Scalable Video Transmission for a Surveillance System

Tomi Räty
VTT Technical Research Centre of Finland

Lassi Lehikoinen
VTT Technical Research Centre of Finland

Francois Bremond
Institut National de Recherche en Informatique et en Automatique

Abstract

The Area of Interest (AoI) is a distributed scalable video transmission subsystem, for a surveillance system, which concentrates on decrementing the amount of video information transmitted to the end-user equipped with a mobile device. The video information is processed by the Video Surveillance Intelligent Platform (VSIP) to discriminate the essential images of the indoor area under stationary video surveillance. The AoI system analyzes the output of the VSIP's images and eXtended Markup Language (XML) image information. The AoI system is able to define and extract the essential information, e.g., a tracked individual, and it transmits only this image to the end-user. First, the AoI transmits the entire image of the indoor area to the mobile device of the end-user. Then, the AoI system transmits only the secluded tracked objects' images to the mobile device. The end-user's device portrays the scaled portrait images of the targeted object on top of the background image. The AoI system endeavors to decrease the size of the video images transmitted to a smart phone over a wireless network and to retain the comprehension of a tracking situation. The operability of the constructed prototype indicates that this endeavor is attained. The research is based on the constructive method of the related publications and technologies and the results are derived by the implemented AoI system.

Key Words- Multimedia communication, cooperative information systems, intelligent sensors, mobile communication, and multimedia systems.

1. Introduction

Video surveillance has become a ubiquitous aspect of the modern urban landscape [1]. Video surveillance is a significant market [2]. The systems must be network-connected, entail multiple cameras, and the complete system has to be reliable and robust [2]. Video surveillance applications must be real-time, which entail low delay and timing constraints for processing [2]. Target detection and tracking is a fundamental technology to develop real-world computer vision systems [3].

The Area of Interest (AoI) system comprises of the AoI server and the AoI client. The AoI server resides in a desktop and the AoI client resides in a surveillance personnel's mobile device. The AoI server utilizes images and eXtended Markup Language (XML) files, containing image information, that are received from Video Surveillance Intelligent Platform (VSIP). The images are snap-shots from a stationary camera of a surveyed indoor area. The XML files contain information about the tracked entities of the surveyed indoor area. This information includes the location of the tracked entity on the snap-shot image. During the initial transmission from the AoI server to the AoI client, one entire image of the surveyed indoor area is transmitted. This image is utilized as a background image by the AoI client and it is displayed on the security personnel's end-device. After the first image transmission, the AoI server distinguishes the tracked image from each image received from the VSIP. The AoI server extracts the tracked object from each image and the extracted image is transmitted to the AoI client. Upon reception of an extracted image, the AoI client displays the extracted image on the correct location, i.e., where the tracked object actually resides, of the background image. This procedure of extracting the tracked object by the AoI server, transmitting it to the AoI client, and displaying the extracted image at the correct location of the background image is conducted until the AoI system is shut down. By extracting the tracked object from the image and forming an extracted image decreases the amount of information required to be transmitted to the end-device.

The intent of the AoI system is to ultimately abate the quantity of information required to be transmitted to the security personnel while retaining all the required information for the security personnel to be fully cognizant about the surveyed indoor area. The operability of the constructed AoI system prototype indicates that this endeavor is attained.

The structure of this paper is the following. First a general overview of contemporary video monitoring is presented. Then a concise presentation of the VSIP is rendered. This is followed by a presentation of the implemented AoI system, containing detailed information regarding the structure of the AoI system and the

transmission paradigm of the AoI system. The conclusion denotes the image subsidization in examples and summarizes the paper.

2. Video monitoring

Video monitoring typically deploys multiple video cameras, channeling video signals to a central monitoring room, where multiplexing is utilized to render a subset of the images to security personnel [1]. Modern video-based surveillance systems utilize real-time image analysis techniques for efficacious image transmission and event-based attention focusing [4].

