DSPcam: A Camera Sensor System for Surveillance Networks

Arvind Kandhalu, Anthony Rowe, Ragunathan (Raj) Rajkumar
Real-time and Multimedia Systems Laboratory
Carnegie Mellon University, Pittsburgh, PA
{akandhal, agr, raj}@ece.cmu.edu

Abstract—**Surveillance is emerging as a key application of camera networks. In this paper, we describe our wireless smart camera system called DSPcam designed for real-time surveillance purposes. DSPcam has a Blackfin processor, CMOS image sensor and 802.11 b/g communication module. It also integrates with a standard sensor network node such as Firefly [1], through which IEEE 802.15.4-based communication is available. An open-source image processing library ported to our DSPcam platform allows for the development of many computer vision applications. We use the in-network processing ability of each DSPcam in our system to annotate video streams with meta-data information that succinctly describes key elements of the visual data being transmitted. The meta-data can be used to trigger alerts and for conducting rapid ex post facto searches. The DSPcams use a time-synchronized communication protocol called TSAM [2] that enables transmission of a large number of video flows and dynamic allocation of bandwidth for high-priority video streams. Finally, we also describe the integration of DSPcam with Sensor Andrew, a large-scale sensing network [4] deployed across the campus of Anonymous University.**

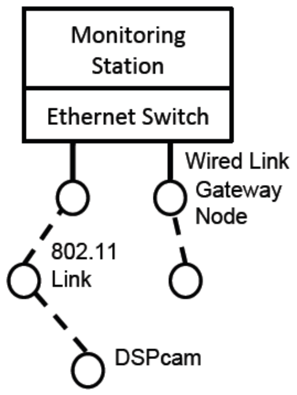


Fig. 1. Typical deployment of DSPcam mesh network.

I. Introduction

Video Surveillance is turning into a vital tool for many organizations for safety and security applications. Yet the cost and difficulty of deployment has not reduced. Typical surveillance systems involve continuous monitoring of multiple video streams from wired IP-camera systems by human operators. Such systems coupled with manual browsing of thousands of video frames for crime scene and forensic analysis is not reliable neither scalable. This has brought about the need for collaborative effort from the systems and vision research communities to develop a surveillance system that is low-cost, reliable, easy to manage, easy to deploy and can process video data for automated real-time alerts and effective retrieval of archived footage. An important component to make such a system viable is a smart camera system.

The sheer amount of raw data generated from camera networks precludes it from human analysis in many applications. Hence distributed intelligent algorithms supported by in-node image processing are required to successfully operate scalable camera networks. This in-node processing of video data is enabled by smart camera systems. Real-time image processing and distributed reasoning made possible by smart cameras can immensely enhance security and surveillance in public or corporate buildings.

In this paper, we describe the design of a smart camera sensor system called DSPcam. The system has been developed with the aim of facilitating distributed intelligent surveillance. With this application in mind, our sensor system architecture targets the provision of sufficient processing power and an adequate imaging system. A typical network configuration of DSPcams is shown in Figure 1. The DSPcam has an 802.11 radio that communicates over multiple hops to form a mesh network. The traffic generated on this network is primarily directed to a single point in the network, namely the operators observation station. Since each camera has local processing capabilities, it can detect the event unfolding in its field of view and appropriately annotate the video for the operator. For example, if a DSPcam detects a human walking, it can label the video data with a tag to represent human motion. These tags can, not only draw attention to the situation on the operators screen, but they can also be used for conducting rapid ex post facto searches. Searches can now include high-level requests for images of a particular object as it moves across multiple camera views. This type of quick decision-making helps provide the best quality information where it is needed.

An important requirement in intelligent surveillance networks is automated tracking. Multi-camera vision systems might fail tracking in cluttered environments due to loss of line-of-sight with the target. Having infinite resolution and zooming capabilities would make the job easier, but it would exponentially increase the computational load and cost of the system. In this context, the integration of a widely distributed low-cost wireless sensor network and a coarsely distributed higher level of intelligence that can be exploited by computer vision systems may overcome many troubles in tracking of large cluttered environments.

