A Video Analysis Framework for Surveillance System

Nikom Suvonvorn

Department of Computer Engineering, Prince of Songkla University,

Hat Yai, Songkhla 90112, Thailand

kom@coe.psu.ac.th

***Abstract*—an on-line video processing for surveillance system is a very challenging problem. The computational complexity of video analysis algorithms and the massive amount of data to be analyzed must be considered under real-time constraints. Moreover it needs to satisfy different criteria of application domain, such as, scalability, re-configurability, and quality of service. In this paper we propose a flexible/efficient video analysis framework for surveillance system which is a component-based architecture. The video acquisition, re-configurable video analysis, and video storage are some of the basic components. The component execution and inter-components synchronization are designed for supporting the multi-cores and multi-processors architecture with multi-threading implementation on .NET Framework. Experimental results on real-time motion tracking are presented with discussion.**

I. Introduction

For many reasons, including security, the popularity of IP-based cameras for video surveillance is growing rapidly. In third-generation video surveillance systems, video information, as a stream or sequence of digital images, provides an intelligent and efficient approach for automatic system analysis. Motion detection, object tracking, and abnormal event detection are some of the video analysis techniques needed as basic functions for modern surveillance systems. Many researches [1][2][3][4] have proposed new surveillance system architectures for supporting these technologies. For example, [5][6][7] present cooperative target tracking methods using multiple cameras. The surveillance systems offered by private companies and government organizations range greatly in price and quality. However, most of them support only their own products, and camera infrastructure.

We propose here a novel component-based framework of on-line video processing for surveillance system designed to work independently with camera devices and network infrastructure. Our goal is first to obtain a flexible/efficient system for real-time video analysis with some applicative aspects of surveillance domain, such as, acquisition, analysis, storage, alert and playback components. The inter-components synchronization and inner-component execution are introduced in order to deal with real-time processing. For supporting the parallelization on the multi-cores and multi-processors architecture, the multi-threading programming with event-based paradigm is implemented under .NET framework.

The paper is organized as follows: section 2 discusses on system requirement and its specification. The architecture design of overall system shows in section 3, section 4 is devoted to software architecture, and section 5 and 6 present system performance evaluation and conclusion respectively.

II. System Requirement & Specification

Content-based video analysis for surveillance system requires basically real-time processing with respect to some quality of services such as accuracy, spatial/temporal resolution, and latency. Effectively, the challenging task needs a particular architecture to be supported.

In real situation, multiple sensors can connect to system for several analysis tasks: detection, tracking, recognition and etc. System's resources, for example CPU memory and network bandwidth, are necessary to be allocated dynamically. Certainly, the scalability of architecture is required in order to guarantee a certain quality of services. As well as the capability of distribution of analysis tasks to match with the allocating additional resources in an optimum way will increase systematically the efficiency of system. A distributed scalable architecture is basically adopted as our conceptional application design implementing by parallelizing and distributing different components to different cores and processors. The re-configurability of system is another important requirement in order to reflect the necessary of usages in the real world. Content-based video analysis for surveillance system needs to be adaptable for the specific tasks, for example, local analysis with signal sensor, cooperative analysis with multiple sensors, and global analysis for overall decision system.

The specification of our system is defined based on the surveillance application which concerns initially the video sensors. This should satisfy the criteria as follows:

- Support the multiple cameras platform such as Axis, D-Link, Panasonic, Sony, Stardot, Pixord and etc.
- Support the multi-cores and multi-processors architecture.
- Support the standard functions of Network Video Recorder (NVR) such as viewing, recording (full, motion based), and playback.
- Support the additional plug and play modules of image analysis filters and encoders.

III. System Architecture

The architecture of our system can be divided into five important components: video acquisition, video analysis, video encoding, event alert, and playback. These components lay on three layer architectures: Application Layer (AL), Communication Layer (CL) and Processing Layer (PL), shown in figure 1. Note that 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.

