Framework for Research on Detection Classifiers

Michal Hradiš*
Faculty of Information Technology, Brno University of Technology

Abstract

Detection of patterns in images with classifiers is currently one of the most important research topics in computer vision. Many practical applications such as face detection exist and recent work even suggests that any specialized detectors (e.g. corner-point detectors) can be approximated by very fast detection classifiers. In this paper, we analyze the requirements on tools which are needed when experimenting with detection classifiers and we present a general framework which was created to fulfill these requirements. This framework offers high performance for training, high variability, elegant handling of configuration and it is able to meet all the requirements which arise when experimenting with almost all possible kinds of detection classifiers. The framework offers good testing support, full supporting infrastructure and some useful training algorithms and features. We offer this framework for research and educational purposes and we hope it will allow lower initial investments when experimenting with detection classifiers.

CR Categories: I.5.2 [Design Methodology]: Classifier design and evaluation

Keywords: Detection, Face Detection, Classification, Image Processing, Computer Vision, AdaBoost, WaldBoost, Cascade of Classifiers, Corner Points, Classifier Evaluation

1 Introduction

Detection of 2D patterns in images has many real-world applications ranging from camera orientation to computer-human interaction. In past years, many purpose specific and application specific detectors were proposed. Many of those classifiers were carefully designed by experienced scientist and engineers to achieve high detection rates and at the same time low computational cost. Examples of such detectors are various corner detectors or other interest point detectors [Mikolajczyk and Schmid 2004, Kadir and Brady 2001], road sign detectors [Escalera et al. 1997], and the frontal face detector based on a cascade of boosted classifiers by Viola and Jones [2001]. Some of the detectors (mostly the face detectors) use structure of classifiers which employs a mechanism of attention of focus. Such detectors use simple and fast classifiers to reject the most common negative samples and then they use gradually more complex classifiers to deal with the more difficult and rare negative samples.

When designing detectors of 2D patterns for real-time applications, the demands for high precision and low computational cost and the effort to optimize both of them at the same time, consumes most of the development time. Recently Šochman and Matas [2007] have proposed a unified approach how to emulate behavior of any existing detector by sequential classifier which is optimal in terms of computational complexity for desired detection precision. They argue that, when using their approach, it is possible to skip the process of optimization and finding a fast and still precise enough approximation to the original detector, which can be sometimes very difficult for humans. Instead, the main effort is put into finding a suitable set of features which are then automatically combined into a WaldBoost ensemble. In their work, they report that they managed to automatically create classifiers emulating two interest point detectors, Hessian-Laplace [Mikolajczyk and Schmid 2004] and Kadir-Brady [2001] saliency detector, while achieving 70x faster detection times over the original Kadir-Brady detector.

Considering the wide area of application of classifier-based 2D pattern detectors and the amount of research effort invested into this area in recent years, it is surprising that only minimal publicly available support for researchers in this field is available. This fact is in high contrast to classification and pattern recognition in general, where research tools and libraries like LIBSVM, Matlab toolkits, Yet Another Learning Environment and many others are generally known and used.

The lack of specialized tools is most restraining at the beginning of the research, as training and testing of detection classifiers has some special characteristics which result in the fact that it is necessary to invest a lot of effort before any experiments can be conducted. Usually, some set of tools or a framework providing annotation and extraction of samples from images, bootstrapping, handling of special representations of samples (e.g. integral image, frequency image), feature extraction, non-maxima suppression, testing, etc. has to be created in advance. Some care has to be also given to memory and computational efficiency. Otherwise the experiments may be constrained by available time or equipment.

To our knowledge, the only publicly available and useful tool for training of detection classifiers is the implementation of cascade of boosted classifiers with extended set of Haar-like features [Lienhart, and Maydt 2002] which is similar to the original algorithm used by Viola and Jones [2001] in their frontal face detector. This implementation is available as a part of the OpenCV library. Unfortunately its use as a base for research efforts is relatively limited, as it is not well documented, it is not modular enough and it does not provide good support for evaluation of results.

Due to all the facts mentioned before, we have created a general framework which can be used as a base for research in the field of detection classifiers.

The rest of this paper is structured as follows. First, some of the current approaches to training of detection classifiers are described in Section 2. In Section 3, demands for an experimental framework are formulated and analyzed. General design ideas of the created framework are presented and some of its interesting parts are described in more detail in Section 4. In Section 5, some results achieved with current algorithms implemented in the framework are presented together with discussion of learning efficiency. Finally, the paper is concluded and ideas for future work are presented in Section 6.