An important requirement in intelligent surveillance networks is automated tracking. Multi-camera vision systems might fail tracking in cluttered environments due to loss of line-of-sight with the target. Having infinite resolution and zooming capabilities would make the job easier, but it would exponentially increase the computational load and cost of the system. In this context, the integration of a widely distributed low-cost wireless sensor network and a coarsely distributed higher level of intelligence that can be exploited by computer vision systems may overcome many troubles in tracking of large cluttered environments.

II. Related Work

Wolf et. al [5] outline the importance of smart cameras as embedded systems and the various design aspects of smart cameras. In [6], the authors have developed a smart camera using an Altera FPGA and LUPA 4000 CMOS image sensor. It can generate up to 200 frames per second of VGA data but this implies that it requires correspondingly large amounts of bandwidth. This high data-rate requirement may not be suitable for bandwidth constrained wireless networks. A neuromorphic temporal contrast vision and a signal processor based smart camera is described in [7]. In [8], CMUcam3 along with FireFly sensor nodes have been used to form an energy-efficient sensor network to perform in-home activity clustering. Such a network is extremely resource constrained to perform complex vision tasks and was not intended for video transmission. Cyclops [17] was designed for low power operation and connects a complex programmable logic device (CPLD) directly to the camera for basic image processing. However the 8-bit, 7.3 MHz low power CPU and 64KB RAM limits the computation capability for supporting higher-level computer vision algorithms. MeshEye [12] consists of a low resolution stereo vision system that continuously determines the position, range and size of moving objects entering its field of view. This information triggers a color camera module to acquire a high-resolution image sub-array containing the object, which can be efficiently processed in subsequent stages. The CITRIC [9] hardware platform consists of an image sensor and a frequency scalable (up to 624MHz) PXA270 processor. The device connects with a standard sensor network mote to form a camera mote.

Since cameras have limitations due to limited field of view or resolution, there have been efforts to fuse camera sensors with information from other sensors. In [20], wireless sensor networks with Passive Infrared (PIR) sensors are integrated with multiple cameras to detect and track people. The simple but reliable outputs from the PIR sensor nodes are exploited to improve the accuracy of the vision system. In [21], data from multiple PIR sensors, two color cameras and a pair of microwave sensors are collected, processed and fused to track human presence in the vicinity of a robot.

III. DSPcam Architecture

A. Hardware

The DSPcam has been designed with a modular architecture. This allows for easy hardware updates, for example the core module can be switched for a module with more RAM. At the same time, a tight integration of imaging, processing and networking components is achieved. The key components of DSPcam hardware platform consist of a image sensor module, Core processor module and 802.11 wireless module. Figure 2 shows the DSPcam hardware platform and Figure 3 shows the block diagram of the hardware.

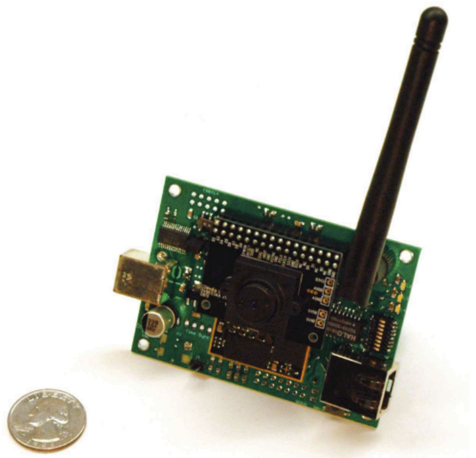


Fig. 2. The DSPcam hardware.

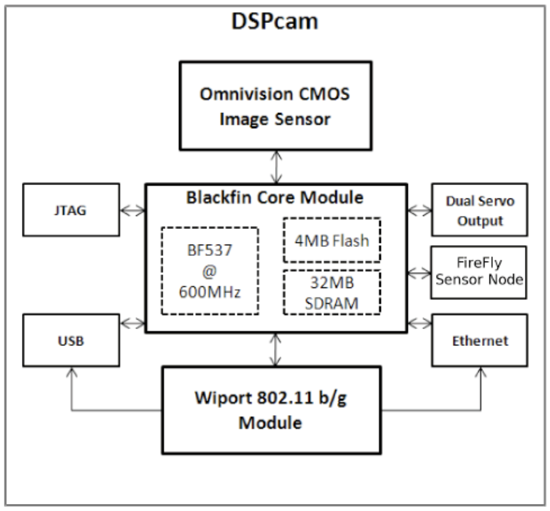


Fig. 3. Block Diagram of DSPcam.

