Real-Time Face Detection/Identification for Surveillance System

Nikom Suvonvorn, Anant Chocksuriwong
Department of Computer Engineering, Prince of Songkla University,
Hat Yai, Songkhla 90112, Thailand
Email (kom@coe.psu.ac.th, ant@coe.psu.ac.th)

Abstract

Face processing is one among the popular problems of detection and recognition. In surveillance system application, the massive data of video is analyzed by complex algorithms that must consider the real-time constraints. In this paper we propose a face processing framework as a component-based architecture for dealing with face online processing. Face detection and classification algorithms are realized based on a Haar-like features and Support Vector Machine (SVM) respectively as a way to efficiently extract a face image and enhance face recognition rates with robustness on different orientation Experimental results are presented with discussion.

1. INTRODUCTION

Face detection and classification is one of video processing techniques needed as basic functions for third generation surveillance systems. System detected a present of faces in scene for identify known person in database and stored in case unknown. Face recognition acquires biometric information from a person to be verified in a less intrusive manner than other biometric recognition techniques, such as fingerprint recognition, without requiring the person to directly contact a recognition system with part of his or her body. Many architectures [1][2][3] has proposed for dealing with real-time processing. We propose here a component-based framework of on-line face processing for surveillance system. Our goal is first to obtain a real-time face detection and classification framework with some applicative aspects of surveillance domain, such as, acquisition, storage, alert and playback components. The processing system architecture based on the inter-components synchronization and the inner-component execution model is adopted in order to deal with the parallelization on the multi-cores and multi-processors. The seconde objective is to deal with the face detection and classification. The system is realized based on a Haar-like [12] features algorithm and a Support Vector Machine (SVM) [10] algorithm method as a way to efficiently extract a face image and enhance face recognition rates with robustness on different orientation (scale, rotation and affine transform) and occlusions problem [13].

The paper is organized as follows: section 2 discusses on the system architecture of on-line face processing, section 3 present the face detection and classification method, and section 4 and 5 show performance evaluation and conclusion respectively.

2. ON-LINE FACE PROCESSING FRAMEWORK

The architecture of our framework is divided into six components: video acquisition, video analysis (face detection), decision component (face identification), video recording, event alert, and playback. These components lay on the three layer architectures: Application Layer (AL), Communication Layer (CL) and Processing Layer (PL), shown in figure 1. In this framework, the Processing Component (PC) means alls component located on PL, and respectively for Communication Component (CC). The CL facilitates the communication between components, and also with external sources such as cameras, alert system and database server. It emphasizes how images would be transferred from one to another component. Let’s notice that the processing time of each component is different. Generally, displaying images from acquisitions module can be done with no latency. However, in the situations such as complex images analysis or image encoding with different qualities will logically spend more processing time than the acquisitions rate which leads to the problem of image exchange between components. This problem will take care by the inter-component synchronization mechanism of CL layer.

In contrary, the PC component is considered as the internal unit of component on CL. There is no communication between components on PL layer. CC can load/unload PC in order to execute a specific task.
The PC of acquisition component corresponds to different drivers of IP-camera. In the case of video analysis and decision components, PC represents the face detection and identification algorithm respectively. The acquisition component requests image streams from IP-camera. Decoded Images are then transferred to the video analysis and decision component for face detection and identification. The results as image regions of faces and its features are then submitted to corresponding components for alerting and recording.

In order to decrease the computation time for face processing, the inner-component execution for supporting the multi-cores and multi-processors is introduced. Note that the face detection and identification algorithms are implemented in low-level programming language C.

Figure 1. Component-based Framework for on-line face processing

A. Inter-Components Synchronization

The synchronization between components is the key point of our framework. In order to profit fully the parallelization on the multi-cores and multi-processors architecture, each instance of component is designed to be independently executed and acts as consumer and producer of image data. At each communication between any two CCs, a buffer is used as resource exchange zone that play the important role for the best effort service, note that the two components may be processed at different frame rate depending on a specific task. The synchronization guarantees that only one component can access to the buffer at a time for reading or writing. Number of image data in buffer is limited at a value, if there is more data than that value the oldest one will be automatically removed. The mechanism is implemented using event interaction model with the publish/subscribe paradigm. The components can interact in different ways, such as one-to-one, one-to-many (multicast), many-to-one, and many-to-many.

Figure 3 shows the synchronization of two components (thread#1 and thread#3) with an event service (thread#2). The image data \( VsImage \) of thread#1 is published asynchronously to thread#2 for collecting in a buffer. For an interval of time, thread#3 requests thread#2 for \( VsImage \) and remove it from the buffer. For the safety of resource sharing in multithreading programming some lock/unlock procedures are applied.

B. Inner-Component Execution

Each communication component contains a processing component representing algorithm to be executed. A clock inside CC component is implemented for fixing an interval of execution time. For each elapsed time, new thread is created for running the same instruction code. Lock/unlock procedures are used for preventing the thread from others in sharing resource conflicts during the receiving and sending image data. Note that \( VsImage \) is copied as private resource to each thread, and then the process can be executed in parallelized ways for multi-cores or multi-processors architecture.

