A Scalable Distributed Architecture for Intelligent Vision System

Guojian Wang, *Member, IEEE*, Linmi Tao, Huijun Di, Xiyong Ye, and Yuanchun Shi, *Senior Member, IEEE*

***Abstract*—The complexity of intelligent computer vision systems demands novel system architectures that are capable of integrating various computer vision algorithms into a working system with high scalability. The real-time applications of human-centered computing are based on multiple cameras in current systems, which require a transparent distributed architecture. This paper presents an application-oriented service share model for the generalization of vision processing. Based on the model, a vision system architecture is presented that can readily integrate computer vision processing and make application modules share services and exchange messages transparently. The architecture provides a standard interface for loading various modules and a mechanism for modules to acquire inputs and publish processing results that can be used as inputs by others. Using this architecture, a system can load specific applications without considering the common low-layer data processing. We have implemented a prototype vision system based on the proposed architecture. The latency performance and 3-D track function were tested with the prototype system. The architecture is scalable and open, so it will be useful for supporting the development of an intelligent vision system, as well as a distributed sensor system.**

***Index Terms*—Distributed architecture, ontology, service share, system integration.**

I. Introduction

IN RECENT years, there has been an increasing research interest in human-centered computing. The main motivating principle is that computers should adapt to people rather than *vice versa* [1]. An intelligent vision system that is adapted to this new computing mode would greatly facilitate the development of intelligent surveillance systems and it could be applied in more fields. CCTV cameras have become cheaper in recent years, but considerable human resources are required to view the CCTV output and this remains expensive. There is a pressing need for automated surveillance systems with commercial, military, and law enforcement applications [1]. They are continually used in some industrial fields, but surveillance systems can also be used to assist human life and in different healthcare fields. For example, smart home systems can be used to care for older people's health, where subjects can move freely in a scene and the system seamlessly provides suitable services or useful information to a remote monitoring system. A key functionality of these applications is the detection of abnormal events using a multicamera system. However, several challenges face intelligent video systems. First, vast volumes of data must be managed. The limitations of the visual fields of single camera means that these systems need to employ multiple video sensors and this produces very large volumes of video data. Second, the freedom of human actions mean the system is confronted with many unpredictable tasks and this leads to unlimited computing requirements. Third, the provision of suitable services requires that the system knows the scene context, which means that even more complex tasks need to be processed. To overcome these challenges, we propose that a system should virtualize the different computing tasks into limited layer service spaces. After such virtualization, the application does not care how the computing resources are managed and higher layers do not need to know how basic services are provided, such as capturing sensor data, compressing video data, and transforming data. This will make the system sufficiently scalable that it can be applied to a wide variety of applications.

Activity in the field of computer vision algorithm research has accelerated rapidly during recent decades, and this makes it more pressing to research architectures for vision systems that can easily support the deployment of common, new vision processing algorithms or methods that are specific to an industrial field. Most previous vision systems [3], [5]–[7] are adapted for processing a specific task and the lack of focus on system development issues has forced each system developer to explicitly consider low-level data-processing details. This has led to insufficient reusability of the developed modules. Therefore, cooperative distributed vision systems with multiple cameras are required and more scalable architectures should be considered. The main goal of this study is to propose a general architecture for a scalable intelligent vision system that can more readily integrate diverse types of algorithms into a working system, especially vision algorithms.

II. Related Work

In recent years, computer vision algorithm research has placed greater emphasis on practical applications and the development of hardware. However, the adaptability of most computer vision algorithms is determined by specific physical conditions. For example, it is difficult to design a general tracking algorithm for different environments and targets. To

Manuscript received September 17, 2011; accepted October 09, 2011. Date of publication October 28, 2011; date of current version January 20, 2012. This work was supported in part by the National Natural Science Foundation of China under Grant 60873266 and Minister of Science and Technology of China under International Science and Technology Cooperative Program. Paper no. TII-11-533.

The authors are with the Key Laboratory of Pervasive Computing, Ministry of Education, Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: wgjxx001@gmail.com; linmi@tsinghua.edu.cn; ajon98@gmail.com; ye-xy07@mails.tsinghua.edu.cn; shiyc@tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2011.2173945

be used in different application, fixed component structures are not capable of dealing with all situations [29] and a system needs the flexibility to support a variety of integrated computing modules and researchers have proposed several models or architecture to facilitate integration. Muja *et al.* [8] proposed a framework for vision based object recognition. The proposed framework treats each specific algorithm as a black box that defines several standard interfaces, including a set of inputs, outputs, and parameters. Using this framework, systems can dynamically load various detectors as plug-ins and execute them sequentially or in parallel. However, this framework is only applicable to visual detection algorithms for use with static images. Quigley *et al.* [9] proposed a framework that was mainly applicable to robot operating systems (ROS framework). The proposed ROS framework provides a structured communication layer that can quickly integrate existing systems and a variety of different processing modules (including some computer vision processing algorithms). The main goal of the ROS framework was to hide much of the hardware and the underlying software architecture, so as to maximize the reuse of existing code. However, this paper mainly dealt with framework design goals and the integration of robotic handling modules. To address the key issues in vision systems, the ROS framework simply proposed several specifications for integrating general vision algorithms, particularly, the OpenCV code library, so it cannot be directly used with vision systems. The current study is limited from the perspective of integrating multiple processing modules in the computer vision field, because it is mainly focused on providing a framework for integrating typical vision algorithms.

