REDUCED SUPPORT VECTOR MACHINES APPLIED TO REAL-TIME FACE TRACKING

Benjamin Castañeda

Department of Computer Engineering
Rochester Institute of Technology
Rochester, NY
bcastane@ieee.org

Juan C. Cockburn

Department of Computer Engineering
Rochester Institute of Technology
Rochester, NY
jcck@ieee.org

ABSTRACT

This paper presents the implementation of a real-time face tracker to study the integration of Support Vector Machines (SVM) classifiers into a visual real-time tracking architecture. Face tracking has a large number of applications, especially in the fields of surveillance and human-computer interaction, which requires real-time performance. Even though SVM have previously been applied to face detection, their use in real-time applications is a challenge due to the computational cost implied in the SVM's evaluation stage. We address this problem by reducing the number of support vectors with almost no loss in accuracy of the classifier. Experiments showed that classification performed by the original SVM without reducing the number of support vectors took 42% of the total computation time of the face tracker and less than 2% after the reduction was performed.

1. INTRODUCTION

Human-Computer Interaction has received lately renewed interest due to advances in technology. Applications, which could have been considered fiction a decade ago, are now possible due to smaller, cheaper and more powerful computing devices. For example, smart rooms could have smart cameras in charge of tracking the user so his/her gestures and movements are evaluated as input commands. It is apparent that visual tracking should be a fundamental capability of the system, and since the interaction between the room and the user is required, this should be accomplished in real-time.

A common approach to tracking an object in a video stream is to use an object detector, a classifier and a motion estimator or tracker in sequential order. The object detector scans a frame from the video stream and selects the candidates to be analyzed by the classifier. The classifier evaluates every candidate assigning it a measure that indicates the likeliness of the candidate to be the object searched. The candidate with the best score is then locked and the tracker is used to follow it through the field of view. This standard tracking architecture has been successfully implemented in a variety of applications [1, 2].

The classifier plays an important role in the overall performance of the tracking system. In recent years, Support Vector Machines (SVM) have been a breakthrough in the machine learning community. They have been proved effective in a variety of tasks related to classification such as Character Recognition, Image Rotation Detection, Gesture Recognition, Face Recognition, Bioinformatics, and Spam Classification. However their use has been hampered by the complexity and computational time involved in the test (classification) stage.

The goal of this work is to analyze the integration of a SVM classifier into a visual real-time tracking architecture. For this purpose, a face-tracking system based on SVM is implemented. The computational time of the SVM test phase is improved by reducing the number of support vectors. Face tracking was selected among different applications because it represents a prototype visual tracking problem. A practical face tracking application should detect and track faces on complex backgrounds, be insensitive to head orientation, scale changes, illumination changes, partial occlusions and shadows [3], plus the additional real-time constraint. Since the objective of this work is focused on the use of SVMs as classifiers in a tracking architecture, the problem was limited to tracking faces of people looking into the camera at a known fixed distance from the camera, without occlusion and with controlled illumination.

The paper is divided as follows: Section 2 describes the architecture implemented for the face-tracking system. Section 3 presents the details on the SVM classifier used and the method to reduce the number of support vectors. Section 4 shows the experimental results of the time performance of the face-tracking architecture. Finally, Section 5 closes the paper with conclusions.

2. VISUAL TRACKING ARCHITECTURE

The face-tracking system is based on the architecture reported in [4]. The architecture consists of several tracking modules, one per each feature, and a Data Fusion Stage (see Figure 1). Each of the modules can be configured independently in order to take advantage of the characteristics of a specific feature. The data fusion algorithm exploits the relationships among different features to improve the overall (face) detection, tracking and reacquisition. A visual tracking module is composed of pre-processing, classification and motion estimation stages, and has two operational modes: Detection and Tracking. In detection mode, the pre-processing and classifier combination is used to obtain an initial position of the feature of interest with high confidence. Then the module is switched to tracking mode, where a motion estimator is used to track the feature detected by the classifier. Both modes of operation give as a result a list of vectors (f_i, s_i,x_i) containing a candidate identifier (f_i), a score indicating the likelihood of the candidate to be the actual feature (s_i) and its position (x_i).

