Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras, September 2008, pp. 1-10

CITRIC: A LOW-BANDWIDTH WIRELESS CAMERA NETWORK PLATFORM

Phoebus Chen[†*], Parvez Ahammad[†], Colby Boyer[†], Shih-I Huang[§], Leon Lin[§], Edgar Lobaton[†], Marci Meingast[†], Songhwai Oh[‡], Simon Wang[§], Posu Yan[†], Allen Y. Yang[†], Chuohao Yeo[†], Lung-Chung Chang[§], J.D. Tygar[†], and S. Shankar Sastry[†]

[†] *Electrical Engineering and Computer Sciences, University of California, Berkeley; CA 94720, USA*
[‡] *Electrical Engineering and Computer Sciences, University of California, Merced; CA 95344, USA*
[§] *Industrial Technology Research Institute; Chutung, Hsinchu, Taiwan 310, R.O.C.*

ABSTRACT

In this paper, we propose and demonstrate a novel wireless camera network system, called *CITRIC*. The core component of this system is a new hardware platform that integrates a camera, a frequency-scalable (up to 624 MHz) CPU, 16 MB FLASH, and 64 MB RAM onto a single device. The device then connects with a standard sensor network mote to form a *camera mote*. The design enables in-network processing of images to reduce communication requirements, which has traditionally been high in existing camera networks with centralized processing. We also propose a back-end client/server architecture to provide a user interface to the system and support further centralized processing for higher-level applications. Our camera mote enables a wider variety of distributed pattern recognition applications than traditional platforms because it provides more computing power and tighter integration of physical components while still consuming relatively little power. Furthermore, the mote easily integrates with existing low-bandwidth sensor networks because it can communicate over the IEEE 802.15.4 protocol with other sensor network platforms. We demonstrate our system on three applications: image compression, target tracking, and camera localization.

Index Terms— Wireless Sensor Network, Camera Sensor, Sensor Architecture, Embedded System.

1. INTRODUCTION

Wireless sensor networks (WSNs) have emerged as a new class of information technology infrastructure where computing is embedded into the physical world [9, 1, 11]. A WSN consists of a large number of spatially distributed devices with computing and sensing capabilities, i.e., motes, which form an ad-hoc wireless network for communication. Applications of WSNs include building control [13], environmental monitoring [29], traffic control [20], manufacturing and plant automation [34], service robotics [17], and surveillance [23]. The standardization of communication protocols for sensor networks, namely IEEE 802.15.4 and ZigBee, has facilitated the effort to commericalize WSNs.

The research in WSNs has traditionally focused on low-bandwidth sensors (e.g., acoustic, vibration, and infrared sensors) that limit the ability to identify complex, high-level physical phenomena. This limitation can be addressed by integrating high-bandwidth sensors, such as image sensors, to provide visual verification, in-depth situational awareness, recognition, and other capabilities. This new class of WSNs is called *heterogeneous sensor networks* (HSNs). The integration of high-bandwidth sensors and low-power wireless communication in HSNs requires new in-network information processing techniques and networking techniques to reduce the communication cost for long-term deployment.

In this paper, we describe the design and evaluation of a wireless *camera mote* for HSNs, called the *CITRIC mote*, which is a wireless camera hardware platform with a 1.3 megapixel camera, a PDA class processor, 64 MB RAM, and 16 MB FLASH. This new platform will help us develop a new set of in-network information processing and networking techniques for HSNs. Since wireless camera networks performing in-network processing are relatively new, it is important for our platform to balance performance with ease of development of in-network computer vision algorithms to enable a wider base of applications. Modularity is a key tenet of our design, reflected in the separation of the image processing and networking hardware on the CITRIC mote and in the separation of functions in our client/server back-end software architecture for the entire CITRIC system. Surveillance is used as an example scenario throughout this paper.

Figure 1 shows a typical network configuration for our

*Corresponding author. Email: phoebusc@eecs.berkeley.edu. This work is partially supported by ARO MURI W911NF-06-1-0076 and by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: AFOSR (#FA9550-06-1-0244), BT, Cisco, ESCHER, HP, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies.

978-1-4244-2665-2/08/$25.00 ©2008 IEEE

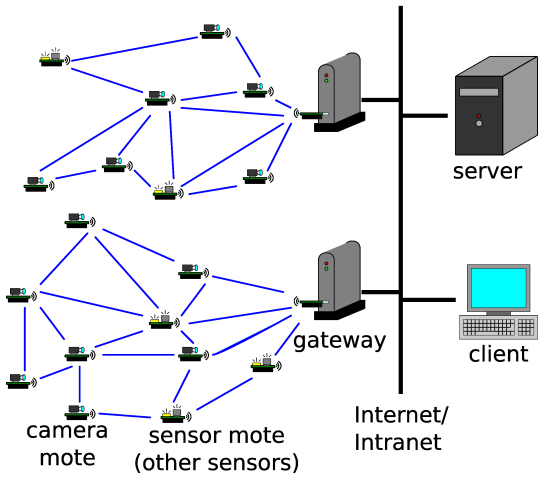

Fig. 1. Architecture of our wireless camera network.

surveillance system. The CITRIC motes are wirelessly networked with each other and possibly with other types of motes over the IEEE 802.15.4 protocol. Some motes also communicate with gateway computers that are connected to the Internet. The motes first perform pre-processing functions on images captured from the camera sensors and then send the results over the network to a central server, which routes the information to various clients for further processing and visualization. The server itself may also provide some centralized processing and logging of data. This architecture allows various clients to interact with different subsets of the motes and support different high-level applications.