Object tracking is an essential task for many applications in the region of video surveillance. Every detected object is tracked and their trajectories are analyzed to derive their movement in the scene. Detected objects are recognized and their behavior is analyzed to verify if state is potentially dangerous or normal. [5]

2.2. Situation awareness

The key factor to security is situation awareness, which requires information and spans multiple scales of space and time [6]. Multi-scale techniques evoke a completely novel region of research, in addition to challenges in performance modeling and evaluation [6]. Visual surveillance in dynamic scenes endeavors to detect and track certain objects from image sequences, and to understand object behaviors [7]. The goal of visual surveillance is to achieve the plenary surveillance task as automatically as possible [7].

2.3. Middleware

Enabling a group of video surveillance algorithms to cooperate in the monitoring of a large surveillance network presents substantial challenges [8]. Middleware can assist with the general aspects of video surveillance network construction, containing support for communication and computation [8]. The drawbacks in many current video surveillance systems contain lower Quality-of-Service (QoS) in video transmission. [9]

Researchers concentrate mainly on the vicissitude of content comprehension, e.g., detecting and tracking. They have not heeded the scalability of video surveillance systems. They typically utilize a centralized architecture and posit the required system resources. [10]

Digital surveillance systems disclose restrictions regarding delay and visual quality that pose demands on the video codec. Flexible composition of the compressed video data is required. In a large surveillance system, the digital network enables interconnected LANs with distinct bandwidths and QoS. [11]

3. Video Surveillance Intelligent Platform

A demanding problem in the domain of computer vision and artificial intelligence is video comprehension. The first step utilizes typically extensive usage of methods for data analysis while the second step conducts structural analysis of the symbolic data collected at the antecedent step, as Figure 1 illustrates. [12]

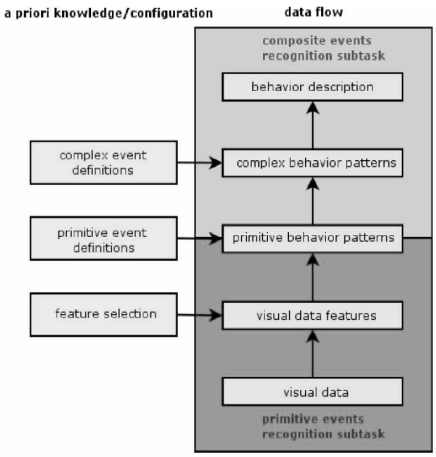

Figure 1. A generic architecture of a video comprehension system. [12]

This approach is available as a platform for image sequence comprehension named VSIP. VSIP has been developed by the research group ORION at INRIA (Institut National de Recherche en Informatique et en Automatique), Sophia Antipolis. VSIP is a generic environment for amalgamating algorithms for processing and analysis of videos which enables to combine and exchange miscellaneous techniques at different stages of the video comprehension process. VSIP is oriented to assist developers depicting their own scenarios and systems capable of monitoring behaviors, dedicated to specific applications. [12]

VSIP elicits primitive geometric features, such as areas of motion. Objects are then recognized and tracked. At the second level the events, in which the detected objects participate, are discriminated. To perform this task, an event description language is used. [12]

4. Area of Interest system

The intention of the AoI (Area of Interest) system is to transmit merely the important dynamic video image information, gathered from a surveillance point by a stationary camera, to a mobile device. The static information, a.k.a. the environment background of the video images, is transmitted only once during inception. The system sequesters the essential objects from .jpg images and sends the separated objects to a mobile device. In the Figure 2, a moving object is traced from a

surveillance camera perspective. The traced object is extracted from the background as depicted in the Figure 3. Finally, the traced object and its deployment pixel coordinates, meaning the coordinates of the traced object in the whole image, are transmitted to a mobile device which displays it at the right location on the screen. The end-device view of the extracted image placed on the background image is presented in Figure 4. When the AoI system is executing, the outcome at the mobile device is a continuous video stream in which only the traced dynamic areas are received from the server via a network connection and merged into the background. The mobile device is the Nokia N95 phone, equipped with S60 3rd Edition SDK for Symbian OS 9.2, Supporting Feature Pack 1. The server's software is implemented in Windows XP OS utilizing Microsoft .NET Framework SDK v2.0.

