Face Tracking Using Adaptive Appearance Models and Convolutional Neural Network

Boguslaw Rymut and Bogdan Kwolek

Rzeszów University of Technology
W. Pola 2, 35-959 Rzeszów, Poland
http://www.prz.edu.pl

Abstract. One inherent problem of online learning based trackers is drift consisting in a gradual accommodation of the tracker to non-targets. This paper proposes an algorithm that does not suffer from the template drift inherent in a naive implementation of the online appearance models. The tracking is done via particle swarm optimization algorithm built on adaptive appearance models. The convolutional neural network based face detections are employed to support the re-diversification of the swarm in the course of the tracking. Such candidate solutions vote simultaneously towards true location of the face through correcting the fitness function. In particular, the hybrid algorithm has better recovery capability in case of tracking failure.

Keywords: Swarm intelligence, Hybrid intelligent algorithms

1 Introduction

Face tracking is an important problem for various applications, like video surveillance, user friendly interfaces, emotion recognition, biometrics and face recognition. It is an active area of the research because of the lack of a satisfactory tracking system that can deal with intrinsic and extrinsic distortions. The face can be tracked as single entity or as several individual facial features that were selected in advance [1]. Tao et al. [2] present a system, which tracks the location of the face and facial features using a probability network encoding the spatial relationships between facial features. A method proposed in [3] uses shape constrained search technique in combination with a set of feature templates, which are updated using a nearest neighbor approach. Some methods combine several techniques in a single tracking system. An example of such an approach is in [4], where fast motion-based face tracking and neural network based face detection are combined to achieve better tracking. In [5], Haar cascade based face detection is used to correct the proposal distribution for the particle filter.

Online appearance models were successfully used for facial features tracking [6] on account of their strong capability to adapt to variations of the appearance. However, the discussed algorithm like other template-based algorithms is not drift free. Template drift is a common phenomenon in which the target gradually shifts away from the template and the template is occupied progressively step by step by background objects due to template update by background pixels. Thus, if something happens wrong in some number of consecutive frames, for example due to complete occlusion of the target or very fast motion, the tracking can be lost and the algorithm should be re-initialized in order to continue the tracking of the target. On the other hand, because the observations are noisy, the error grows gradually over time and this in turn leads to accumulation of the drift, which can lead to difficulties in any long term tracking.

Several remedies were proposed to ameliorate the above mentioned difficulties. An algorithm proposed in [7] corrects template drift by the use of robust weights that are based on evidence, accumulated over many frames. Although such an algorithm is able to cope with template drift, it can still fail due to fast motion of the face or sudden appearance changes. In order to cope with drift accumulation as well as fast motion of the face we propose an algorithm that combines face tracking and face detection. The face finder permits face re-detection as well as reduces the accumulation of the drift over time. The tracker and detector are combined elegantly using swarm intelligence. Moreover, face detections are used to support the re-diversification of the particle swarm in the course of the tracking. Such candidate solutions vote simultaneously towards real location of the face through correcting the fitness function. Thanks to synergistic combination of different techniques [8], the hybrid algorithm is better. In particular, it has better recovery capability in case of tracking failure.

Convolutional neural network [9][10] integrates feature extraction and classification into single structure. It extracts two-dimensional features at increasing scales, and by comparison to relevant techniques it is relatively tolerant to local geometric distortions in the image. Thanks to such generalization capability it is used in our face tracking algorithm. The face tracker is based on adaptive appearance models and particle swarm optimization [11].

2 Visual appearance modeling using adaptive models

Our intensity-based appearance model consists of three components, namely, the W-component expressing the two-frame variations, the S-component characterizing the stable structure within all previous observations and F-component representing a fixed initial template. The model $A_t = \{W_t, S_t, F_t\}$ represents the appearances existing in all observations up to time $t-1$. It is a mixture of Gaussians [6] with centers $\{\mu_t^{(l)}\}_{l=w,s,f}$, their corresponding variances $\{(\sigma_t^{(l)})^2\}_{l=w,s,f}$ and mixing probabilities $\{m_t^{(l)}\}_{l=w,s,f}$.