We envision our surveillance system to be deployed in a perimeter (e.g., a building or a park) where security can be administered by a single entity. Multiple surveillance systems can also be connected over the Internet. The central server should not be a significant bottleneck in the system because much of the image processing and computer vision algorithms will be run on the motes, meaning the wired backend system will not be processing raw image streams. Also, by not streaming images over the network, the system provides better security against eavesdropping and better privacy protection to those under surveillance.

The rest of this section surveys existing camera mote platforms and motivates why a new design is necessary to meet all our design requirements. We believe that our platform provides the best balance between performance, cost, power consumption, ease of development, and ease of deployment.

1.1. Related Work

Similar to the design of our platform, many of the existing camera motes consist of a camera-and-processor board and a networking mote. A comparison of some representative platforms with our platform is shown in Table 1. A good treatment on the baseline computation requirements for in-network image processing can be found in [8]. The networking motes have minimal on-board processing, typically not suitable for running image processing or computer vision algorithms.

Some platforms in the past focused on streaming video to a centralized server for processing, such as eCAM [25], a small wearable camera platform consisting of an image compression module (no programmable CPU) and a networking node.

One of the earliest camera motes with significant on-board processing is Panoptes [10]. The latest version of the Panoptes platform consists of a Stargate "gateway mote," an 802.11b PCMCIA wireless card, and a USB camera. Panoptes is targeted at applications where one would selectively stream video to conserve bandwidth. To this end, the platform has a priority-based adaptive buffering scheme, a filter to remove uninteresting video frames, a video/camera query system, and video compression. The use of commercial devices in Panoptes, instead of a tightly integrated design, imposes extra limitations. Most notably, the frame rate of the camera is limited by the USB bus speed, which forces the USB camera to compress the image and the Stargate processor to decompress the image to perform processing, thus consuming extra computation and power.

On the other hand, the Cyclops [27], WiSN [8], and WiCa [15] platforms have much tighter camera and on-board processor integration. Cyclops was designed for low power operation and connects a complex programmable logic device (CPLD) directly to the camera for basic image processing such as background subtraction and frame differencing. However, the 8-bit, 7.3 MHz low-power CPU and 64 KB RAM limits the computation capability for supporting higher-level computer-vision algorithms. WiSN uses a more powerful 32-bit, 48 MHz CPU and also 64 KB RAM, but the processor is shared between networking and image processing processes. Similar to the Cyclops, the second generation WiCa mote speeds up low-level image processing using an 84-MHz Xetal-II SIMD processor, which has a linear processor array of 320 parallel processing elements and a 16-bit global control processor for higher-level sequential processing. It uses a separate 8051 MCU and ZigBee module for networking [14].

The platform most similar to the CITRIC mote is a prototype platform used by [30], which consists of an iMote2 [5] connected to a custom-built camera sensor board. The platform consists of an XScale CPU running at a slightly lower clock speed, 32 MB RAM, 32 MB FLASH, and an OmniVision camera. Unlike the CITRIC mote, the networking and image processing functions are both performed on the XScale processor, and the platform does not have a built-in microphone. The separation of the image processing unit from the networking unit in the CITRIC mote allows for easy development and testing of various image processing and computer vision algorithms.

Finally, multi-tiered camera networks have also been proposed to use low cost/power/resolution camera motes to wake up higher cost/power/resolution cameras to capture and pro-

Table 1. Comparison of existing wireless camera mote platforms with the new CITRIC mote platform.

Platform	Processor	RAM	ROM	Camera	Wireless
eCAM [25]	OV528 Serial Bridge (JPEG Compression only)	N/A	N/A	COMedia C328-7640 board uses OV7640 camera (640 × 480 pixel @ 30 fps)	Eco node uses nRF24E1 radio+MCU (1 Mb/s, 10 m range)
Panoptes [10]	Intel XScale PXA255 (400 MHz, 32-bit CPU)	64 MB	32 MB	Logitech 3000 USB Camera[1] (640 × 480 pixel @ ≈ 13 fps) (160 × 120 pixel @ ≈ 30 fps)	802.11 PCMCIA Card (11 Mb/s for 802.11b)
Cyclops [27]	Atmel ATmega128L (7.3728 MHz, 8-bit CPU) Xilinx XC2C256 CoolRunner (16 MHz CPLD)	64 KB	512 KB	ADCM-1700 (352 × 288 pixel @ 10 fps)	Mica2 mote uses TR1000 radio (40 kbps)
WiSN [8]	Atmel AT91SAM7S (48 MHz, 32-bit ARM7TDMI CPU)	64 KB +32 KB[2]	256 KB +2 MB[2]	ADCM-1670 (352 × 288 pixel @ 15 fps) ADNS-3060 (30 × 30 pixel @ 100 fps)	built-in CC2420 radio (802.15.4, 250 kbps)
WiCa (Gen 2) [15, 14]	Xetal-II (84 MHz, 320 PE LPA + GCP)	1.75 MB	N/A	Unknown (640 × 480 @ 30 fps)	Aquis Grain ZigBee+MCU uses CC2420 radio (802.15.4, 250 kbps)
iMote2+Cam [30]	Intel XScale PXA271 (up to 416 MHz, 32-bit CPU)	32 MB	32 MB	OV7649 (640 × 480 pixel @ 30 fps) (320 × 240 pixel @ 60 fps)	built-in CC2420 radio (802.15.4, 250 kbps)
CITRIC	Intel XScale PXA270 (up to 624 MHz, 32-bit CPU)	64 MB	16 MB	OV9655 (1280 × 1024 pixel @ 15 fps) (640 × 480 pixel @ 30 fps)	Tmote Sky mote uses CC2420 radio (802.15.4, 250 kbps)

cess interesting images. One such notable multi-tier camera network system is SensEye [16], which consists of 3 tiers of cameras. In the future, we also envision deploying our CITRIC mote in a multi-tier network, particularly ones composed of heterogeneous sensors (e.g., passive-infrared motion sensors and microphones).