2 Detection classifiers

The first practically applicable real-time detector of 2D patterns in video based on classifiers was the frontal face detector proposed

*e-mail: ihrais@fit.vutbr.cz

by Viola and Jones [2001]. This detector combines a cascade of boosted classifiers with Haar-like features and a novel image representation called the integral image to reach low false positive rates and very low average computational cost.

Each stage of the cascade in the Viola and Jones detector (see Figure 1) is a single classifier trained by AdaBoost algorithm [Freund and Schapire 1997] which is tuned to reject fair portion of background samples (e.g. true negative rate (TNR) = 0.5) while keeping almost all face samples (e.g. false negative rate (FNR) = 0.001). All the samples which are not rejected are further processed by following stages of the cascade. During training, background samples are bootstrapped after each stage from a large pool of images to keep the size of negative training set constant. This cascade reaches very low average classification time thanks to the fact that the majority of samples classified during scanning of images correspond to background. Also very low false positive rate (FPR) is reached this way.

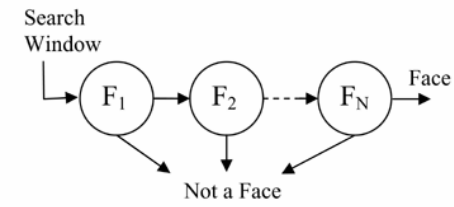

Figure 1. The schema of detection cascade.

The cascade structure of the detector provides good trade-off between precision of classification and average decision time; however, it still is not enough for real-time applications. Viola and Jones achieved further significant speed-up by using simple weak classifiers which use Haar-like features (see fig 2). These features can be computed very fast and in constant time for every scale when integral image is used.

Figure 2. The basic Haar-like features which were used by Viola and Jones [2001] in their frontal face detector.

The original approach by Viola and Jones was subsequently extended by many researches who mostly focused either on modifying the cascade structure to make better use of gained information [Šochman and Matas 2004a, Xiao et al. 2003] or on extending the set of Haar-like features [Lienhart and Maydt 2002, Mita et al. 2005], optimizing the precision-speed trade-off [Šochman and Matas 2005], on better search in the feature space [Li et al. 2002, Šochman and Matas 2004b], or on using more discriminative features (e.g. PCA) in the later stages [Zhang et al. 2004] where Haar-like features may not bring much benefit anymore due to the very difficult decision problem.

Approaches to face detection which differ significantly more from the Viola and Jones detector were also proposed. Some of them reach high detection rates, but they usually do not provide real-time performance. Some of such approaches use neural networks [Rowley et al. 1998] or support vector machines (SVM) [Rätsch et al. 2004].

The area of face detectors based on classification of 2D patterns has been already very well explored and the possible needs of future research can be very well extrapolated from the previous research. On the other hand, the use of detection classifiers for other tasks, such as interest point detection, is a relatively new topic. Fortunately, the ideas from face detectors are in most cases general enough to apply also to detection of other patterns. Evidence for this conclusion can be found in the work of Šochman and Matas [2007] where they use WaldBoost algorithm with Haar-like features to emulate the Hessian-Laplace detector. This approach is the same as they use for face detection [Šochman and Matas 2005]. The only difference when they emulate the Kadir-Brady [2001] detector is that they extend the set of features by energy features.

3 Research Framework Design Choices

Experimenting with 2D pattern detection classifiers has many specifics which need to be considered when designing an experimental framework. Some of the specifics arise directly from the nature of the task. These include the need of annotating objects in images, handling of image data, testing the created classifiers by scanning of images and also using some kind of non-maxima suppression to post-process the raw detections from scanning. Other requests come from the training algorithms which are used. These requests include the need of a bootstrapping mechanism and the need of an importance sampling mechanism. Further, multiple representations of samples may be needed to be able to evaluate different types of features efficiently. The framework should be also modular enough to allow simple addition of new features and classification algorithms and to allow the coexistence and combination of these individual parts in classifiers. Let's now look at some of the aspects in more detail.

The basic function of the experimental framework is loading of samples. There are some fundamental differences in how the positive samples (patterns of interest) and negative samples (background) are obtained.

The positive samples are usually extracted from hand-annotated images. The framework should be able to process such annotations and cut out the samples. On the other hand, when conducting larger number of short experiments (e.g. tuning of parameters), it is desirable to cut the samples as a preprocess step. Some of the works [Xiao et al. 2003] suggest that it is beneficial to extend the positive dataset by applying random transformations (e.g. affine transformation or noise) to the annotated positive samples. In such case, it may be desirable to use importance sampling or even bootstrapping [Šochman and Matas 2005] on the positive set. To make this possible, applying of the random transformations must be integrated into the framework instead of applying the transformations as an of-line preprocess step. When emulating corner point detectors and other similar detectors, it may be desirable to integrate the emulated detectors into the framework to gain more flexibility for experiments.