Current vision systems have introduced several different architectures that can be divided three types according to the centralized level of computing. 1) *Centralized processing:* in this model, the system often has a central host that processes all the data (message), including video data from different cameras, using only one program (thread). This model is often only applicable to a specific case and the level of distributed processing is the lowest in this model [19], [20]. 2) *Layered distributed processing:* in this model, the system consists of many processing nodes (units), where each node is assigned a suitable task, and concurrently processed results from nodes are integrated by a higher level node that takes charge of advanced tasks. The different nodes form a distributed computing system. NIST's smart data flow system [16] is middleware that supports sensor data transmission in distributed environments, but it cannot explore problems over the whole lifetime of video streams. Many other distributed vision based systems also adopt this processing model [21]–[24]. 3) *Autonomous computing:* in this model, the system is divided into equal computing units depending on their logical function. Matsuyama *et al.* developed a real-time cooperative multitarget tracking system consisting of a group of Active Vision Agents (AVAs), where an AVA was a logical model of a network-connected computer with an active camera [15]. However, the system required a complex communication protocol and this demanded greater computing resources. The different units had to communicate with each other to integrate the processing results from other units to deliver an advanced result. This model has the most flexible architecture, although it requires a highly complex communication protocol [25], [26]. The third distributed computing mode can only be used during common computer vision processing in a vision system and it requires the addition of centralized processing to integrate the outputs from common vision processing modules in an actual application.

Previous studies have presented several frameworks or solutions that can be useful in the development and implementation of computer vision based systems. Afrah *et al.* [11] addressed two aspects of vision based system development that are not fully exploited in current frameworks, i.e., abstraction above low-level details and high-level module reusability. They proposed a systematic classification of subtasks in vision based system development and defined a clear scope for vision based system development framework. Vrěcko *et al.* [12] proposed a general method for integrating visual components into a multimodal cognitive system. Their main goal was that integration should be very generic so it could work with an arbitrary set of modalities. However, its ability to extend to cross-modal concepts is currently quite limited. Tsinghua's SISS [13] is a platform for agent management and interagent collaboration in a multiagent system. It was designed for pervasive computing, so it does not meet some requirements of video information analysis. USCs MFSM [14] is middleware that can support efficient real-time media data processing, but it pays little attention to issues of the distributed environment.

Thus, it would be useful to design a flexible software architecture for cooperative distributed visual computation that is based on scalable configurations. We implemented a flexible multiserver platform for distributed visual information processing in our previous research. Using the proposed platform, developers or researchers can ignore the details of low-level data manipulation and focus on the research application [17]. Based on previous work, we propose an Application-oriented Service Share Model (A-SSM) specifically for developing intelligent vision systems with a scalable architecture that facilitates distributed, easy-access, vision processing.

III. Generalization of Intelligent Visual Information Processing

An intelligent video system, such as a smart home system, is based on computer vision technology that generally includes video data capture, transmission, analysis, storage, and retrieval [28]. These different components form a pipeline for video stream processing that requires a common platform. However, most researchers only focus on the computer vision algorithm when analyzing the pipeline. Thus, our basic concept was to initially model generalized vision processing, because this can provide an easy-use interface for plugging in further applications.

A. Generalized Vision Information Processing

Object tracking is a key technique for developing intelligent vision systems [18]. Without knowing the location of an object, systems cannot provide users with an active service and this is an essential characteristic of an intelligent system. Thus, we start the discussion of analysis using a typical vision processing method known as 3-D locating.

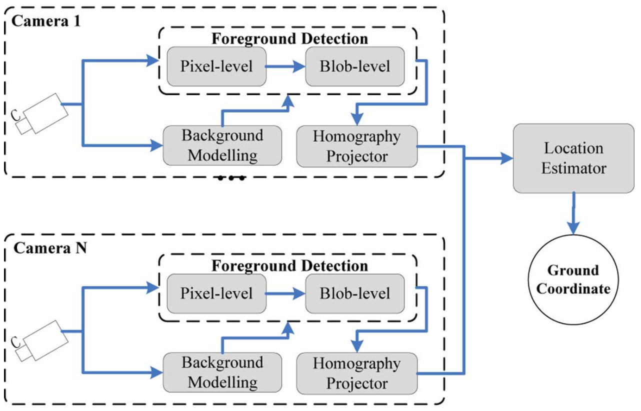


Fig. 1. Process of multicamera 3-D positioning.

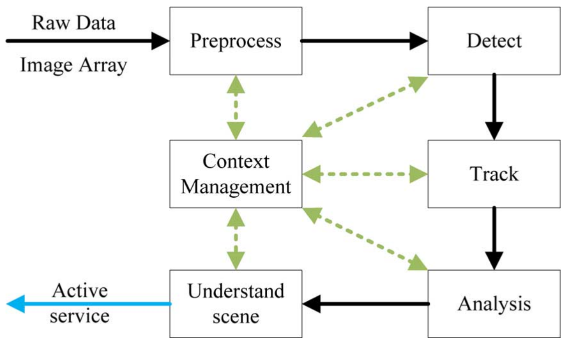


Fig. 2. Information processing in a generalized vision system.

In a 3-D tracking process, the system uses a background model to acquire the foreground information based on the image data obtained by each camera. It then uses the appropriate mapping cameras while ground homograph pixels project foreground pixels onto the common ground plane to form a ground plane in the projection of a collection of pixels. Each angle is detected based on a multicamera image with appropriate geometric constraint conditions and this information can be combined to estimate the location of the body. Fig. 1 shows a simplified version of the process of body positioning using a multicamera vision system.

To promote the process described earlier, we can acquire the information processed by a generalized visual system, as shown in Fig. 2. When a visual system processes information, it generally follows a process of detecting, tracking, and analyzing the object. An intelligent video system should also combine this with contextual information to understand the scene and provide users with suitable proactive services. This requires that a system provides a variety of bottom-up visual information processing functions and a function of high-level semantic reasoning to alleviate tedious manual tasks, such as an operator monitoring the screen of a video surveillance system. Semantic reasoning requires that the system knows the context, because the same state may convey different meanings in a different context. In addition to semantic reasoning, the system must effectively process the low-level data under the guidance of specific context information. As shown in Fig. 2, context management is added to the traditional vision process.