2. ARCHITECTURE AND DESIGN

2.1. Camera Mote

The CITRIC platform consists of a camera daughter board connected to a Tmote Sky board (see Figure 2, left). The Tmote Sky [19] is a variant of the popular Telos B mote [26] for wireless sensor network research, which uses a Texas Instruments MSP430 microcontroller and Chipcon CC2420 IEEE 802.15.4-compliant radio, both selected for low-power operation.

The camera daughter board is comprised of a $4.6\,\text{cm} \times 5.8\,\text{cm}$ processor board and a detachable image sensor board (see Figure 2, middle). The design of the camera board uses a small number of functional blocks to minimize size, power consumption, and manufacturing costs.

To choose a proper onboard processor, we have the option to use either field-programmable gate arrays (FPGAs) or general-purpose processors running embedded Linux. Although FPGAs have advantages in terms of speed and low-power consumption, the user would need to program in a hardware description language, making algorithm implementation and debugging a time-consuming process. On the other hand, many well-studied image processing and computer vision algorithms have been efficiently coded in C/C++, such as the OpenCV library [2]. Therefore, we chose to use a general-purpose processor running embedded Linux (as opposed to TinyOS [32]) for the camera board for rapid prototyping and ease of programming and maintenance.

2.1.1. CMOS image sensor

The camera for our platform is the OmniVision OV9655, a low voltage SXGA (1.3 megapixel) CMOS image sensor that offers the full functionality of a camera and image processor on a single chip. It supports image sizes SXGA (1280×1024), VGA, CIF, and any size scaling down from CIF to 40×30, and provides 8-bit/10-bit images. The image array is capable of operating at up to 30 frames per second (fps) in VGA, CIF, and lower resolutions, and 15 fps in SXGA. The OV9655 is designed to perform well in low-light conditions [24]. The typical active power consumption is 90 mW (15 fps @SXGA) and the standby current is less than $20\,\mu A$.

[1] Frame rate limited by compression and USB bandwidth.

[2] External memory extension. Extending both RAM and ROM not permitted.

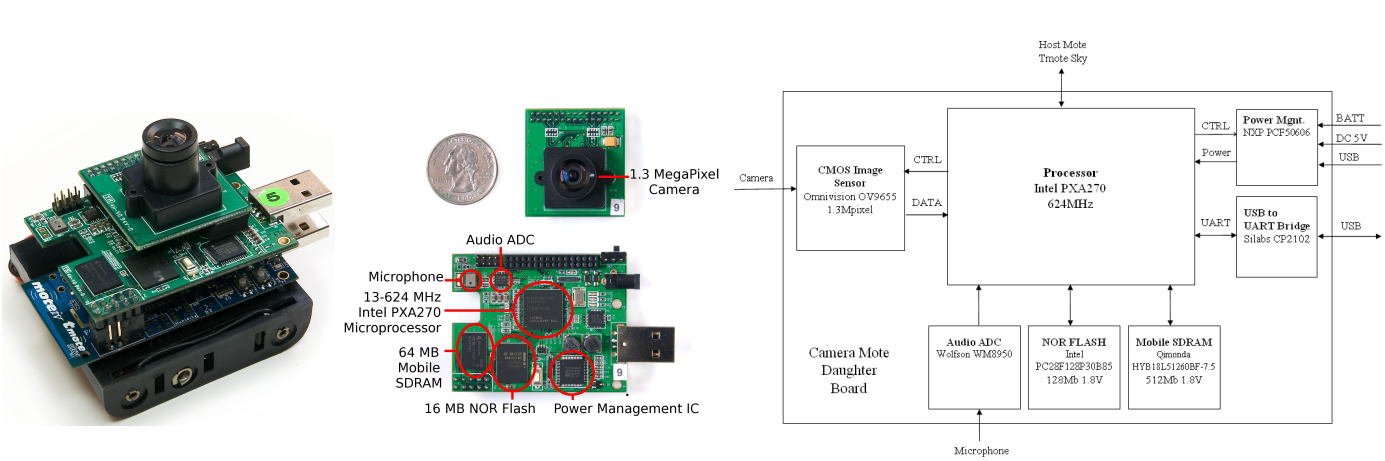

Fig. 2. (Left) Assembled camera daughter board with Tmote. (Middle) Camera daughter board with major functional units outlined. (Right) Block diagram of major camera board components.

2.1.2. Processor

The PXA270 [12] is a fixed-point processor with a maximum speed of 624 MHz, 256 KB of internal SRAM, and a wireless MMX coprocessor to accelerate multimedia operations. The processor is voltage and frequency scalable for low power operation, with a minimum voltage and frequency of 0.85 V and 13 MHz, respectively. Furthermore, the PXA270 features the Intel Quick Capture Interface, which eliminates the need for external preprocessors to connect the processor to the camera sensor. Finally, we chose the PXA270 because of its maturity and the popularity of its software and development tools. The current CITRIC platform supports CPU speeds of 208, 312, 416, and 520 MHz.