1) CMOS image sensor: The image sensor module for our platform is OmniVision OV9653 [10], a low voltage 1.3 megapixel CMOS image sensor. It supports the following resolutions: SXGA (1280x1024), VGA (640x480), QVGA (320x240) and CIF. The OV9653 provides full-frame, sub-sampled or windowed 8 bit/10 bit images in a wide range of formats, controlled through the Serial Camera Control Bus (SCCB) interface. The image array is capable of operating at up to 30 frames per second (fps) for image resolutions up to VGA and at 15 fps in SXGA. The typical power consumption is 50 mW (15fps@SXGA) and the standby power is 30 μW.

The other important component of the imaging sensor is the lens system. The focal length of the lens will determine the Field Of View (FOV). The aperture number also called the f number specifies the amount of light passing through the lens. The lower the number, the more light that will pass through the lens, thus better the performance in low light conditions. For DSPcam, we have three kinds of lens system: (a) $2.2mm$ f2.5 (b) $6.5mm$ f3.2 (c) $3.6mm$ f2.0. Lens (a) provides 90-degree FOV while lens (b) gives 40-degree FOV, which is an average human beings FOV and lens (c) gives 120-degree FOV. Depending on the location, lighting level and orientation of the DSPcam deployment, the appropriate lens system is used.

2) Core Processor: CM-BF537E module from Bluetechnix has been chosen as the core processor module. It consists of the low power DSP/RISC Blackfin family of processors from Analog Devices, 32MB of SDRAM clocked up to 133MHz, and 4MB of Flash. The size of this module is extremely small (36.5 x 31.5) mm. We choose the Blackfin processor as it has the critical capability of doing DMA (Direct Memory Access) which enables high-performance, low overhead block transmission of captured video frames from the camera to the internal memory of the processor. Hence this leaves enough CPU time for running the Operating System and processing the incoming and outgoing data. Besides other interfaces, the Blackfin processor provides a Parallel Peripheral Interface (PPI) that can connect directly to CMOS image sensor directly. Also the Blackfin processor architecture is SIMD complaint and includes instructions for accelerated video and image processing.

3) Networking Module: The wireless module integrated into the DSPcam is WiPort from Lantronix. The WiPort uses the widely accepted 802.11g protocol to connect to a wireless infrastructure or ad-hoc network, and is fully backward compatible with legacy 802.11b wireless networks. The module is compact and houses a 10/100 Ethernet transceiver. The DSPcam has on-board Ethernet and combined with the bridge mode operation of wiport provides wireless functionality. The blackfin processor communicates to the wiport through the serial interface in order to set the wiport module to appropriate operating mode.

4) USB to UART bridge: The DSPcam uses the FT232 USB-UART bridge controller from Future Technology Devices International (FTDI) [13] to connect the UART port of the BF537 with a USB port on a personal computer for programming and data retrieval. FTDI provides royalty free Virtual COM Port (VCP) device drivers that allow the DSPcam to appear as a COM port to PC applications. The FT232 is USB 2.0 full speed (12 Mbps) complaint. DSPcam also has a JTAG interface for programming and debugging.

B. Software

The software architecture of DSPcam is built on a complete open-source development environment comprised of the uClinux embedded operating system, the gcc compiler and a very large set of Linux utilities. Figure 4 depicts the software architecture of the system. The uClinux OS provides open-source PPI drivers for interfacing with image sensors. An

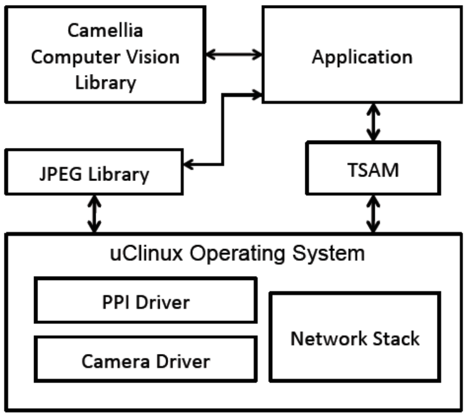


Fig. 4. Software Architecture.