The data fusion stage combines temporal and spatial information from different modules to determine the position of the tracked features and to decide whether the detection mode or the

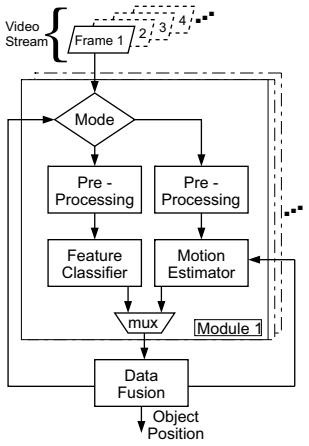

Fig. 1. Feature Based Tracking Architecture

tracking mode of each module should be used. The current and previous data from the list of vectors given by the modules is used to restrict the number of candidates to be classified, and to weight the candidates in order to have the most accurate classification and to choose the mode of operation (detection or tracking). Once the final position of the individual features has been determined, they are used to update the motion estimators from the different modules.

2.1. Face Tracking Implementation

The pre-processing stage consists of two steps. The first step uses a combination of skin color segmentation [5], density maps [6] and geometric filtering to extract face candidate regions from a frame. The second step extracts facial features candidates from each face candidate region. The facial feature candidates are obtained by using a combination of skin color information and the fact that facial features are of lower intensity than the rest of the face [2]. More details of the pre-processing stage can be found in [4].

A Kalman Filter was designed for motion estimation and combined with a template matching technique. For each feature, two Kalman Filters were used, one per coordinate axis. For example, each eye has its own pair of filters. The parameter selection for the Kalman Filter was based on the guidelines presented in [7]. The data fusion stage was designed on biometric heuristics based on the geometric relationships between eyes and lips. The focus of this work is on using SVM classifiers as feature detectors. In this particular case, the features are eyes and lips. Therefore, two SVMs were designed: one to recognize eyes, and another to recognize lips. In Figure 2, an example of eyes and lip detection is shown. The face region is highlighted by a bounding box and the features by rectangular boxes.

3. SVM CLASSIFIERS

SVM are used for eye and lip detection. SVM have been applied successfully in face detection [8, 1] and facial feature extraction [9]. The design of SVM essentially involves choosing a Kernel and designing a training set and a test set. Since the classifier needed to distinguish among facial features, we created a Facial Feature

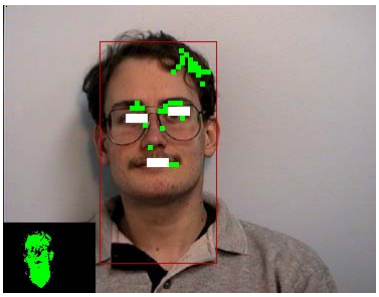

Fig. 2. Example of eyes and lip classification

Data set (FFDS). It consisted of 10×20 pixel images grouped in five classes: eyes, lips, eyebrows, nostrils and hair. Two programs were written to perform this task. The first one followed the algorithm described in the pre-processing stage to obtain facial feature candidates. For each facial feature candidate, a 10x20 image was extracted and classified by a human operator. The second program was designed to guarantee the integrity of the data sets; the already classified features grouped by class were shown to an operator who verifies the classification. 13 subjects from different ethnicity went through this process, in which they were asked to move while looking at a video camera. Only small movements such as yaw, roll and pitch of ± 5 degrees were allowed. The FFDS was then divided in a training set consisting of $15,308$ images and a test set with $5,347$ images.

Two classifiers, one for eyes and one for lips, were trained using the Matlab toolbox [10] based on Platt's SMO Algorithm. To choose the machine kernel and the cost of misclassification parameter (C), experiments were performed with linear, polynomial and radial basis function kernels using a smaller training set ($5,000$ samples) and varying C from 0.01 to 100. From an experimental performance and computational complexity study a polynomial kernel of second degree and $C = 0.01$ were selected. The training process gave as a result 2415 support vectors for eyes and 2261 for lips.