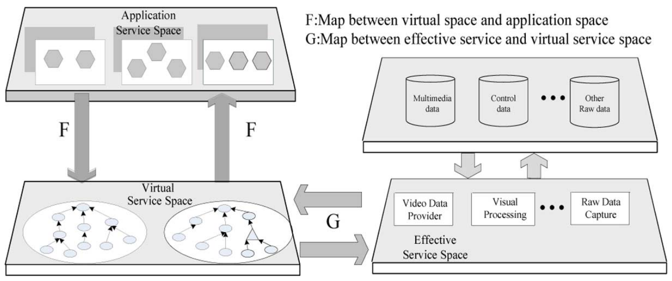


Fig. 3. Brief overview of the architecture of A-SSM. In general, it is divided into three layers, each composed of one specific space.

The intelligent vision system will always be confronted with a disparity between unlimited computing requirements and limited computing resources, because of dynamic changes in the contextual information. To resolve this disparity, we propose an A-SSM, as explained in the following section.

B. A-SSM

Computer vision algorithms are characterized by instability and high computational complexity, and we considered that high-layer applications will require dynamic computing services depending on variations in the environment and requirements. This means that the system must provide a dynamic customized computing service for the high-layer application. We propose an A-SSM that abstracts the different services used in system and virtualizes different computing sources into various service spaces, thereby enabling services that are more standardized and transparent. In this model, the service resources are distributed and high-layer applications can be provided via more ready-accessible transparent services to ensure the scalability of the system.

In general, the A-SSM can be divided into a server domain and application domain. In the server domain, various types of servers are connected and they communicate with each other. All servers can be classified into one of the three service spaces, as shown in Fig. 3. The effective service space is responsible for managing all the actual computing resources, such as video compression and computer vision computing. The virtual service space virtualizes the actual computing from the effective service space as virtual services and provides these services to the application service space via a standard and unified interface. Modules are treated as independent applications in the application service space and they communicate with the server via a standard interface defined in the virtual service space. This separation of server space and application space brings two further benefits as well as the separation into effective and virtual services. First, it divides the tasks of high-layer analysis from low-layer tasks, such as data collection and many preprocessing tasks. This allows the application concentrate to process more of its own tasks with greater transparency. Second, the servers and applications are designed as separate processes that communicated in a consistent mode. This means that many modules can be dynamically plugged in, even during runtime, and application crashes will not result in the breakdown of servers or other applications.

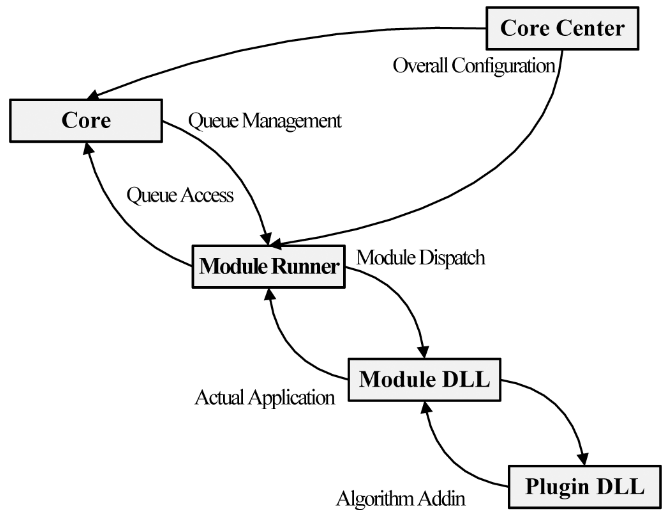


Fig. 4. Overview of the implemented components and their relationships.

Different servers, such as the capture server, archive server, and analysis server, can be allocated to different hosts although they belong in the same space. Thus, all servers in a system have equal weight and no central server exists. In a distributed system, the services provided by various servers are transparent to all applications. Applications simply collaborate with a suitable server or several suitable servers, although they remain unaware of the source of the computing resources or how resources are managed. Thus, the server acts like an agent that serves applications and collaborates or negotiates with other servers. We summarize the main services that should be provided as follows, although the system is not limited.

1) Interface with sensors including cameras, collect sensor data, locate sensor data streams, and collect them, before buffering them for applications.
2) Compress and decompress video data for transmission via a network.
3) Locate applications, route their metadata, and manage the running of their local applications.
4) Manage the buffer pool, refresh it for real-time streams, and acquire the specified buffer for applications.
5) Provide context services to applications, so they can run in a context-aware manner.

IV. Intelligent Vision System Architecture

Our main goal was to create a more flexible architecture that can conveniently integrate CV algorithms with intelligent vision systems and make such systems smart enough to help people with unobtrusive computing. Computer vision algorithms are complex and sensitive to natural condition, such as illumination variation. Thus, our development of the CV algorithm led to a recognition that the system layer support is of great importance.

A. Overview of the Components

Fig. 4 shows that five components are used in the proposed architecture. 1) *Core Center:* this component maintains the overall configuration of the system, e.g., the number of video streams and modules deployed in the system. Processing using this configuration can provide other components with information that allows them to initiate or communicate with each other. 2) *Core:* this component mainly manages different services, such as capturing sensor data, encoding and decoding video streams, and transforming different types of data. A Core mainly fulfills its tasks by managing multiple types of dynamic Queues that applications use for reading or writing. 3) *Module Runner:* this component mainly provides a standard interface for a module, such as a CV processing unit that accesses a specific Queue in a Core. Using the Module Runner, the system can determine the module (especially vision processing) that should be loaded and run. 4) *Module DLL:* this includes the actual vision algorithm that will be dynamically loaded by the Module Runner in the system, according to specified requirements. Using the DLL mode ensures that the system is more scalable. 5) *Plug-in DLL:* this indicates that a plug-in model will be used to help the Module DLL acquire more accurate and specific results. The CV algorithm is often sensitive to conditions and sometimes it will produce different behaviors when using different parameters, such as different shapes, and this will be especially helpful when running detection or tracking tasks.