Let $I(z,t)$ denote the brightness value at the position $z=(x,y)$ in an image $\mathcal{I}$ that was acquired at time t. Let $\mathcal{R}$ be a set of J locations $\{z(j)\}_{j=1}^{J}$ defining a template. $Y_t(\mathcal{R})$ is a vector of the brightness values at locations $z(j)$ in the template. The fitness score has the following form:

$$f(x_t) = \prod_{j=1}^{J} \sum_{l=w,s,f} \frac{m_t^{(l)}(j)}{\sqrt{2\pi(\sigma_t^{(l)}(j))^2}} \exp\left[-\frac{1}{2}\left(\frac{Y_t(j)-\mu_t^{(l)}(j)}{\sigma_t^{(l)}(j)}\right)^2\right]. \quad (1)$$

In the objective function we utilize a recursively updated appearance model, which depicts stable structures seen so far, two-frame variations as well as initial object appearance. The update of the current appearance model A_t to A_{t+1} is done using the Expectation Maximization (EM) algorithm [12][11].

3 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based algorithm that exploits a set of particles representing potential solutions of the optimization task [13]. The particles fly through the n-dimensional problem space with a velocity subject to both stochastic and deterministic update rules. The algorithm seeks for the global best solution through adjusting at each time step the location of each individual according to personal best and the global best positions of particles in the entire swarm. Each particle i keeps the position $p^{(i)}$ in the problem space, which is associated with the best fitness it has achieved personally so far. Additionally, when a particle considers all the population as its topological neighbors, each particle employs g location, which has been obtained so far by any particle in the swarm. The new positions are subsequently scored by a fitness function f. The velocity of each particle i is updated in accordance with the following equation:

$$v^{(i)} \leftarrow wv^{(i)} + c_1 r_1 (p^{(i)} - x^{(i)}) + c_2 r_2 (g - x^{(i)}) \tag{2}$$

where $v^{(i)}$ is the velocity of the ith particle, c_1, c_2 denote the acceleration coefficients, r_1 and r_2 are uniquely generated random numbers in the interval [0.0, 1.0]. The new position of a particle is calculated in the following manner:

$$x^{(i)} \leftarrow x^{(i)} + v^{(i)} \ . \tag{3}$$

The local best position of each particle is updated as follows:

$$p^{(i)} \leftarrow \begin{cases} x^{(i)}, \text{ if } f(x^{(i)}) < f(p^{(i)}) \\ p^{(i)}, \qquad \text{otherwise} \end{cases} \tag{4}$$

and the global best position g is defined as:

$$g \leftarrow \arg\min_{p^{(i)}} \{f(p^{(i)})\} \ . \tag{5}$$

The value of velocity $v^{(i)}$ should be restricted to the range $[-v_{max}, v_{max}]$ to prevent particles from moving out of the search range.

At the beginning of the optimization the PSO initializes randomly the locations as well as the velocities of the particles. Then the algorithm selects $p^{(i)}$ and g values. Afterwards, equations (2)-(5) are called until maximum iterations or minimum error criteria is attained. After that, given $\hat{x}_t = g$ the algorithm calculates $\hat{Y}_t$. Then the algorithm uses it to update the model.

In the simplest solution the object tracking can be realized as deterministic searching of window location, whose content best matches a reference window content. PSO allows us to avoid such time consuming exhaustive searching for the best match. It provides optimal or sub-optimal match without the complete knowledge of the searching space. In PSO based tracking, at the beginning of each frame, an initial position is assigned to each particle:

$$x_t^{(i)} \leftarrow \mathcal{N}(g_{t-1}, \Sigma) \tag{6}$$

given the location g_{t-1} that has been estimated in the previous frame $t-1$.

4 Convolutional Neural Networks

A convolutional neural network [9] is a special kind of a feedforward neural network. It incorporates prior knowledge about the input signal and its distortions into its architecture. Convolutional neural networks (CNNs) are specifically designed to cope with the variability of 2D shapes to be recognized. They combine local feature fields and shared weights as well as utilize spatial subsampling to ensure some level of shift, scale and deformation invariance. Using the local receptive fields the neurons can extract simple visual features such as corners, end-points. These elementary features are then linked by the succeeding layers to detect more complicated features.