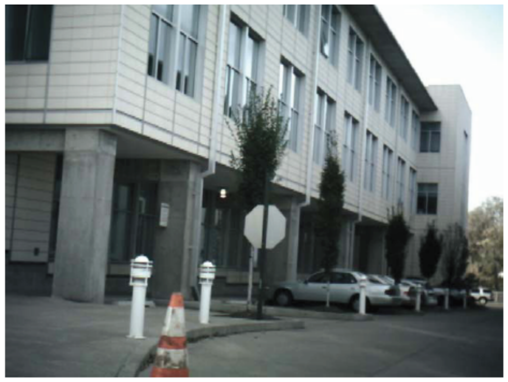


Fig. 5. Original VGA image from DSPcam.

open-source image processing and computer vision library called Camellia [16] has been ported to DSPcam. Figure 5 and Figure 6 show the original VGA image and the image after applying harris corners using the api provided by camellia library respectively. The familiar Linux environment combined with the computer vision library enables quick development of software necessary for surveillance. We have developed an application level MAC protocol that operates over the existing 802.11 MAC protocol. We have developed an api for the protocol using which the appropriate network function calls are made instead of using the standard linux network calls. The MAC protocol is described below.

1) Time Synchronized Application Level MAC (TSAM) Protocol: With existing wireless IP camera systems, as the number of hops to the base station increases, the throughput of the network rapidly decreases. We find that the throughput falls off once the channel has reached 45% of its capacity. Once the network is saturated, the system experiences starvation, preventing some cameras from transmitting any data at all. To avoid excessive and costly over-provisioning of the system, the bandwidth of the video streams needs to be dynamically managed. For this purpose, we have developed a Time Synchronized Application-level MAX protocol (TSAM) [2] that operates over the existing 802.11 MAC protocol. TSAM

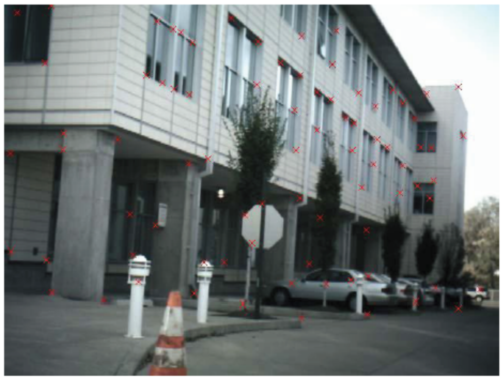


Fig. 6. Image after applying Harris-Corner.

provides a real-time, scalable communication infrastructure for video streaming on commodity hardware. TSAM eliminates contention between nodes by disabling 802.11 collision back-off and allocating exclusive communication slots, thereby able to provide bounded endend delay across multiple hops and collision free operation. An added benefit of the time synchronization comes in the form of temporal registration of images, which simplifies coordination among camera-nodes. As depicted in Figure 4, TSAM operates on top of the existing linux network stack.

IV. Event Tagging on DSPcam

Before any image data is transmitted, the required vision algorithms are processed over the image in order to identify the event in the image to appropriately tag it and then compressed. We will discuss the design trade-offs in the compression scheme. We provide details on a particular vision algorithm required for motion detection and background subtraction that is necessary to extract a moving object that has entered the FOV of the camera.

A. Compression Scheme

Compression of video data is necessary in order to reduce usage of network bandwidth and storage space. Video is essentially a stream of still images. The uClinux OS includes the IJG [19] library that implements JPEG compression. JPEG compression is a common technique for reducing the size of an image for transmission over a network. JPEG provides control over image size using a quality parameter given a particular resolution. In DSPcam we have set the default video output to be QVGA (320x240) in YUV format. This image is then converted to RGB and JPEG-compressed. JPEG compression with a quality of 70 reduces the size of the image by almost 90% while still maintaining good visible quality. This is chosen as the default compression setting. Also although the CMOS sensor can produce 30 FPS, the actual FPS transmitted depends on the number of TSAM communication slots assigned to the node.

B. Motion and Object Detection

Object and motion detection is done through background subtraction on a frame-by-frame basis. That is, the frame difference between the current frame I(x, y) and the Background B(x, y) is calculated. This is, given by:

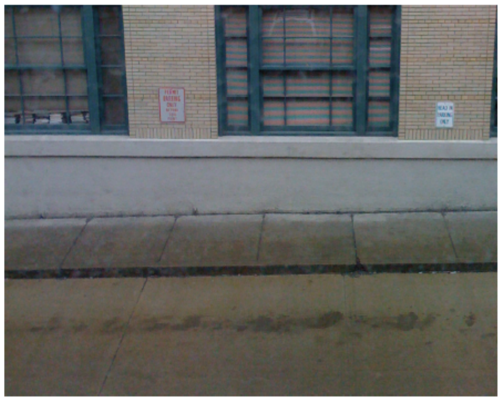


Fig. 7. The background image.

