A DETECTOR TREE OF BOOSTED CLASSIFIERS FOR REAL-TIME OBJECT DETECTION AND TRACKING

Rainer Lienhart, Luhong Liang, and Alexander Kuranov

Microcomputer Research Labs, Intel Corporation
Santa Clara, CA, 95052
{rainer.lienhart, lu.hong.liang, alexander.kuranov}@intel.com

ABSTRACT

This paper presents a novel tree classifier for complex object detection tasks together with a general framework for real-time object tracking in videos using the novel tree classifier. A boosted training algorithm with a clustering-and-splitting step is employed to construct branches in the nodes recursively, if and only if it improves the discriminative power compared to a single monolithic node classifier and has a lower computational complexity. A mouth tracking system that integrates the tree classifier under the proposed framework is built and tested on XM2FDB database. Experimental results show that the detection accuracy is equal or better than a single or multiple cascade classifier, while being computational less demanding.

1. INTRODUCTION

Object detection and tracking in video sequences have been intensively researched in recent years due to their importance in applications such as content-based retrieval, natural human computer interfaces, object based video compression, and video surveillance. In these areas statistical learning methods such as neural networks [1,2] and SVMs [3,4] have attracted much attention.

Recently Viola et al. have proposed a boosted cascade of simple classifiers for rapid object detection [5]. Their approach uses Discrete AdaBoost [8] to select simple classifiers based on individual features drawn from a large and over-complete feature set in order to build strong stage classifiers of the cascade. The structure of a cascade classifier is shown in Figure 1(a): At each stage a certain percentage of background patches are successfully rejected, while (almost) all object patterns are accepted. This approach has been successfully validated for frontal upright face detection [5,6].

For visually more complex and diverse object classes such as multi-view faces and mouths, however, a single cascade classifier is overstrained to accommodate all in-class variability without compromising the discriminative power between the objects of interest and the background. One intuitive solution is to divide the object patterns manually into several, more homogeneous sub-pattern classes, construct multiple parallel cascade classifiers each handling a specific sub-pattern, and merge their individual results. This classifier structure is shown in Fig.1(b) and used together with a *coarse-to-fine* strategy in [7] to develop a detector pyramid for multi-view face detection.

Two challenging problems, however, remain and will be addressed by our novel tree classifier: (1) It is empirical and difficult work to determine the right object sub-pattern classes in most cases. For example, intuitively the openness and the appearance (with/without facial hair) are two primary factors of in-class variability of mouth patterns (see Fig.5). However, in practice it is often difficult to group individual patterns into the right sub-pattern class due to ambiguity such a mouth with a shaved, but still visible beard. (2) Multiple specialized classifiers increase the computational complexity conflicting with the real-time requirement in the applied object detection and tracking system.

Contribution: Firstly, a novel detector tree of boosted classifiers is introduced considering both the characteristics of the patterns in feature space and the computational efficiency in order to address the two aforementioned problems. At each node in the tree a clustering-and-splitting step is embedded into the training algorithm to construct branches in the classifier stages recursively, if and only if branching is advantages from a detection accuracy and computational complexity point of view. Secondly, a general integrating framework for object detection and tracking is proposed and empirically validated by means of a real-time mouth tracking system. Experimental results on the XM2FDB database [9] show that the proposed system is at least 15× faster than a system using multiple SVMs [10], 45% faster than a two cascade classifier, and 12% faster than a single cascade classifier while preserving or even exceeding their detection accuracy.

2. DETECTOR TREE OF BOOSTED CLASSIFIERS

A single classifier is often overstrained to learn a complex object pattern class. This difficulty can be overcome by multiple specialized object detectors given that the object patterns have been grouped into appropriate subclasses. However, classification complexity grows linearly with the number of subclasses. Our detector tree can be viewed as merging early stages of multiple specialized cascade classifiers to preserve classification accuracy while reducing the computational complexity (coarse-to-fine strategy). It will start to grow specialized branches if this is beneficial with respect to classification accuracy and computational complexity (divide-and-conquer strategy) (see Fig.1(c)).

