The Sixth PSU Engineering Conference
8-9 May 2008

A Generic Visual Surveillance System

Nikom SUVONVORN[1*]
[1]Department of Computer Engineering, Prince of Songkla University,
Hat Yai, Songkhla 90112
E-mail: kom@coe.psu.ac.th[*]

Abstract

We introduce in this paper a generic software for real-time video surveillance system to support ongoing research in motion detection, tracking and abnormal event recognition. The proposed system is a multi-camera platform that able to connect with different standard of video inputs for many manufacturers. The architecture of our platform is designed in an efficient way for video display, storage, rapidly searching and playback. The algorithms for video analysis can be developed separately from platform as plug-in. In the current version, some classic algorithms are integrated such as motion detection, and tracking method. The storage system for video encoding and management is functionally based on motion events that invoking by motion detector for staring/stopping video encoding. Some preliminary tests of system are performed in real outdoor/indoor situations.

Keywords: surveillance system, video content analysis, motion detection, object tracking.

1. Introduction

In this article, we introduce a novel platform of video surveillance system working with IP-based cameras. Our goal is first to obtain an efficient complete system for video viewing, video analysis, storage and searching and playback.

For many reasons including especially security, it can be noticed that popularity of IP-based cameras for video surveillance is growing really fast. In the third-generation of video surveillance system, illustrated in figure 1, 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 video analysis techniques needed as basic functions for modern surveillance system. Many researches [1][4][5][6] proposed novel surveillance system architecture for supporting the technologies. [2][3][7] presented the cooperative target tracking methods using multiple cameras of surveillance system.

The surveillance systems existing in the market using by private companies and government organizations provide a large range in terms of prices and qualities. However, most of them support only its own products, system infrastructure for their cameras. We proposed here a system which is designed for working independently with camera devices and network infrastructure.

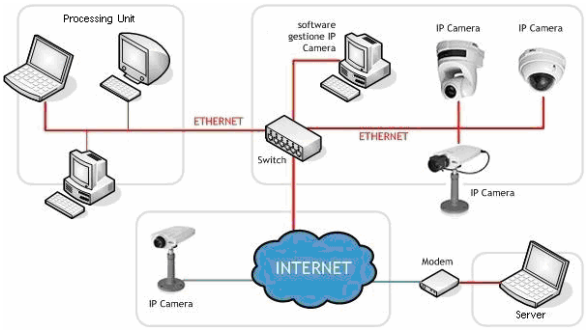


Figure 1. Third generation of video surveillance System.

The paper is organized as follows: section 2 describes architecture design of overall system, section 3 is devoted to the videos content analysis, section 4 presents video storage system, and the last two sections show system performance evaluation and conclusion respectively.

2. Overall system overview

The principle functions of our system can be divided into six important components: acquisition, display, video analysis, storage, alert, searching and playback. These components lay on two processing layer architecture: High Processing Layer (HPL) and Low Processing Layer (LPL), shown in figure 2. Note that High Processing Component (HPC) means alls component located on HPL, and respectively for Low Processing Component (LPC). The HPL facilitates the communication between HPC components, and also with external system such as cameras, alert system and database server. It emphasizes how images would be transferred form one to another component. Let's notice that the processing time of each component is different. Normally, displaying images from acquisitions module can be terminated at the same time with no problem. However, in some situation such as images analysis by complex algorithms or image compression by encoders with different qualities, these naturally spend more processing time than the acquisitions rate. This problem of images exchange between components is take care by HPL mechanism. Each communication between any two

HPCs, a buffer is used as resource exchange zone. Only one component can access to buffer at a time for reading or writing.

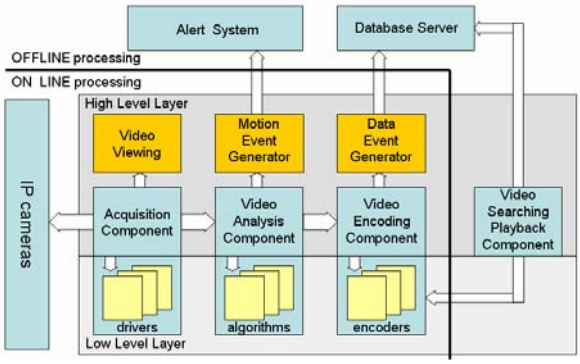


Figure 2. System Architecture.