Figure 2. A traced dynamic object segregated by a square.

Figure 3. A traced dynamic object extracted from the static background.

Figure 4. View from the end-device. The object images are placed on the background image.

4.1. The server structure of the AoI system

The software components of the AoI server are delineated in Figure 5. The main components are the following: 1) AoIMain, 2) ImageSender, 3) XMLParser and 4) ImageProcessor. The AoIMain component is the main executable, which has the responsibility of controlling all the components. The ImageSender component is employed for establishing TCP/IP socket communication with AoI client(s) and sending images through the connection. The XMLParser component is responsible for parsing the .xml file containing object tracing information. The ImageProcessor component has the responsibility of separating traced objects from the background environment. The separation is exerted with the .jpg images.

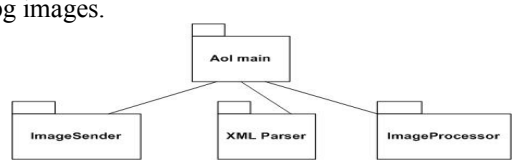

Figure 5. The component diagram of the AoI server.

After the activation of the AoI server, initialization procedures are executed. All the appropriate class instances are created and a listening socket is established for a remote client connection. Once a client has connected to the server, the server's 'Start' method is called and the main functionalities begin. If the server is transmitting objects for the first time, the environment background image is sent by the server. The server parses the frame elements from the .xml file, which contains the object tracing data. After that, the image objects are respectively separated from an image file. Next, the separated objects are transmitted to the client through the socket connection. This procedure is repeated until the .xml file is processed completely. When the file is processed to the end, the AoI server is suspended and de-initialized.

4.2. The client structure of the AoI system

The software components of the AoI client are rendered in Figure 6. The main components are the ensuing: 1) UI, 2) ImageConverter, 3) ImageViewer and 4) SocketCommunicator. The UI component is used for user interactions. Additionally, the UI component controls the ImageConverter, ImageViewer and SocketCommunicator. The ImageConverter converts the 8-bit image descriptors, received from the server, into bitmaps which are displayable on the end-device's screen. The ImageViewer component is responsible for joining the bitmaps of the separated image objects and the background environment bitmap into one merged view, which may be displayed on the screen of the end-device. The SocketCommunicator component is employed in receiving images and control messages from the AoI server via TCP/IPv4 socket connection.

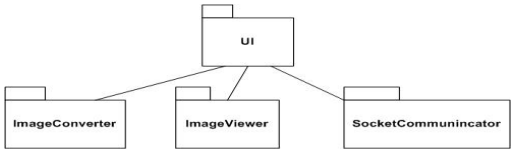

Figure 6. The component diagram of the AoI client.

After the launch of the AoI client application, initialization procedures are exerted. All the appropriate class instances are created and a socket connection to the AoI server is formed. Once the client has connected to the server, it receives a background image and an instruction message indicating if the background was successfully transmitted. Then the background is displayed on the screen. Next, the client receives separated image objects and coordinate instruction messages from the server. The separated images are displayed on top of the background image. When the client receives an instruction message indicating the frame was sent successfully, the client clears the screen from the previous separated objects, and the client prints the background image again to the screen. If a human user presses the "Exit" button from the user interface, the application exits, de-initializes and shuts down.

4.3. The AoI server's main execution sequence

The AoI server reads the .xml file, separates traced dynamic objects from the images and transmits objects to the AoI client. See Figure 3 for an example of a transmitted object. As preconditions to the AoI server execution sequence, the AoI client has already connected to the server and the AoI server has XML file with the object tracing data and the corresponding .jpg image files.
Description of sequence's events, illustrated in Figure 7:
1. The Start() function is called when an AoI client has connected to the server.
1.1 The OpenXMLFile() function opens the .xml file for reading operations.
1.2 The ParseNextXMLFrameContent() function parses next frame element from the .xml file and stores the content to a CXMLFrameContent object.
1.2.1 The CXMLFrameContent object reference is returned.
1.3 The SeparateObjectsFromImage() function is called. The function secludes traced dynamic objects from a .jpg image. The function receives the CXMLFrameContent object reference as a function parameter.
1.4 The SendObjectsToClient() function transmits separated dynamic objects images to the AoI client. Additionally, after transmitting an image, the server transmits a deployment coordinate instruction to the client. When all the objects and coordinate instructions are transmitted, the server sends an instruction to the client denoting that all the objects of the frame are sent.