A typical CNN contains a set of layers each of which consists of one or more planes. Each unit in the plane is connected to a local neighborhood in the previous layer. The unit can be seen as a local feature detector whose activation characteristic is determined in the learning stage. The outputs of such a set of units constitute a feature map. The units in a feature map are constrained to perform the same operation on different parts of the input image or previous feature maps, extracting different features from the same image. A feature map can be obtained in a sequential manner through scanning the input image by a single unit with weights forming a local receptive field and storing the outputs of this unit in corresponding locations in the feature map. This operation is equivalent to a convolution with a small kernel. The feature map can be treated as a plane of units that share weights. The subsampling layers, see Fig. 1, which usually follow layers with local, convolutional feature maps introduce a certain level of invariance to distortions and translations. Features of decreasing spatial resolution and of increasing complexity as well as globality are detected by the units in the successive layers.

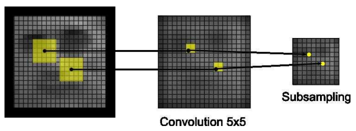

Fig. 1. Convolution and subsampling process in convolutional neural networks

The convolutional neural network we use consists of 7 layers, see Fig. 2. Layer C1 performs a convolution on gray images. The weights in the convolution mask are shared by all the neurons of the same feature map. The receptive fields of neighboring units overlap. The size of the scanning windows was chosen to be 24x24 pixels. The size of the mask is 5x5 and the size of the feature map of this layer is 20×20. The layer has 156 trainable parameters. Layer S1 is the averaging/subsampling layer. It consists of 6 planes of size 20 by 20. Each unit in one of these planes receives four inputs from the corresponding plane in C1. Receptive fields do not overlap and all the weights are equal within a single unit. Therefore, this layer performs a local averaging and 2 to 1 subsampling. The number of trainable parameters utilized in this layer is 12. Once a feature has been extracted through the first two layers its accurate location in the image is less substantial and spatial relations with other features are more relevant. Therefore layers S1 and C2 are partially connected, and the task of such a configuration is to discover the relationships between different features.

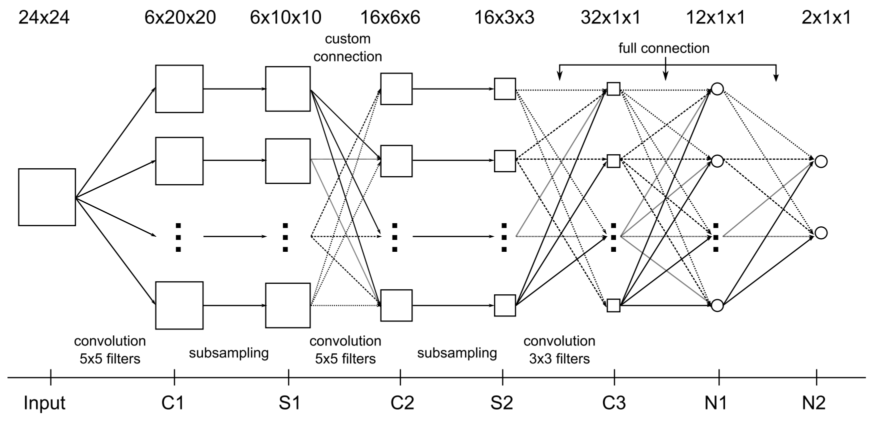

Fig. 2. Convolutional neural network for face detection

Layer C2 is composed of 16 feature maps. Each unit contains one or two receptive fields of size 5×5, which operate at identical positions within each S1 maps. The first eight feature maps use single receptive fields. They form two independent groups of units responsible for distinguishing between face and non-face patterns. The remaining eight feature maps take inputs from every contiguous subsets of two feature maps in S1. This layer has 416 free parameters. Layer S2 plays the same role as the layer S1. It is constructed of 16 feature maps and has 32 free parameters. The next layer C3 is the convolution layer, which consists of 32 neurons and has 320 free parameters. The next fully connected layer consists of 12 neurons and has 396 trainable parameters. Finally, the output layer has two nodes that is fully connected to the all the nodes from the previous layer. The network contains many connections but relatively few free trained parameters. Weight sharing allows us to reduce considerably the number of free parameters and improves the generalization capability.