In contrary, the LPC is considered as an internal unit of HPC component. There is no communication between LPC components on LPL layer. HPC can load/unload LPC in order to execute a specific task. For example, LPC of acquisition component corresponds to different drivers of IP-camera. In the case of video analysis and storage components, LPC represents different kind of image filters and encoders respectively. Actually, there is no LPC for displaying component. All LPC components are developed separately from the main system and acts as plug-ins. In order to decrease the computation time for video analysis, the algorithms of LPC components are mostly implemented in C.

It is important to be noticed that the first fives 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 and then transferred to video processing component. It produces the results as image or sequence of images or image's features that will be used by storage and alert modules.

The last component of system corresponds to the searching and playback which is an offline component. It uses the same drivers of encoder for decoding the images sequence and display. Some facilities for searching are actually developed in order to select the events of interest.

2.1 Acquisition

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. In general, the native video format is the most suitable for such system. Our acquisition and displaying component inspired the work of Andrew Kirillov [8].

2.2 Graphic User Interface

Proposed system is a multiple-camera platform that allows viewing many different cameras from different manufacturers at a time. User design should be easy as most as possible for using.

At the first step, user has to add new cameras by configuring the camera's parameters and video analyzers. For the current version, this is only one encoder, so there is no need to define. The available camera's drivers and image analyzers are automatically listed in the configuration dialog.

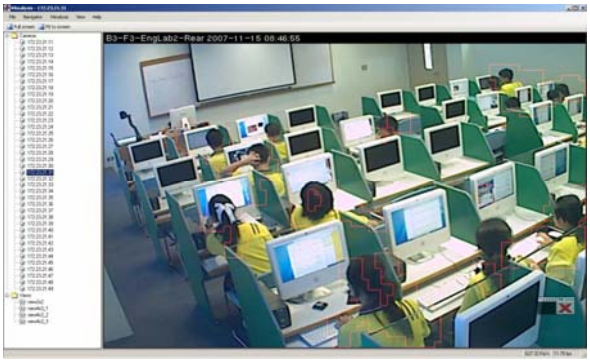


Figure 3.1. Connect to a signal camera at PSU Wittayanusorn School.

A view for grouping the interest cameras can be created by using a dialog wizard in order to simultaneously display. User can define number of cameras and position in the view that can be complete simply by drag and drop style: selecting available pre-defined cameras into view.

All configurations about application, cameras, analyzer, encoders and views are saved automatically in XML configuration files. The figure 3.1-3.3 shows examples of user interface.

Figure 3.2. View definition by regrouping cameras into 3x4 matrix.

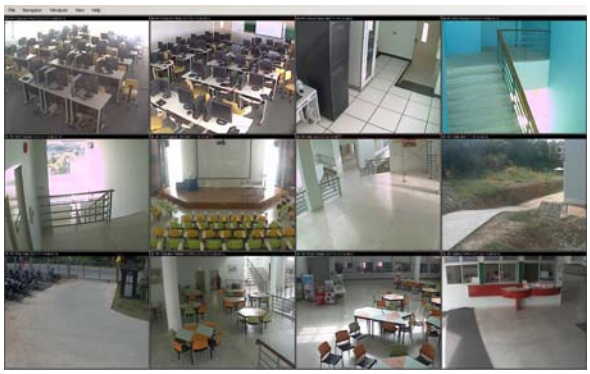


Figure 3.3. View in full screen.

3. Videos Analysis

For the video surveillance system, techniques for motion detection, object tracking and recognition are mostly required. From these basic algorithms, the more complex applications can be developed, for example, object removal detection, direction control, preventive area alert and etc.

In order to test our system, an algorithm for motion detection is implemented. Note that this algorithm has to develop and test separately from the principle system. Then, by adding some interface codes (LPC), the system can recognize them as its plug-ins.

3.1 Motion History Detection/Segmentation

Mainly purpose of motion detection is to search the pedestrian or vehicle presences in images sequence. The "Background subtraction" method is employed in this paper which is suitable for fixed-cameras. It is a classic technique for detecting moving object in images sequence. The basic idea is that moving object O can be detected by subtracting a stationary background B_t from the current image I_t.

$$O = \min(|I_t - B_t|, I_{\max})$$

The accuracy of method depends crucially on the quality of background normally degraded by the following factors: predefined threshold setting for assigning pixels to background/foreground, quick contrast/illumination changes in scene, shadows and etc. Lots of researches have been carried out in order to improve the background subtraction method such as mixture of Gaussian, Pixel-Wise Gaussian distribution. However, these lead to more complex algorithm which increases the inevitable computation time. By this reason, we chose a simplistic background model that updating its current background to the new one by average.

