Real-time Video Surveillance for Large Scenes

Hanyu Liu, Chong Tang, and Shaoen Wu

School of Computing, University of Southern Mississippi, Hattiesburg, MS 39401, USA

{ hanyu.liu, chong.tang, shaoen.wu }@usm.edu

1. Introduction

With the advances in networking technologies, more and more Internet protocol (IP) cameras are being adopted to support a variety of requirements and applications, such as security surveillance, traffic control [1] and online courses [2]. Compared to the traditional analog closed circuit television (CCTV) cameras, IP cameras can exchange data via wired or wireless computer networks. Therefore, the captured real-time video can be easily shared with remote clients. However, constrained by optical technologies and costs, cameras can only provide limited angles of view, namely covers only partial of a scene, especially when it is large.

A widely used solution to the above limitation is to deploy multiple IP cameras and generate the whole scene in separate displays or viewing windows in a large display. This intuitive solution stems from the traditional CCTV surveillance system, which connects all cameras to a central control where there are a number of displays [3]. However, two problems exist in this solution. The first is the difficulty of synchronizing the frames from different cameras on a packet-switching network [4]. The other is that the separation in display of views from different cameras provides offensive user experience and undermines the exploitation of useful detail information from those cameras [5] [6].

In this article, we briefly present a novel framework, which takes the advantages of recent advances in Computer Science and other disciplines, to address the problems mentioned above. In the rest of the article, we first describe the architecture of the system. Then, discussed is the hierarchical real-time video stitching. Finally, a novel working set oriented user interface is presented, which provides end users a more friendly and efficient way to accomplish their work with multiple IP cameras.

2. Environment

The distributed system is depicted in Figure 1, Panasonic residential IP network cameras, BL-C210, are used to capture video. The cameras support H.264, MPEG-4 and Motion JPEG compression format and can stream compressed video via RTSP [7], a popular protocol used for establishing and controlling media sessions. The embedded device consists of an x86 based AMD embedded CPU and the PC has an Intel i3 processor with 4G RAM. All those devices are connected via gigabit Ethernet or 802.11n Wi-Fi wireless network.

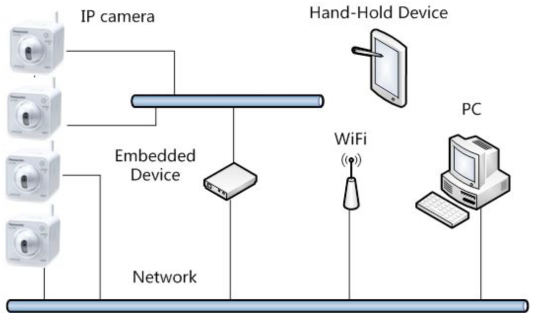


Figure 1. The System Architecture. All IP cameras are connected to Internet. The Embedded Device preforms stitching remotely and streaming to end users.

3. Video Stitching

To provide a panorama view of a monitored scene, we introduce a Computer Vision technique, video stitching. This technique exploits the image stitching that has been investigated for decades [8] and a number of software have been released [9]. With video stitching, the video frames from different cameras covering an entire scene are combined together. In addition to the details from each camera, the user can have a panorama video of these cameras, which could support a better overall understanding the entire target scene. We implemented the video stitching based on homography [10], a concept wildly used in

Computer Vision that can be viewed as an invertible transformation from one projective plane to another. Denote p_a as a point p in frame (or projective plane) F_a and p_b in frame F_b. The transformation can be described as follows:

$$p_a = K_a \cdot H_{ba} \cdot K_b^{-1} \cdot p_b$$

Where H_{ba} is the homography mapping matrix. K_a and K_b are the intrinsic parameter matrices of the cameras. Before calculating remapping matrix, we apply Scale-Invariant Feature Transform (SIFT) [11] to extract associating points and Random Sample Consensus (RANSAC) [12] to eliminate outliners.