Of the five components, Core Center, Core, and Module Runner run independently, while Module DLL is usually loaded by the Module Runner and sometimes by the plug-in DLL. The Core Center must maintain the whole configuration of the system from a file or another mode before other components can start to work. The Core and Module Runner components connect to the Core Center which initiates itself and begin work and acquire on how to collect inputs and communicate with other components. A specific module can communicate with a Core and accomplish some specific tasks after being loaded by a Module Runner. The Core Center's main responsibility is to manage the whole configuration of system, hence its name the "Core Center." There is only one Core Center, whereas there can be more than one Core in a system, and Cores can be flexibly located at one host or different hosts. Some common services are provided by the Core, including raw data acquisition, video compressing/uncompressing, and net transmission, which simplified system development by reusing these services. Each Module Runner can share the Core's services without knowing where the Core is or how the Core is working.

With this architecture, the main model of providing/getting services and communication is abstracted as operations of writing/reading different Queues. The system has the characteristic of hierarchical transparency, which means that there is less dependence among the different components. Furthermore, implementations based on the proposed architecture allow the system's combinational development to be broken down into the independent development of modules, which improves efficiency.

B. Dynamic Queue Management

The system's state and environment conditions are variable, so the system must be able to adapt itself to the dynamics of the system. When the system works, its components must cooperate to function together. Thus, several problems must be considered as follows.

1) *Variable inputs:* some modules have to wait for other processing results as their input.

2) *Source management:* some modules may utilize the same source, such as video.
3) *Latency:* different latencies in image processing and semantic reasoning mean that vision processing modules may not recognize each other.

To solve these problems and produce real-time applications, we propose a dynamic queue model, where the modules run in parallel and dynamically exchange information via a specialized shared memory known as the *Dynamic Queue*.

In addition to some descriptive information, such as the type and name, the queue data item is a binary sequence described by modules, which makes it easy to import a new type of queue into the system. Currently, queues are classified into three types according to the data item. 1) *Video Queue:* these are composed of images sequence from cameras, files, or other sources. The images item in these queues are often used as the input for computer vision processing modules. 2) *Stream Queue:* this type of queue is used for video compression. It is mainly used to transmit to other hosts throughout the network and it is decompressed to a Video Queue. 3) *Text Queue:* this type of queue mainly stores the results of module processing. Some modules may utilize these queues as inputs. By combining various Text Queues from different module processing results, the system can share and integrate much more useful information.

The queue is designed to enable a writer and multiple readers with different reading gaps, e.g., a blob detector processes each frame $(\text{reading gap} = 1)$, while a face tracker may process only once every five frames $(\text{reading gap} = 5)$. Using this mechanism, the computer vision algorithm can be organized more effectively. In this scenario, multiple different CV algorithms can process multicamera data and their processing results can be flexibly integrated. This makes it easier for high-layer applications to reason about the actual meaning of a scenario. Basically, using this mechanism can provide the precise location of a user, despite the user "vanishing" from the field of view of a camera and appearing in other cameras.

C. Virtualization of Computer Vision Computing

In the vision system, many modules apply various CV algorithms that work together to fulfill a system task. Among these processing modules, some are connected in series, some are connected in parallel, and some are connected indirectly. In traditional vision systems, these modules are coupled tightly to resolve a specific issue that limits the scalability of system. If each module is transparent to others, with no regard for how they are connected with others, inputs can be allocated automatically by the system when adapting to variations in conditions and outputs can be readily available to others. Thus, transparency facilitates the scalability of the system. We consider that virtualization of these modules is the best way of obtaining transparency, especially with common computer vision algorithms such as Blob Detect, Face Detect, Background Generation, and Foreground Detection, and that it has an important role in distributed vision systems. Virtualization has two aspects. One is the virtualization of the module, while the other is the virtualization of module input/output. Virtualization of the module is mainly concerned with its key properties and function descriptions. After the virtualization of the module, the system can judge whether a module meets the requests of an application. The virtualization of inputs/outputs is mainly concerned with the standardization of how a module interacts with others. This virtualization makes it possible for the system to configure modules automatically. Furthermore, it will be possible to combine a series of results from modules to get new results. Thus, the communicating system can integrate information more flexibly. For example, by combining the information of a human's head and leg, the system can infer the information of a human's body even when some parts of the body are occluded. This is useful for improving the efficiency and accuracy of CV human-tracking algorithms. A key function of intelligent vision systems based on the proposed A-SSM is that transparency provides the results of various computer vision algorithms as services for applications. This means that the researcher can focus more energy on resolving the actual problem in their research field. In the virtualization process, we are inclined to import an ontology for virtualization.

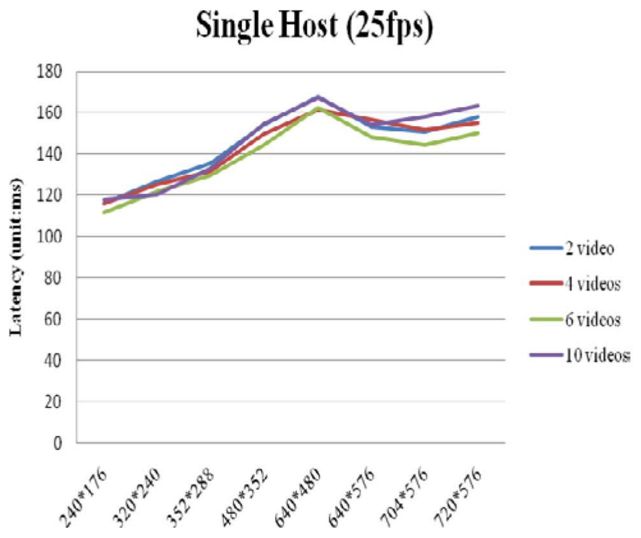
The term "ontology" was borrowed from philosophy and it was introduced into the knowledge engineering field as a means of abstracting and representing knowledge. More recently, ontology has been applied in many fields of computer science, such as the semantic Web, e-commerce, and information systems [6]. Moreover, ontology has been introduced in some context-aware systems in the pervasive computing domain, e.g., CoBrA, Gaia, and SOCAM. Poppe *et al.* introduced an application of Semantic Web Technology to overcome metadata interoperability problems in a surveillance system [27].