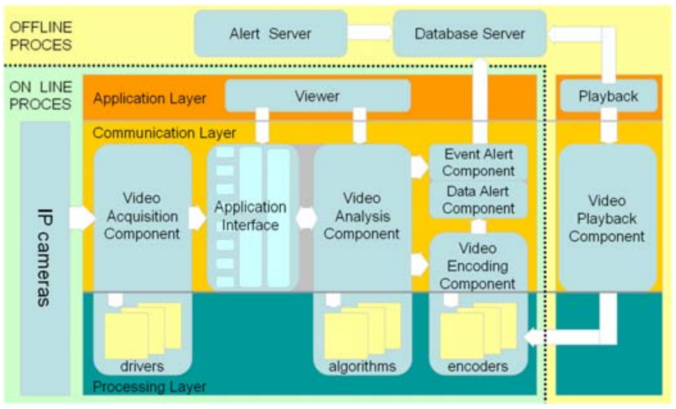

Fig. 1. Third generation of video surveillance System

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. For example, PC of acquisition component corresponds to different drivers of IP-camera. In the case of video analysis and storage components, PC represents different kind of image filters and encoders respectively. All PC components are developed separately from the framework and acts as plug-ins. It is important to be noticed that the first four components are some kind of online processing. The acquisition component requests image streams from IP-camera localized on internet or on local area network. Images stream are decoded as sequence of image frames and then transferred to video analysis component. The analysed results as image or sequence of images or image's features will be used later by event alert and encoder components. The last component of system corresponds to the playback which is an offline component. It uses the same drivers of encoder for decoding the images sequence and display.

In order to decrease the computation time for video analysis, we introduce the *inner-component execution* for supporting the multi-cores and multi-processors. Note that the algorithms of PC components are preferred to be implemented in low-level programming language like C or C++.

A. *Video Acquisition Component*

A lot of manufacturers provide a great range of IP video cameras. The simplest video format supported by almost all IP cameras is just JPEG. Cameras allow retrieving a single image by accessing a special URL. The disadvantage is that it is required to send a new HTTP request to the camera's web server each time you need to retrieve a new image, which makes loss of speed because of extra HTTP headers being sent/received. The advantage is that the acquisition component can easily control the frame rate.

The second popular format is Motion JPEG (MJPEG). This format allows downloading a stream of JPEGs. In this case, the acquisition component does an HTTP request to a special camera's URL. Then, the camera replies to request with a stream of JPEGs delimited with a special delimiter specified in HTTP headers. MJPEG approach seems to be much better than JPEG, because it speed up the acquisition rate with lower bandwidth. Cameras from most of manufacturers support these two basic formats such as Axis, D-link, Panasonic, Bosch and etc.

MPEG-2 and MPEG-4 video format are actually the most popular. Normally, the encoders/decoders of MPEG-2 and MPEG-4 are specifically defined by the provider, called native format. By adding some special watermark filters, unusual manipulation of video can be detected. Most of manufacturers provide the ActiveX control for accessing to their native video formats.

B. *Video Analysis Component*

The PC components represent basic techniques or algorithms for content-based video analysis such as motion detection, object tracking and recognition. By applying these basic components, the more complex applications can be developed, for example parking surveillance system, entrance security system, and etc. In our architecture, the application interface attached with video analysis component play the role of re-configurable application. The analysis can be defined into three levels depending on the complexity of situations consist of local analysis (signal sensor), cooperative analysis (multiple sensors), and global analysis (local and cooperative together).

Figure 2 shows an example of entrance security system. It composes of two local analyses for face and license plate detection, one cooperative analysis for under vehicle scanner, and one global analysis for decision marking.

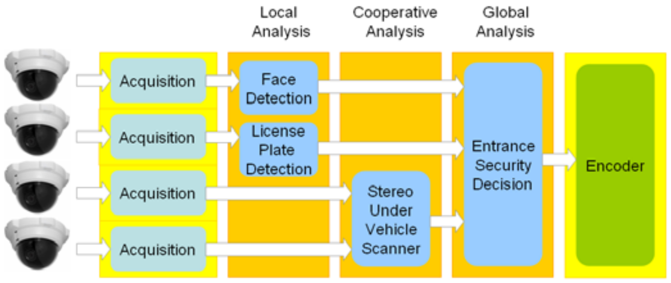

Fig. 2. Re-configurable application: example for the entrance security system

C. Video Encoding Component

Video encoding component of our surveillance system is designed to store image sequences when only if the events of interest on the scene, such as motions, are detected. Like the other PC components, the encoders can be developed (or using an existing one) and interfaced to the principal framework by adding some codes that act as plug-ins. In the current version, window media encoder and AVI encoder with windows media codec are implemented. The most important step for using encoder is how to describe the encoding profiles. In general, the codecs for quality-based encoding or bit rate-based encoding needs to be explicitly specified. In our case, 100% quality-based encoding is used that gives the best video quality with less time consumption of encoding process.

