing a soccer ball (490 images per test/training set each) and randomly cropped picture regions where no ball is present at all (4145 images per test/training set each). The ball images have the size of 19x19 pixels.

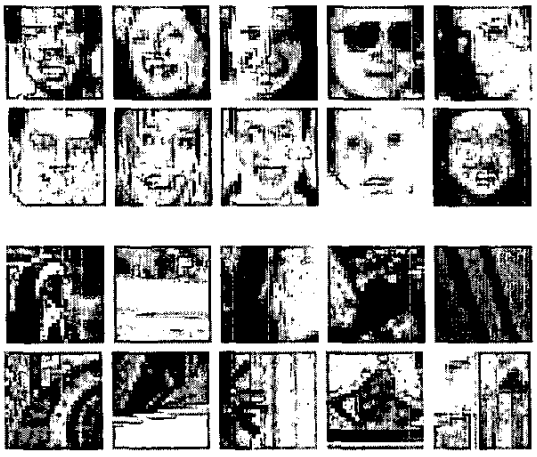

Fig. 6. Images from face/non face set

With both algorithms we trained face and ball detectors using the given training sets. Training was stopped when the resulting strong classifier labeled all examples in the training set correctly. Parameters used for evolution were: Population size $P = 250$, 20% of all individuals undergo crossover ($p_c = 0.2$), 80% of all individuals were mutated ($p_m = 0.8$) and the population was initialized randomly. In the following,

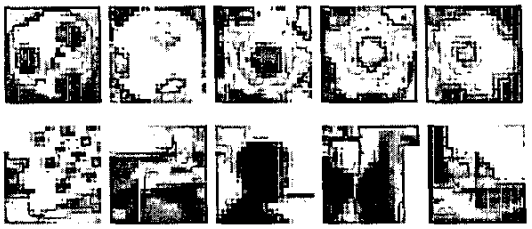

Fig. 7. Examples from the image set with balls/non balls

averaged results over 20 runs of the EABoost experiments are shown. The EA terminated if the population was converged to a good solution so that no better individual was found within the next 50 generations. If convergence did not occur within 400 generations, the EA was stopped as well. Experiments were carried out on an Pentium III 650MHz processor.

The classification rates and the false positive rates for both algorithms during training are shown in figure 8 and 9. For both training sets EABoost is able to find classifiers with a lower number of features compared to ExBoost. Table I shows the best, average and worst number of features. Note that the classifiers learned with EABoost use only 75%/82% (face set/ball set) of the number of features of ExBoost. Therefore, as shown in table IV, the classifiers can be evaluated faster.

Algorithm	best	average	worst
ExBoost (face set)	227	227	227
EABoost (face set)	158	171	184
ExBoost (ball set)	132	132	132
EABoost (ball set)	99	108	121

TABLE I
NUMBER OF SELECTED FEATURES

Algorithm	average time per iteration	average total time
ExBoost (face set)	185.2s	42040s
EABoost (face set)	63.5s	10868s
ExBoost (ball set)	57.8s	7627s
EABoost (ball set)	42.2s	4568s

TABLE II
TRAINING TIMES

The results show, that training times are also reduced by the use of EABoost (see table II). The mean time for the search for one weak classifier on the face set was 64 seconds for EABoost compared to 185 seconds for ExBoost which is a speedup of 2.9. In the ball set, we have smaller images, so that the search space is not as large in the face set. In this case, training times for one iteration with EABoost are comparable to ExBoost. Note that the total training times for EABoost are much shorter due to the reduced iteration times and the reduced total number of classifiers.

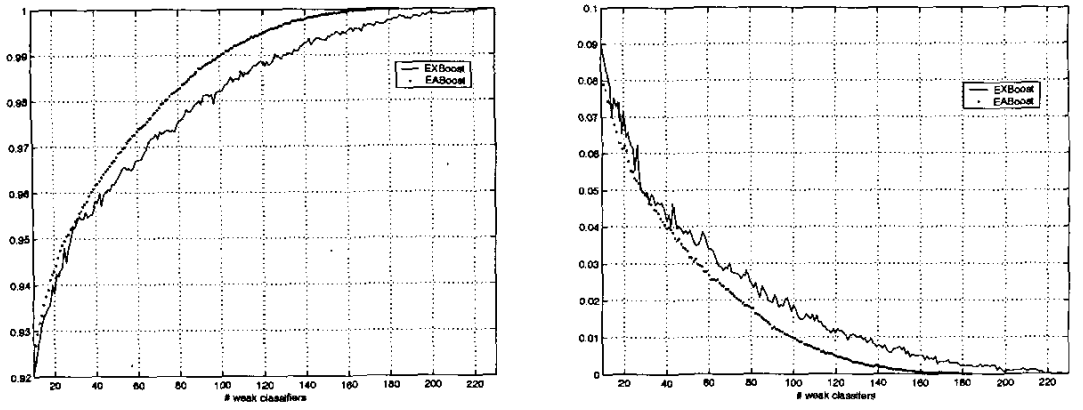

Fig. 8. Classification rates (left) and false positive rates (right) on training set (face set)