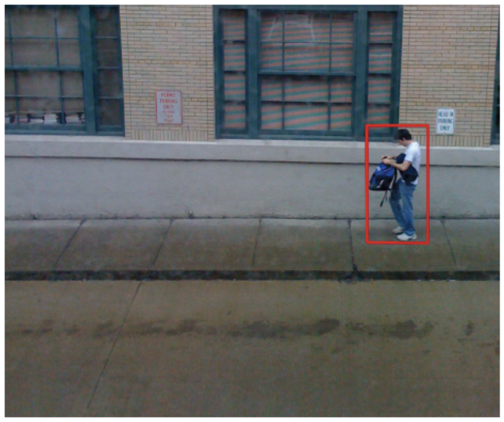


Fig. 8. Image after detection of motion.

$$D(x, y) = |I(x, y)B(x, y)|$$

where x, y denote the pixel coordinates.

All pixels whose difference given by $D(x, y)$ exceed a particular preset threshold which is determined heuristically are set in a binary mask $M(x, y)$ as potential candidates of motion. A blob search algorithm is run over $M(x, y)$ to extract the blobs seen in the binary image. After this is done, small pattern changes could be eliminated as noise. Then, a summation of the pixels covered by the remaining blobs is taken. If this is greater than a preset threshold, we deem that motion has been detected. Also the blobs that give the most significant contribution to motion can be extracted by computing the dot product between the image $I(x, y)$ and $M(x, y)$. Figure 7 and Figure 8 show the images where the background and motion detection is shown respectively.

The background image cant be chosen as a static image. It needs to be updated periodically to adapt to illumination changes, motion changes caused by high-frequency background object such as tree branches and changes in the background geometry due to objects such as parked cars. For this purpose the background is computed as a running average

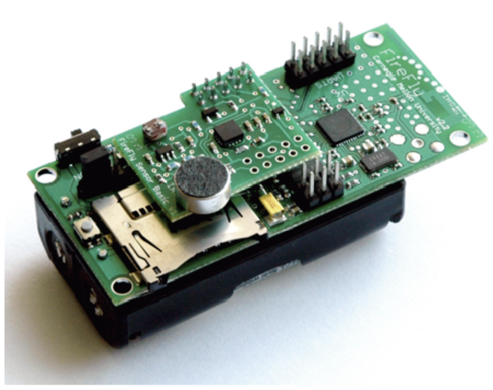


Fig. 9. FireFly sensor networking hardware platform.

given by the following equation:

$$B_{t+1} = K * I_t + (1 - K) * B_t$$

where K is the learning rate and is 0.05, t denotes the current time instant. This gives higher weightage to the most recent image frame as compared to the previous frames.

V. Sensor Fusion

Wireless sensor networks typically consist of low-cost resource constrained nodes called motes. These motes consist of various low-level sensors such as temperature, light and sound sensors. The fusion of these low-bandwidth sensor networks along with information rich high-bandwidth camera networks can help in overcoming the limitations of surveillance systems using camera sensors alone. For example in case of cluttered scenes where direct line of sight is not available, other sensors could inform the presence/absence of people in the area. Thus the information obtained from low-level sensors can aid in the decision making process of high-level camera sensors.

A. FireFly Sensor Networking Platform

The Firefly platform is a low-cost, low-power hardware platform which is shown in Figure 9. It uses an Atmel ATmega1281 8-bit microcontroller with 8KB of RAM and 128KB of ROM along with Chipcons IEEE 802.15.4 standard-complaint radio transceiver for communication. The FireFly supports a sensor expansion board that provides light, temperature, audio, dual axis acceleration and battery-voltage level sensing. The FireFly sensor node runs nano-RK [11] a fully preemptive reservation-based real time operating system (RTOS).