D. Event and Data Alert Component

Event and data alert component are utility of our system in order to communicate with the external database server. An event will be raised systematically by alert component when an event of interest in the video occurs. Event detection is analysed by video analysis component. The events are necessary for driving the other components of system such as encoder, database server and alert system.

Data alert component captures the alerts from the video encoding component. After finishing the encoding process of images sequence of current event, an alert is immediately raised by encoder component that invoke the data alert component. It sends a metadata containing video file description to the external database server. The following items are indicated in metadata: name, size and location of video file, staring and ending time of event, algorithm's name and encoder type. This information will be used by the playback application and alert system.

E. Inter-Components Synchronization

The synchronization between components is the key point of our architecture design. 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.

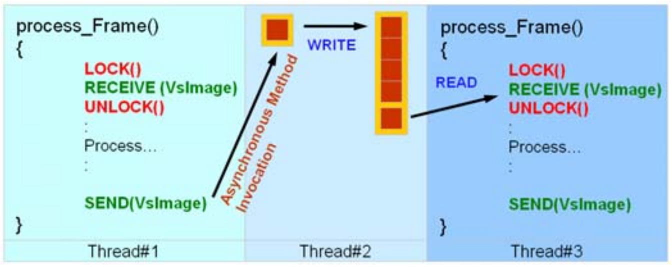

Fig. 3. Asynchronous Inter-Components Synchronization

F. Inner-Component Execution

Each communication component contains a processing component representing algorithm to be executed. A clock insides 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.

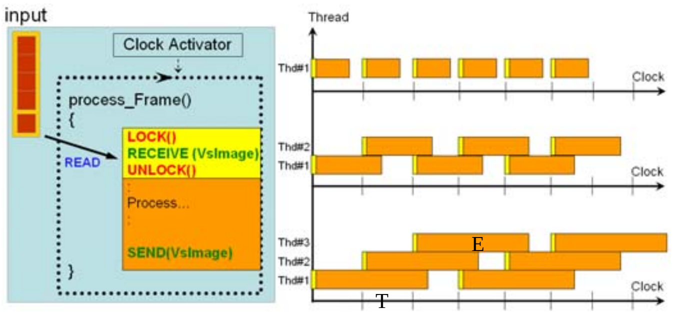

Fig. 4. Inner-Component Execution

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 the situation when some serial manipulation is strict, the design doesn't profit the parallelize configuration. Figure 5 shows the encoder component, it 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) .

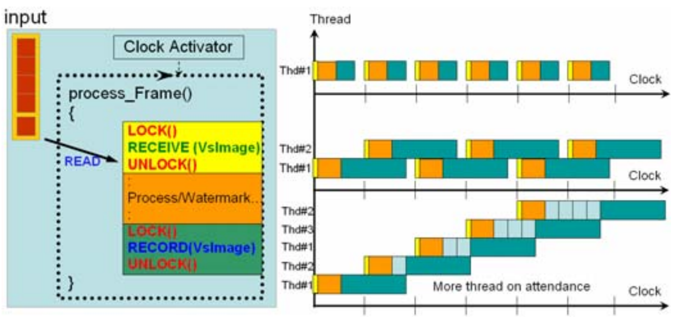

Fig. 5. Inner-Component Execution

IV. Software Architecture

In this section we describe the software architecture of our framework which is entirely implemented under the .NET framework version 2.0. Our architecture consists four layers: application layer, application core layer, system core layer, and external module layer. The *VsCore* in the system core layer is the most important library that implements how to synchronize between communication components for supporting the three possible types of analysis: local (*VsCamera*), cooperative (*VsChannel*) and global analysis (*VsPage*). The system core also defines how external processing modules can be interfaced and executed during the runtime service. Four types of interfaces are predefined: *VsProvider* for image acquisition, *VsAnalyzer* for any kind of image filters, *VsEncoder* for image encoding into files, and *VsData* for database connection.

