A flexible multi-server platform for distributed video information processing

Yao Wang[1, 2], Linmi Tao[1], Qiang Liu[2], Yanjun zhao[1], Guangyou Xu[1]
[1] Department of Computer Science, Tsinghua University
Beijing 100084, China
[2] School of Software, Tsinghua University,
Beijing 100084, China
yao-wang04@mails.tsinghua.edu.cn linmi@tsinghua.edu.cn

Abstract. The complexities of advanced computer vision systems call for an infrastructure which is capable of integrating various computer vision algorithms into a working system with high flexibility. The distributed cameras and the need for real-time applications in current systems both lead to a distributed architecture. This paper presents a platform which acts as a framework for video information processing and analysis applications to plug in. The platform is composed by a set of servers which collaborate with each other to accomplish the tasks like video capture, transmission, buffering and synchronization. A set of inherit classes are designed for the sake of simplicity in sharing data among applications through sockets. We also suggested an approach to evaluate the delay of the platform. The platform is now running as a part of our On-the-Spot archiving system, and the results show that this platform can support up to 5 cameras system for real-time video information processing running on dual Xeon computers. As the platform is flexible, it can support the development of various application systems.

Keywords: multi-server, on-line video content analysis, middleware, system integration.

1 Introduction

Intelligent computer vision systems are getting more and more complex such as the systems presented in [2, 3]. The systems include not only distributed array of cameras that offer wide area monitoring, but also a set of computer vision algorithms designed for scene analysis at multiple levels of abstraction [1]. The computer vision algorithms in such systems can be used to derive indexed metadata for the large amount of raw video data generated by the camera networks, or to create user-aware services [4]. Though there are many video content analysis and scene understanding algorithms developed in laboratories, they are generally designed for some specific applications and for operating in single machine environment. Therefore, to integrate diversiform algorithms into a working system and deploy them into a distributed environment are the urgent needs in the development of advanced computer vision systems which calls for a platform to support the distributed information processing and algorithms' integration.

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

Some previous works have presented a few architectures or middleware for the development and implementation of computer vision systems. For example, Tsinghua's SISS [5] presented a platform to help agents' management and inter-agent collaboration in a multi-agent system. It is designed for pervasive computing and do not meet some of the requirements of video-based information analysis systems. USC's MFSM [6] is a middleware which supports efficient real-time media data processing and user immersion. It didn't pay enough attention to issues on the distributed environment. NIST's smart data flow system [7] is a middleware that supports sensor data transmission in distributed environment, but it didn't explore the problems with the whole lifetime of video streams. Hereby, we present a server-based platform to enable distributed, scalable, easy-access video information processing. The following of this paper are organized as this, section 2 describe our basic idea of the system architecture. Section 3 is some detail design aspects of the platform. Section 4 is the experiments and some results, followed by section 5, the conclusion and future work.

2 Overview of system architecture

An advanced computer vision system such as an intelligent video surveillance system generally includes video data capture, transmission, analysis, storage and retrieval [8]. These different parts form a pipeline of the video stream processing which requires a common platform. However, most researches only focus on the analysis part of the whole pipeline. In this regard, our idea is to provide a ready-for-use framework. The framework can run in minimal configuration which is mainly a video record system that collects cameras' image data, compresses it and stores in files for retrieving. For building an intelligent computer vision system, the platform also provides an easy-use interface for high level video analysis applications to plug in and work without much modification. We provide two access points which correspond to on-line and off-line processing. For an on-line application, the video frames are directly captured form cameras, and for an off-line application, the video frames are retrieved from storage files. The applications use a query like methods to get video frames, and to get metadata from other applications by means of callback. All metadata is synchronized to video frames and can be used as indexes of video streams to provide more efficient search as well as more efficient storage.

Figure 1 shows an overview of the system. The software architecture can be divided into a server domain and an application domain. In the server domain, there are various type of servers connected and communicate with each other. This is the main part of the platform. In the application domain, algorithms are treated as independent module applications and communicate with the server through sockets. This separation of server domain and application domain brings two benefits. First, it divides the tasks of video analysis from the other front and end tasks such as data fetching and result record. Now the data transmission of whole system can be managed by the platform and be transparent to the applications. Second, the server and applications are designed as separate processes and communicated by standard socket. This means that applications can be written in any language and the crash of

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

one application will not result in the brake down of other applications or servers. This is important especially in prototype stage when the algorithms are not very robust and efficient.