B. Integration of DSPcam with FireFly

The DSPcam uses a modified version of the SLIP (serial line IP) [14] protocol called SLIPstream [15] to communicate to the FireFly sensor nodes through the serial pins. Figure 10 depicts the integration of DSPcam and Firefly. Figure 11 shows the integrated hardware and Figure 12 depicts the DSPcam Graphical User Interface (GUI) showing both video data from DSPcam and sensor information from FireFly.

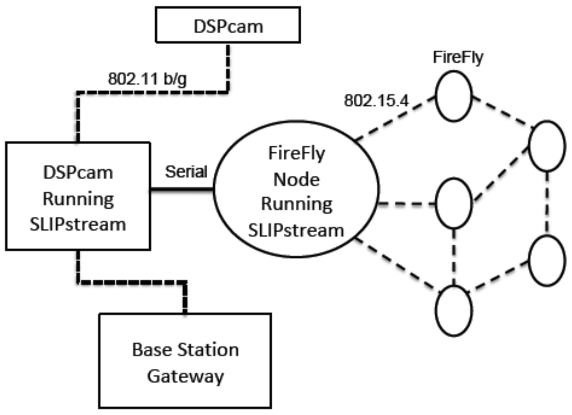


Fig. 10. Integration of DSPcam with FireFly

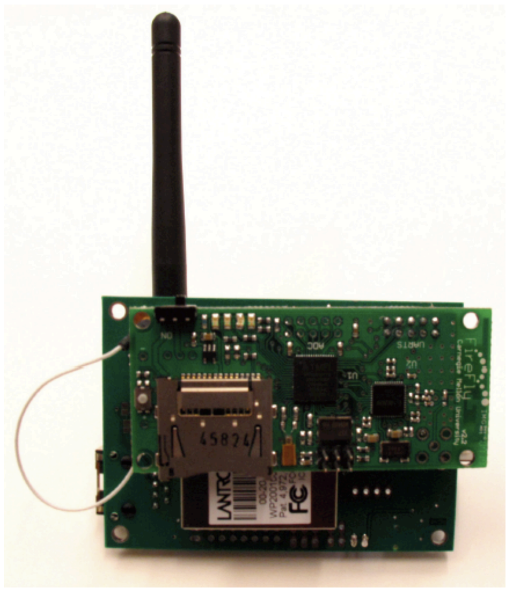
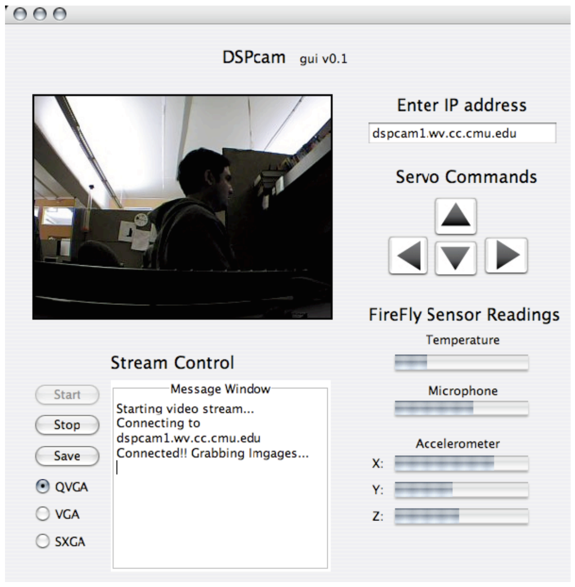


Fig. 11. DSPcam and FireFly integrated hardware.

VI. Sensor Andrew: Large Scale Campus-Wide Sensing

Sensor Andrew [4] is a multi-disciplinary campus-wide scalable sensor network that is designed to host a wide range of sensor, actuator and low-power applications. The goals of Sensor Andrew are to support ubiquitous large scale monitoring and control of infrastructure in a way that is extensible, easy to use, and secure while maintaining privacy. Sensing devices that are used range from cameras and battery-operated sensor nodes to energy-monitoring devices wired into building power supplies. We briefly describe the architecture of Sensor Andrew and then describe the integration of DSPcam in detail.

Figure 13 shows the three-tiered architecture of Sensor Andrew, with a front-end server layer, a gateway layer and a transducer layer. The servers and gateways operate as part of the campus network. The gateway layer is comprised of medium to desktop-class computing devices and DSPcams that



Starting video stream...
Connecting to
dspcam1.wv.cc.cmu.edu
Connected!! Grabbing Imgages...

