ENGINE FOR REAL-TIME 2D OBJECT DETECTION

Roman Juránek
Doctoral Degree Programme (1), FIT BUT
E-mail: ijuranek@fit.vutbr.cz

Supervised by: Pavel Zemčík
E-mail: zemcik@fit.vutbr.cz

ABSTRACT

This work was motivated by project CareTaker on which the Department of Computer Graphics and Multimedia on FIT participates. The project focuses on development of surveillance system for underground stations. The system for real-time object detection is one of the project goals. This paper proposes detection engine that uses WaldBoost based image classifiers to process video. Before the engine is presented, boosting training methods are summarized and principle of object detection with classifiers is explained.

1 INTRODUCTION

The common approach to object detection is to use a binary classifier and scan input image with sliding window representing a sub-image, which is classified. This approach was first used in combination with Adaptive Boosting (AdaBoost) by Viola and Jones [1]. In their work they have used classifier consisting of small set of Haar-like features to detect faces. To reduce average number of evaluated weak hypotheses they have used classifier cascade. Another approach to reduce classification time was presented in [2] where early-termination version of AdaBoost was introduced as training method suitable for time constrained applications - i.e. online object detection.

2 ADABOOST TRAINING

The AdaBoost was introduced by Freund and Schapire in [3]. Schapire and Singer [4] introduced real AdaBoost which allows confidence rated predictions and is most commonly used in combination with domain partitioning weak hypotheses (e.g. decision trees). Many modifications were proposed since then e.g. FloatBoost [5] or Total Corrective Step [6].
The basic algorithm incrementally builds strong classifier from simple weak classifiers (or weak hypotheses). The weak classifier is simple function that is at least slightly better in decision than a random function. Input for the algorithm is labeled training set of samples and set of weak hypotheses (e.g. Haar-like features with threshold or decision tree).
AdaBoost keeps weight distribution for samples in training set. The weight of a sample is higher when the sample is hard to classify to its class. Samples with lower error (easier to classify) has

low weight. In each step AdaBoost first optimizes parameters of weak hypotheses for current sample distribution and then selects weak classifier with lowest error rate. The distribution is then updated according to the selected hypothesis.
AdaBoost proved to be resistant to over-fitting which is due the fact that with growing complexity of the classifier it increases margins between the samples of different classes, but still can over-fit in presence of noise.

2.1 WALDBOOST

The WaldBoost uses real Adaboost to select weak hypotheses in stages. For every stage it adds two early-termination thresholds. Thresholds calculation is based on parameters α (false negative rate) and β (false positive rate). In detection tasks, β is usually set to 0 because we do not want any false detections. Value of α is set to low value because we usually do not mind when some objects are undetected. Higher values of α result in faster and less precise classifiers and lower values result in slower classifiers with low error rate. Speed of classifier is usually measured as an average number of features evaluated per sample.
Example classifiers for this paper were generated by research boosting framework [7] on DCGM FIT. The framework supports various training algorithms (AdaBoost, WaldBoost, etc.), different types of weak learners (Decision tree, Histogram, etc.) and a number of types of image features. The training process is set up by XML configuration where the used algorithm, weak learners and training data are specified. Classifier in XML format is produced as an output.

2.2 OBJECT DETECTION USING A CLASSIFIER

Binary classifier can only decide whether a sample is in *background* or *object* class. To detect objects the image scan with sliding window is performed. In case the objects are of various sizes the image must be processed several times with different sizes of classifier. When detection of differently rotated objects is desired the original image must be rotated and scanned with axially aligned window.

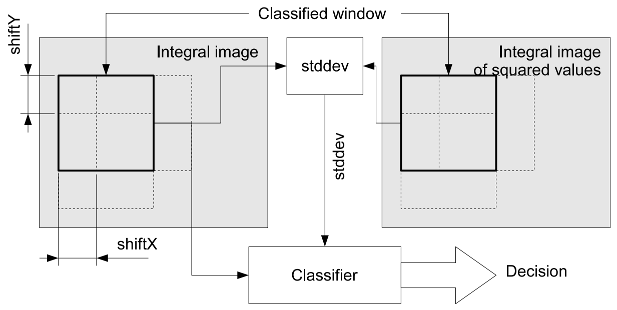

Figure 1: Image scanning principle

The Figure 1 illustrates the scanning process. Input are two integral images [1] on which the feature response is calculated. Actual response is calculated on normal integral image (left). The second one is used only for rapid standard deviation calculation which is needed for Haar feature response normalization.