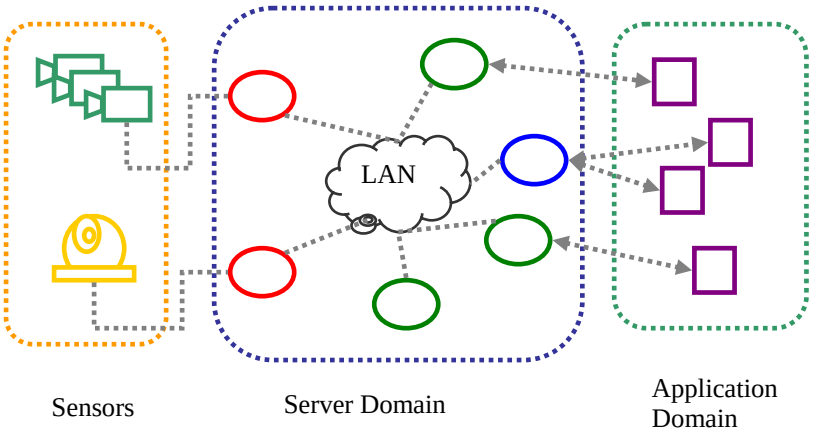


Fig. 1. A brief view of the platform's architecture. The server domain is comprised of various types of servers which act as agents to serve there local applications. There are three types of servers. The capture servers (red) have threads which interface with the sensors (cameras) and distribute sensor data to other servers for analysis. It also records the original data files for retrieval. The process servers (green) buffer the sensor data and provide scalable video frames according to the application requirements. The archive server (blue) holds a full connection with the other servers to collects metadata form analysis applications and fetch original data files for end users.

For a distributed system, the servers are allocated in different hosts. Each host runs one server to provide services for applications on the same host. On the other hand, the distributed environment is transparent to the applications. They just collaborate with its local server as if all other resources are local. In this regard, the server acts like an agent which serves local applications and collaborates or negotiates with other servers. We summarize the main requirements to the servers as follow:

1. Interface with the cameras and fetch the sensor data.
2. Negotiate with each other, locate the sensor data streams, then collect them and buffer them for applications.
3. Locate applications, route metadata for them as well as manage their local applications' run.
4. Compress and decompress the video data for transmission through network.
5. Manage the buffer pool, refresh it for real-time streams and fetch specified buffer for applications.
6. Provide context information to the applications so that they can run in a context-aware manner.
7. Provide necessary privacy enhancement for the system so as private data will not be abused. (This requirement is not discussed in the paper)

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

3 Specifications

3.1 Multi-thread servers

As the servers have to carry out several tasks and serve multiple applications, its structure is highly multi-threaded (as shown in figure 2). There are three main types of threads in the servers. A sensor data collecting thread runs in a loop to receive sensor data from data source (capture card, file, socket, etc.) and inserts it into a buffer pool. It works in push mode and discards oldest data in pool when new data comes. A sensor data consuming thread may be a decoding thread which decompress the video data and put into another frame buffer pool, or a data sending thread which sends the sensor data to other hosts. A client serve thread serves an application and fetches corresponding data from the buffer pools. The buffer pools are essentially queues shared by reader and writer threads. We utilize the reader-preference lock to synchronize the threads.

All servers are designed under this multi-thread model which gives the server good scalability in design. For example, when we add metadata sending and receiving function to the server, what have to do is just add new threads in it. Similarly, if we need sensor data recording, a new sensor data consuming thread can be integrated in the server conveniently.