Fig. 12. The DSPcam Graphical User Interface.

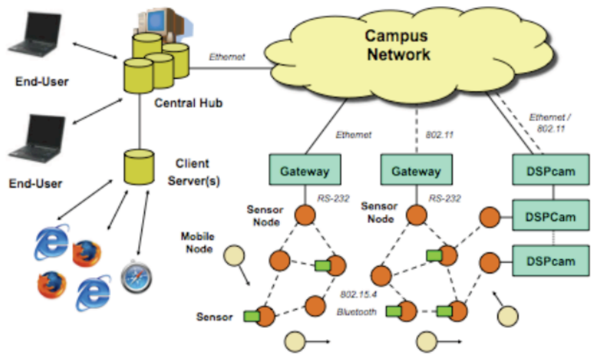


Fig. 13. Sensor Andrew Architecture.

have internet access. The gateways run adapters that shapes the transducer layer information for the server layer. At its core, the server layer has a set of high-performance systems with extensive storage capabilities. This layer acts as the administration front-end to the entire system for configuration, control, data aggregation and archival. The server layer also consists of agents which can subscribe to sensor data to provide high-level services.

The communication between gateways and user applications is done through the eXtensible Messaging Presence Protocol (XMPP) [18]. XMPP is an open XML-inspired Internet protocol traditionally used for online chat communications. XMPP supports publish-subscribe messaging through what are called event nodes. Event nodes are addressable data channels that allow clients to publish and/or subscribe to event feeds. Nodes may also maintain a history of events, provide meta-information about the event feed as well as contain access control lists. This push model of communication provides a powerful mechanism, for distributing sensor data to any interested application or user. For DSPcam the publish messages

```
<DeviceInstallation id ="0x4352" type="DSPCAM"
     timestamp="2009-03-10T10:49:00">
<Event name="Motion">
   <Cause object="Vehicle">
   <Attribute
                  Type="Car"
                  Color="Red"/>
</Event>
```

Fig. 14. SOX message for a DSPcam node publishing data.

will typically occur when an event is detected and the tags describing the event will be published to a particular event node. The application/user interested will then subscribe to this event node to be alerted whenever there is a tag published. A Sensor Over XMPP (SOX) library has been built as a layer on top of XMPP that provides a set of common tools as well as a uniform interface for all Sensor Andrew Applications. A schema of the SOX messages from DSPcams has been designed. As shown in Figure 14, the scheme describes the location of the DSPcam, the event detected and a description of the objects causing the event. Storing this meta-data in a structured fashion is important, as different users might need to query the system in different ways, such as finding the events that have occurred in last 5 hours, events that have been caused by pedestrians and so on.

A. Security and Privacy

Given the nature of the information collected from the DSPcams one has to be naturally concerned about security and privacy. A technique commonly followed is to run a face detection algorithm and mask the face. Although this might solve the problem, the over written face-data is lost forever which is not suitable for forensic applications. In [3], a Privacy through Invertible Cryptographic Obscuration (PICO) technique has been used to addressed this privacy problem. They detect the region of interest, which is the face region of the image and uses AES technique for selective encryption on the region to protect. This encrypted data is passed on to the JPEG process line. The session encryption key, the encrypted regions definitions are all stored inside the JPEG file header. The decryption details can be publicly read but not used without the private key, thereby preserving the privacy details. Another way of handling the privacy issue is by communicating only the tag data i.e. meta data information alone. This way, alerts could be generated and the video can be made available only under emergency.

All server-layer communication in Sensor Andrew takes place over XMPP using SSL connections. All client applications are required to authenticate with a user-name and password. Any guest access to the network is automatically restricted by the access control lists to anonymous data that could not be used to identify individuals.