When training detectors, large amounts of background samples are needed (up to bilions). It is not possible to keep all of these samples in memory or to store them individually. The background samples must be definitely obtained at random positions from larger images on demand during the training. However, it still may not be effective to keep all the source images in memory and some mechanism to exchange the images should be provided. When training face detectors, the background samples are ususally extracted from images which do not contain any faces. However, being able to cut background samples from images with annotated objects of intereset may provide better classification performance and it is necessary when training for example corner

point detectors. The loading of background samples may consume a significant part of training time, especially considering that during bootstrapping, it is necessary to classify the samples by the partially trained classifiers. Fortunately, it is possible to paralelize this part efficiently.

Some of the algorithms, especially the boosting algorithms which are used to train detection classifiers, are very simple. Compared to that, the bootstrapping mechanism which is used by all of the algorithms, may pose a significant additional effort to implement. This fact suggests that the framework should provide a common bootstrapping mechanism which could be used by the training algorithms. The same is true for importance sampling.

The fact, that many of the detection classifiers use Haar-like features which can be evaluated in constant time independent on their size, offers a possibility to represent the samples as indexes into integral representation of the source images from which the samples are cut. This approach is, however, very constraining. It requires all of the source images to be loaded in memory at all times and it significantly reduces possibilities of using other types of features and other types of classifiers. Such solution also limits the possibility of applying random transformations to the annotated data. A more suitable solution is to rescale all samples to single size.

There also should be multiple representations of the samples available. The integral images are suitable for Haar-like features, but other features need different representations for efficient evaluation. Basic gray-scale representation should be available to be used by algorithms like SVM and neural networks and features like Local Binary Patterns (LBP) [Ojala and Pietikäinen 1999] and PCA. Frequency representation is suitable for Gaabor wavelets and for other filters. Even color information can be needed in some cases. It is not desirable to keep all of the representations at all times, but rather to create them on demand of features and classifiers. In some cases, it may be even beneficial for the features or classifiers to have the possibility to attach some kind of "user" data to the samples. For example, in the case of the Local Rank Differences (LRD) [Zemčík et al. 2007], sums of rectangular areas of only few different sizes can be computed only once and stored to significantly speed-up the training process.

The main testing procedure for detection classifiers consists of multi-scale scanning of images, non-maxima suppression and comparing the detections with a ground truth. This basic testing procedure is used for all kinds of detectors and also the reported measures are usually the same. That is the reason for which any framework for experimenting with detection classifiers should provide this method of testing for any classifier that can be created in it. To make it possible, the samples have to be loaded and represented during testing in the same way as during training (rescaled to single size, the same representations available). This may not be the most efficient way to scan images, especially when using Haar-like features, but it is the only way to keep the framework general.

The general test provided by the framework should be ideally able to produce ROC, DET and precision-recall curve for multiple lengths of a classifier (if it has multiple stages). It should also provide average speed of a classifier and it should plot a curve describing the amount of samples which reach the individual stages of the classifier.

It is desirable that the framework provides some support for logging information about the training process. This information may for example include the fraction of area under precision-recall curve after each stage, amount of already classified positive and negative samples and speed and duration of training.

In general, it is necessary that the framework is modular as much as possible. It must be possible to add new learning algorithms, features, test and sources of data individually without any need of understanding the rest of the framework. It must be also possible to combine different features and classification algorithms into single detector. Only when these conditions are met, it is possible to efficiently use such framework for research.

4 Framework Description

Based on the analysis which is discussed in the previous section, we have created a framework for research on detection classifiers of 2D patterns. The main design objective was to provide a flexible and general framework which can be used as a high-performance basis for experiments in this field. Another objective was to allow good portability between hardware and software platforms. Finally any inherent dependencies on commercial libraries and tools were avoided to allow using of the framework for research and possibly educational purposes without any expenses. The last requirement is valid especially in the case of developing countries.

Considering all of the objectives (especially portability, high performance and modularity), C++ was chosen as the most appropriate development language. Significant advantage of using C++ is also the fact that many freeware or open source C/C++ libraries and tools exit. Currently library libxml2 is used in the framework to load configuration and store results, GNUPlot is used to visualize results and some optional parts of the framework use the OpenCV library. Additionally OpenMP application programming interface can be used to boost performance on shared-memory multiprocessor platforms.