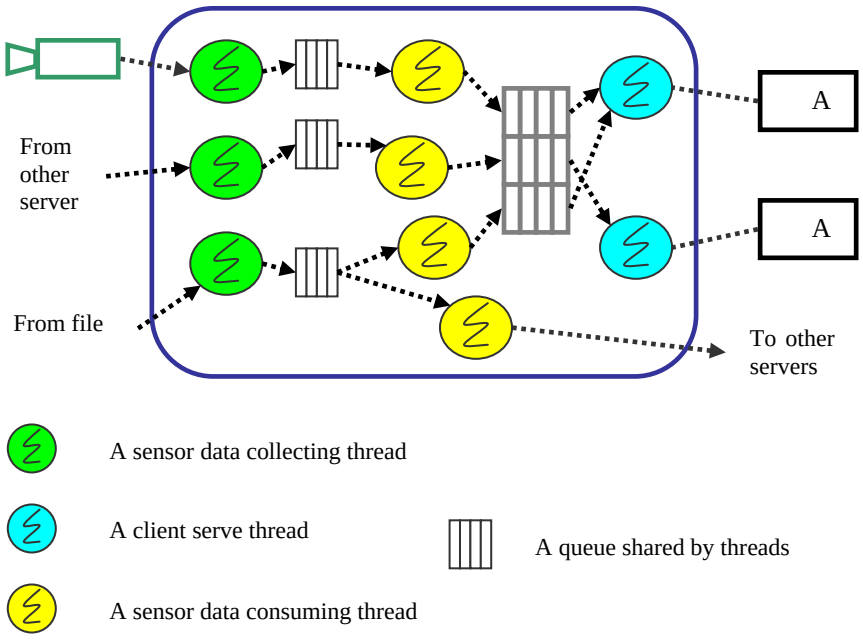


Fig. 2. The multi-thread structure of the servers.

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

3.2 Video data and metadata distribution

In out current system, the data distribution among different hosts is based on TCP/UDP. The UDP is adopted for video streams distribution and TCP is used for metadata transportation. We use MPEG-2 as our video codec and the metadata is constructed in a MPEG-7 compliant XML data structure. The TCP connections are built and maintained by servers. We are currently using a global configure file to enable the servers to find each other and locate the resource.

Since XML data structure is widely used in the platform for metadata sharing and application-server communications, we design a set of classes for the transparency in application programming. The inherit structure of the classes are specified as figure 3.

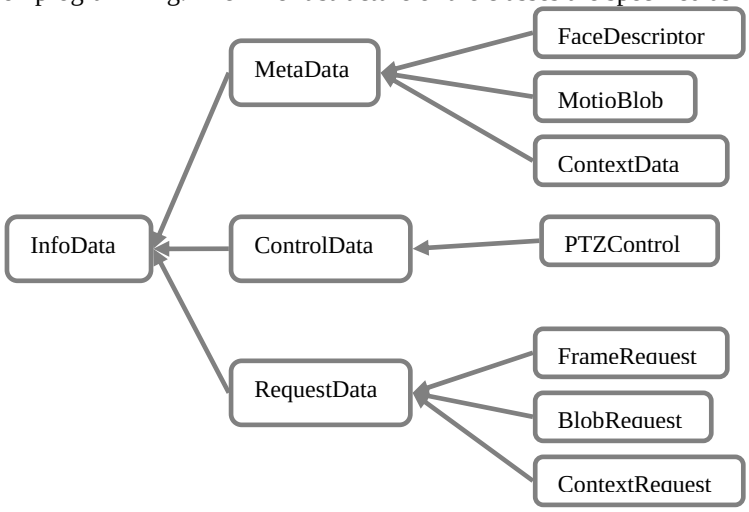


Fig. 3. The inherit structure of the assistant classes used for XML message generating and sharing. Other classes can be added under this structure due to new requirements.

3.3 Video scalability in application serves

Scalability is required when serve various applications. The scalability can be studied from three aspects, temporal, spatial and random access. [9] Every time an application requests a frame from the server, it creates an instance of the FrameRequest class (shown in figure 3), fills the properties and sends it to the server. The server then parses the request, pick matching data from buffer pools and do some transformation before send it to the application.

Table 1. The main properties of the FrameRequest class.

Property name	Property description
Camera ID	Describe the request video stream.
Frame interval	Describe the frame interval