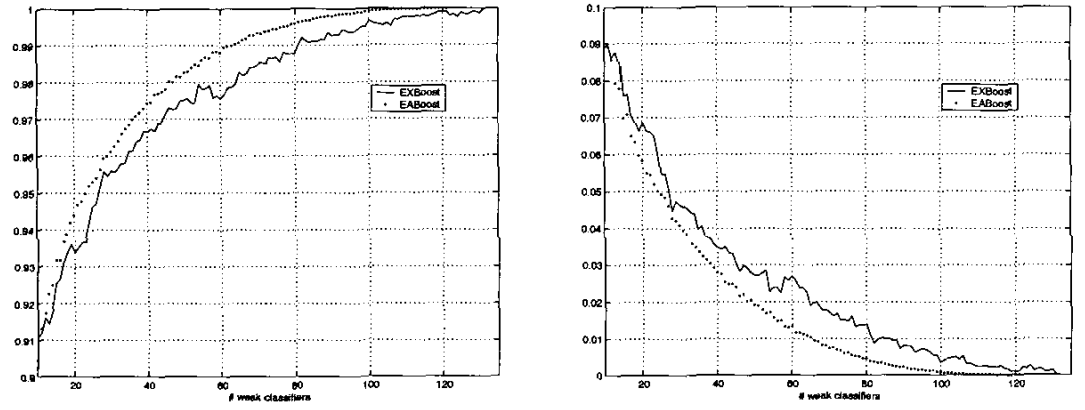

Fig. 9. Classification rates (left) and false positive rates (right) on training set (ball set)

Algorithm	best	average	worst
ExBoost (face set)	96.1%	96.1%	96.1%
EABoost (face set)	96.9%	96.7%	96.4%
ExBoost (ball set)	97.5%	97.5%	97.5%
EABoost (ball set)	98.3%	98.0%	97.7%

TABLE III
CLASSIFICATION RATES ON TEST SETS

Algorithm	best	average	worst
ExBoost (face set)	2.44s	2.44s	2.44s
EABoost (face set)	2.10s	2.28s	2.43s
ExBoost (ball set)	1.10s	1.10s	1.10s
EABoost (ball set)	0.98s	1.08s	1.19s

TABLE IV
RUNTIMES FOR FINAL CLASSIFIERS ON COMPLETE TEST SETS

The learned detectors are evaluated on the two test sets to compare detection and false positive rates (see figures 10 and 11). We can see that although the detectors that are learned with EABoost use a lower number of features, they are able achieve higher detection rates with lower false positive rates on the test sets. For the face set, the best evolved detector (concerning detection rate on test set) uses only 163 features and classifies 96.9% of the face set correctly, compared to a classification rate of 96.1% achieved by the detector with 227 features found by ExBoost. On the set with ball images, the best evolved detector uses 119 features and classifies 98.3% of the set correctly, whereas the detector with 132 features found by ExBoost achieves a classification rate of 97.5%.

It is interesting to have a look at some of the features, that EABoost evolved. Figure 12 shows the first six features that were selected for one of the strong classifiers by EABoost. As one can see, the features mainly cover the regions around the eyes due to the fact that these regions are characteristically for faces. The newly proposed feature type with two unconnected regions can be found within the first significant features, too. This feature pays attention to the fact, that mostly the area in the middle of the forehead (spotlight) is brighter than the area of the hair. Because those situations can better be tracked by loosely coupled features, it is harder to find an