2.1.3. External Memory

The PXA270 is connected to 64 MB of 1.8 V Qimonda Mobile SDRAM and 16 MB of 1.8 V Intel NOR FLASH. The SDRAM is for storing image frames during processing, and the FLASH is for storing code. 64 MB of SDRAM is more than sufficient for storing 2 frames at 1.3 megapixel resolution (3 Bytes/pixel $\times$ 1.3 megapixel $\times$ 2 frames $=$ 8 MB), the minimal requirement for background subtraction. 64 MB is also the largest size of the Single Data Rate (SDR) mobile SDRAM components natively supported by the PXA270 currently available on the market. As for the FLASH, the code size for most computer vision algorithms falls well under 16 MB. Our selection criteria for the types of non-volatile and volatile memory are access speed/bandwidth, capacity, power consumption, cost, physical size, and availability.

Our choices for non-volatile memory were NAND and NOR FLASH, where the former has lower cost-per-bit and higher density but slower random access and the latter has the capability to execute code directly out of the non-volatile memory on boot up (eXecution-In-Place, XIP) [6]. NOR FLASH was chosen not only because it supported XIP, but also because NAND Flash is not natively supported by the PXA270 processor.

Our choices for volatile memory were Mobile SDRAM and Pseudo SRAM, both of which consume very little power. Low power consumption is an important factor when choosing memory because it has been demonstrated that the memory in handsets demands up to 20 percent of the total power budget, equal to the power demands of the application processor [33]. Mobile SDRAM was chosen because of its significantly higher density and speed.

We had to forgo using multi-chip packages (MCPs) that incorporate a complete memory subsystem (ex. NOR + Pseudo SRAM, NAND + Mobile SDRAM, NOR + NAND + Mobile SDRAM) in a single component due to their availability, but they may be used in future versions of the platform.

2.1.4. Microphone

In order to run high-bandwidth, multi-modal sensing algorithms fusing audio and video sensor outputs, it was important to include a microphone on the camera daughter board rather than use a microphone attached to the Tmote Sky wireless mote. This simplified the operation of the entire system by dedicating the communication between the Tmote Sky and the camera daughter board to data that needed to be transmitted over the wireless network. The microphone on the board is connected to the Wolfson WM8950 mono audio ADC, which was designed for portable applications. The WM8950 features high-quality audio (at sample rates from 8 to 48 ks/s) with low-power consumption (10 mA all-on 48 ks/s mode) and integrates a microphone preamplifier to reduce the number of external components [35].

2.1.5. Power Management

The camera daughter board uses the NXP PCF50606, a power management IC for the XScale application processors, to manage the power supply and put the system into sleep mode. When compared to an equivalent solution with multiple discrete components, the PCF50606 significantly reduces the system cost and size [21]. The entire camera mote, *including* the Tmote Sky, is designed to be powered by either four AA batteries, a USB cable, or a 5 V DC power adapter cable.

2.1.6. USB to UART bridge

The camera daughter board uses the Silicon Laboratories CP2102 USB-to-UART bridge controller to connect the UART port of the PXA270 with a USB port on a personal computer for programming and data retrieval. Silicon Laboratories provides royalty-free Virtual COM Port (VCP) device drivers that allow the camera mote to appear as a COM port to PC applications [28]. The CP2102 is USB 2.0 full-speed (12 Mbps) compliant, and was chosen because it minimizes the number of physical components on the PCB.

The camera daughter board also has a JTAG interface for programming and debugging.

2.2. Wireless Communications

As shown in Figure 1, sensor data in our system flow from the motes to a gateway over the IEEE 802.15.4 protocol, then from the gateway over a wired Internet back-end to a centralized server, and finally from the server to the client(s). The maximum data rate of 802.15.4 is 250 kbps per frequency channel (16 channels available in the 2.4 GHz band), far too low for a camera mote to stream images back to the server at a high enough quality and frame rate for real-time applications. A key tenet of our design is to push computing out to the edge of the network and only send post-processed data (for instance, low-dimensional features from an image) in real-time back to the centralized server and clients for further processing. If an event of interest occurs in the network, we can then send a query for the relevant image sequence to be compressed and sent back to the server over a slightly longer period of time. Since we are using commercial off-the-shelf motes running TinyOS/NesC, we can easily substitute different standard routing protocols to suit an application's particular needs. For instance, the real-time requirements of surveillance imply that typical communication does not need to run over a reliable transport protocol.

2.3. Client/Server Interface

From a more abstract point of view, the sensor network can be modeled as a shared computing resource consisting of a set of nodes that can be accessed by multiple users concurrently. As such, users logging into the system can assign different tasks to different nodes. The first user to log in to a node becomes a "manager" of that node (see Figure 3). Other users logging in to the system can assign tasks to any unmanaged nodes, but will only be able to listen to the data output by managed nodes. For example, if there is a node performing a tracking task for a given user then a new user will not be able to assign a face recognition task to the node but s/he will still be able to log in and listen to the output data for the tracking task. As users log out of the nodes, the manager role is passed on to the next logged in user. In the future, the system will be extended to allow users to negotiate the use of the resources by integrating an interface for users to yield the role of manager to others.

Note that in this shared computing model, the server is "invisible" to the users. In reality, the server will provides some services (such as database storage for more memory intensive tasks) but its role will be hidden from the user.

3. CAMERA MOTE BENCHMARKS

3.1. Energy Consumption

The power consumption of the camera mote was determined by logging the current and voltage of the device when it was connected to four AA batteries (outputting $\approx$ 6 V). A Tektronix AM 503B Current Probe Amplifier was used to convert current to voltage, and a National Instruments 9215 USB data logger was used to log both the voltage of the batteries and the voltage of the current probe.