Training: Training starts with the root tree node. The root tree node distinguishes itself from over nodes by not having any parent. The positive training set for the root tree is set to the complete positive training set (see Fig. 2).

As shown in Fig.2, the proposed algorithm is a recursive procedure. At each node all positive and negative training samples specified by the parent node are used for train a boosted classifier (step 4) [5,6]. The result of the training is a strong classifier with a given false alarm rate (e.g., 50%) and given hit rate (e.g., 99.9%). Its computational complexity is linear to its number of weak classifiers. Then, in step 6 a *k*-means clustering algorithm is utilized to divide positive samples into *k* subsets. The *k* positive subsets together with all the negative samples are used to train *k* strong classifiers. If the total number of features used by these *k* classifiers is less than that used in the monolithic classifier, the *k* strong classifiers are computational more efficient than the monolithic classifier. Therefore the current cascade is split into *k* branches. Each branch receives only the corresponding subset of positive samples together with a new filtered set of negative samples. Otherwise, the monolithic classifier is used preserving the cascade structure at this node. This procedure is recursively applied until a given target depth of the tree is reached.

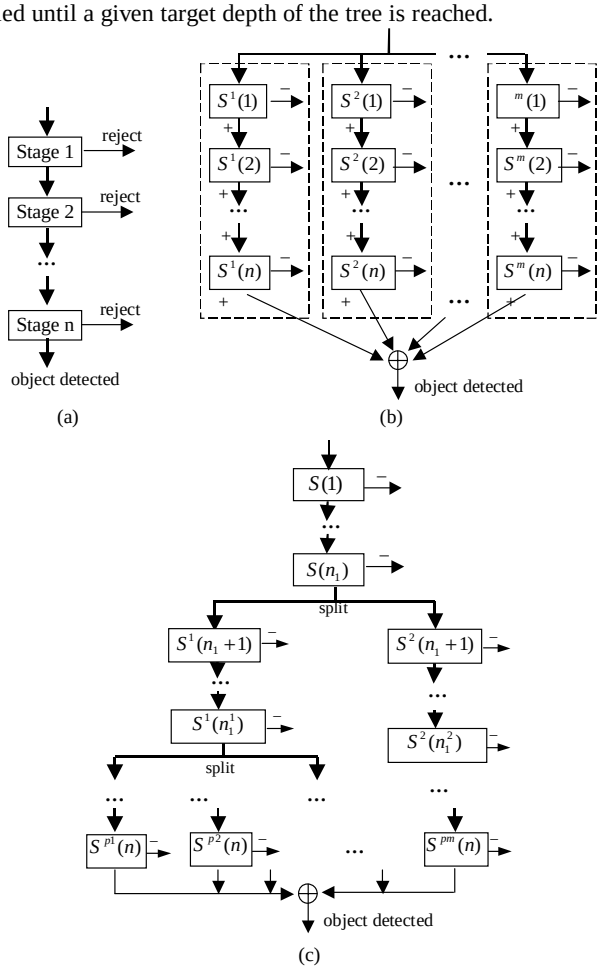

Fig.1: Three different classifier structures: (a) cascade classifier; (b) multiple cascade classifiers; (c) tree classifier

Classification: During classification a depth-first search algorithm is applied to find an *acceptance* path from the root to a terminal node of the detection tree. If the input pattern is rejected by a node's strong classifier, the search will trace back to the nearest upper split point and try another branch, until an *acceptance* path is found or all the possible paths have been searched. If an *acceptance* path is found, the input pattern will be labeled positive, otherwise negative.

Splitting Criterion: Our splitting criterion is based on the minimal number of features (= lowest computational complexity) needed to achieve a given training hit and false alarm rate ignoring detection performance. This is reasonable for the following reasons:

1. A single N stage cascade classifier with a hit rate h_{stage} and false alarm rate f_{stage} per stage will have approximately an overall hit rate of h = pow (h_{stage},N) and false alarm rate of f = pow (f_{stage},N).
2. M parallel cascades will exhibit an overall hit rate of $h \geq$ pow (h_{stage},N) and false alarm rate of $f \leq N$*pow (f_{stage},N). The N times higher false alarm rate can be compensated by training ΔN = log (1/N) / log(f_{stage}) additional stages. Since these additional stages are very unlikely to be ever evaluated they hardly influence the overall computational complexity, but result in the same detection performance. In practice, it can be observed that given the same number of stages, a specialized cascade removed even more background patterns than an identically trained monolithic cascade. Thus, in practice additional stages are not necessary at all.
3. A detection tree with M terminal nodes can be converted into M parallel cascades, thus the reasoning of 2. applies here, too.

```
struct TreeNode {
   BoostedClassifier* bc=0;
   TreeNode* next=0, child=0, parent=0;
   TrainingData *posSampleIdx; // Describes positive training set
   int evaluate( sample ); // Evaluate sample given the tree node by
   // tracing back the path to the root node and constructing a cascade classifier
   TreeNode(TreeNode *_parent, TrainingData *_Idx, TreeNode * _next)
   { parent = _parent; posSampleIdx = _Idx; next = _next; }
};

startTreeTraining()
   1. Create new TN=TreeNode(0, all positive training examples, 0)
   2. nodeTraining(TN, 0, TARGET_HEIGHT_OF_TREE)

nodeTraining(TreeNode* parent, curLevel, stopLevel)
   1. If (stopLevel == curLevel) return;
   2. Load all positive training examples SPOS assigned to the parent node by parent->
      posSampleIdx and filter with parent->evaluate()
   3. Load negative training set SNEG of size CNEG filtered with parent->evaluate()
   4. Train standard stage classifier S^1 with SPOS plus SNEG. Let O(S^1) denote the
      number of features needed for achieving a given performance
   5. BestClassifier = S^1.; BestNoOfFeatures = O(S^1)
   6. For k=2 to K_max
      a. Calculate for SPOS all features used in stage classifier S^1. Do k-means clustering
         on feature data and create k sets SPOS_i of positive training examples.
      b. Train k standard stage classifiers S^k_i on SPOS_i plus SNEG.
      c. If (BestNoOfFeatures > O(S^k_1) + … + O(S^k_k))
         i. BestNoOfFeatures = O(S^k_1) + … + O(S^k_k)
         ii. BestClassifier = { S^k_1, …, S^k_k}
   7. TreeNode* TN_0 = 0
   8. For each classifier S^k_i in BestClassifier
      a. Create new TreeNode * TN_i =TreeNode(parent, SPOS_i, TN_i-1)
      b. nodeTraining(TN_i, curLevel+1, stopLevel)
```

Fig.2: Detection tree training algorithm

3. MOUTH DETECTION AND TRACKING FRAMEWORK

This section describes an approach that integrates the tree classifier into a general framework for object detection and

tracking. The implementation of this framework in this paper focuses on human mouth detection and tracking in video sequences; however it is also applicable to other complex object detection and tracking problems.

As shown in Fig. 3, the kernel of the framework is a finite state machine that consists of two states: detection and tracking. The system starts with the detection state in which the face detector [6,12] followed by the tree classifier for mouth detection is utilized to locate the face of the speaker as well as his/her mouth location. If the detections are successful in several successive frames, the state machine enters the tracking state where only the tree classifier is employed to detect the mouth in the region around the location predicted from previous detection or tracking results. If any detection failure occurs in the tracking state, the state machine switches back to the detection state to recapture the object. In the framework, there is also a post-processing module to smooth the raw mouth locations and conceal accidental detection failures.

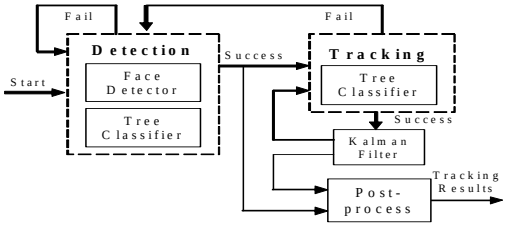

Fig.3: Framework for mouth detection and tracking. The tree classifier represents the mouth detector.

In the detection state, the face detector presented in [6] is used to locate the speaker. Only frontal upright faces are considered in this paper, thus a single cascade classifier is powerful enough for face detection [6]. The search area for the mouth with the tree classifier is reduced to the lower region of the detected face. To accommodate scale variations, a multi-scale search is utilized within a constrained range estimated according to the face detection result.