The structure of the framework

The framework itself consists mostly of definitions of interfaces which allow adding of new parts almost in plug-and-play fashion. These interfaces are available for all parts of the framework which are expected to be experimented with. These parts include sources of samples, features, testing methods, bootstrapping and importance sampling, cascade-like algorithms, stage classifier algorithms, weak learning algorithms, trained classifiers and weak hypotheses.

The part of the framework which provides loading of samples has three levels. The topmost level provides a uniform way to load samples from a dataset and is used by the training algorithms. The middle level corresponds to subsets of samples which are available in a dataset. The objects corresponding to the subsets distribute requests for samples between individual physical sources of samples which form the bottom level. The types of individual sources of samples are hidden behind a general interface and thus new types of sources of samples can be easily added. The structure of this part of the framework can be seen in Figure 3.

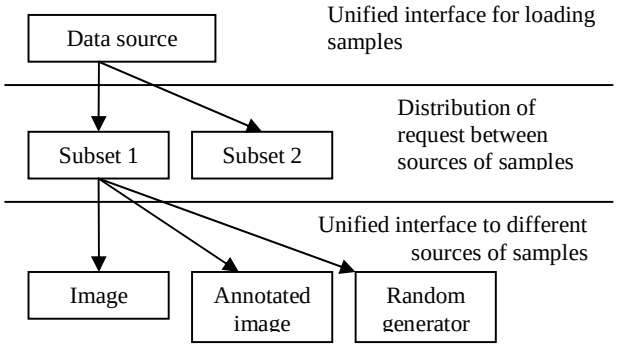

Figure 3. The part of the framework which takes care of loading samples.

Interfaces that form the part of the framework which is dedicated to training of the detection classifiers is shown in Figure 4. These interfaces provide abstraction for most of the independent parts of training algorithms which follow the idea of the face detector by Viola and Jones [2001]. These cascade-like algorithms use a bootstrapping mechanism for loading of samples. The samples can be subsequently passed to stage learning algorithms which create monolithic classifiers. These algorithms operate on fixed set of samples, but can use provided importance sampling to speed-up training.

The weak learning algorithms, which are normally used by the stage learning algorithms, create individual weak hypotheses. Although only domain partitioning weak hypotheses [Schapire and Singer 1999] are supported, the generality of the framework is not reduced. The reason for this is that all the weak hypotheses can be very well approximated by the domain partitioning hypotheses. Not much difference exists between the stage classifiers and weak hypotheses – both can be created by the same algorithms. The reason for the existence of two separate interfaces is that the weak hypotheses contain equivalent of alpha values from the original discrete AdaBoost algorithm [Freund and Schapire 1997] which are set by the boosting algorithms. Finally, the weak learning algorithms use features to transform the image data.

During training, objects which contain multiple features are used and the individual features are accessed by unique integer indices. This allows efficient evaluation of features like Gabor wavelets or LRD which can share certain computations between individual features. When the best feature is selected, an object containing only this single feature is created. Such object is further wrapped by a weak hypothesis and attached to the stage classifier. When the stage classifier is finished, it is attached to the cascade-like classifier. The interfaces of classes which can be combined into a classifier and their relations are shown on figure 5.

The structure of the training part of the framework is specially designed to fit the algorithms similar to the cascade of boosted simple classifiers; however, this structure is also able to accommodate most other approaches by skipping some of its parts. For example, SVM or neural networks can be implemented as stage learning algorithms and they can use the data from samples directly. Another example is the WaldBoost algorithm which does not use stage learning algorithms, but uses directly weak learners.

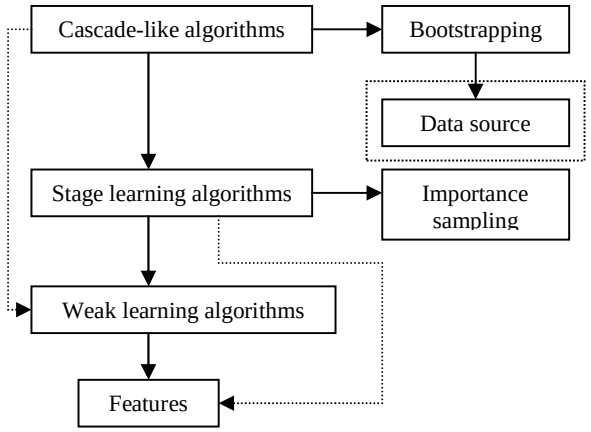

Figure 4. The interfaces which form the training part of the framework together with their dependencies.