3.1. Reduced Support Vectors

Reducing the number of support vectors is an approximation problem that can be posed as a minimization problem as follows: The decision boundary of the original SVM is defined implicitly by:

$$\Psi = \sum_{i=1}^{N_s} w_i \Phi(x_i) \tag{1}$$

where $\Psi \in \mathcal{F}$ is characterized by the support vectors $x_i \in \mathcal{X}$, $w_i \in \mathbb{R}$ and $\Phi(x) \in \mathcal{F}$ is the non-linear mapping implicitly defined by the kernel of choice.

Similarly, an approximation to that decision boundary can be written as:

$$\Psi' = \sum_{i=1}^{N_z} \beta_i \Phi(z_i) \tag{2}$$

with $N_z \ll N_s$, $\beta_i \in \mathbb{R}$, and the reduced set of vectors $z_i \in \mathcal{X}$.

Then, the reduced support vector problem can be posed as:

$$\min_{\Psi'} \|\Psi - \Psi'\|^2 \tag{3}$$

where $\|\cdot\|$ denotes the Euclidian Norm.

To accomplish this task, we followed the work presented in [11] and extended it to be applicable to polynomial kernels. First, we consider the case approximating Ψ with one vector z. Then (3) will transform into:

$$\min_z \| \frac{(\Psi \cdot \Phi(z))}{\Phi(z) \cdot \Phi(z)} \Phi(z) - \Psi \|^2 = \min_z \|\Psi\|^2 - \frac{(\Psi \cdot \Phi(z))^2}{(\Phi(z) \cdot \Phi(z))} \quad (4)$$

which is equivalent to:

$$\max_z \frac{(\Psi \cdot \Phi(z))^2}{(\Phi(z) \cdot \Phi(z))} \quad (5)$$

For the case of polynomial kernels, the following iterative equation was deduced to find z:

$$\begin{aligned} z &= \frac{(z^T z + d)(\sum_{i=1}^{N_s} w_i (x_i^T z + d)^{p-1} x_i)}{(\sum_{i=1}^{N_s} w_i (x_i^T z + d)^p)} \\ z_{n+1} &= \frac{(z_n^T z_n + d)(\sum_{i=1}^{N_s} w_i (x_i^T z_n + d)^{p-1} x_i)}{(\sum_{i=1}^{N_s} w_i (x_i^T z_n + d)^p)} \end{aligned} \quad (6)$$

Once z is found through (5), the associated weight β is computed by:

$$\beta = \frac{(\Psi \cdot \Phi(z))}{(\Phi(z) \cdot \Phi(z))} \quad (7)$$

In order to calculate higher order reduced set vectors z_m, $m > 1$, Ψ_m is introduced as the decision boundary to minimize in every iteration of the algorithm. For the first one, Ψ_m is initialized as the original decision boundary Ψ. In the next iterations, Ψ_m is computed as:

$$\begin{aligned} \Psi_m &= \Psi - \sum_{i=1}^{m-1} \beta_i \Phi(z_i) \\ \Psi_m &= \sum_{i=1}^{N_s} w_i \Phi(x_i) - \sum_{i=1}^{m-1} \beta_i \Phi(z_i) \\ \Psi_m &= \sum_{i=1}^{N_x} \omega_i \Phi(\chi_i) \end{aligned} \quad (8)$$

where m is the iteration number, $N_x = N_s + m - 1$, $(\omega_1, \ldots, \omega_{Nx}) = (w_1, \ldots, w_{N_s}, -\beta_1, \ldots, -\beta_{m-1})$, and $(\chi_1, \ldots, \chi_{Nx}) = (x_1, \ldots, x_{N_s}, z_1, \ldots, z_{m-1})$.