Figure 4 shows diagrams displaying the relation between the time interval (T), execution time of a frame (E), and number of threads (N). We can noticed that if the execution time of process is less than time interval, only one reused thread is sufficient, in contrary more threads \( N=1+E/T \) are needed. However, in case when the serial manipulation is stricted, such as the encoder component, this design
doesn’t profit the parallelize configuration. For example, the encoder receives the sequence of image data for encoding into a file which is entirely a serial process. Only a small piece of code (for example the watermarking process) before encoding that can have a change to be parallelized. If the execution time of algorithm is longer than the time interval, the number of threads on the stack waiting for execution will explode that may cause the serious problem for the stability of system. Note that the time interval defining by application represents the temporal resolution which is considered as quality of services. This value concerns how the system is sensitive to the event of interest. We fix the value by default at 250 milliseconds (4 fps).

3. FACE DETECTION AND CLASSIFICATION

In this section we focalize on the video analysis and decision component of our proposed framework of on-line face processing that corresponds to face detection and identification algorithm. The problem was the use of face as biometric data is that images obtained are strongly influences by any conditions: orientation, illumination, occlusion and expressions of face. The illumination correct with the processes of normalization. Problems about expression are managed by the data processing algorithms / extraction. The largest problems are concern about orientation of face and his occlusion [4][5][6].

C. Training (Offline Processing)

Lots of keypoints detectors have been proposed in the literature [7]. They are either based on a preliminary contour detection or directly computed on grey-level images. The Harris detector [8], that is used in this article, belongs to the second class. It is consequently not dependant of a prior success of the contour extraction step. This detector is based on statistics of the image and rests on the detection of average changes of the auto-correlation function. Figure 4 presents the interest points and the associated neighborhood obtained for one individual extracted from the AR face database [9]. Local descriptors are the computed on the neighborhood of interest points.

D. Face Detection (Video Analysis Component)

Haar-like features are widely used in face searching and numerous prototypes have been trained to accurately represent human faces. In a Haar-like feature approach, feature values are obtained by summing up the values of pixels in each region of a face image and weighting and then summing up the regional sums, instead of directly using the values of the pixels of the face image. Fig. 6 illustrates various prototypes of Haar-like features.
1. Edge features
   ![Edge features]

2. Lines features
   ![Lines features]

3. Center-surround features
   ![Center-surround features]

Figure 6. Prototypes of Haar-like features

Fig. 7 illustrates a block diagram of a face detection method using Haar-like features. A face area of an image of one frame is determined using a trained group of face images while gradually decreasing the size of the image in a pyramid manner. A plurality of candidate areas for the face area of the image are generated in the process of recovering the size of the image, and an average of the candidate areas is output.

![Block diagram of Haar Face Detection]

Figure 7. Block diagram of Haar Face Detection

E. Identification (Decision Component)

The class of each key point is obtained by the SVM that takes into account the invariant descriptors computed within its neighborhood. The class of the unknown object corresponds to the majority class after the voting for each key point (Fig. 8). This approach permits the recognition of an object even for example it is occluded by another one.

![Block diagram of Recognition System]

Figure 8. Block diagram of Recognition System

4. PERFORMANCE

The system is executed for face detection and identification with event-based recording that runs for a few days continuously. The image resolution is fixed at 640x480 with 4 fps.

![Software User Interface]

Figure 9. Software User Interface

Firstly, we observed that our framework is robust to either the number of cameras or the frame rate changes, depending on the traffic of network. The load balance between the two cores show every encourage results at around 50±5%. Table 1 shows the performance.

<table>
<thead>
<tr>
<th>C</th>
<th>S</th>
<th>N(Kb/s)</th>
<th>M (Mb/s)</th>
<th>P (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>640x480</td>
<td>330</td>
<td>80</td>
<td>50±05</td>
</tr>
<tr>
<td>6</td>
<td>640x480</td>
<td>550</td>
<td>100</td>
<td>75±05</td>
</tr>
<tr>
<td>9</td>
<td>640x480</td>
<td>770</td>
<td>180</td>
<td>90±10</td>
</tr>
</tbody>
</table>

Table 1. Performance (C=camera, S=size, N=network, M=memory, P=CPU)

We can notice that if the CPU is used at maximum performance, the number of images to be analyzed is automatically reduced which will systematically lead...
to the increasing of memory and latency. We found that up to 9 cameras can be supported by our framework at around 4 fps. We can notice that the framework acts the role of the best effort service. However, this will cause inevitable the increasing of latency.

Figure 10. Face Detection Results

### F. Face Detection and Identification Performance

The performance of a face recognition system realized based on a AR face database (126 personnes [Fig 11], 70 men et 56 women )

<table>
<thead>
<tr>
<th>Method</th>
<th>Recognition Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenface</td>
<td>79.11%</td>
</tr>
<tr>
<td>Local Inv+SVM*</td>
<td>97.48%</td>
</tr>
<tr>
<td>Fisherface</td>
<td>98.85%</td>
</tr>
<tr>
<td>DCV</td>
<td>99.65%</td>
</tr>
</tbody>
</table>

Table. 2 Recognition rate comparison with another methods

The time taken for the development system to obtain the name of each person to be verified from the server was measured that the system can complete face verification about less than a second. The system is also efficient for a face recognition system that operates on a large data base requiring high computational load and a large memory size.

### 5. CONCLUSIONS

We presented a component-based framework of face processing applied for surveillance system. Face detection and face classification algorithms base on Haar-like and SVM method are implemented on the framework and tested with respect to the processing time and recognition rate. The result shows that the testing system can perform as the best effort nice cameras in parallel with up to 99.65% of recognition rate.

![Figure 11. Sample of face with different occlusions](image)

### REFERENCES