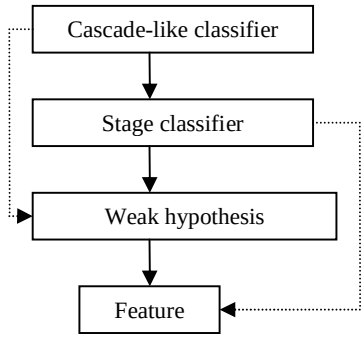

Figure 5. The interfaces which are defined in the framework to provide support for created classifiers.

Samples

All the samples in the framework are represented by individual objects which are chained into linked lists. This is much different to general classification environments, where datasets are usually represented by matrices of floating-point values.

Representing of samples as individual objects provides possibility to structure the information and store multiple representations of the original data as well as any other additional information. Samples can be currently represented in the framework by gray-scale images, integral images, and frequency images. Also, information, such as class of the sample, its weight (which is used by boosting algorithms), standard deviation of pixels, or identification of the exact origin of the sample (e.g. the name of the source image and position within it) can be stored into the sample object. Additionally, the samples provide general purpose storage which can be used to store temporal information or even additional non-visual information connected to the sample.

Testing

The framework provides an interface which allows users to create test of their own. Currently, the framework contains implementations of two tests. One of them is designed to test individual samples and the other is a scanning test. When testing, samples are loaded from a data source in the same way as during training. This implies that the test does not have direct control over which samples are loaded and does not necessarily know

where the samples are from exactly. This is not a problem when testing on individual samples, since a single subset from the data source can in this case contain samples of only single class. But it must be dealt with in the case of scanning test, as the subset from which the samples are loaded has to contain all sub-windows from the scanned images. The sub-windows which are classified to contain the pattern of interest have to be further processed by some non-maxima suppression algorithm and finally the resulting detections have to be compared to a ground truth annotation.

The implemented scanning test achieves this thanks to the fact that the framework provides the possibility to load samples in linear order and that the samples that are cut from images receive an identification string which contains the name of the source image file, position in the image and size of the original sub window. Based on the textual identification, an appropriate annotation file is found and loaded. Also, the position and size of the sample is extracted from the identification string.

Configuration

The framework, as it has been described up to this point, lacks one crucial feature to become really useful research tool. The missing feature is the possibility to define experiments in some external configuration file. Such task would be very simple if it was not necessary to maintain the independency between the individual parts of the framework. However, the framework provides quite an elegant support for configuration.

To store the configuration information, the framework uses XML files in which each XML node corresponds to a single object. During initialization, the XML file is parsed by libxml2 library which produces a tree structure representing the content of the XML file. This tree structure is consequently processed by constructors of objects in the framework. When a constructor processes an xml node, it offers the child nodes to other constructors. An object of specific class is created only when the constructor receives xml node of appropriate name. Each interface which is defined in the framework has a function assigned to it which, when given an XML node, tries to create objects of classes which implement that interface. This way, it is sufficient to register new classes in appropriate loading functions and then all configuration information can be processed in their constructor.

Parallelization

When analyzing algorithms which are similar to the original Viola and Jones algorithm, it can be identified that selection of weak classifiers consumes the most significant part of the training time. The framework offers a way to reduce this time by parallel execution of this part. When multiple weak learners are defined, the framework is able to assign each of them to separate thread and execute them in parallel while achieving almost linear speed-up.

Another part of the training algorithms which consumes significant amount of time is bootstrapping. This part is also parallelized in the framework. During bootstrapping, each thread is loading samples from separate physical sources of samples and classifies the samples by the provided classifier.

5 Results

Currently, the whole basic infrastructure of the framework is implemented as well as many of the variable parts. The available boosting algorithms are AdaBoost and GentleBoost. From the detection classifiers, the framework contains classifier cascade and the WaldBoost algorithm [Šochman and Matas 2005]. Also some weak learners were implemented – e.g. decision trees and histogram weak learners. Haar-like features, LRD and LBP are also available. The framework further provides support for loading of samples from multiple sources and support for testing by scanning images.

The testing procedure which is currently implemented in the framework is able to scan images and use non-maxima suppression which was suggested by Šochman and Matas [2007]. The testing procedure creates receiver operating characteristic (ROC), detection error trade-off curve (DET) and precision-recall curve for multiple stages of the tested classifier. It is also able to plot the dependency of average speed of the classifier on the length of the classifier.