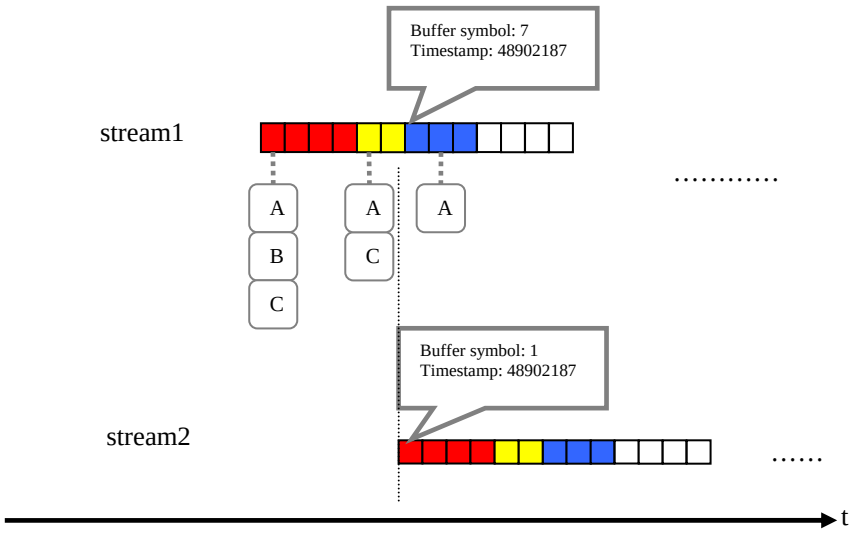
Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

	from last frame. This enables applications to get frames in different frame rate.
Frame size	Describe the request frame size.
Request region	Request a special region in the frame.
Frame type	Describe the frame type like grew level or color image.

3.4 Synchronization of streams

The distribute processing brings synchronization challenges which need to be solved before the platform can work. Here we give two examples to illustrate the challenges. In the first example, an application detects the moving area of every frame as its output. This output is collected and further recorded in an archive file in MPEG-7 format. For an off-line browsing and retrieval application, when an aim object is found, it has to locate the frames described in the archive file. In the second example, a data fusion application uses outputs of different applications to analysis the object's location. This means the fusion application has to synchronize different metadata derived form different camera data.

We use both time and buffer symbol to synchronize the streams and metadata. When a compressed buffer is obtained from a capture card, a buffer number and a timestamp will be added to the head of the buffer. To synchronize the video data with the metadata, the buffer symbol is adopted while to synchronize different video streams, the timestamp is utilized. An NTP (Network Time Protocol) server is established to ensure all capture servers to be synchronized to be synchronized with a common time source. The details are shown in figure 4.



Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

Fig. 4. Because the applications are not synchronized with each other, the frames in stream may be treated differently. As shown in the figure, the first frame is treated by application A, B and C. The second one is treated by application A and C. And the third frame is only treated by application A. Two streams which may start at different time. Each stream marks every buffer with a buffer symbol and a timestamp. The buffer symbol can not be used for cross stream synchronization, but can be used for single stream synchronization.

3.5 Storage and retrieval parts

In the platform, the media streams are stored in capture servers while they are generated. The files are divided into sections in constant time length and named after the start time and stream ID. The capture server also includes a thread which responses to other servers' requests and creates TCP data streams based on the storage files. Currently an application is built on the archive server to collect metadata from other servers and construct a MPEG-7 file which describes the video data. Another application is developed to provide a user interface which help the user efficiently use the archive files (figure 8).

```
<Descriptor xsl:type="MovingRegionType">
  <MediaLocator>
    <MediaUrl>Video\Cam1_48902187.mpg</MediaUrl>
  </MediaLocator>
  <TextAnnotation>
    <FreeTextAnnotation>Pedestrian</FreeTextAnnotation>
  </TextAnnotation>
  <Descriptor xsl:type="SpatioTemporalLocator">
    <FigureTrajectory type="rectangle">
      <MediaTime>
        <MediaTimePoint>T00:01:33</MediaTimePoint>
        <MediaDuration>PT1M10S</MediaDuration>
      </MediaTime>
      <Descriptor xsl:type="TemporalInterpolation">
        <WholeInterval>
          <MediaDuration>PT1M10S</MediaDuration>
        </WholeInterval>
        …
      </Descriptor>
    </FigureTrajectory>
  </Descriptor>
</Descriptor>
```

Fig. 5. A sample MPEG-7 code created by the retrieval application which store a moving object's trajectory and type.

4 Implementation and results

The servers were implemented with ACE [10] and the MPEG encode and decode part were using FFmpeg [11]. We currently apply our platform on the project named On-The-Spot Archiving system. It is an application system operated in the meeting

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

room environment and aims to create meeting archive in on-line. We use two computers run as capture hosts. One equipped with two capture cards and the other with three. Several other computers run as process servers and archive server. They are all connected to a hub through 10Mbps ports.