In the tracking state, only the tree classifier is used to detect the mouth. A linear Kalman filter (LKF) is employed to predict the center of the search region in the next frame and correct the result in the current frame. The LKF addresses the general problem of estimating the state X of a discrete-time process that is governed by a linear stochastic difference equation

$$X_{k+1} = AX_k + w_k$$

with a measurement Z, that is

$$Z_k = HX_k + v_k$$

The random variables w_k and v_k are assumed to be independent of each other and have normal probability distributions. In this paper the Newton dynamics model similar to [11] is employed, i.e.,

$$X = \begin{pmatrix} x_c \\ y_c \\ \dot{x}_c \\ \dot{y}_c \\ \ddot{x}_c \\ \ddot{y}_c \end{pmatrix} \quad A = \begin{pmatrix} 1 & 0 & \Delta t & 0 & \Delta t/2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \Delta t/2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad Z = \begin{pmatrix} x_c \\ y_c \end{pmatrix} \quad H = (I \quad , \quad 0)^T ,$$

where $\Delta t = 0.04$ based on a frame rate of 25Hz. In practice the search region in the next frame t+1 is centered around (x_c, y_c) obtained from the *time update* with a width and height of 40% larger than the detected mouth at time t.

In the framework there is also a post-processing module to refine the trajectory of mouth in three phases: First a linear interpolation is employed to fill in the gaps in trajectory caused by detection failures. Then a median filter is used to eliminate incorrect detections under the assumption that outliers only occurs individually. At last a Gaussian filter is utilized to suppress the *jitter* in the trajectory. Fig.4. shows the effectiveness of the post-processing module.

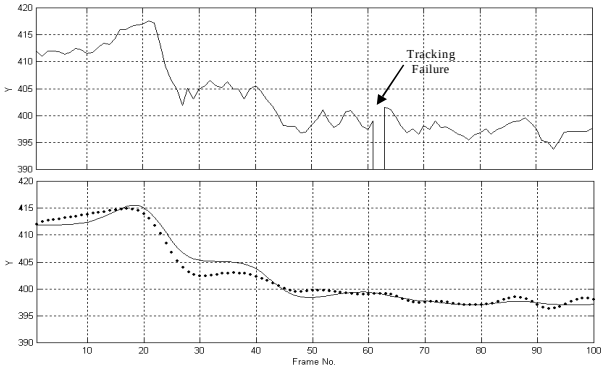

Fig.4: The Y positions of the tracked mouth in the first 100 frames of sequence 276_1_4to5 in the XM2FDB database (Top: before post-process; bottom: after post-process; the actual Y position is shown as a dotted line)

4. EXPERIMENTAL RESULTS

For training 1,050 mouth images were extracted from the sequences of the 'Client' subset of XM2FDB database [9]. These sample images were manually classified into 250 images of speakers with beard and 800 without beard. By randomly mirroring, rotating, and re-scaling these images 6,000 positive training samples of speakers with beard and 9,000 without beard were generated. Negative training examples were randomly extracted from a set of approximately 16,500 face-free and mouth-free images. Fig.5 shows some training samples of mouth regions without beard (top row), mouth regions with beard (middle row), and difficult non-mouth samples (bottom row).

Fig.5: Mouth and Non-mouth samples

In our experiments, three mouth tracking systems were built:

- System 1 was based on a cascade classifier (Fig.1(a)) with 18 stages trained on all positive mouth samples (15,000 in total) and 10,000 negative examples at each stage.

- System 2 was based on two specialized cascade classifiers with 17 stages (Fig.1(b)): one for mouth regions of speakers with beard and one for mouth regions of speakers without beard. For each classifier, all positive samples of the respective type plus 10,000 negative examples where used for training at each stage.
- System 3 was based on a tree classifier (Fig.1(c)) with 17 stages and 2 branches (split point at stage 3) and was trained with the same data set as used for system 1.