The face detector was trained on 3000 non-face patches collected from about 1500 images and 1500 frontal faces covering out-of-plane rotation in the range $-20^\circ, \ldots, 20^\circ$. All faces were manually aligned by eyes position. For each face example the synthesized faces were randomly generated by in-plane rotation in the range $-10^\circ, \ldots, 10^\circ$, random scaling about $\pm 10\%$, random shifting up to ± 1 pixel and mirroring. All faces were then cropped and re-scaled to windows of size 20×20 pixels while preserving their aspect ratio. The training collection contains also images acquired from our video cameras. To provide more false examples we utilized a training with bootstrapping [14]. By using bootstrapping we iteratively gathered examples, which were close to the boundaries of face and non-face clusters in the early stages of training. The activation function in the network was a hyperbolic tangent.

5 The Algorithm

In the particle swarm optimization algorithm each particle in the population represents a candidate solution to the optimization problem. Much of the success of PSO algorithms arises from the tendency of the individual particles to deviate from the best known position in any given iteration, enabling them to neglect local optima, while the swarm as a whole gravitates towards the global extremum. In a dynamic optimization problem the aim is not only to seek the extrema, but also to follow their progression through the space as closely as possible. Since the face tracking is a dynamic optimization problem, the tracking can be accomplished by means of incorporating the temporal continuity information into the conventional PSO algorithm. In consequence, the tracking can be achieved by a sequence of static PSO-based optimizations to seek the best object location, followed by re-diversification of the particles using (6) to cover the possible object state in the next time step.

Our tracking algorithm estimates the location of the face and the size of the template. In the second image in upper row of Fig. 3 we can see the locations of the particles after re-diversification. As we can see in this illustrative example, despite covering by the swarm almost the whole face, the tracker lost the target, see last image in the upper row of Fig. 3.

In order to reduce the computation overhead of the face detection algorithm we extract skin like patches, see 1st image in the bottom row of Fig. 3. Through the histogram back-projection we calculate the skin probability images, which are then thresholded to extract the skin like patches. The 2D histogram of the skin color is constructed in rg color space and consists of 8×8 bins. The skin patches are then refined using morphological closeing. Afterwards, the connected components labeling takes place. Using the extracted connected components we calculate areas and then remove small components. The neural network is then executed only on such refined patches, which likely contain faces. Encountered detections are passed to a non-maxima suppression algorithm, which removes possible multiple detections of the same face.