To demonstrate the capabilities of the framework, we have trained a WaldBoost classifier for frontal face detection task. The classifier was trained on 5000 hand-annotated faces which were divided into training and validation sets. Additionally, random affine transformations were applied to the original samples until the sizes of both sets reached 20000 samples. In each iteration of the training algorithm, non-face samples were bootstrapped from a large pool of images without any faces to keep the number of non-face samples in training and validation set at 20000. The final size of the classifier was set to be 1000 and the alpha value which defines the False Negative Rate of the resulting classifier was set to 0.1.

The progress of fraction of area under precision-recall curve on the training and validation set can be seen in Figure 6. Figure 7 shows the amount of negative samples which reach the individual stages of the classifier.

The classifier was tested on MIT+CMU frontal face dataset. The classifier needed to evaluate on average 5.4 weak hypotheses per window to make a decision. The graph describing speed of the classifier can be seen in Figure 8. The ROC and the precision-recall curve which describe the performance of the classifier on the test dataset can be seen in Figure 9 and in Figure 10. Finally, graphical representation of the detections can be seen in Figure 11.

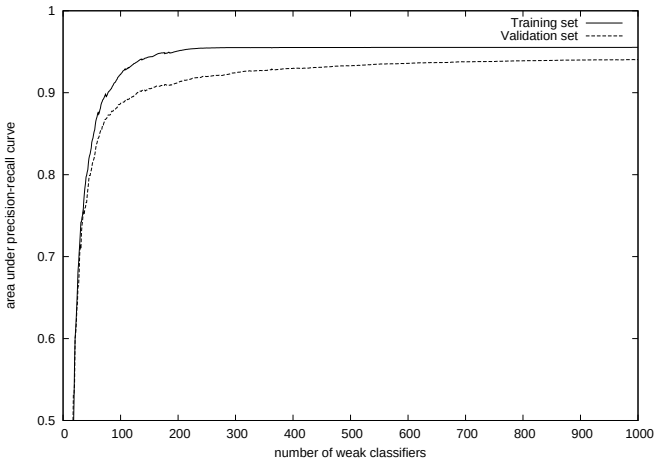

Figure 6. The area under precision-recall curve on the training and validation sets for the in individual stages of the classifier.

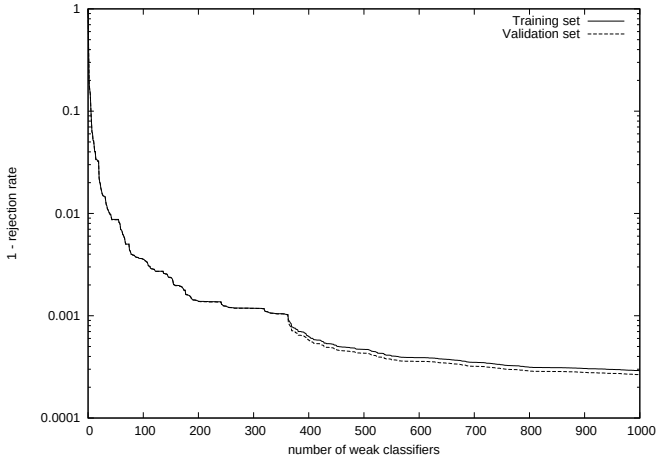

Figure 7. The fraction of negative samples of the training and validation which reach individual stages of the classifier.

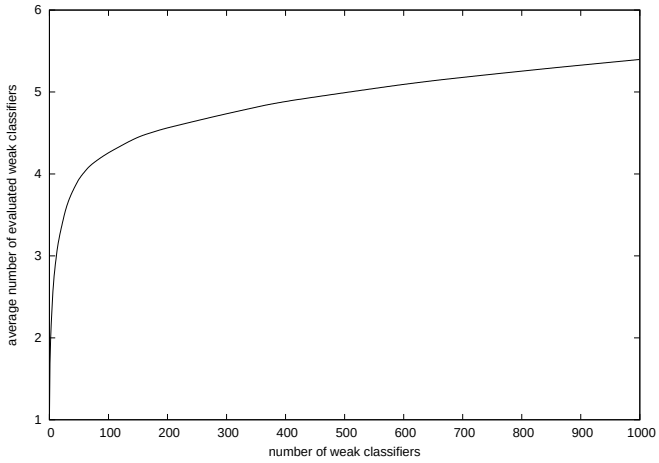

Figure 8. Speed of the classifier.

