REAL-TIME AND MULTI-VIEW FACE TRACKING ON MOBILE PLATFORM

Lei Xu, Jiangwei Li and Kongqiao Wang
Nokia Research Center

ABSTRACT
The development of mobile platform has raised an emergent requirement for face-related multimedia applications. However, as the basis of such applications, face detection and tracking still suffers from large facial pose variation and limited computation resource. In this paper, a multi-view and real-time face tracking system is presented for mobile platform. First, three basic detectors are trained by local binary pattern (LBP) and Boosting algorithm. These detectors are then flexibly expanded for multi-view face detection through the rotation facility of LBP. To further accelerate the face tracking process, a robust facial pose estimation algorithm and the face matrix partition scheme is proposed. The experimental results on handset show the superior performance of the proposed method.

Index Terms— multi-view, real-time, mobile platform

1. INTRODUCTION
Face-based technology has shown its promising perspective and attracted lively interest from both scientific and industrial community. More recently, increasing attention is given to how to utilize facial information to enrich mobile multimedia applications. The potential applications include: 1) eye gaze or facial expression can be extracted for real-time interaction, such as game; 2) the facial key points in video call can be localized, transmitted and reconstructed (e.g. as a cartoon avatar) instead of real video stream, such that the required bandwidth can be dramatically reduced; 3) face recognition can be used for identity verification and automatic photo clustering or tagging.

As the foundation of the above-mentioned applications, face detection and face tracking has become relatively mature for PC platform and controllable condition. However, many difficulties must be taken into account for mobile environment. On the one hand, large pose variation (including pitch, yaw and roll) should be effectively managed, because usually the camera position is not fixed and users might freely hold the handset without much restriction. On the other hand, most handsets are only equipped with very limited computation resources (low CPU frequency, and lack of arithmetic units of floating point operations). Therefore, applicable mobile face tracking system must have two main characteristics: multi-view and real-time.

Multiple facial views will inevitably cause large within-class variation for the face category, which will then make it rather difficult to train a single face detector to cover all the views. To solve this problem, many researchers have adopted the view partition technique [4, 8, 3, 2, 9]. The basic idea is to divide the whole facial view set into a series of subsets with compact distribution, and train a high-performance detector for each subset.

Despite of the feature and training algorithm of single detector, the system performance is greatly affected by the connection between multiple detectors. Basically, three kinds of structures are used: parallel [8, 9], pyramid [4, 2] and exclusive [3]. The parallel scheme is more accurate, but large computation cost is required. The exclusive scheme is fast, but it is difficult to precisely switch among detectors for large pose variation. Compared with these two schemes, the pyramid scheme gets better trade-off between accuracy and efficiency, such as Li’s [4] coarse-to-fine search and Huang’s width-first-search [2].

In some practical mobile applications like [6], the skin color is employed to speed up the detection and tracking process. However, the skin color lacks the discrimination ability for other body parts and background object with similar color. Therefore, the performance may be largely decreased under some mobile environment with varying illumination.

In this paper a robust face tracking system is presented which works very well on mobile platform, i.e. Nokia N95 and N82. The main characteristics of the system are: (1) adaptation for multi-view problem, including $360^\circ$ roll, $[-90^\circ, +90^\circ]$ yaw and $[-20^\circ, +20^\circ]$ pitch; (2) very high speed - up to 20 FPS in case of one face. The rest of the paper is organized as follows: the whole system is briefly describes in Section 2. Section 3 presents the implementation process of the proposed method. The experimental results and conclusion are given in Section 4 and 5.

2. METHODOLOGY OVERVIEW
The face tracking system in this paper follows the detect-then-track scheme. Both detection stage and tracking stage are based on statistical detectors. Basically, three boosted face detectors with local binary pattern (LBP) [5] and cascade structure [7] are trained for different facial views. By
flexibly using the rotation facility of LBP, these models can be extended to a larger set of virtual models and cover faces with large view variation.

Face detection stage aims at finding new faces through exhaustive search. To achieve better user experience, a robust facial pose estimation algorithm and the face matrix partition technique are employed to speed up this process.

Face tracking stage only perform local search for knowing faces with specific models and scales. Herein we utilize classification instead of other unsupervised techniques (e.g. MeanShift) for tracking. There are two reasons for this selection: 1) the accuracy of the face detectors is already very high; and 2) the color information is less discriminative and suffers from varying illumination condition.

3. IMPLEMENTATION

3.1. Learning method for face detector

LBP is robust for monotonic lighting change and only needs very low computational cost. More importantly, it is inherently convenient to handle the rotation problem without explicitly rotating the image itself. Suppose an image has 4 rotation orientations $0^\circ, 90^\circ, 180^\circ$ and $270^\circ$. After we compute the LBP map of one orientation, we can easily get the other 3 LBP maps by circularly shifting the LBP codes to 2, 4 and 6 bits. Due to above advantages, we choose LBP to construct the face detectors in the face tracking system.

The training process for single face detector basically follows the Gentle Boost framework[1], as is shown in Algorithm 1. The weak classifier structure and feature selection criterion are critical for the training process.