First, we measured the power consumption of the camera daughter board alone running Linux but with no active processes (Idle). We then took the same measurement but with the Tmote attached, although no data was sent to the Tmote (Idle + Tmote). In this test, the Tmote was running an application that waits to receive any packets from the camera board and transmits over the radio. On average, Idle consumes 428 – 478 mW, and Idle + Tmote consumes 527 – 594 mW, depending on the processor speed.

We also measured the power consumption of the mote running a typical background subtraction function. The test utilizes all the components of the mote by both running the CPU and using the Tmote to transmit the image coordinates of the foreground. At the processor speed 520 MHz, the power consumption was 970 mW. Note that the power consumption may be reduced by enabling power management on the Tmote.

The current draw is relatively constant over time, even though the voltage of the batteries decreases with time. The calculations were made using the nominal voltage of 6 V in order to be consistent, since each experiment starts and ends with a different voltage. If we assume the camera mote consumes about 1 W and runs on batteries with 2700 mAh capacity, we expect the camera mote to last over 16 hours under continuous operation.

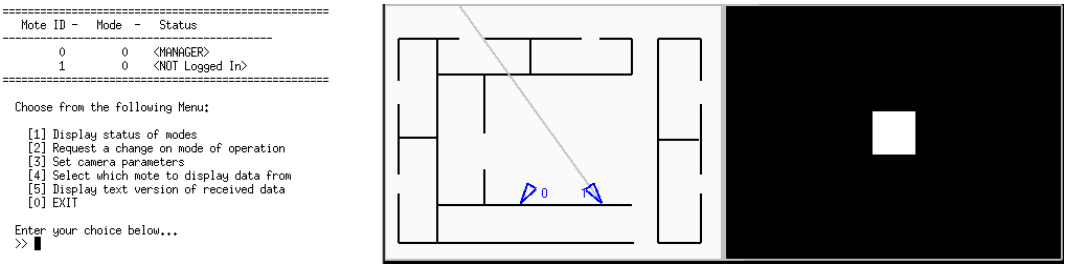

Fig. 3. Screenshot of user interface. (Left) A text menu is available for assigning tasks to the motes. At the top of this menu is a table showing that the user is the "manager" of one node and "not logged in" on the other node. (Right) The graphical visualization of the network physical layout and a detected foreground region from a selected mote.

3.2. Speed

The speed benchmarks for the camera board were chosen to reflect typical image processing computations. We compared the benchmarks with and without the Intel Integrated Performance Primitives (IPP) library to evaluate whether IPP provides a significant performance increase.

All benchmarks were performed on 512×512 image arrays. The *Add* benchmark adds two arrays. The *Background Subtraction* benchmark computes the difference of two arrays and then compares the result against a constant threshold to get a boolean array (mask). The *Median Filter* benchmark performs smoothing by taking the median pixel value of a 3×3 pixel area at each pixel. The *Canny* benchmark implements the first stage of the Canny edge detection algorithm. The benchmark results for *Add* and *Background Subtraction* were averaged over 1000 trials, while those for *Median Filter* and *Canny* were averaged over 100 trials.

Figure 4 shows the average execution time of one iteration for each benchmark. Note that the IPP versions of the functions are not necessarily always faster than their non-IPP counterparts. For example, the *Background Subtraction* benchmark consists of an arithmetic operation and a comparison. Implemented in IPP, this requires two function calls and thus two iterations through the entire array. But implemented without IPP, we can perform both operations on the same iteration through the array, resulting in only one iteration and fewer memory accesses. Such non-IPP optimizations should be taken into consideration when building future applications in order to obtain maximum performance. Also, the non-linear performance curve for different CPU frequencies can be attributed to the constant speed of memory access (the bus speed is 208 MHz regardless of the processor speed).

4. APPLICATIONS

In this section, we demonstrate the capacity and performance of the proposed platform via three representative low-level vision applications: image compression, target tracking, and camera localization. The algorithms are implemented in C/C++ on the camera motes and a base-station computer. In all the experiments, the processor speed is set at 520 MHz.

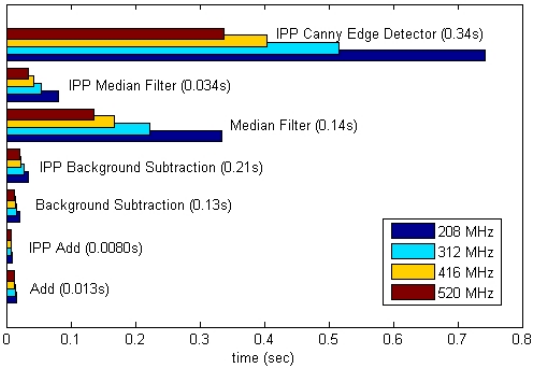

Fig. 4. Average run time of basic image processing functions on 512×512 images (over $\geq$ 100 iterations). Execution time at 520 MHz is shown in parentheses.

4.1. Image Compression

Image compression is an important function to many camera-based applications, especially in networks designed to push captured images to a back-end server for further processing. We quantitatively measure the speed of image compression on the CITRIC mote platform, and the rate-distortion and time-distortion trade-offs between two compression schemes: JPEG and Compressed Sensing (CS) [4].

The embedded Linux OS includes the IJG library that implements JPEG compression.[3] Since the onboard CPU only has native support for fixed-point arithmetic, we used the integer DCT implementation.