3 THE ENGINE DESIGN

The engine was implemented in C++ language. The core is object oriented and the structure of classes corresponds to basic classifier elements - features (`TFeature`), weak hypothesis (`TWeakHypothesis`) and classifier (`TClassifier`). Object oriented development allows to add arbitrary type of feature and weak hypothesis. The engine is therefore easily extendible. When the detection is executed, first, the integral representations of input frame are calculated. Image is then scanned with the classifier for several times according to number of defined sizes of scanning window. Before each scan, the parameters of the classifier are adjusted for particular scan size and the image is then processed. Encountered detections are then processed by non-maxima suppression algorithm [8] which removes possible multiple detections of same object.

3.1 CLASSIFIERS, WEAK HYPOTHESES AND IMAGE FEATURES

The very basic element of a classifier is image feature. The engine can handle features of two types – *discrete* which response is an integer, and *continuous* where the response is float. Supported are continuous Haar-like features based on difference of adjacent rectangular regions of image (corresponding to shapes in Figure 2), and one discrete type which quantizes response of a continuous feature $g(X)$ to N levels on specified interval $\langle min; max \rangle$ (Formula 1).

$$f(X) = \left\lfloor \frac{N(g(X) - min)}{max - min} \right\rfloor \tag{1}$$

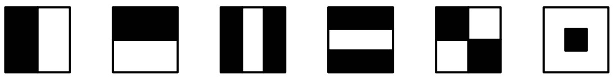

Figure 2: Shapes of Haar features

On higher level there are weak classifiers. There are supported two types of weak hypotheses. Basic hypothesis with one threshold T and given values of $alpha$ and $parity$ (Formula 2). The other type is more general domain partitioning weak hypothesis that assigns a real number to each possible value of some discrete feature – the evaluation is explained in Formula 3, where $f(X)$ is response of a discrete feature and $alpha$ the table with classifier responses.

$$h(X) = sign(f(X) - T) * alpha * parity \tag{2}$$

$$h(X) = alpha\,[f(X)] \tag{3}$$

The classifier is represented by sequence of weak hypotheses and thresholds for each weak hypothesis. The class `TClassifier` provides methods to evaluate classifier on single sample and to scan entire input frame. The class also provides a method to recalculate parameters of all features for a specified scan window size.
The input for a classifier is intensity image and integral image. The images are provided by detector where input intensity image is automatically integrated.
From user point of view a classifier is represented by XML structure which contains sequence of stages with one weak classifier each. In Figure 3 is shown one stage with domain partitioning hypothesis which contains discretized Haar feature.

```
<Stage posT="1E+10" negT="-1.3">
  <DomainPartitionWeakHypothesis
    binMap="0 0 1 2 3 3" alpha="0.0 0.1 0.2 0.3">
    <Discretize min="-2.0" max="2.0" bins="6">
      <HaarHorizontalDoubleFeature x="2" y="4" bw="5" bh="8" />
    </Discretize>
  </DomainPartitionWeakHypothesis>
</Stage>
```

Figure 3: Example of classifier stage in XML

The interface to the detector provides methods for detector initialization from XML structure, multi-scale image scan, result access and parameter setting.

4 DETECTION RESULTS AND APPLICATION

The detection engine was tested during the development on the face detection (which is very typical task) and the dog detection (which was motivated by the CareTaker project). Performance of the detector was measured on 2.33GHz Intel Core2 Duo processor. The performance depends mainly on number of sub-images classified during the image scan which is determined by input image size, scan window size and its shift in position during the scan.
Example of a dog detection is shown in Figure 4. The detection was executed on image of size 512×288px. The performance reached 40 fps with 10 features evaluated per sample in average.