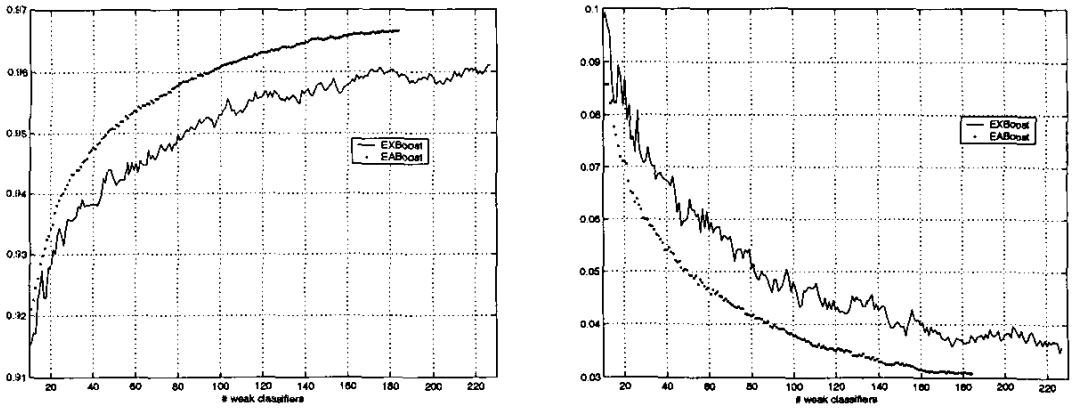

Fig. 10. Classification rates (left) and false positive rates (right) on test set (face set)

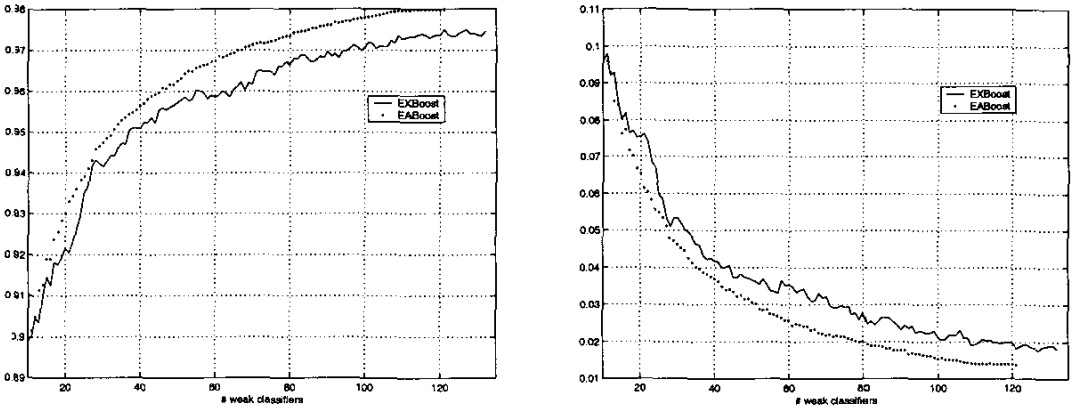

Fig. 11. Classification rates (left) and false positive rates (right) on test set (ball set)

appropriate classifier with one of the four base types with directly connected regions of interest.

V. Conclusion and Future Work

In this paper we extended the approach of Viola et al. [13], which is one of best known methods to train real time object detectors. We presented a new combination of Adaboost learning with an Evolutionary Algorithm to build robust and fast classifiers. Our approach can cope with a large set of possible features that cannot be searched exhaustively. This makes it possible to train classifiers that have higher detection rates, which can even be evaluated faster, compared to the combination of Adaboost with exhaustive search on small initial feature sets. Training times were also reduced, making it possible to use larger feature sets on a high number of training examples.

Nevertheless, there are many directions to go for further research. First of all, we would like to extend our EA to a so called Memetic Algorithm [11]. In this case, every individual is improved by a heuristic Local Search operator after mutation and crossover has been applied. In our algorithm, improvement could be done e.g. by translating or scaling the feature in small steps for a number of iterations to see if the classification error can be reduced.

Combinations of evolutionary search and other boosting algorithms like e.g. GentleBoost [7] or FloatBoost [17] should also be analyzed in the future.

In the second part of their paper, Viola et al proposed the use of a cascade of multiple strong classifiers to increase detection speed. The use of detector cascades was not addressed within this paper. It was our aim, to show how to improve the feature selection procedure that is a substantial part of the cascade training so that improving feature selection implies improving the whole cascade. However, learning the structure of a cascade of detectors is a difficult optimization problem that we plan to cope with more sophisticated heuristics in the future.