The procedure will end when $\|\Psi - \Psi'\|^2$ is less than a predefined tolerance ρ, or $m = N_z$.

4. RESULTS

Three people were asked to participate in the experiments with the face tracking application. Five videos per person were timed and analyzed. Each of the videos lasted approximately 2 minutes yielding information for 50000 frames. Table 1 shows the maximum, minimum and average time in milliseconds to perform a classification with the support vectors resulting from the SMO training. The average time to classify a single feature candidate is also presented. Both classifiers, for eyes and for lips, need more than one millisecond per candidate. Their average classification time in both cases exceeds 33.3 milliseconds, which means these classifiers would not be able to achieve real-time performance by themselves. Table 2 summarizes the time performance of the support vector machines classifiers in combination with Kalman filters and template matching. The classifier is used to detect the initial position of the feature. This initial step may introduce a noticeable delay in the video stream, which is then recovered in the subsequent frames where template matching is used. The average time per frame for this latter step is 4.1 milliseconds. Therefore, the average process time per frame is reduced to less than 8 milliseconds in both features.

	#SV	Min(ms)	Max(ms)	Avg(ms)	Avg Time/point(ms)
Eyes	2415	45.58	167.00	102.28	1.21
Lips	2261	34.09	197.74	72.73	1.13

Table 1. SVM Classification Time

	#SV	Avg(ms)	Avg fr on track
Eyes	2415	7.54	26.76
Lips	2261	6.46	26.98

Table 2. SVM and Kalman Filters Tracking Time

The feature-based architecture includes a latter step of data fusion, which basically combines the information of all the features to improve reacquisition when the track of one of them is lost. This final stage helps to reduce the number of candidate features to be evaluated when reacquiring one feature, improving time performance. Figure 3 presents the average distribution of the computation in the feature-based architecture based on the two-minute videos. Even though, support vector machines are used on average once every 20 frames, the complexity of evaluating them takes 42% of the computation time. The computational time spent in the motion estimation step is similar (46%), however the motion estimator subroutine is called 20 times more often than the support vector machines and its computation is divided among 20 frames in average while support vector machines concentrate in only one frame. The pre-processing stage and the data fusion stages combined take 12% of the computation. Reducing the computational time spent in SVM classification would clearly improve the overall time performance of the tracking architecture.

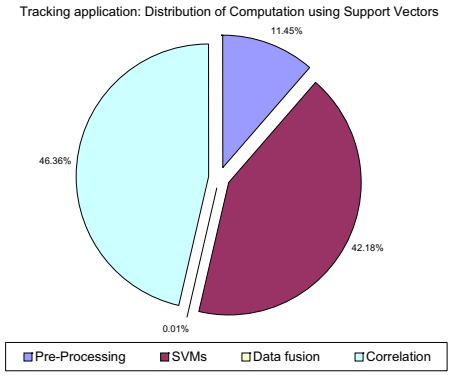

Fig. 3. Distribution of Computation

After applying the support vector reduction, results show that with far less number of vectors, the accuracy in test and training sets are similar to the accuracy of the original SVM (see Table 3 below).

	# SV	Accuracy Training Set	Accuracy Test Set
Eyes	2415	98.8	91.1
Lips	2261	98.7	91.1
Eyes	40	98.7	90.4
Lips	37	98.3	89.5

Table 3. Results of reducing the number of Support Vectors

Table 4 shows the improvement in time performance when the reduced support vectors are used. These results even suggest that SVM classifiers could be used in every frame and real time performance could still be achieved. Table 5 shows the results of integrating the classifiers with Kalman filters and template matching. The average time per feature is now less than 4 milliseconds. Comparing with the results showed in Table 2, the average time per frame has been lowered by half in each feature. With the

	#SV	Min(ms)	Max(ms)	Avg(ms)	Avg Time/point(ms)
Eyes	40	0.9139	3.3485	2.0509	0.0242
Lips	37	0.6850	3.9729	1.4613	0.0228

Table 4. Reduced Support Vectors Classification Time

	#SV	Avg(ms)	Avg fr on track
Eyes	40	3.9241	24.6875
Lips	37	3.8981	23.9143

Table 5. RSV and Kalman Filters

introduction of the reduced support vectors into the feature-based architecture, the distribution of the computation changes significantly (see Figure 4). The computational time spent in the classifiers is reduced from 42% to 1.4%.