Using ontology may have some benefits as follows. 1) Making systems better at understanding multimedia data. It will be convenient for analyzing and reasoning with data [10]. 2) It will be useful for fusing different modules and systems. Using the inference capability, a new module can be created by integrating results from other modules. 3) It will help developers to understand and evolve the system. After introducing an ontology, the system can load suitable modules depending on various task ontology descriptions. For example, if an application wants to know a man's orientation, it simply proposes a task to system. Upon receiving a task about detecting a man's orientation, the system runs relevant face tracker modules according to the available camera data. Thus, the application does not care how the system schedule is processed. This transformation can be achieved using the following description.

```
Setting a task:
  <Task xsi:type="HumanActivity">
    <Participant Person/>
  <Output Orientation of Head/>
  </Task>
```

The corresponding episode of the module running configuration may be as follows:

```
<ModuleConfig>
<RunModule tag="FaceDetect1">
<QuReading>
  <Queue name="1_Cam" manner="FileMap" gap="1"
  />
</QuReading>
```



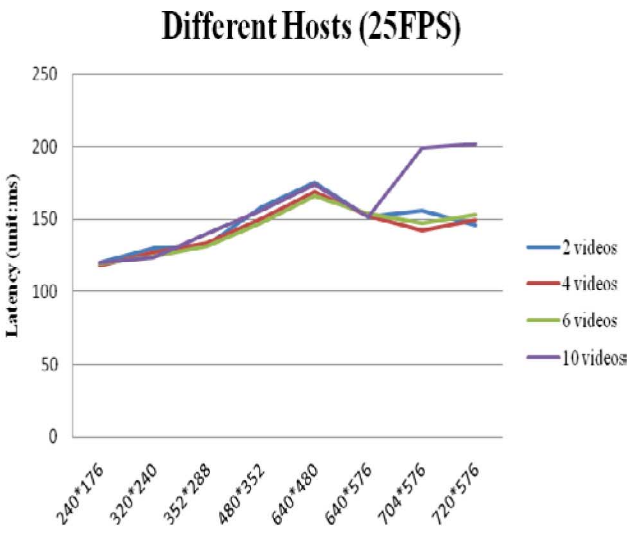


Fig. 5. Impact of video compress/uncompressing and net transmission.

```
<QuWriting>
    <Queue name="1_Face_Dectect" manner="TCP" />
</QuWriting>
<Module>
<DLL>HaarObjDetector.dll</DLL>
    <QuReading cnt="1" qu1="1_Cam" />
    <QuWriting cnt="1" qu1="1_Face_Dectect" />
    <Plugins>
        <DLL>ShapePluginRectangle.dll</DLL>
    </Plugins>
</Module>
</RunModule>
......
</ModuleConfig>
```

V. Experimental System and Results

In this section, we developed a prototype vision system based on the proposed architecture and we present the results of performance tests. Four video sensors were installed in the prototype system. The system's goal was to monitor an object and its trajectory, which can be used to reason about its semantic meaning. The system supports distributed multimedia data capturing, compressing/uncompressing, transferring, and integrates some common vision algorithms such as blob detect, track, and skin detect. We also implemented a runtime switcher mechanism to select queues from candidate queues according to signals emitted by a module, which is a commonly used technique in a multiple camera system.

Latency is a basic parameter for real-time applications, so we conducted an experiment to test the latency of video data from when it was captured to when it was used as an input of a module. We wrote some testing code to record the interval from when the system captured the video data to when image in the module received the same data. We determined the latency for 500 frames and computed their average latency in each experiment. When video data were needed by different modules and these modules were not on one host, the typical latency was the total interval of the capture, compress, transmit, buffering, and decompress actions. To test the affect of compress/uncompress and net transmission, we determined the latency of video data in a single host and in different hosts. The results are shown in Fig. 5.

Fig. 5 shows that the latency from the system capturing the video data to the module acquiring it via net transmission was slightly larger than on a single host, showing that the net transmission had very little affect upon the latency. The latency was mainly affected by the processing latency, i.e., the processing power of the CPU, mainly because of the video compression/decompression. This implies that when more and more applications are running, the processing power of the CPU will be a bottleneck. To handle such situations, it will be convenient to use video sensors with compression functions or to allocate applications to many more hosts using the proposed architecture.

We tested the performance of the implemented components working on the proposed architecture by deploying a 3-D tracking module in the system, as shown in Fig. 6. For every video data stream captured by camera, the core created a corresponding video queue for vision processing modules to be used as the input. Three FG detectors created three FG Queues, using video queue as inputs, while three Blob Detectors will create Blob Queues for a human body, using the FG Queue as the input. When the 3-D tracker is running, it combines the three Blob Queues to reconstruct the 3-D information from the object so as to acquire its real-time location and trajectory. In our experiment, we configured two cores in one computer (Intel Duo CPU E8400 @ 3.00 GHz, 4G memory) with one core in another computer (Intel i7 CPU @ 3.07 GHz, 6.00 GB memory). Three types of Run modules were running on these two computers, as shown in Fig. 7. A screenshot of the running process is shown in Fig. 7. In this configuration, the 3-D track can run normally at a frame rate of 21 frames/s and a video resolution of 320×240, showing the system can meet real time needs. In fact, the cores and modules can be configured