VII. Conclusion

We have presented the modular architecture of DSPcam, a new camera sensor system designed specifically for surveillance networks. The system enables tagging of video data. Such tagging helps in triggering automated alerts and increasing ease of information retrieval. Also these tags can be used as a means of preserving privacy. The DSPcam has a powerful Blackfin processor and open source development environment, which provides an ideal platform for developing computer vision applications. The system has IEEE 802.11b/g and ethernet networking capabilities. DSPcam uses a Time Synchronized Application level MAC (TSAM) protocol for video streaming which allows for dynamic bandwidth allocation based on the priority of the video streams. Additionally the DSPcams transmit data only on the occurrence of an event, which conserves bandwidth. DSPcam has been integrated with FireFly sensor networking platform, thus providing a platform for application of sensor fusion techniques to surveillance. We have integrated DSPcam with Sensor Andrew, a campus wide sensing system. Sensor Andrew provides the necessary authentication and infrastructure for surveillance networks. We plan to deploy a large number of DSPcams as part of the Sensor Andrew Surveillance application and implement a host of distributed vision algorithms.

References

[1] Anthony Rowe, Rahul Mangharam, Raj Rajkumar, "FireFly: A Time Synchronized Real-Time Sensor Networking Platform, *Wireless Ad Hoc Networking: Personal-Area, Local -Area, and the Sensory-Area Networks*, CRC Press Book Chapter, 2006.

[2] Arvind Kandhalu, Anthony Rowe, Raj (Ragunathan) Rajkumar, "Real-Time Video Surveillance over IEEE 802.11 Mesh Networks, *Real-Time Applications Symposium*, 2009.

[3] Chattopadhyay, A., Boult, T.E, xi"PrivacyCam: a Privacy Preserving Camera using uClinux on the Blackfin DSP, *Computer Vision and Pattern Recognition*, 2007.

[4] Anthony Rowe, Mario Berges, Gaurav Bhatia, Ethan Goldman, Raj Rajkumar, Lucio Soibelman, "Demonstrating Sensor Andrew: Large-Scale Campus-Wide Sensing and Actuation, *Demonstration International Conference on Information Processing in Sensor Networks*, 2009.

[5] Wayne Wolf, Burak Ozr, and Tiehan Lv, "Smart cameras for embedded systems, *IEEE Computer*, 2002.

[6] F. Dias, F. Berry, J. Serot, F. Marmoiton, "Hardware, Design and Implementation Issues on a FPGA-Based Smart Camera, *International Conference on Distributed Smart Cameras (ICDSC)*, 2007.

[7] Litzenberger, M., Belbachir, A. N., Schon, P. Posch, C., "Embedded Smart Cameras for High Speed Vision, *ICDSC*, 2007.

[8] Anthony Rowe, Dhiraj Goel, Raj Rajkumar, "FireFly Mosaic: A Vision Enabled Wireless Sensor Networking System, *IEEE Real Time Systems Symposium (RTSS)*, 2007.

[9] Chen, P., Ahammad, P., Boyer, C., Shih-I Huang, Leon Lin, Lobaton, E., Meingast, M., Songhwai Oh, Wang, S., Posu Yan, Yang, A.Y., Chuohao Yeo, Lung-Chung Chang, Tygar, J.D., Sastry, S.S., "CITRIC: A low bandwidth wireless camera network platform, *ICDSC*, 2008.

[10] Omnivision Technologies Inc. OV9653 Color CMOS SXGA Camera Chip with *OmniPixel* Technology Datasheet, 2006.

[11] A. Eswaran, A. Rowe and R. Rajkumar, "Nano-RK: An Energy-Aware Resource-Centric Operating System for Sensor Networks, *RTSS*, 2005.

[12] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "Mesh- Eye: A Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance, *Information Processing in Sensor Networks (IPSN-SPOTS)*, 2007.

[13] Future Technology Devices International. FT232R USB to UART bridge. *wwww.ftdichip.com/Products/FT232R.htm.*

[14] http://tools.ietf.org/html/rfc1055

[15] http://nanork.org/wiki/SLIPstream

[16] http://camellia.sourceforge.net

[17] M. Rahimi, R. Baer, O.Iroezi, J. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: In situ image sensing and interpretation, *Embedded Networked Sensor Systems*, 2005.

[18] http://www.xmpp.org

[19] http://www/ijg.org

[20] Andrew Prati, Roberto Vezzani, Luca Benini, Elisabetta Farella, Piero Zappi, "An Integrated Multi-Modal Sensor Network for Video Surveillance, *Video Surveillance and Sensor Networks*, 2005.

[21] Yucong Lu, Lingqi Zeng, and Gary M. Bone, "Multisensor system for safer human-robot interaction, *IEEE International Conference on Robotics and Automation*, 2005.