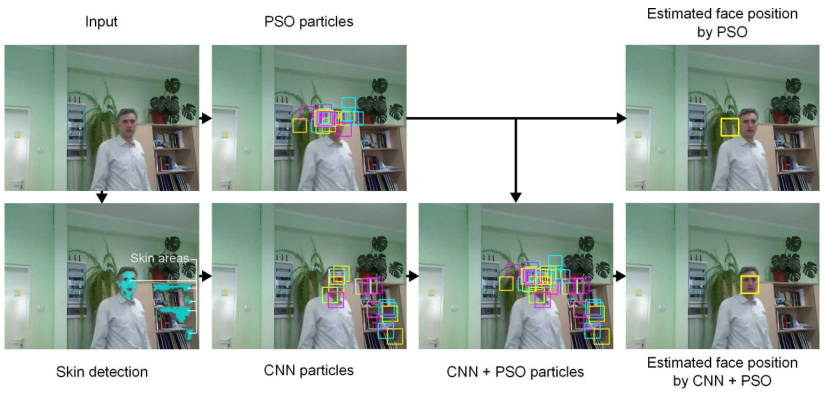

Fig. 3. The main stages of image processing in the hybrid algorithm

At the beginning of each frame, using the probabilities generated by softmax function $h(\tilde{x})$, where $\tilde{x}$ denotes the location determined by the face detector, we select 32 best face candidates. Given the estimate of the face location in the previous frame we perform re-diversification using locations generated by the face detector, see 2nd line in the pseudo-code as well as 3rd image in bottom row of Fig. 3, and locations obtained by (6), see 3rd line in the pseudo-code. In line #10, the algorithm selects the nearest face candidate for the currently considered particle i as well as a face candidate that is closest to g_t. The term $[1-\mathcal{N}(x_t^{(i,k+1)}-\tilde{x}_t^{(j^*)},\Sigma_d)h(\tilde{x}_t^{(j^*)})]$, see code in line #11, allows the algorithm to prefer the particles, which are situated near the detected faces. In turn, the term $[1-\mathcal{N}(x_t^{(i,k+1)}-\tilde{x}_t^{(l^*)},\Sigma_d)h(\tilde{x}_t^{(l^*)})]$ allows the algorithm to prefer such a global particle, which has in the proximity a detected face. The w_1 and w_2 are

```
1.  initialize $v_t^{(i,0)}$, detect faces, determine $\{\tilde{x}_t^{(i)}\}_{i=1}^{32}$
2.  $\{x_t^{(i,0)}=\tilde{x}_t^{(i)}, \hat{f}_t^{(i)}=(1-h(\tilde{x}_t^{(i)})\}_{i=1}^{32}$
3.  $\{x_t^{(i,0)}\sim\mathcal{N}(g_{t-1},\Sigma_0), \hat{f}_t^{(i)}=f(x_t^{(i,0)})\}_{i=33}^{N}$
4.  $\{p_t^{(i)}=x_t^{(i,0)}\}_{i=1}^{N}$
5.  $i^*=\arg\min_{(i)}\hat{f}_t^{(i)}, \quad g_t=p_t^{(i^*)}, \quad \hat{f}_t^{(g)}=\hat{f}_t^{(i^*)}$
6.  For $k=0,1,\ldots,K$
7.    For each particle $i$
8.      $v_t^{(i,k+1)}=wv_t^{(i,k)}+c_1r_1(p_t^{(i)}-x_t^{(i,k)})+c_2r_2(g_t-x_t^{(i,k)})$
9.      $x_t^{(i,k+1)}=x_t^{(i,k)}+v_t^{(i,k+1)}$
10.     $j^*=\arg\min_j d(x_t^{(i,k+1)},\tilde{x}_t^{(j)}), \quad l^*=\arg\min_l d(g_t,\tilde{x}_t^{(l)})$
11.     $f_c^{(i,k)}=w_1f(x_t^{(i,k+1)})+w_2^{(k)}\left[1-\mathcal{N}(x_t^{(i,k+1)}-\tilde{x}_t^{(j^*)},\Sigma_d)h(\tilde{x}_t^{(j^*)})\right]+$
            $(1-w_1-w_2^{(k)})\left[1-\mathcal{N}(x_t^{(i,k+1)}-\tilde{x}_t^{(l^*)},\Sigma_d)h(\tilde{x}_t^{(l^*)})\right]$
12.     If $f_c^{(i,k)}<\hat{f}_t^{(i)}$ then $p_t^{(i)}=x_t^{(i,k+1)}, \quad \hat{f}_t^{(i)}=f_c^{(i,k)}$
13.     If $\hat{f}_t^{(i)}<\hat{f}_t^{(g)}$ then $g_t=p_t^{(i)}, \quad \hat{f}_t^{(g)}=\hat{f}_t^{(i)}$
```

weighting factors. The weighting factor w_2 depends on k and is used to balance the influence of the discussed terms. This way, the tracker and the detector are combined within the particle swarm optimization algorithm.

6 Experimental Results

The algorithm has been tested on real images that were acquired by a laptop built in camera. In Fig. 4 we can see some selected images that were utilized in the experiments. In the upper row we can see the results that were generated by adaptive appearance based tracker. The tracking was done on images acquired at rate 24 fps. In the images from middle row of Fig. 4 we can see the results, which were generated by the same algorithm, but using every third image of the input sequence. As we can observe, due to faster motion of the person in such an image sequence, the algorithm fails in frame #240. The tracking recovered in frame #360, see Fig. 4, and the algorithm continued successfully the tracking of the person. In the images from bottom row of the Fig. 4 we can see the experimental results, which were obtained by our hybrid algorithm. As we can observe, the tracker is capable of tracking the person despite fast motion as well as wooden bookshelf in the background.

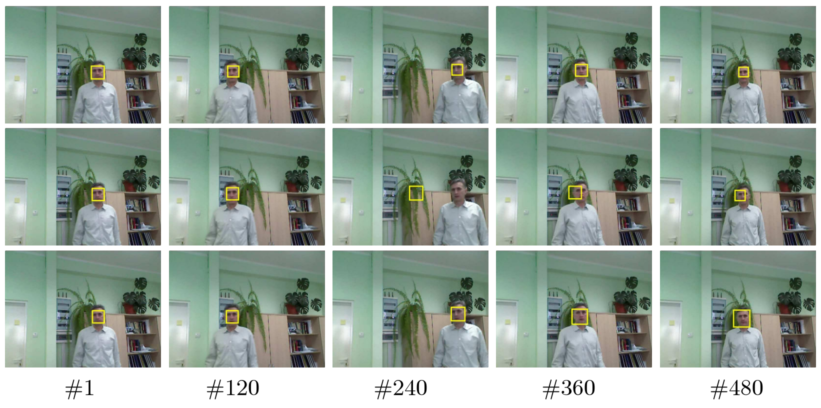

Fig. 4. Face tracking by adaptive appearance models (images in upper row), by adaptive appearance models on every third frame of the input sequence (middle row), and by hybrid algorithm on every 3rd frame (bottom row)