As latency is a key parameter for soft real-time applications. We design a method to test the latency form a physical event happens to video frames which describes this event reach an application. As shown in figure 6, an application is running on processing server which includes several threads. One thread creates black-white picture sequence on the LCD screen. This thread simulates an event (which is "screen become white" event) happens periodically and record the exact time when the event happens. A camera focuses the LCD screen, and the data is captured by a capture server and sent back to the process server. Other threads of the application acquire video frames from the server and check whether they are black or white. These threads simulate the applications which acquire video streams for analysis. When a white frame is detected by a thread, it also records the time. The difference between the show-thread's time record and the check-threads' time record is the total time interval of capture, compress, transmit and buffering. We use two capture servers to generate 5x1Mbps video streams. One of the streams is used as delay test stream. A third computer is used for the process server and tests the delay. The delay are recorded for seven times and shown in figure 7.

From figure 7, it can be found that on a single processor machine, the latency becomes large when four applications request the same stream. However, on a multi processor machine, the latency is acceptable even five applications request the same steam. This implies that when multiple applications running on the same host, the processing ability of CPU becomes the bottleneck. Allocating some applications to additional hosts could handle such situations.

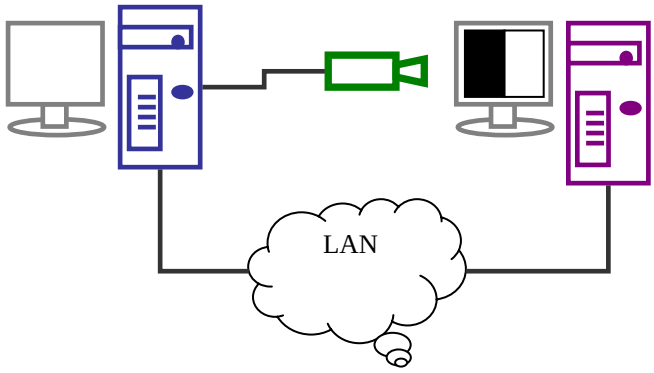


Fig. 6. Latency test. An application runs on the process server. It shows black-white frames and detects them to compute the latency. Black frame lasts for 2s while white frame lasts 40ms.

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

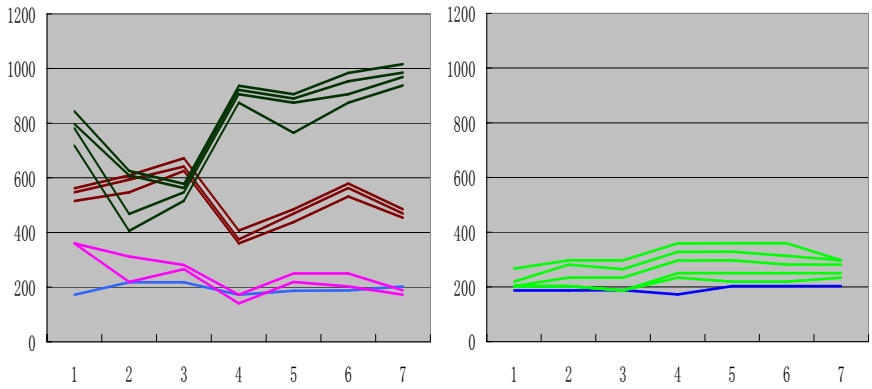


Fig. 7. Some results of the delay test. The left one shows the result of the test program running on an AMD Athlon 2000+ computer, and the right one is the result running on a dual Xeon 2.4GHz computer. Each colored line represents a thread which simulates an application running on the platform. For the left one, we test the delay of 1, 2, 3, and 4 threads. For the right one, we test the delay of 1 and 5 threads.