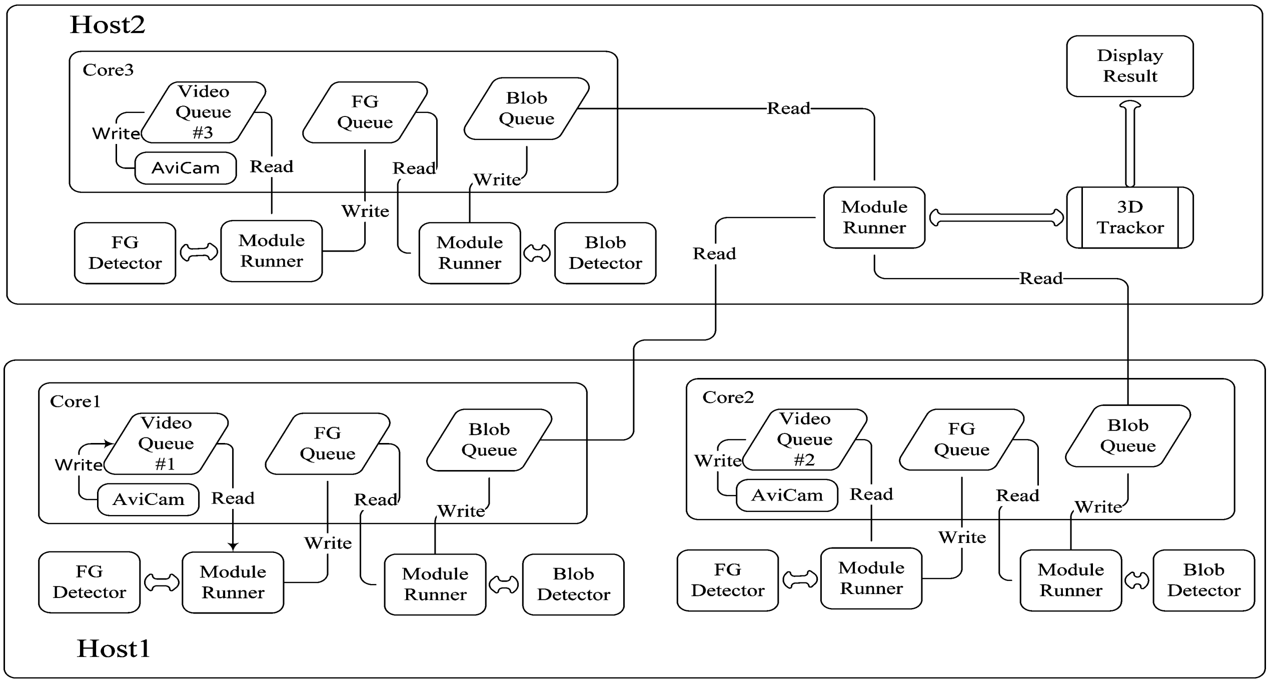


Fig. 6. Components configured for 3-D tracking.