The demonstrated above results were obtained by PSO consisting of $N = 128$ particles in $K = 5$ iterations. The weighting coefficient w_1 has been set to 0.6, and $w_2^{(k)}$ assumed the values $0.4, 0.3, \ldots, 0.0$. The average error of the template location on the sequence depicted in Fig.4 (on every third frame, until the tracking is lost) is equal to 11 pixels. Such error arose due to fast motion of the person as well as noisy images acquired by a laptop camera. The average error of the hybrid algorithm is about 5 pixels. Future work will concentrate on implementation of the complete algorithm on GPU. We intend to use our GPU implementation of the adaptive appearance based tracker [11].

7 Conclusions

This paper proposes a hybrid algorithm consisting of face tracking and face detection modules. In the first module we employ particle swarm optimization and the adaptive appearance models. In order to increase the resistance of the tracking module to the drift we employ in the second module the convolutional neural network. The face detections support the re-diversification of the swarm in the course of the tracking. Such candidate solutions vote also simultaneously towards true location of the face through correcting the fitness function. In particular, the hybrid algorithm has better recovery capability in case of unsuccessful tracking.

References

1. Karlsson, S., Taj, M., Cavallaro, A.: Detection and tracking of humans and faces. J. Image Video Process, 2008, pp. 1–9 (2008)
2. Tao, H., Lopez, R., Huang, T.: Tracking facial features using probabilistic network. In: IEEE Int. Conf. on Aut. Face and Gesture Recognition, pp. 166–170 (1999)
3. Cristinacce, D., Cootes, T.F.: Facial feature detection and tracking with automatic template selection. In: Proc. of the 7th Int. Conf. on Automatic Face and Gesture Recognition, pp. 429–434 (2006)
4. McKenna, S., Gong, S.: Tracking faces. In: Int. Conf. on Automated Face and Gesture Recognition, pp. 271–276 (1996)
5. Kwolek, B.: Face tracking using color, elliptical shape features and a detection cascade of boosted classifiers in particle filter. In: Int. Conf. on Computer Vision and Graphics. Computational Vision and Graphics 32, Springer, pp. 287–292 (2004)
6. Jepson, A.D., Fleet, D.J., El-Maraghi, T.: Robust on-line appearance models for visual tracking. IEEE Trans. on PAMI, vol. 25, pp. 1296–1311 (2003)
7. Schreiber, D.: Robust template tracking with drift correction. Pattern Recognition Letters, vol. 28, pp. 1483–1491 (2007)
8. Corchado, E., Abraham, A., de Carvalho, A.: Hybrid intelligent algorithms and applications. Information Sciences, vol. 180, pp. 2633–2634 (2010)
9. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech, and time-series. In Arbib, M., ed.: The Handbook of Brain Theory and Neural Networks. MIT Press (1995)
10. Garcia, C., Delakis, M.: Convolutional face finder: A neural architecture for fast and robust face detection. IEEE Tran. on PAMI, vol. 26, pp. 1408–1423 (2004)
11. Rymut, B., Kwolek, B.: GPU-supported object tracking using adaptive appearance models and particle swarm optimization. In: Int. Conf. on Computer Vision and Graphics. LNCS, vol. 6375, pp. II:227-234 (2010)
12. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. of the Royal Statistical Society. Ser. B 39, pp. 1–38 (1977)
13. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of IEEE Int. Conf. on Neural Networks, IEEE Press, Piscataway, NJ, pp. 1942–1948 (1995)
14. Rowley, H.A., Baluja, S., Kanade, T.: Neural network-based face detection. IEEE Tran. on Pattern Analysis and Machine Intelligence, vol. 20, pp. 23–38 (1998)