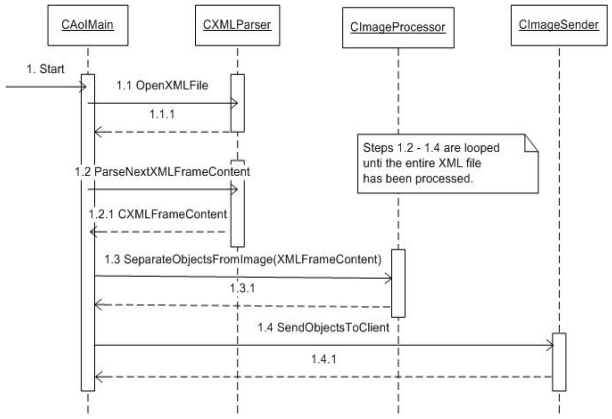

Figure 7. The AoI server execution sequence diagram.

The outputs of the AoI server are the image files of the separated dynamic objects. The separated image files are deleted from the AoI server's file system after the sending has executed successfully. There are two notable exceptions regarding: 1) an addition to the step 1.4, when transmitting image objects for the first time to the AoI client, the background environment image and a notification instruction of successful sending are sent. After the first time, the background is not sent; and 2) if the .xml file in not processed completely, sequence restarts from the step 1.2 after the execution of step 1.4. If the .xml file is processed entirely, Start() method returns.

4.4. The AoI server's main execution sequence

The AoI client receives TCP packets from the AoI server. The packets are parsed and addressed appropriately. First, a background image is received. Then an arbitrary amount of separated images are received. In the steps 1 – 2.1.2, a background image is received from the AoI server. These steps are performed only once. In the steps 3 – 4.1.2, the separated image objects are received from the server with appropriate coordinate instruction messages. These steps are looped until the AoI server stops, i.e., the xml file has been read to the end. As a precondition to the AoI client execution sequence, the AoI server is running.

Description of sequence's events, illustrated in Figure 8:
1. The MessageReceived() callback is called by the CCommunicator class instance when a message is received from the connected socket. When receiving messages for the first time, the content of the message is added to the background image buffer. Since the background image can be large, the MessageReceived() function is called often before the whole image is completely in the image buffer. The AoI server transmits a notification when the background image is successfully sent to a client. Upon reception of the notification, step 1.1 is executed.

1.1 The ConvertDesL() function is called to convert the serialized 8-bit background image buffer into bitmap image which can be displayed on the devices screen.
2. The ConversionComplete() function is called after the image conversion is ready.
2.1 The SetBitmap() function sets the converted bitmap for the CAoIAppView object.
2.1.1 The Draw() function is called. It displays the converted bitmap, in this case the background image, on the end-device's screen. Now the background is displayed successfully on the screen. Then the server begins to send the separated image objects.
3. The MessageReceived() callback is called by CCommunicator class instance. It is called as many times as required until the whole separated image is stored into the image buffer. When the AoI server has transmitted the whole separated image, it transmits a coordinate instruction message to a client. This message contains the deployment coordinates for the separated images and the image length for a validation check at the client side.
3.1 The ParseXCoordinate() is called to parse the X coordinate from the coordinate instruction message. This value is stored into a variable.
3.2 The ParseYCoordinate() is called to parse the Y coordinate from the coordinate instruction message. This value is stored into a variable.
3.3 The ParseImageLength() is called to parse the image length from the coordinate instruction message. This value is stored into a variable.
3.4 The SetImageTopLeftX() function is called to set the X coordinate value for the CAoIAppView class instance.
3.5 The SetImageTopLeftY() function is called to set the Y coordinate value for the CAoIAppView class instance.
3.6 The ConvertDesL() function is called to convert the serialized 8-bit image buffer into a bitmap image which can be displayed on the devices screen.
4. The ConversionComplete() function is called after the image conversion is ready.
4.1 The SetBitmap() function can be called now, when the coordinates of the separated image has set to the CAoIAppView class instance.
4.1.1 The Draw() function is called. It draws the separated image object at the correct coordinates. The image is added onto the background image.