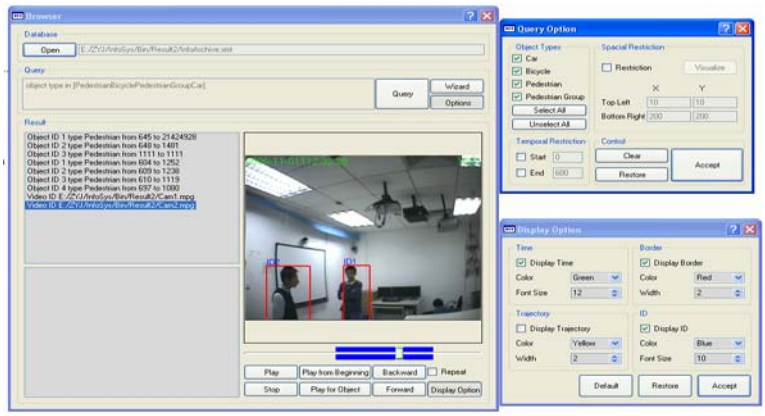


Fig. 8. The user interface of the off-line browsing and retrieval application.

5 Conclusion and future work

The distributed processing of the video streams improves the flexibility of a computer vision system. In this paper, we presented a distributed architecture for complex computer vision application systems. We use multi-servers to meet the common requirements from most software modules like collecting sensor data streams, routing metadata and sharing context information. Some module applications

Proceedings of the 5th International Conference on Computer Vision Systems (ICVS 2007)
Published in 2007 by Applied Computer Science Group, Bielefeld University, Germany, ISBN 978-3-00-020933-8
This document and other contributions archived and available at: http://biecoll.ub.uni-bielefeld.de

have been built on the platform. For example, a motion detection application has been developed on the platform. The application is originally designed for the project in an off-line environment, and it performs well on the platform for our On-The-Spot Archiving system.

Though the proposed architecture is capable to work well as mentioned above, the performance will be degraded when high resolution video streams are processed. We adopt 352*288 resolution, 1~2Mbps bit rate as our present video stream parameters. The decoding of one stream consumes approximately 20% CPU power on an AMD 2000+ computer. For a 640*480 streams, both transportation latency in a 10/100M LAN and the resource consumption of decoding become high. So the real-time performance becomes challenging.

Two approaches can be helpful to this problem. First, we leave the high resolution applications on the capture server and use the hardware decoder of the capture card. Second, adopt a scalable video codec such as MPEG4-SVC, and run the platform with high performance switches and multi-core CPUs.

Acknowledgments. This research is supported by NSFC project 60433030 and 60673189. We are thankful for the contributions of our colleagues Luo Sun, Liang Jiang and Jun Zhang to the platform. We are also thankful for the thoughtful comments and suggestions of our reviewers.

References

1. Valera, M. Velastin, S.A.: Intelligent distributed surveillance systems: a review. Vision, Image and Signal Processing, IEE Proceedings-Volume 152, Issue 2, 8 April 2005. P192-204
2. Mohan M. Trivedi, Tarak L. Gandhi, Kohsia S. Huang.: Distributed Interactive Video Arrays for Event Capture and Enhanced Situational Awareness. IEEE Intelligent Systems, Sep. 2005
3. Ellis, T.J. Black, J.: A multi-view surveillance system. IEE Colloquium (Digest), 2003 3-10062 59-63
4. Guangyou Xu, Linmi Tao, David Zhang and Yuanchun Shi.: Dual relations in physical and cyber space. Chinese Science Bulletin, 2006, 51(1): 121-128
5. Weikai Xie, Yuanchun Shi, Guanyou Xu, et al.: Smart Platform - a software infrastructure for Smart Space (SISS). Proceedings Fourth IEEE International Conference on Multimodal Interfaces, 429-34 2002
6. R. J. Francois, Gerard G. Medioni.: A modular middleware flow scheduling framework. Proceedings of the eighth ACM international conference on Multimedia (2000)
7. M. Michel, V. Stanford, O. Galibert.: Network transfer of control data: An application of the NIST SMART DATA FLOW. CCCT 2003 VOL, 2
8. D. Doermann, A. Karunanidhi, N. Parkeh, M. A. Khan, S. Chen, H. T. Ozdemir, M. Miwa, K. C. Lee.: Issues in the transmission, analysis, storage and retrieval of surveillance video. Proceedings of the 2003 International Conference on Multimedia and Expo (2003)
9. Ziliani, F.: The importance of 'scalability' in video surveillance architectures. Imaging for Crime Detection and Prevention, 2005, P29-32
10. Stephen D. Huston, James CE Johnson, Umar Syyid.: ACE Programmer's Guide, The: Practical Design Patterns for Network and Systems Programming
11. http://ffmpeg.mplayerhq.hu