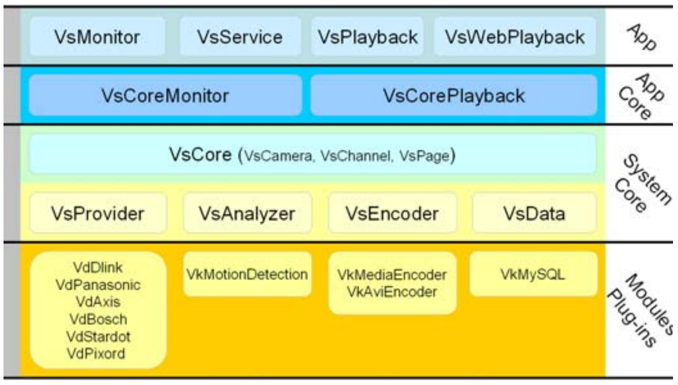

Fig. 6. Software Architecture

The external module represents the algorithms which are implemented in different languages and can be plugged into system as plug-ins. For the application core, its role is to define the functionality of a specific application that implements under the system core. The application layer interfaces the application core in different ways to the users which allows, for example, the same function can be called from either standalone or web application. Figure 6 shows overview of our software architecture. Figure 7 shows the standalone user interface that allows viewing analysing and encoding many different cameras from different manufacturers at a time.

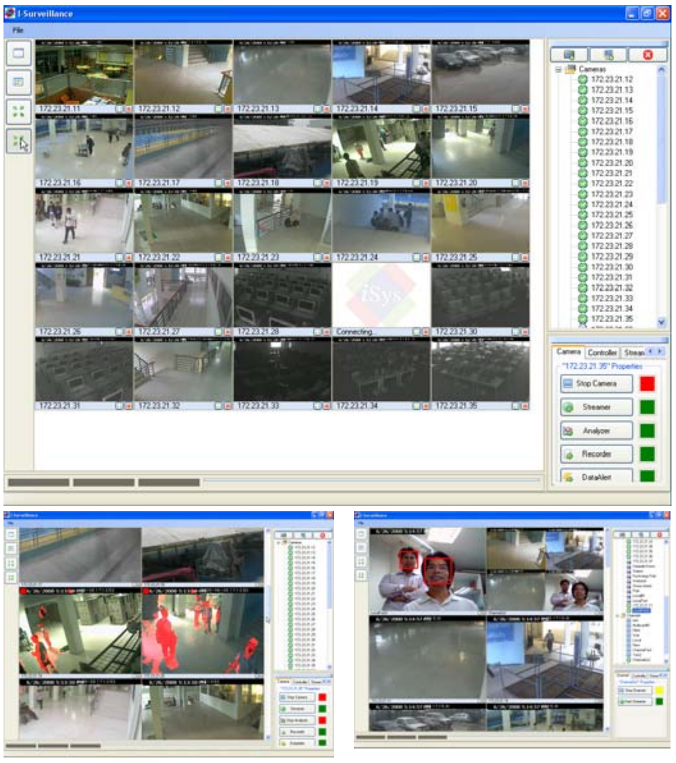

Fig. 7. Software User Interface and Results

V. Performance

In order to test our system, the algorithms for motion detection (figure 8) and face detection are implemented. Motion detection is based on background subtraction technique which the stationary background is updated by running average. Motion history of moving object over time is computed in order to increase the robustness of detection rate. Then, object direction is determined from its gradient. Face detection is based on Haar-like features proposed by Paul Viola[8] and improved by Rainer Lienhart[9].

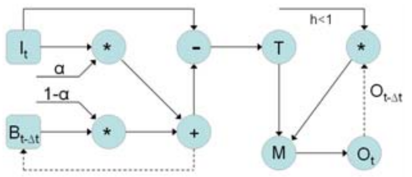

Fig. 8. Motion Detection and Tracking: **I** current image, **B** background image, **T** threshold, **O** history image, **M** maximum, ***, - , +** operators, **h, α** constants.

The test is done with some 25 IP-cameras from Axis Panasonic and D-Link manufacturers. A desktop PC with Intel Core 2 Duo processor cadenced at 1.7 Ghz, 1 Gbyte memory and 100 Mb/s Ethernet card is used for testing.