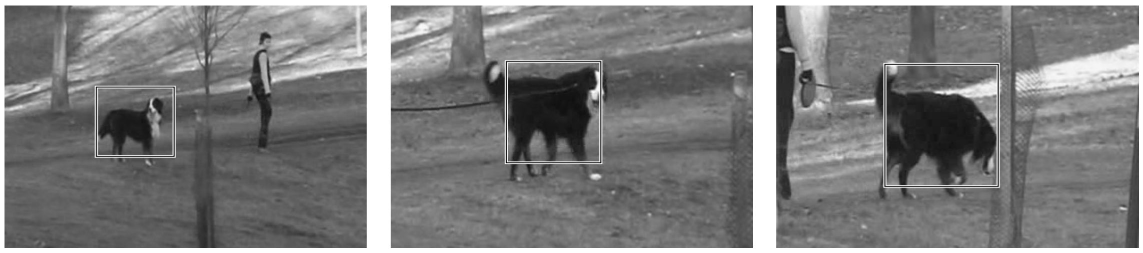

Figure 4: Dog detection in outdoor scene

In the case of face detection (Figure 5), the performance on 320×240px image was 30fps which was due the limited frame rate of used camera. Frame rate achieved on recorded video reached 50fps. The average number of features evaluated per sample was 6.
The detection of 2D patterns has wide range of real world applications from camera orientation to computer-human interaction. Recently Šochman and Matas proposed unified approach how to emulate behaviour of corner point detector by sequential classifier [8].

5 CONCLUSION AND FUTURE WORK

The paper summarized some of typical methods for training of detection classifiers and presented real-time 2D object detection engine that uses XML based WaldBoost classifiers. The engine was successfully tested on face detection and dog detection tasks where the performance

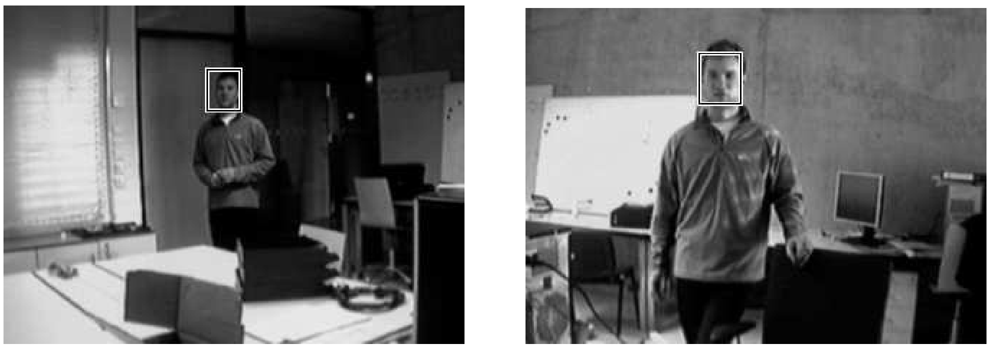

Figure 5: Face detection example

was measured. The engine can process real-time video with low computational requirements. According to experiments the performance is sufficient to process high resolution video. Future effort on the engine development will focus on detection of rotated objects, further engine optimization and multiprocessor environment support.

REFERENCE

[1] Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple Feaures. In CVPR, 2001

[2] Šochman, J., Matas, J.: WaldBoost-Learning for Time Constrained Sequential Detection, In CVPR, 2005

[3] Freund, Y., Schapire, R.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, In Journal of Computer and System Sciences, 1997

[4] Schapire, R., Singer, Y.: Improved Boosting Algorithms Using Confidence-Rated Predictions, In Machine Learning, 1999

[5] Li, S. et al.: FloatBoost Learning for Classification, In: S. Thrun S. Becker and K. Obermayer, editors, NIPS 15, MIT Press, 2002

[6] Šochman, J., Matas, J.: AdaBoost with Totally Corrective Updates for Fast Face Detection, Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004

[7] Hradiš, M.: Framework for Research on Detection Classifiers, In SCCG, Budmeřice, Slovakia, 2008, To be published

[8] Šochman, J., Matas, J.: Learning A Fast Emulator of a Binary Decision Process, In ACCV, 2007