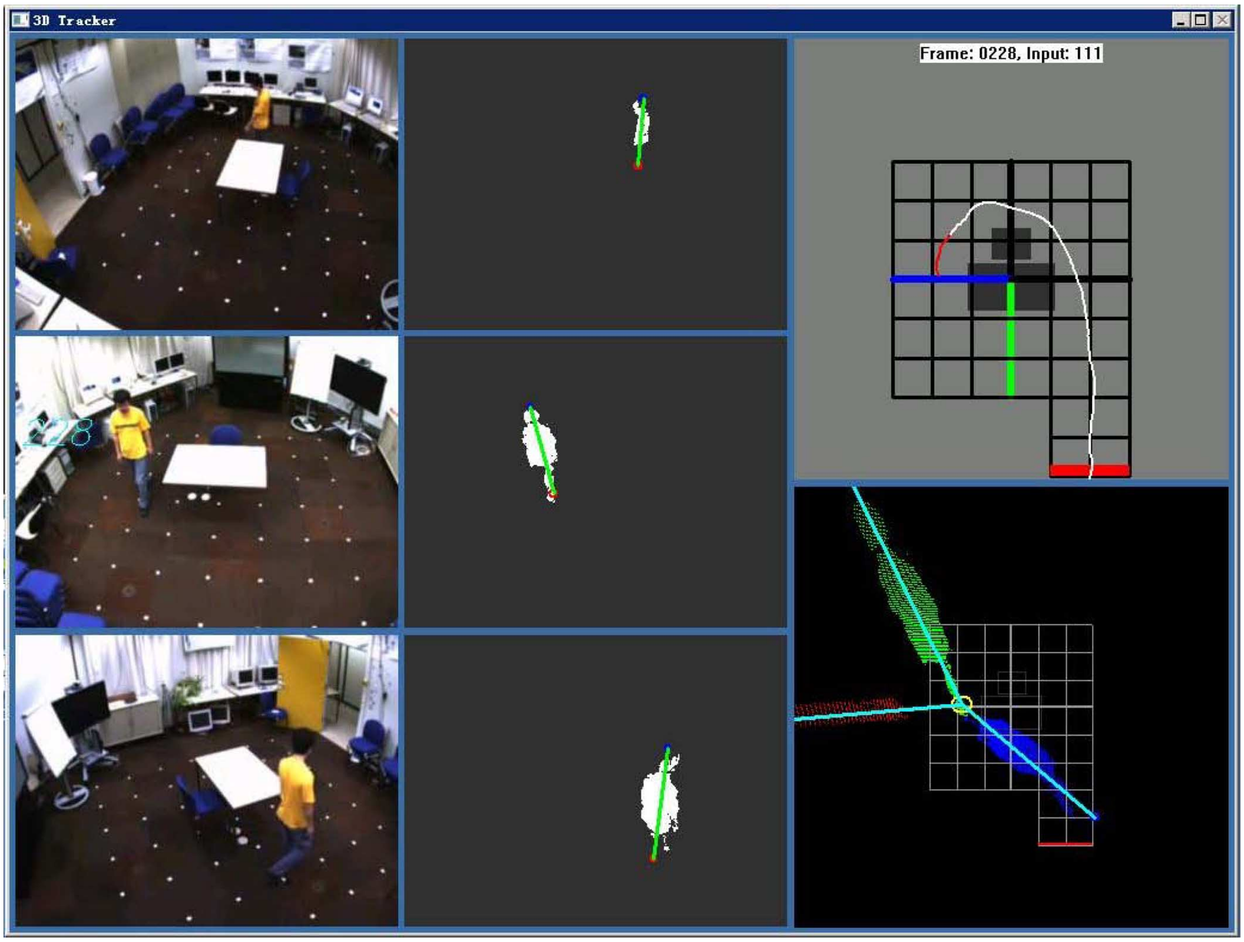


Fig. 7. Screenshot of a 3-D Tracker Module running on the proposed system. The left column shows the original video, while the center shows the processing results for the respective video. The right graph shows the results of 3-D locating and tracking.

very flexibly in many more computers or a single computer, because they run independently and they are transparent to each other, because they connect to each other locally or remotely automatically depending on the configuration.

VI. Conclusion

In this study, the generalization of computer vision processing is summarized using a common model that can be used in intelligent video systems. A novel flexible and real-time distributed architecture was proposed based on our model. In the proposed architecture, the core data manipulation is abstracted into three abstract elements, i.e., the queue, queue reader, and queue writer, where the behavior of acquiring inputs and sending outputs for data processing (including visual processing) is generalized as different types of queue accessing. The queue is managed concurrently, so it can be treated as a bridge that enables cooperation among different processing units in a system. Fewer elements with higher generalization will ensure the scalability and transparency of the system. These three elements provide a basic IO structure ($\mathrm{Queue\ Reader} \leftarrow \mathrm{Queue} \leftarrow \mathrm{Queue\ Writer}$) that can be integrated to give the system smart control over resources and processing modules, with low computational and communication costs. Using this architecture, vision systems can automatically load various processing modules according to different tasks, so they can easily exhibit intelligent characteristics. Context management is a very important role and the use of an ontology representing metadata in the system has just been initiated. Further research will investigate how the system applies the ontology with more inference power to make the system more smart and extensible.

Acknowledgment

The authors gratefully acknowledge the anonymous reviewers for their instructive comments. They would like to thank Sami and MoYou for their suggestion to improve the presentation of this paper.

References

[1] M. Weiser, "The computer for the 21st century," *IEEE Pervas. Comput.*, vol. 99, no. 1, pp. 19–25, Jan.–Mar. 2002.

[2] T. D. Räty, "Survey on contemporary remote surveillance systems for public safety," *IEEE Trans. Syst., Man, Cybern. C: Appl. Rev.*, vol. 40, no. 5, pp. 493–515, Sep. 2010.

[3] M. Valera and S. A. Velastin, "Intelligent distributed surveillance systems: A review," *IEE Proc. Vision, Image Signal Process.*, vol. 152, no. 2, pp. 192–204, Apr. 2005.

[4] G. Xu, L. Tao, D. Zhang, and Y. Shi, "Dual relations in physical and cyber space," *Chinese Sci. Bull.*, vol. 51, no. 1, pp. 121–128, 2006.

[5] H. A. Rowley and J. M. Rehg, "Analyzing articulated motion using expectation maximization," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, San Juan, Puerto Rico, Jun. 17–19, 1997, pp. 935–941.

[6] J. Ye, L. Coyle, S. Dobson, and P. Nixon, "Ontology-based models in pervasive computing systems," *Knowl. Eng. Rev.*, vol. 22, no. 4, pp. 315–347, 2007.

[7] O. Akman, A. A. Alatan, and T. Ciloglu, "Multi-camera visual surveillance for motion detection, occlusion handling, tracking and event recognition," presented at the presented at the Workshop Multi-Camera Multi-Modal Sens. Fusion Algorithms Appl., Marseille, France, 2008.

[8] M. Muja, R. B. Rusu, G. Bradski, and D. Lowe, "REIN—A fast, robust, scalable recognition infrastructure," presented at the presented at the Int. Conf. Robot. Autom., Shanghai, China, 2011.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. Int. Conf. Robot. Autom. (ICRA'09)*, 2009.

[10] R. Nevatia, J. Hobbs, and B. Bolles, "An ontology for video event representation," in *Proc. IEEE Workshop Event Detect. Recognit.*, Jun. 2004, p. 119.

[11] A. Afrah, G. Miller, and S. Fels, "Vision system development through separation of management and processing," in *Proc. 11th IEEE Int. Symp. Multimedia*, Dec. 14–16, 2009, pp. 612–617.

[12] A. Vrecko, D. Skocaj, N. Hawes, and A. Leonardis, "A computer vision integration model for a multi-modal cognitive system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 10–15, 2009, pp. 3140–3147.

[13] W. Xie, Y. Shi, G. Xu, and Y. Mao, "Smart platform—A software infrastructure for smart space (SISS)," in *Proc. 4th IEEE Int. Conf. Multimodal Interfaces*, 2002, pp. 429–434.

[14] R. J. Francois and G. G. Medioni, "A modular middleware flow scheduling framework," in *Proc. 8th ACM Int. Conf. Multimedia*, 2000, pp. 371–374.

[15] T. Matsuyama and N. Ukita, "Real-time multitarget tracking by a cooperative distributed vision system," *Proc. IEEE*, vol. 90, no. 7, pp. 1136–1150, Jul. 2002.