The CS scheme using random matrices [7, 3] has been shown to provide unique advantages in lossy compression, particularly in low bandwidth networks with energy and computation constrained nodes. While a detailed discussion about CS is outside the scope of this paper, we briefly describe the key elements used in our experiments. Suppose an $n \times n$ image or image block $I \in \mathbb{R}^{n^2}$ stacked in vector form can be written as a linear combination of a set of basis vectors; i.e.,

[3] Available from `http://www.ijg.org`

$I = F\boldsymbol{x}$, where $F \in \mathbb{R}^{n^2 \times m}$ is some (possibly overcomplete) linear basis. If one assumes that only a sparse set of basis vectors is needed, i.e., all but a small percent of the coefficients in $\boldsymbol{x}$ are (close to) zero, then it has been shown that, with overwhelming probability, $\boldsymbol{x}$ can be stably recovered from a small number of random projections of the original image I, $\boldsymbol{y} \doteq RI = RF\boldsymbol{x} \in \mathbb{R}^d$, where $R \in \mathbb{R}^{d \times n^2}$ is a random projection matrix, and d is the number of coefficients used in compression [7, 3]. The sparse representation $\boldsymbol{x}$ can be recovered via ℓ^1-minimization:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \text{ subject to } \boldsymbol{y} = RF\boldsymbol{x}. \quad (1)$$

The reconstruction of the image is then $\hat{I} = F\boldsymbol{x}^*$.

In this paper, the components of the random matrix R are assigned to be ± 1 with equal probability (i.e., the Rademacher distribution). Therefore, computing the random projections only involves addition and subtraction, which is particularly suitable for fixed-point processors.[4] We apply an Exponential-Golomb coding method to encode the random projection coefficients. At the decoder, the inverse DCT matrix is used as the image basis F to perform reconstruction via ℓ^1-minimization.

Compared to image compression based on block DCT transform or wavelets, there is no surprise that random projection requires more coefficients to achieve the same compression quality. On the other hand, the random projection scheme has the following unique advantages:

1. Transmission of the random projections $\boldsymbol{y}$ is robust to packet loss in the network. Even if part of the coefficients in $\boldsymbol{y}$ is lost during transmission, the receiver can still adaptively create the appropriate measurement matrix R' and carry out ℓ^1-minimization at the expense of less accuracy.

2. It is straightforward to implement a progressive compression protocol using random projection, e.g., one can construct additional random projections of the image signal I to improve the reconstruction accuracy.

3. In terms of security, if (part of) the projection signal $\boldsymbol{y}$ is intercepted but the random seed used to generate the random matrix is not known to the intruder, it is more difficult to decipher the original signal I than using coefficients by DCT and wavelets.

We use two standard test images, "Lena" and "Barbara", for our compression evaluations. Each test image is a 512×512 grayscale picture. To evaluate compression performance, we measure the reconstruction quality with peak signal-to-noise ratio (PSNR), the byte rate, and the compression time. The byte rate indicates the size of the data transmitted by the mote while the compression time gives an indication of the amount of energy expended while compressing. Each test image is compiled into the program code and copied to a memory buffer in run-time before compression on the mote, while the reconstruction is performed on the computer. All time measurements are averaged over 1000 trials.

Figure 5 shows selected results from our experiments. Since JPEG is a widely used compression scheme with well understood performance, we leave out its Rate-Distortion curve for space reasons. For the Time-Distortion curve using JPEG, an important observation is that if we extrapolate the curve to the point where quality goes to 0, we arrive at the computational overhead of JPEG, i.e., we still require a computational time of 70 ms per image. In contrast, there is almost no overhead in CS: If we look at the its Time-Distortion curve, as the reconstruction quality tends towards zero, the computation time required also tends towards zero.

4.2. Single Target Tracking via Background Subtraction

Due to the inherent richness of the visual medium, video-based scene analysis (such as tracking, counting, and recognition) typically requires a background-subtraction step that focuses attention of the system on smaller regions of interest in order to reduce the complexity of data processing. A simple approach to background subtraction from video data is via frame differencing [27]. This approach compares each incoming frame with a background image model and classifies the pixels of significant variation as part of the foreground. The foreground pixels are then processed for identification and tracking. The success of frame differencing depends on the robust extraction and maintenance of the background model. Some of the known challenges include illumination changes, vacillating backgrounds, shadows, visual clutter, and occlusion [31].

Because there is no single background model that can address all these challenges, the model must be selected based on application requirements. In the context of camera motes, the computational complexity of the algorithm and run-time are also crucial factors, which justifies our choice of a simple background subtraction technique such as frame differencing in this demonstration.

The data flow in Figure 6 shows the target-detection algorithm based on background subtraction. The first component performs two tasks: a background-foreground segmentation and an update on the background model B_t. An initial mask $M_t^0(i,j) := |I_t(i,j) - B_t(i,j)| > \tau$ is set based on a specified threshold τ, and then post-processed by using median filtering (a 9×9 block is used in our examples). All contiguous foreground regions in M_t^0 are grouped together as blobs and any blob smaller than a specified threshold is removed. The result is a segmentation output M_t, a binary array with a value of 1 for foreground and 0 for background. Finally, a set

[4] We have compared the reconstruction quality using ± 1 random coefficients with general real Gaussian coefficients, and we found the difference is minimal.