$$B_t = \alpha I_t + (1-\alpha) B_{t-\Delta t}$$

Motion history detection has objective for increasing the robustness of moving object detection method. Instead of using just only the difference values between the current frame and background, the motion history of moving object over time is considered. Moving object detected at time *t* becomes history of its own at time *t+1*. Motion history image O_t of moving object will disappear from time-to-time which is represented by the multiplication by *h* (*h*<1) illustrated by the following equation.

$$O_t = \max(h \times O_{t-\Delta t}, O)$$

By applying the gradient method to motion history image, we can determine the direction of moving object. Then, motion segmentation can be done by classifying its direction. Figure 4 shows the summary diagram for motion history detection and segmentation.

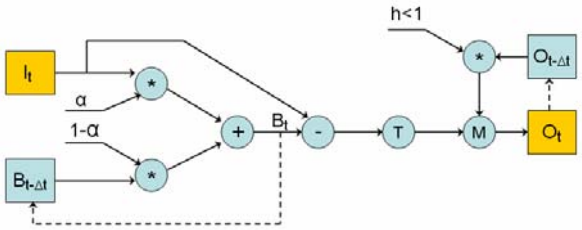


Figure4. Motion history detection and segmentation.

With the algorithm described above, we obtain position size and direction of moving objects which is necessary for testing our system. However, by defining inappropriate parameters, algorithm will give undesirable results. First remark, with *h* close to 1, we got a long history of moving object that finally leads to wrong detection for its size and position, but inversely improve the direction. And when *h* close to 0, we got opposite results. Second remark, the method will give result incorrect when moving object is not rigid. In this case, object can be perceived as multiple parts with different directions. For our specific case about pedestrian/vehicle detection, algorithm gives comfortable results, especially when it stays far from camera.

3.2 Motion Event Generator

Event Generator in this circumstance means that events will be raised systematically when motion occurs or disappears. These events are necessary for driving the other components of system such as

encoder and external alert system.

A serious problem about motion detection technique described above is that when object stop moving for a while, which occurs very often in real situation, object is undetectable and then will be immediately recovered. This situation will provide eventually unforeseen event separating two motions of the same object. We encounter the problem simply by introducing an event smoother which is actually implemented by a self-decreasing timer. Motion detection event invoked by detector for indicating stopping motion will not be raised immediately; the generator will wait for some seconds.

4. Video encoding and storage

Video storage component of our surveillance system is designed by default to store images sequence when only having motions on the scene. Encoder starts encoding images after capturing a motion event and stop when new event is raised.

4.1 Encoders

Like the other LPC components, the encoders can be developed (or using an existing one) and interfaced to the principal platform by adding some codes that act as plug-ins.

In our current version, Window Media Encoder SDK is used by default (a license windows can get it directly from Microsoft's website). The most important step for using WME is the description of encoding profiles. In general, the codecs for quality-based encoding or bit rate-based encoding needs to be explicitly specified in the profile. In our case, the Windows Media Audio and Video codecs version 9.0 is used for 100% quality-based encoding. This configuration gives the best video quality with less time consumption of encoding process, but needs more disk space for storage. The choice is made based on the fact that main CPU should be used further in the video analysis process.

4.2 Data Event Generator

Data Event Generator is a HPC utility of our platform to communicate with the external offline searching and playback application. After finishing the encoding process of images sequence of current motion, an event is immediately raised by encoder component that invoke the Data Event Generator. This HPC will send a metadata file 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 motion, algorithm's name and encoder type. This information will be used by searching and playback application, alert system.

There are two external systems that do not mention here in details are alerts system and searching/playback application. Both of them are very necessary for completing the global need of video surveillance system.

6. Performance and discussion

The test of system is performed into two parts: component testing and global system testing. 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. The PC is located at COE department.

6.1 Acquisition and viewing

The test is done with some 47 cameras from Axis manufacturer (model 206) locating at PSU Wittayanusorn School, 7 cameras of Bosch Company at PSU entrance gates, 1 Panasonic camera at COE department, 1 D-Link IP-camera (install locally) and 2 USB webcam.