The three systems were tested on the "imposter" subset of the XM2FDB database [9] with 759 sequences recorded from 95 speakers (Fig.6.) using a Pentium 4 computer with 1.7GHz and 1GB RAM. Table 1 lists the accuracy and the average execution time per frame obtained by each system, together with the results obtained by the SVM based system [10]. Our results indicate that the tree classifier is superior to the cascade classifier with respect to accuracy, while having the shortest execution time of all four systems. Only the detection accuracy for multiple specialized cascade classifiers was slightly better but at a significantly higher computational cost (45% more demanding). In addition, compared with the SVM based system, the tree classifier based system is 66 and 15 times faster in detection and tracking, respectively, while preserving at least the same accuracy. Fig.6 shows some tracking results.

Fig.6: Some sample results of the tree classifier based system (top 2 lines: results of frame 5, 15, 25, 35, 45, 55 of sequence 313_1_4to5.avi; bottom 2 lines: results of different speakers)

Table.1: Experimental results on the "impostor" subset with 759 video sequences of 95 speakers (15 speakers with beard, 80 speakers without beard)

Type for Classifier	Correct	Correct Rate	Execution time / frame	
			Detection	Tracking
Single Cascade (1)	713	93.9%	38.0 ms	7.3 ms
Parallel cascades (2)	732	96.4 %	42.7 ms	9.4 ms
Detection Tree (3)	722	95.1%	33.8 ms	6.5 ms
SVMs[10]	699	92.1 %	2,232 ms	99 ms

5. CONCLUSION

This paper presented a novel detector tree of boosted classifiers together with a general framework for complex object detection and tracking in real-time. Dissimilar to the widely used cascade classifier, the tree classifier allows the stages of a cascade to split into several branches in order to deal with the potential diverse clusters of complex object patterns as they may occur, for example, in multi-view face or mouth detection. A clustering-and-splitting approach is embedded into the training algorithm to determine the split point and construct branches, which improve the discriminative power compared to a single cascade classifier and has lower computational cost than a single or multiple cascade classifier. An integrating general framework for object detection and tracking is also proposed, and a mouth tracking system is implemented and tested on the XM2FDB database. Experimental results show that the proposed algorithm is more than 15 times faster than our previous algorithm based on SVM [10], respectively, while having better tracking accuracy at the same time. At a better detection performance, it is also 12% faster than a single cascade classifier. Although the proposed approach has been validated only for human mouth tracking, it can be applied to other complex object tracking problems as well.

REFERENCES

[1] H. Rowley, S Baluja, T. Kanade, Neural network-based face detection, IEEE Trans. PAMI, 20(1): 23~38, 1998

[2] K. Sung, T. Poggio, Example-based Learning for view-based human face detection, IEEE Trans. PAMI, 20(1): 39~51

[3] E. Osuna, R. Freund, F. Girosi, Traing support vector machines: an application to face detection, In Proc. of CVPR, Puerto Rico, pp.130~136, 1997

[4] C. Papageorgiou, M. Oren, and T. Poggio, A general framework for object detection, International Conference on Computer Vision, Bombay, India, pp. 555~562, 1998

[5] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, IEEE CVPR, pp. 511~518, 2001

[6] R. Lienhart, J. Maydt, An extended set of Haar-like features for rapid objection detection, IEEE ICIP, pp.900~903, 2002

[7] Z. Zhang, L. Zhu, S. Li, et al., Real-time multi-view face detection, 5th International Conference on Automatic Face and Gesture Recognition, Washington, DC, USA, 2002

[8] Y. Freud and R. Schapire, A short introduction to Boosting, J. of Japanese Society for AI, 14(5): 71~780, 1999

[9] J. Luettin and G. Maitre, Evaluation protocol for the XM2FDB database, In IDIAP-COM 98-05, 1998

[10] L. Liang, X. Liu, Y. Zhao, et al, Speaker independent audio-visual continuous speech recognition, IEEE ICME, Lausanne, Switzerland, 2002

[11] M. D. Cordea, E. M. Petriu, N. D. Georganas, et al, Real-time 2(1/2)-D head pose recovery for model-based video-coding, IEEE Trans. on Instrumentation and Measurement, 50(4): pp. 1007~1013, 2001

[12] http://www.intel.com/research/mrl/research/opencv/