Given a $20 \times 20$ window, a 324-dimensional (the boundary pixels are omitted) feature vector $F$ can be extracted after smoothing. Each element of the vector corresponds to the LBP value of certain pixel within the window. For the sake of low computation cost, every weak classifier is built on one element, and the tree-like structure is adopted for these weak classifiers. Suppose the $k$th element is selected, the weak classifier will have 256 output leaves corresponding to LBP values in $[0, 255]$:

$$W_i(k, i) = \frac{P_i(\text{Face}|F[k] = i) - P_i(\neg\text{Face}|F[k] = i)}{P_i(\text{Face}|F[k] = i) + P_i(\neg\text{Face}|F[k] = i)}$$  \hspace{1cm} (1)

In the above, $t$ is the iteration number and $0 \leq i \leq 255$. $P_i(\text{Face}|F[k] = i)$ is the probability that the window belongs to face when the $k$th dimension value equals $i$, while $P_i(\neg\text{Face}|F[k] = i)$ is the probability that this window belongs to non-face. $W_i(k, i)$ is located in $[-1, +1]$ and larger value indicates that the window is more like a face. Furthermore, $|W(k, i)|$ measures the class separability for position $k$ and LBP value $i$. It is obvious that larger $|W(k, i)|$ implies better separability.

To select the most discriminative element from the feature vector in each round during the training, the following criterion is adopted:

$$J_t(k) = \sum_{i=0}^{255} W_t(k, i) \cdot P_t(F(k) = i), \quad 1 \leq k \leq 324 \quad (2)$$

where $P(F[k] = i)$ is the probability that the $k$th dimension value equals $i$. Therefore, the label of the most discriminative element will be:

$$ID_t = \arg\max_k J_t(k) \quad (3)$$

After a set of weak classifiers ($W_i(ID_t), 1 \leq i \leq t$) have been iteratively selected, the strong classifier will be:

$$S_t(X) = \sum_{j=1}^{t} W_t(ID_t, X[ID_t]) \quad (4)$$

By comparing $S_t(X)$ with an optimum threshold $\Gamma$, the decision can be made whether $X$ is a face.

In practical application, the corresponding probabilities in expression 1 and 2 can be estimated by the dynamic weighted histogram of the sample pool.

Algorithm 1 Learning algorithm for strong classifier

Input: face set $P(|P| = M)$ and non-face set $N(|N| = L)$

Output: weak classifier set $W$, threshold $\Gamma$

1: initial weights: $\theta_0[i] = \frac{1}{2M} |p_i \in P; \eta_0[i] = \frac{1}{2L} |n_i \in N$;
2: $t = 1$; $W = \Phi$;
3: for $t = 1, \ldots, T$ do
4: weights normalization;
5: estimate the required probabilities;
6: select $ID_t$th feature via equation (3)
7: $W = W \cap W(ID_t)$;
8: calculate the confidence $R_t[i](1 \leq i \leq P)$ for the $i$th face sample via equation (4)
9: select the optimum threshold $\Gamma$ based on $R_t$;
10: Check the rejection rate of non-face samples based on $\Gamma$, and decide whether to terminate the loop;
11: update the weights:
$$\theta_t = \theta_t \cdot e^{-S_t(p_t)} \quad (5)$$
$$\eta_t = \eta_t \cdot e^{-S_t(n_t)} \quad (6)$$

end for

3.2. Multi-view face detection

To save ROM space, only three models (detectors) are trained: $0^\circ$ model, $45^\circ$ model and a profile model (this model only contains right profile samples). Each model covers a certain
range of rotation. For example, the $0^\circ$ model should contain samples with $[-30^\circ,+30^\circ]$ roll, $[-22.5^\circ,+22.5^\circ]$ yaw and $[-20^\circ,+20^\circ]$ pitch.

The above 3 models can only meet a small part of requirement for multi-view face detection. However, by exploiting the rotation property of LBP, the above 3 models will be expanded to a larger set of virtual models. For example, by shifting LBP codes to 2, 4 and 6 bits, $0^\circ$ model can be used to detect faces with rotation angle of $90^\circ$, $180^\circ$ and $270^\circ$. Similarly, the $45^\circ$ model will offer the ability for $135^\circ$, $225^\circ$ and $315^\circ$. Meanwhile, the profile model can be expanded to 8 virtual models through mirror operation and shift operation.

So far, we have totally got 16 models (3 original plus 13 virtual), and they will collaborate for the task of multi-view face detection. Moreover, these additional virtual models do not occupy physical memory, which is crucial for mobile application. For each sliding window, if either of the models returns true, the final result will be face.

The LBP-based multi-view FD method is more efficient than the Haar feature used in[7][4][3]. In practical application, we first compute all LBP features of the whole image and save them in the memory. These LBP features can then be repeatedly picked up for different windows. This strategy is called feature-centric[8]. On the contrary, the number of Haar features is very huge due to different size and rectangle layout, so it’s hard to compute and save all required features beforehand. Therefore, the features needed in each window have to be computed individually via the integral image. This strategy is called window-centric and is computationally wasteful, especially for mobile platform.