Firstly, the test is performed according to basic functional design, 16 cameras from different manufacturers are installed with 25-30 fps at 640x460 pixels, and then regrouping as view for connecting and displaying in real-time in the same view simultaneously. The system is executed during 48 hours continuously, no error found. The processes are repeated for a couple of works. By considering the maximum number of cameras viewing at the same time, we found that this is limited to around 16 cameras that may implicitly explain by quality of network connection (though internet) and bandwidth limit of testing PC. In fact, one image in jpeg format corresponds by average 74Kbyte, with 25-30 fps of 16 cameras; bandwidth is then bounded from 29.6 to 35.5 Mbyte.

6.2 Motion history detection/segmentation

The algorithm of motion detection is implemented in C and used OpenCV library. The test is performed under our system.

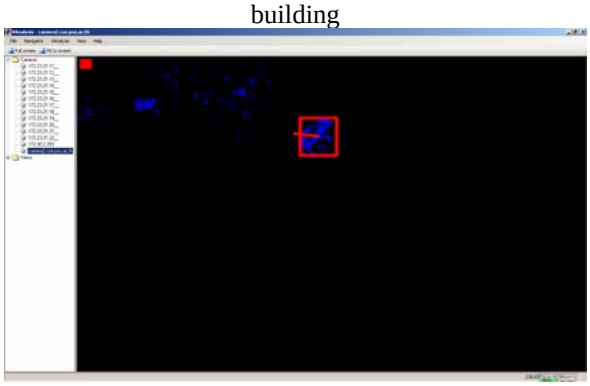
Figure 5.1 illustrates that our system is viewing the sequence of images from Panasonic IP-camera located at the first floor under robot building. Figure 5.2 shows some result of motion detection algorithm.

Like the remarks that I did on the section 3.1, the precision of method (for location, size and direction) is very sensitive to the parameter *h.* Moreover, method will give wrong result when the moving of non-rigid object is too close to the camera. However, it gives satisfied results if this configuration is determined appropriately.

6.2 Overall system testing

Overall system testing is done at first step with only one camera. Image acquisition from Panasonic IP-camera (at the first floor of robot building) is defined at 10 fps of image 640x480 pixels. System is executed continuously during 24 hours. By observing the results, we found that motions of people working in/out from robot building are quite well detected. However, some noises are also registered. We can recognize for two types: contrast/illumination change rapidly due to sunshine and non-static background problem (tree moving because of the winds).

Figure 5.1 Video viewing from camera at robot building



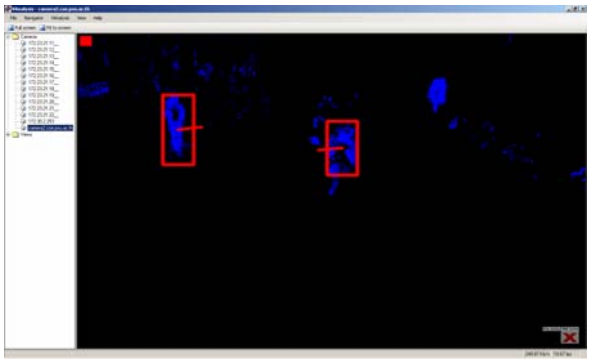


Figure 5.2 motion detection results, red rectangles indicate motions occurring in the area with directions, blue color represents motion history.

Second step, the configuration of testing is setup like before but increasing number of cameras up to 7 (1 Panasonic + 6 Bosch at PSU) and executing for one day. We found that, the acquisition component and video analysis are very well done. However, this test recovers another important problem is that the encoder is mal-functional after a triple hours later. It generates the error "object instance can't be created", and found that more cameras used, faster get the error. We have assumption that maybe there are too much encoder instances that try to write video images onto disk at the same time.

7. Conclusion

We proposed in this article a generic version of video surveillance system. System architecture with details is presented. Some preliminary tests of system show encourage results. Nevertheless, more extra debugging and testing has to be performed in order to improve the robustness that is very important for such system.

Future works, firstly, more complex algorithm of video content analysis for the specific application have to be explored. Secondly, the external alert system must be completed and tested together with this platform, and the same for searching/playback application that use results produced by system. All together the system may become a good choice as SME solution of video surveillance system

References

Conference

[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] 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.

[3] 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.

[4] 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.

[5] 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

[6] 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.

[7] J. Piater and J. Crowley, Multi-modal tracking of interacting targets using Gaussian approximations, Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance. IEEE Computer Society, Dec. 2001

Web

[8] Andrew Kirillov, Camera Vision - video surveillance on C#, http://www.codeproject.com/KB/audio-video/cameraviewer.aspx.