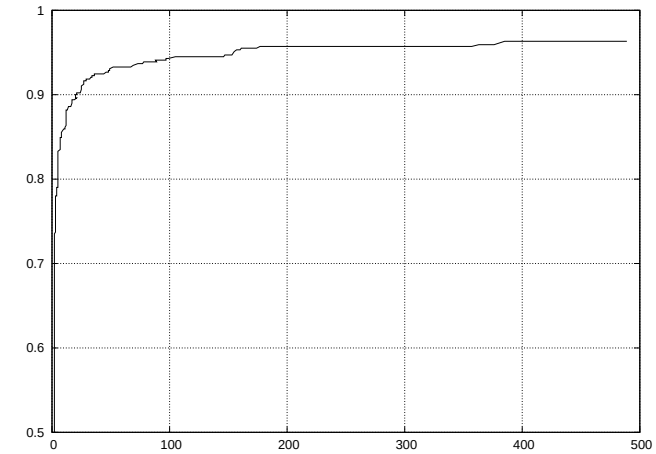

Figure 9. The ROC curve on the MIT+CMU frontal face dataset.

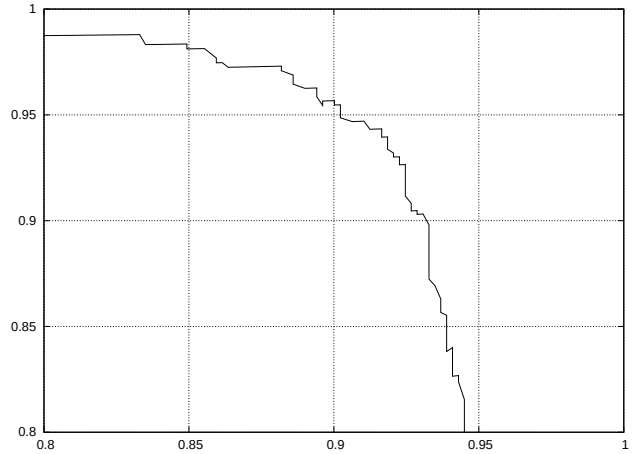

Figure 10. The precision-recall curve on the MIT+CMU frontal face dataset.

One of the objectives of the framework was to reach high performance for training of detection classifiers. This objective was reached by optimal use of cache memory during training of weak classifiers, by optimized evaluation of features and by parallelization. We measure the speed of training of weak classifiers as number of features which are evaluated per second during the training (features per second). All the measurements were performed on a blade server with two Intel Xeon E5245 quad core processors running at 2.33GHz.

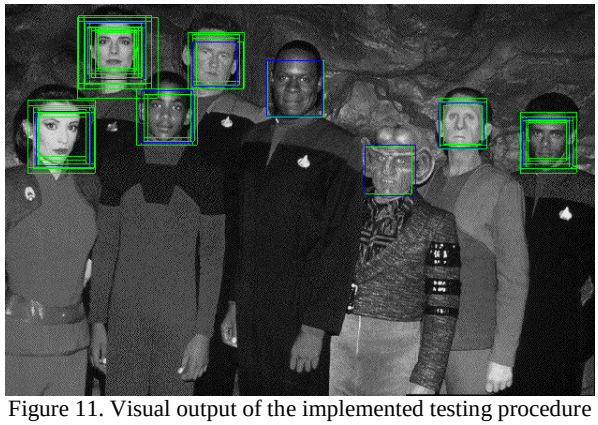

Figure 11. Visual output of the implemented testing procedure

When using Haar-like features with simple weak learners, the framework was able to reach speed 60M features per second while running in a single thread. With LRD, the framework reached speed of 45M features per second. Speeds when running in multiple threads were also measured. For this test, Haar-like features were used with more complex weak classifiers. The results can be found in Figure 11. It can be seen that the speedup is almost linear and reaches 640 % of the speed of single thread when all 8 threads are used. The speed of bootstrapping was measured to be 3,5M samples per minute for the trained classifier when using single thread.

Number of threads	1	2	3	4	5	6	7	8
Speed	42	80	116	148	160	198	236	270

Figure 11. Speed during training of weak classifiers for different numbers of threads. The results are in millions of evaluated features per second.

6 Conclusion and Future Work

This paper presents a framework which was designed to be used as a base for research on detection classifiers of 2D patterns. This framework offers high-performance in training, high variability, good configurability and it is able to meet all the requirements which arise when experimenting with almost all possible kinds of detection classifiers. At this point, the framework offers good testing support, full supporting infrastructure and some useful training algorithms and features which allow training of classifiers with high precision of detection and low computational cost.