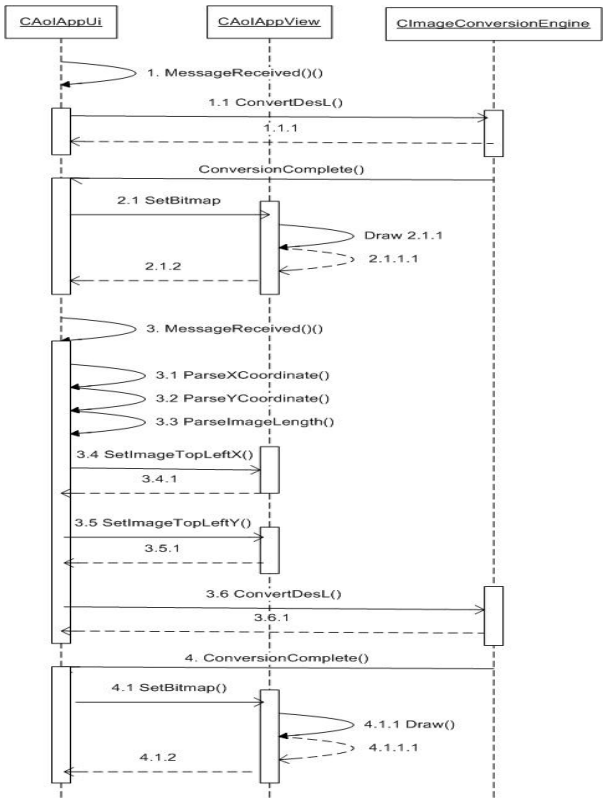

Figure 8. The AoI client execution sequence diagram.

There are two notable exceptions regarding: 1) if the AoI server is not running, then the client initialization fails and application does not start; and 2) in step 3, if the AoI client receives an instruction message indicating that a whole frame content has been transmitted, the background image is printed on the display again before adding the separated objects into it. In this manner, the screen is "cleared" from the previous separated objects.

5. Conclusion

Video surveillance is an important branch in the field of surveillance. With the utilization of advanced video surveillance tools, such as VSIP, it is possible to distinguish images of tracked objects. By abating the amount of image information that needs to be transferred, the images can be transmitted faster to the end users, e.g., surveillance personnel. We have illustrated the implemented design and communication how this endeavor is attained with the AoI system.

The information and structure of the AoI system was modeled on recent journals and conference papers regarding video surveillance and monitoring. There are theories demanding real-time reactivity, low delay and timing constraints from Desurmont et al. and the importance of situation awareness according to Hampapur

et al. which includes challenges in performance modeling and evaluation. The AoI system attempts to reduce the real-time challenges by subsiding the amount of image information that needs to be transmitted. May et al. deem that there is a requirement for the flexible composition for the compressed video data, the AoI system decreases the amount of image information transmitted. The AoI applies to the May et al.'s restrictions of surveillance applications regarding delay, complexity, security, visual quality and QoS predicaments by scaling the image size. This also applies to the drawbacks of lower QoS in video transmission declared by Yan et al. Korshunov et al. state that enough research hasn't been contributed on scalability of video surveillance systems. These typically utilize a centralized architecture and posit availability of all the required system resources, such as computational power and network bandwidth. The AoI system endeavors to transmit as little image information as possible while retaining the quality of the prominent image information.