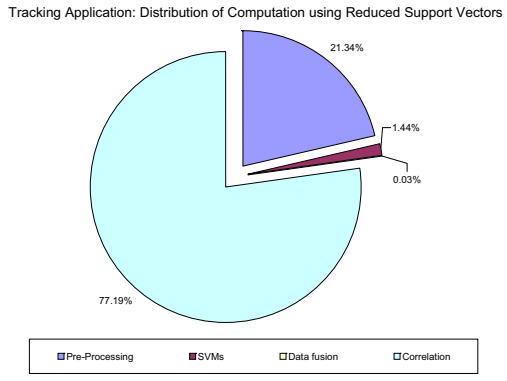

Fig. 4. Distribution of Computation using Reduced Support Vectors

5. CONCLUSION

The face tracking application implemented in this work represents a prototype of the use of Support Vector Machines as classifiers in tracking applications. It also illustrates the impact of reducing the number of support vectors in the overall time performance of the tracking system. This application clearly shows that the use of Support Vector Machines in tracking applications is feasible and that it can be done in real-time. In this particular application, the classification performed by the original SVM without reducing the number of support vectors took 42% of the total computation in the tracking architecture. This percentage was lowered to less than 2% by reducing the number of support vectors involved in the classification process. The number of support vectors was reduced by 98% of the original one. Due to the reduction in the computational cost of the classifier, the detection and tracking of the three features (2 eyes and the lips) can be performed at every frame. Reducing the computational cost of SVM classification gives additional time for other tasks. This is critical if an application runs in real-time. This additional time can be allocated to the pre-processing, motion estimation or data fusion stages.

6. REFERENCES

[1] V. Kumar and T. Poggio, "Learning-based approach to realtime tracking analysis of faces," *Technical Report 1672 - MIT*, 1998.

[2] J. Sobottka and I. Pittas, "Segmentation and tracking of faces in color images," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 236–241.

[3] M. Porta, "Vision-based user interfaces: methods and applications," *Int. Journal on Human-Computer Studies*, vol. 57, pp. 27–73, 2002.

[4] B. Casta neda, Y. Luzanov, and J.C. Cockburn, "A modular architecture for real-time feature-based tracking," in *Proc. 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Quebec, Canada, May 2004.

[5] E. Saber and A. M. Tekalp, "Frontal-view face detection and facial feature extraction using color, shape and symmetry based cost-functions," *Pattern Recognition Letter*, , no. 19, pp. 669–680, 1998.

[6] V. Kravtchenko, "Tracking color objects in real time," M.S. thesis, The University of British Columbia, 1999.

[7] M.R.J. Kohler, "System architecture and techniques for gesture recognition in unconstraint environments," in *International Conference on Virtual Systems and Multimedia*, Switzerland, 1997.

[8] S. Romdhani, B. Schölkopf P. Torr, and A. Blake, "Computationally efficient face detection," in *8th International Conference on Computer Vision*, 2001, vol. 2, pp. 695–700.

[9] B. Heisele, P. Ho, J. Wu, and T. Poggio, "Face recognition: component-based versus global approaches," *Computer Vision and Image Understanding*, vol. 91, pp. 6–21, 2003.

[10] G. C. Cawley, "MATLAB support vector machine toolbox (v0.50β) [http://theoval.sys.uea.ac.uk/˜gcc/svm/toolbox]," University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K. NR4 7TJ, 2000.

[11] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K-R. Müller, G. Rätsch, and A.J. Smola, "Input space vs. feature space in kernel-based methods," in *IEEE Transactions on Neural Networks*, 1999.