3.3. Facial pose estimation

The drawback of multiple models is its time consumption. To solve this problem, we build several facial pose estimators to fast determine the most precise models for current window, such that the other unnecessary models can be neglected to save time. Firstly, the LBP feature vector of the input image patch is extracted. Secondly, the circular shift operation is conducted by 2, 4 and 6 bits to get the other three feature vectors. Thirdly, the 4 vector are fed to the facial pose estimator and 4 confidence values are got. Finally, only the feature vector corresponding to the maximum confidence value is selected for final decision.

In our system, the first layer of each cascade classifier is used as the estimator. Similarly, the first layer of $45^\circ$ model is used for $45^\circ$, $135^\circ$, $225^\circ$ and $315^\circ$, while the first layer of profile model is used for 8 profile orientations. As a result, 3 out of the 16 virtual models are needed for every image patch, such that the overall computation cost is greatly reduced.

The similar facial pose estimation method is used in [3] by selecting the first 3 layers. Compared with that method, two points makes our results more reliable: 1) in [3] the output of different classifiers are directly compared, while in this paper the outputs of the same classifier are compared; and 2) in [3] some similar facial poses work as the input, which makes the task rather difficult. In our method, the variation of the input classes (e.g. $0^\circ$, $90^\circ$, $180^\circ$ and $270^\circ$) is inherently large, which reduces the difficulty of facial pose estimation.

3.4. Face matrix partition

Although LBP feature, cascade classifiers and facial pose estimation is used to accelerate the detection stage, however, in practical applications the speed should be further increased to satisfy real-time requirement. Therefore, the face matrix partition method is proposed to make the whole process faster and smooth.

The face matrix is shown in Fig 1. The column represents faces with particular facial pose, while the row represents faces with similar size. Therefore, each grid in Fig 1 could represent a face subset with particular size and pose. Herein, the facial pose is categorized in the way that the models are defined in 3.2. For example, the $0^\circ$, $90^\circ$, $180^\circ$ and $270^\circ$ faces are grouped into the first column.

The idea of face matrix partition is to firstly cluster the whole face matrix into $n$ subsets, in order that every subset takes nearly the same computation cost for detection. Then during the tracking process, the $n$ subsets are repeatedly allocated to $n$ consecutive frame. Therefore, the average computing cost is greatly reduced. Meanwhile, every frame nearly costs the same time, so the whole process remains smooth. A negative influence of this scheme may be that, it usually takes more than 1 frame to detect a new face. However, considering that FPS is largely increased, such influence can be omitted in practical application.

4. EXPERIMENT AND APPLICATION

4.1. Experiment setup

The experimental platform in this paper is Nokia cell phone N95 and N82. These two phones are both equipped with a 5 megapixel camera and a 322Mhz ARM11 CPU. The software environment is Symbian S60 3.1 Edition. A camera-based
mobile application will run on the S60 platform to help to evaluate the performance of the method in this paper. Image sequence with size QVGA can be continuously extracted in the application for face tracking. The detected face will be indicated by a green rectangle.

For each of the three models (0° model, 45° model and profile model), more than 40000 face samples are collected and the non-faces are automatically generated by the Bootstrap method. After the training, 12, 12 and 11 layers (containing 803, 802 and 634 weak classifiers) are obtained, respectively. The total memory consumption of these models is about 1.1M bytes, which is acceptable for S60 system.

Because the lowest target toward speed is 10 FPS, we partition the face matrix into 5 subsets. Therefore, it will theoretically cost at most 0.5s to find a face of any size and posture. This response speed can be accepted for practical application.

4.2. Face tracking

This experiment will test the real-time characteristic of the face tracking system. The environment includes outdoor, office room and a dark room with controllable lighting. The minimum and maximum face size is set as 25 and 150.

Several scenarios are included during the test: 1) the tester holds the mobile phone at any angle; 2) the object person rotates his face within a large out-of-plane rotation angle; 3) the object person move slightly around the given location; 4) the tester moves phone freely; and 5) multiple faces appear in the image simultaneously.

The recall rate and precision rate are employed for evaluation. A correct tracking can be counted only when a face is robustly marked by the green rectangle till its disappearing. An incorrect tracking is counted when a false alarm appears or the rectangle does not cover the major of the face.

In the test, we get a 96.52% recall rate and a 98.5% precision rate. The missed faces are caused by very fast posture change. The real speed of face tracking is about 18-20 FPS for one face. Even for the limitation of 10 faces, the speed could retain 10 FPS and no clear pause occurred in the video stream. Therefore, this face tracking system can satisfy the requirement of real-time application.

Fig 2 lists several screen shots when the face tracking system is running. Note that these images are captured from a big monitor which is connecting with the mobile phone through a video cable. The content displayed in the monitor is fully the same as is processed in the mobile phone.

5. CONCLUSION

In this paper a multi-view and real-time face tracking system is presented. The main contribution of this paper includes: 1)a set of LBP-based models are designed for fast multi-view face detection; 2)a precise facial pose estimation method is proposed to speed up the detection process; and 3)the facial matrix partition method is adopted to smooth the tracking process and increase the overall FPS. The experimental results of face detection and tracking on Nokia cell phone has proved the feasibility of the proposed method.

6. REFERENCES