Firstly, the system is executed with motion detection and full recording during a few days continuously. The spatial resolution of image sequence is fixed at 640x480 and the temporal resolution is defined for 2 and 4 fps. We observed that the framework is very robust to either the number of cameras or the frame rate changes which depending on the traffic of network. The load balance show every encourage results at around 50±5% for each core. Table I shows the performance in details according to number of cameras (C), image size (S), and number of analyzed image in frame per second (F). The resources used for network bandwidth (N), memory (M), and CPU times (P) are detailed. We can notice that if the CPU is used at maximum performance for 100%, the number of images to be analyzed is automatically reduced which will systematically lead to the increasing of memory and latency.

TABLE I
PERFORMANCES

C	S	F*	N(Kb/s)	M (Mb/s)	P (%)
Motion Detection					
9	640x480	4-3.11	770	180	50±05
12	640x480	4-3.41	1.390	250	80±05
16	640x480	4-3.64	2.500	400	95±05
20	640x480	2-1.60	930	340	60±10
25	640x480	2-1.72	1.340	400	80±10
Face Detection					
4	640x480	4-3.10	330	80	50±05
6	640x480	4-3.25	550	100	75±05
9	640x480	4-3.50	750	180	90±10

*F (4-3.11) means that 4 is the number of expected images to be analyzed and 3.11 is the real value at runtime.

Secondly, the system is executed with motion detection but recording only if motion events are detected. The spatial resolution of image sequence is fixed at 640x480. We found that up to 25 cameras can be supported by our framework at around 4 fps. This may explain that some of image sequences are buffered if there are a large number of motions to be analyzed or encoded. The images in buffer will then be executed later when the CPU is available. We can notice that the framework acts the role of the best effort service. However, this will cause inevitable the increasing of latency.

Thirdly, the system is executed with face detection and full record. We found that only nice cameras running in parallel the system is saturated, shown in table I.

VI. CONCLUSION

We presented a novel component-based framework for the on-line video processing of surveillance system. The system architecture supporting the multi-cores and multi-processors is proposed by introducing the inter-component synchronization and the inner-component execution. The implementation of multi-threading programming is done under .NET Framework. Some preliminary tests of system performance with motion detection and face detection shows encourage results. In the future work, more extra testing with complex algorithms of the cooperative and global analysis need to be performed.

REFERENCES

[1] R.G.J. Wijnhoven, E.G.T. Jaspers, P.H.N. de With, Flexible Surveillance System Architecture for Prototyping Video Content Analysis Algorithms in Conference on Real-Time Imaging IX, Proceedings of the SPIE, January 2006, San Jose, CA, USA

[2] Xiaojing Yuan, Zehang Sun, and Y. Varol, and G. Bebis, A distributed visual surveillance system, Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, 21-22 July 2003, page(s): 199- 204.

[3] B.Georis, X.Desurmont, and all, IP-distributed computer-aided video-surveillance system, Intelligence Distributed Surveillance Systems, IEEE Symposium on (Ref. No. 2003/10062) Volume, Issue , 26 Feb. 2003 Page(s): 18/1 - 18/5

[4] C. Jaynes and S. Webb and R. Steele and Q. Xiong, An Open Development Environment for Evaluation of Video Surveillance Systems, Proceedings of the Third International Workshop on Performance Evaluation of Tracking and Surveillance, 2002.

[5] B. Abreu, L. Botelho A. and all. Video-based multi-agent traffic surveillance system, Proceedings of the IEEE Intelligent Vehicles Symposium, 2000. Page(s):457 – 462.

[6] T.Matsuyama and N.Ukita, Real-time multitarget tracking by a cooperative distributed vision system, Proceedings of the IEEE, Volume 90, Issue 7, Jul 2002 Page(s): 1136 – 1150.

[7] J. Piater and J. Crowley, Multi-modal tracking of interacting targets using Gaussian approximations, n Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance. IEEE Computer Society, Dec. 2001S. M. Metev and V. P. Veiko, *Laser Assisted Microtechnology*, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[8] Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001. The paper is available online at http://www.ai.mit.edu/people/viola/

[9] Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.