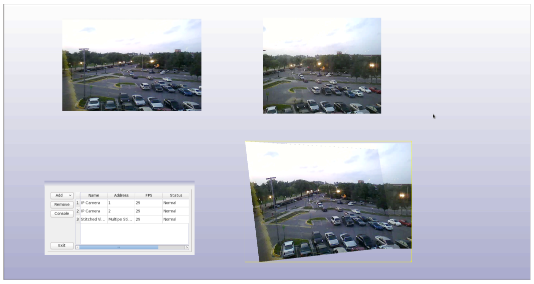
To address the synchronization problem among videos from multiple cameras in complicated networking environments, we design the video stitching system with hierarchical implementation and optimization. In some cases where the networking topology is simple and stable, the videos streamed from the IP cameras can be directly fetched by the computer and then stitched for further use. Otherwise, in the cases where frame synchronization from the IP cameras cannot be guaranteed, or the target mission is critical, the embedded device is topologically deployed nearby the related cameras. Then the video is stitched first at the front-end embedded device and then delivered to the back-end users.

4. Working Set Oriented User Interface

As each camera could only capture a small portion of the whole target scene, it is crucial develop a novel user interface to assist a user to manage the cameras and accelerate the processing of those captured videos. The traditional grid TV wall lacks scalability and capability to customize. We present a novel user interface that encapsulates basic video displaying and camera managing functions into an "item" [13] for each camera. An example of the interface is shown in Figure 2. Each "item" can be freely placed, resized and grouped on top of a flat panel. The user can easily build up a highly customized, full functional user interface to meet different requirements. If the user wants to stitch two videos, s/he can simply drag one "item" to the other with mouse or figures on touch screens. In addition, our user interface is also adaptive to different sizes of display. For large and high-resolution display, multiple "items" can be added and stitched. On the other hand, for small displays or hand hold devices, one "item" can be initialized and support basic video displaying function.

5. Conclusion and Future Works

In our preliminary test, our video stitching on the embedded device works an average of 12 FPS output with two 640x480 video inputs. The user interface can easily boost a 30 FPS output with four 640x480 video inputs via RTSP on a typical IPv4 network. In addition, we conducted a small-scale pilot test and simulated both online course and parking area surveillance scenarios; all test takers respond positively. In the next step, it would be interesting and meaningful to test the performance in practice and collect formal reposes from professional users.



	Name	Address	FPS	Status
1	IP Camera	1	29	Normal
2	IP Camera	2	29	Normal
3	Stitched Vi...	Multipe Sti...	29	Normal

Figure 2. User interface with a floating control panel and three video "items". The bottom-right item is generated from top two. The user could freely manipulate and manage those items.

References

[1]. Industrial Video & Control (IV&C) official website, http://www.ivcco.com.

[2]. Netcom Learning official website, http://www.netcomlearning.com.

[3]. CCTV SecurityPros official website, http://www.cctvsecuritypros.com.

[4]. James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach (5th Edition), Addison Wesley, 2009.

[5]. Trouvain, B, Schlick, C. “A study of audio and visual context switch indicators in a mutirobot navigation task”, IEEE International Conference on System, Man and Cybernetics, 2004.

[6]. Cinvertino, G.; Chen, J.; Yost, B.; Ryu, Y. and North, C. “Exploring context switching and cognition in dual-view coordinated visualizations”, International Conference of Coordinated and Multiple Views in Exploratory Visualization, 2003.

[7]. RFC 2326, Real Time Streaming Protocol, IETF, 1998.

[8]. Szeliski, Richard, “Image Alignment and Stitching”, 2005.

[9]. Hugin official site, http://hugin.sourceforge.net.

[10]. Richard Hartley and Andrew Zisserman, “Multiple View Geometry in Computer Vision”, Cambridge University Press, 2003.

[11]. Lowe, David G. “Object recognition from local scale-invariant features”, Proceedings of the International Conference of Computer Vision, 1999.

[12]. Martin A. Fischler and Robert C. Bolles, “Random Smaple Consensus”, Comm. Of the ACM, 1981.

[13]. Andrew Bragdon, Robert Zeleznik, Steven P. Reiss, Suman Karumuri, William Cheung, Joshua Kaplan, Christipher Coleman, Ferdi ANdepitra, and Jeseph J. LaViola Jr. “Code Bubbles, A Working Set-based Interface for Code Understanding and Maintanence”, International Conference on Human Factors in Computing System, 2010.