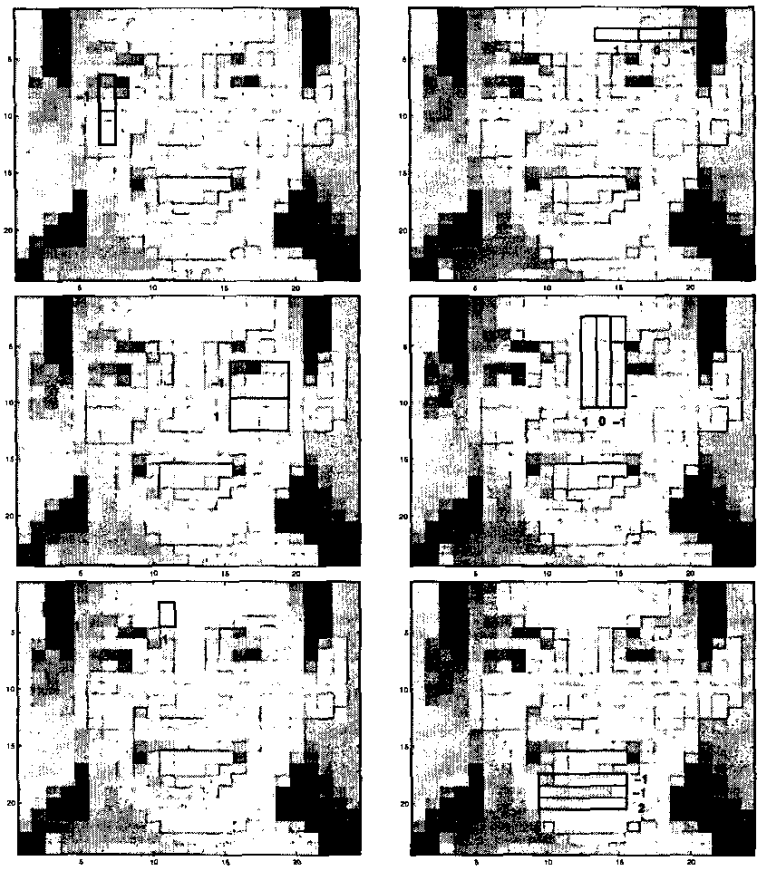

Fig. 12. First six evolved features

REFERENCES

[1] A. Guarda, C. Le Gal and A. Lux. Evolving visual features and detectors. In *International Symposium on Computer Graphics, Image Processing, and Vision*, 1998.

[2] P. Carbonetto. Viola training data [Database]. URL http://www.cs.ubc.ca/~ pcarbo/viola-traindata.tar.gz.

[3] D. Howard, S. C. Roberts and R. Brankin. Evolution of ship detectors for satellite SAR imagery. In *Genetic Programming: Second European Workshop EuroGP'99*, pages 135–148, 1999.

[4] D. Roth, M. Yang and N.Ahuja. A snowbased face detector. In *Advances in Neural Information Processing Systems 12 (NIPS 12)*, volume 12, 2000.

[5] 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.

[6] J. Bala, K. DeJong, J. Huang, H. Vafaie and H. Wechsler. Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation*, 4(3):297–312, 1997.

[7] J. Friedman, T. Hastie and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of statistics*, 38(2):337–374, 2000.

[8] K. Krawiec and B. Bhanu. Visual learning by evolutionary feature synthesis. In *Twentieth International Conference on Machine Learning (ICML 2003)*, pages 376–383, 2003.

[9] Y. Lin and B. Bhanu. Learning features for object recognition. In *Genetic and Evolutionary Computation (GECCO-03)*, pages 2227–2239, 2003.

[10] B. McCane and K. Novins. On training cascade face detectors. In *Image and Vision Computing*, pages 239–244, New Zealand, 2003.

[11] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report 826, California Insititue of Technology, 1989.

[12] M.S. Bartlett, G. Littlewort, I. Fasel and J.R. Movellan. Real time face detection and facial expression recognition: Development and application to human-computer interaction. In *CVPR Workshop on Computer Vision and Pattern Recognition for Human-Computer Interaction*, Vancouver, Canada, 2003.

[13] P. Viola and M.J. Jones. Robust real-time object detection. In *IEEE Workshop on Statistical and Theories of Computer Vision*, 2001.

[14] R. Lienhart, A. Kuranov and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM'03, 25th Pattern Recognition Symposium*, pages 297–304, 2003.

[15] H. Schneiderman and T. Kanade. A statistical method for object detection applied to faces and cars. In *International Conference on Computer Vision and Pattern Recognition*, pages 1746–1759, 2000.

[16] K. Sung and T. Poggio. Example-based learning for view-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:39–51, 1998.

[17] S.Z. Li, Z.Q. Zhang, Harry Shum and H.J. Zhan. Floatboost learning for classification. In *16-th Annual Conference on Neural Information Processing Systems (NIPS)*, Vancouver, Canada, 2002.

[18] Y. Freund and R.E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, September 1999.