The AoI system comprises of the AoI server and the AoI client. The AoI server processes the images received from the VSIP tool and accompanied with the VSIP tool's XML an extraction of the tracked object is performed. The AoI server transmits the images of the tracked objects to the AoI client. The AoI client receives the entire background indoor image in the first transmission from the AoI server. After the first transmission, the AoI server only transmits the images of the tracked objects. The AoI client updates the tracked object image onto the initially received background image. The image sizes of the transmitted tracked objects are subsided in comparison to their entire and original image sizes. For instance, the entire size of Frame003 is 21 814 bytes and the size of the images containing the extracted objects are 924 and 690 bytes. In Frame103, the size of the complete image is 21 834 bytes and the size of the image containing the extracted objects is 624 bytes. In Frame186, the size of the complete image is 21 994 bytes and the size of the images containing the extracted objects are 498, 644, and 736 bytes respectively.

The intent of the AoI system is to ultimately subside the quantity of information required to transmit the security personnel while retaining all the required information for the security personnel to be fully aware about the surveyed indoor area. The operability of the constructed AoI system prototype indicates that this endeavor is attained.

6. Acknowledgement

We would like to sincerely thank the EU-CAVIAR project for the utilization of the images and XML files that were employed in this publication.

12. References

[1] Makris, D. and Ellis, T.: Learning Semantic Sense Models from Observing Activity in Visual Surveillance, *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 35, No. 3, June 2005, pp. 397-408.

[2] Desurmont, X., Bastide, A., Chaudy, C., Parisot, C., Delaigle, J.F., and Macq, B.: Image analysis architectures and techniques for intelligent surveillance systems, *IEE Proc.-Vis. Image Signal Process.*, Vol. 152, No. 2, April 2005, pp. 224-231.

[3] Matsuyama, T. and Ukita, N.: Real-Time Multitarget Tracking by a Cooperative Distributed Vision System, *Proceedings of the IEEE*, Vol. 90, No. 7, July 2002, pp. 1136-1150.

[4] Foresti, G.L., Micheloni, C., Snidaro, L., Remagnino, P., and Ellis, T.: Active Video-Based Surveillance System, *IEEE Signal Processing Magazine*, March 2005, pp. 25-37.

[5] Micheloni, C., Foresti, G.L., and Snidaro, L.: A Network of Co-operative Cameras for Visual Surveillance, *IEE Proc.-Vis. Image Signal Process.*, Vol. 152, No. 2, April 2005, pp. 205-212.

[6] Hampapur, A., Brown, L., Connell, J., Ekin, A., Haas, N., Lu, M., Merkl, H., Pankanti, S., Senior, A., Shu, C.-F., and Tian, Y.L.: Smart Video Surveillance, *IEEE Signal Processing Magazine*, March 2005, pp. 38-51.

[7] Hu, W., Tan, T., Wang, L., and Maybank, S.: A Survey on Visual Surveillance of Object Motion and Behaviors, *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, Vol. 34, No. 3, August 2004, pp. 334-352.

[8] Detmold, H., Dick, A., Falkner, K., Munro, D.S., van den Hengel, A., and Morrison, R.: Middleware for Video Surveillance Networks, MidSens'06, November 27-December 1, 2006, Melbourne, Australia.

[9] Yang, H., Xie, L., and Xie, F.: Research on Cluster Remote Video Surveillance System, 2006 IEEE International Conference on Industrial Informatics.

[10] Korshunov, P. & Ooi, W.T.: Critical Video Quality for Distributed Automated Video Surveillance, MM'05, November 6-11, 2007, Singapore.

[11] May, A., Teh, J., Hobson, P., Ziliani, F., and Reichel, J.: Scalable Video Requirements for Surveillance Systems, 2004, The Institution of Electrical Engineers, printed and published by the IEE, Michael Faraday House, Six Hills Way, Stevenage, SG1, 2AY.

[12] Bremond, F., Thonnat, M., and Zuniga, M.: Video Understanding Framework For Automatic Behaviour Recognition, Behaviour Research Methods, Volume 28, Number 3, August 2006.