[16] M. Michel, V. Stanford, and O. Galibert, "Network transfer of control data," An Application of the NIST SMART DATA FLOW. CCCT 2003 vol. 2.

[17] Y. Wang, L. Tao, Q. Liu, Y. Zhao, and G. Xu, "A flexible multi-server platform for distributed video information processing," in *Proc. 5th Int. Conf. Comput. Vis. Syst.*, 2007.

[18] X. Zhang, H. Liu, and X. Li, "Target tracking for mobile robot platforms via object matching and background anti-matching," *Robot. Auton. Syst.*, vol. 58, pp. 1197–1206, 2010.

[19] Q. Cai and J. Aggarwal, "Tracking human motion in structured environments using a distributed-camera system," *IEEE Trans. Pattern Recognit. Mach. Intell.*, vol. 21, no. 11, pp. 1241–1247, Nov. 1999.

[20] A. Hoover and B. D. Olsen, "Sensor network perception for mobile robotics," in *Proc. Int. Conf. Robot. Autom.*, San Francisco, CA, 2000, vol. I, pp. 342–348.

[21] N. Thanthry, I. Emmuadi, A. Srikumar, K. Namuduri, and R. Pendse, "SVSS: Intelligent video surveillance system for aircraft," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 24, no. 10, pp. 23–29, Oct. 2009.

[22] C.-M. Huang and L.-C. Fu, "Multitarget visual tracking based effective surveillance with cooperation of multiple active cameras," *IEEE Trans. Syst., Man, Cybern. B: Cybern.*, vol. 41, no. 1, pp. 234–247, Feb. 2011.

[23] M. M. Trivedi, K. S. Huang, and I. Mikic, "Dynamic context capture and distributed video arrays for intelligent spaces," *IEEE Trans. Syst., Man Cybern. A*, vol. 35, no. 1, pp. 145–163, Jan. 2005.

[24] D. A. Fidaleo, H.-A. Nguyen, and M. Trivedi, "The networked sensor tapestry (NeST): A privacy enhanced software architecture for interactive analysis of data in video-sensor networks," in *Int. Multimedia Conf. Arch. Proc. ACM 2nd Int. Workshop Video Surveill. Sensor Netw.*, 2004, pp. 46–53.

[25] Y. Cho, S. O. Lim, and H. S. Yang, "Collaborative occupancy reasoning in visual sensor network for scalable smart video surveillance," *IEEE Trans. Consum. Electron.*, vol. 56, no. 3, pp. 1997–2003, Aug. 2010.

[26] R. G. J. Wijnhoven, E. G. T. De Jaspers, and P. H. N. De With, "Flexible surveillance system architecture for prototyping video content analysis algorithms," in *Proc. SPIE—Int. Soc. Opt. Eng.*, 2006, vol. 6073, p. P60730R.

[27] C. Poppe, G. Martens, P. De Potter, and R. Van de Walle, "Semantic web technologies for video surveillance metadata," in *Multimedia Tools and Applications*. Dordrecht, The Netherlands: Springer, Sep. 2010 [Online]. Available: http://www.springerlink.com/content/n514j052014675p2/fulltext.pdf, [Online]. Available:

[28] D. Doermann, A. Karunanidhi, N. Parkeh, M. A. Khan, S. Chen, H. T. Ozdemir, M. Miwa, and K. C. Lee, "Issues in the transmission, analysis, storage and retrieval of surveillance video," in *Proc. 2003 Int. Conf. Multimedia Expo*, 2003, pp. 161–164.

[29] S. ChenY. LiN. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *Int. J. Robot. Res.*, vol. 30, no. 11, pp. 1343–1377, Sep. 2011.

GuoJian Wang (M'11) received the B.E. degree in communication engineering from the Guangzhou Institute of Communications, Guangzhou, China, and the M.E. degree in computer science from Zhengzhou Information Engineering University, Zhengzhou, China. He is currently working toward the Ph.D. degree in the Key Laboratory of Pervasive Computing, Ministry of Education, Department of Computer Science and Technology, Tsinghua University, Beijing, China.

His research interests include computer vision, machine learning, and software engineering.

Linmi Tao received the B.S. degree in biology from Zhejiang University, Zhejiang, China, the M.S. degree in cognitive science from the Institute of Biophysics, Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China.

He is currently an Associate Professor in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He has studied and worked at the International Institute for Advanced Scientific Studies, University of Verona and Tsinghua University on computational visual perception, 3-D visual information processing, and computer vision. His research covers a brand spectrum on computer vision, computational cognitive vision, and affective computing based on his cross-disciplinary background. His current research interests include vision-assisted brain–computer interface and human-centered interaction supported by National Key Scientific Research Projects and International Cooperative Projects with Intel, Siemens, and IBM. He teaches two courses: Human–Computer Interaction and Patter Recognition, to undergraduate students.

Dr. Tao is a member of the Association for Computing Machinery, and served as Vice Chair, Chair of the ACM SIGCHI China Chapter.

Huijun Di received the B.E. and Ph.D. degrees in computer science from Tsinghua University, Beijing, China.

He is currently a Postdoctoral Researcher in the Key Laboratory of Pervasive Computing, Ministry of Education, Department of Computer Science and Technology, Tsinghua University. His research interests include computer vision, probability modeling, and machine learning.

Xiyong Ye is currently working toward the Ph.D. degree in the Department of Computer Science, Tsinghua University, Beijing, China.

His research interests include the areas of computer vision and machine learning.

Yuanchun Shi (SM'07) received the B.S., M.S., and Ph.D. degrees in computer science from Tsinghua University, Beijing, China.

She is currently a Professor in the Department of Computer Science, Tsinghua University. She has been guiding the Smart Space research, which has made progresses on multiuser interaction, multimodality fusion, context awareness, and scalable group communication. Her research interests include pervasive computing, human computer interaction, and distributed multimedia processing.