Anyone who plans to participate in research of detection classifiers can find in this paper a basic analysis of the requirements for tools that are needed for conducting experiments in this field. This information should be useful to decide if it is effective to create new tools or it is more effective to search for tools which have been already created. If the later is found to be true, we encourage you to contact us, as the framework which we described in this paper is freely available for research and educational purposes.

At this point, the framework itself is finished and further work will be mostly focused on increasing precision of the trained classifiers. This may be achieved by implementing new algorithms and new features or by integrating libraries like libsmv into the framework.

Currently, data which is suitable for frontal face detection is prepared and ready to use. In the future, we want to extend the experiments to other detection problems such as detection of out-of-plane rotated faces or emulation of existing corner point detectors.

Acknowledgements

This work has been supported by the “Centre of Computer Graphics” (CPG-LC06008), Czech Ministry of Education, Youth, and Sports, CareTaker, IST EU project number 027231, and Czech Grant Agency, project GA201/06/1821 “Image Recognition Algorithms”.

References

BAE, H., KIM, S. 2005. Real-time face detection and recognition using hybrid-information extracted from face space and facial features, IVC(23), No. 13, 29 November 2005, pp. 1181-1191.

ESCALERA, A., MORENO, L. E., SALICHS, M. A., ARMINGOL, J. M. 1997. Road traffic sign detection and classification. *Industrial Electronics, IEEE Transactions on*, Vol. 44, No. 6. (1997), pp. 848-859.

FREUND, Y., SCHAPIRE, R. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119--139.

KADIR, T., BRADY, M. 2001. Saliency, Scale and Image Description. International Journal of Computer Vision, Volume 45, Number 2 / November

LI, S., ZHANG, Z., SHUM, H., ZHANG. H. 2002. FloatBoost learning for classification. In S. Thrun S. Becker and K. Obermayer, editors, NIPS 15. MIT Press.

LIENHART, R., MAYDT, J. 2002. An extended set of Haar-like features for rapid object detection**.** Image Processing. 2002. Proceedings. 2002 International Conference on, Volume 1, Issue , 2002 Page(s): I-900 - I-903 vol.1

MATAS, J, ŠOCHMAN, J. 2007. Wald's Sequential Analysis for Time-constrained Vision Problems. In ICRA.

MIKOLAJCZYK, K., SCHMID, C. 2004. An affine invariant interest point detector. International Journal of Computer Vision, Volume 60.

MITA, T., KANEKO, T., HORI, O. 2005. Joint Haar-like features for face detection. Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, Volume 2, Issue , 17-21 Oct. 2005 Page(s): 1619 - 1626 Vol. 2

OJALA, T., PIETIKÄINEN, M. 1999. Unsupervised texture segmentation using feature distributions. Pattern Recognition, 32(3), 1999, s. 477-486.

ROWLEY, H., BALUJA, S., KANADE, T. 1998. Neural Network-Based Face Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 1, January, 1998, pp. 23-38.

RÄTSCH, M., ROMDHANI, S., VETTER, T. 2004. Efficient Face Detection by a Cascaded Support Vector Machine Using Haar-Like Features. In proceeding of DAGM-Symposium, Lecture Notes in Computer Science.

SCHAPIRE, R., SINGER, Y. 1999. Improved boosting algorithms using confidence-rated predictions. In: Machine Learning, (1999) 37(3):297-336.

ŠOCHMAN, J., MATAS, J, 2004a. Inter-stage Feature Propagation in Cascade Building with AdaBoost. In ICPR 2004.

ŠOCHMAN, J., MATAS, J, 2004b. AdaBoost with Totally Corrective Updates for Fast Face Detection. Sixth IEEE International Conference on Automatic Face and Gesture Recognition, p. 445.

ŠOCHMAN, J., MATAS, J. 2005. WALDBOOST — Learning for Time Constrained Sequential Detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2, s. 150.

ŠOCHMAN, J., MATAS, J. 2007. Learning A Fast Emulator of a Binary Decision Process. In ACCV.

VIOLA, P., JONES, M. 2001 Rapid object detection using a boosted cascade of simple features. In CVPR.

XIAO, R., ZHU, L. AND ZHANG, H.J. 2003. Boosting Chain Learning for Object Detection, ICCV03, Nice, France.

ZEMČÍK, P., HRADIŠ, M., HEROUT, A. 2007. Local Rank Differences - Novel Features for Image Processing. Poster MLMI.

ZHANG, D., LI, S. Z., GATICA-PEREZ, D. 2004. Real-Time Face Detection Using Boosting in Hierarchical Feature Spaces. Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02.