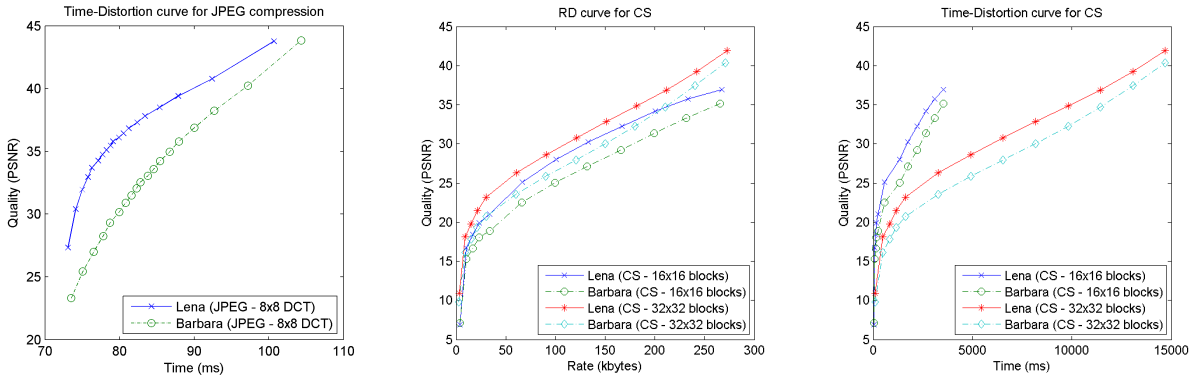

Fig. 5. (Left) Time-Distortion curve for JPEG. (Middle) Rate-Distortion curve for CS with random projections. (Right) Time-Distortion curve for CS with random projections.

of boxes S_t bounding the resulting blobs in M_t are computed and used for tracking. The tracking results from a single camera view are displayed in Figure 7.

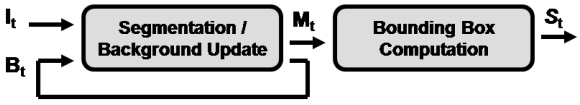

Fig. 6. Data-flow diagram of mote target-detection algorithm.

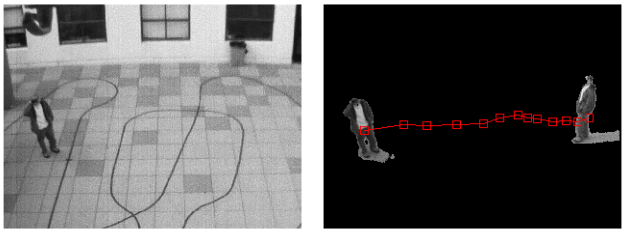

Fig. 7. Tracking results for a single camera view. (Left) Sample image. (Right) The path represents the motion of the center of mass of the target.

Figure 8 illustrates the data flow of the entire system. In this example, two CITRIC motes perform background subtraction in real-time. The resulting images are stored locally on the motes and downloaded offline (left side of Figure 8). Features based on the observations are sent to the server via radio, i.e., the coordinates of the bounding box. A client can then log on to the server and receive information streamed from the motes. On the right side of Figure 8 we observe a screenshot of the current visualization available on the client. The left plot in this visualization is a diagram of the floor plan where the camera motes are located, with rays specifying the angles in which there was a detection. The right plot depicts bounding boxes for the corresponding camera mote.

The execution time per frame for background subtraction and the bounding box computation is typically 0.2 – 0.4 s at

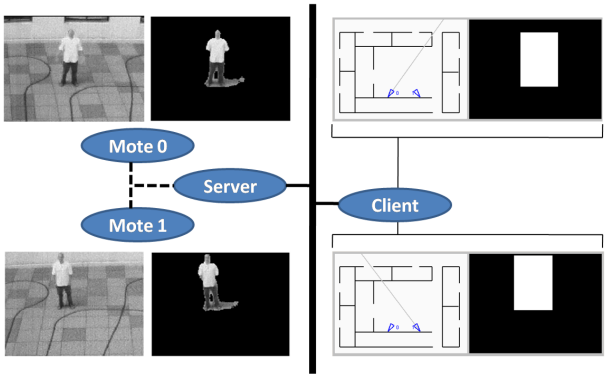

Fig. 8. Data diagram for Target Tracking application. Motes with corresponding local observations are displayed to the left, and client visualization to the right.

a resolution of 320×240, and 0.3 – 0.8 s at 640×480. The frame rate is not fixed due to the variable execution time of the algorithm depending on the number of foreground pixels.

4.3. Camera Localization using Multi-Target Tracking

Most camera networks benefit from knowing where the sensors are located relative to one another in a common coordinate frame, i.e., localization. Camera localization is the process of finding the position of the cameras as well as the orientation of each camera's field of view. In this subsection, we demonstrate a localization method based on [18]. The method simultaneously tracks multiple objects and uses the recovered tracks as features to estimate the position and orientation of the cameras up to a scale factor.

The tracks of the moving objects are formed in the image plane using a single point per object at each time instance. In our experiment, this point is the center of the bounding box around a detected foreground object. Over the whole surveillance sequence of duration T, there are K unknown number of objects. Denote the number of independent objects detected at time t as n_t and the set of their coordinates in the

image as $y_t = \{y_t^j \in \mathbb{R}^2 \ : \ j = 1, \ldots, n_t\}$. Then, the set of all objects during the whole sequence is $Y = \cup_{t=1,\ldots,T} y_t$. The problem of multi-target tracking is defined as a partition of Y

$$\omega = \{\tau_0, \tau_2, \ldots, \tau_K\},$$

where τ_0 collects the false alarm samples, and τ_i, $i = 1, \ldots, K$, collect the samples that form the K tracks, respectively.

One common drawback for target tracking via direct background subtraction is that the method is not stable enough for tracking multiple moving objects. In [22], the multi-target tracking algorithm models each of the K tracks using a linear dynamic model. Then, the maximum a posteriori (MAP) probability $P(\omega|Y)$ is optimized using Markov Chain Monte Carlo Data Association (MCMCDA). Hence, the optimal partition ω^* corresponds to the tracks of K objects in the image sequence. The solution allows us to build tracks from multiple moving objects at any given time and use more information from the dynamics of the scene than just using the background subtraction results directly.

In our experiment, we positioned two camera motes 8.5 feet apart and pointed them at an open area where people were walking, as shown by the top row of pictures in Figure 9. Each camera mote ran background subtraction on its current image and then sent the bounding box coordinates back to the base station for each detected foreground object. The center of each bounding box was used to build the image tracks over time on the base station computer, as shown in Figure 10. It can be seen that multiple tracks are successfully estimated from the image sequence.

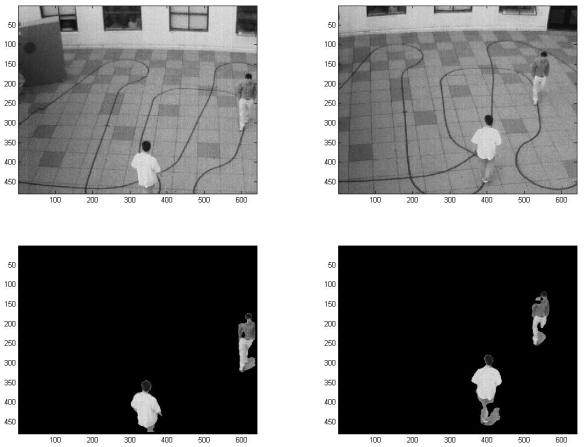

Fig. 9. (Top) Image frames from the left and right camera motes, respectively, viewing the scene. (Bottom) The detected foreground objects from the scene.

The localization algorithm then takes these tracks and performs multiple-view track correspondence. This method is particularly suitable for a low-bandwidth camera network because it works well on wide-baseline images and images lacking distinct static features [18]. Furthermore, only the coordinates of the foreground objects need to be transmitted, not entire images. In implementing the localization, tracks from the two image sequences are compared, and we adapt the method such that minimizing reprojection error determines which tracks best correspond between images. We used 43 frames from the cameras at an image resolution of 640×480. Foreground objects were detected in 22 of the 43 frames and tracks were built off these detected foreground objects. Four tracks were built in the first camera and five tracks were built in the second camera. Using the adapted localization method, we were able to determine the localization of the two cameras relative to one another with an average reprojection error of 4.94 pixels. This was based on the matching of four tracks between the two cameras which minimize the reprojection error.

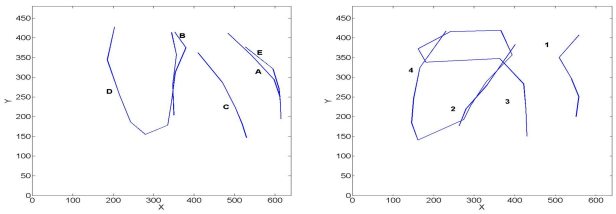

Fig. 10. The tracks of the moving objects in the image planes of the left and right camera motes, respectively, formed by MCMCDA.

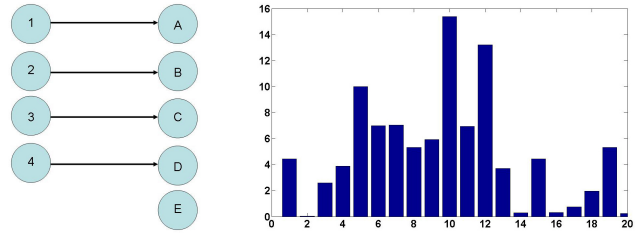

Fig. 11. (Left) The matching of tracks between the cameras that were used for localization. (Right) The reprojection error measured in pixels for each of the 20 points of the tracks.

The accuracy of the camera localization estimate is affected by a few factors. First, the choice of the (low-cost) camera has an effect on the quality of the captured image. Second, the precision of the synchronization between the cameras affects the accuracy of the image correspondence. Last, we only used a small number of frames to estimate track correspondence. Using a longer image sequence with more data points can reduce the estimation error.

5. CONCLUSION AND FUTURE WORK

We have presented the architecture of CITRIC, a new wireless camera mote system for low-bandwidth networks. The system enables the captured images to be processed locally on the camera board, and only compressed low-dimensional features are transmitted through the wireless network. To this

end, the CITRIC mote has been designed to have state-of-the-art computing power and memory (up to 624 MHz, 32-bit XScale processor; 64 MB RAM; 16 MB ROM), and runs embedded Linux. The mote communicates over the IEEE 802.15.4 protocol, which also makes it easy to integrate with existing HSNs.

We plan to improve the usability of our system by enabling clients to manage and interact with clusters of motes instead of individual motes. We will also expand the available C library of image processing functions on our camera motes and evaluate their performance. The platform will enable investigators to explore different distributed camera network applications.

6. REFERENCES

[1] I. Akyildizand, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Comm Mag*, 40(8):102–116, 2002.

[2] G. Bradski, A. Kaehler, and V. Pisarevsky. Learning-based computer vision with Intel's Open Source Computer Vision Library. *Intel Technology Journal*, May 2005.

[3] E. Candès. Compressive sampling. In *Proc of Int Congress of Math*, 2006.

[4] E. Candes and J. Romberg. Practical signal recovery from random projections. In *Wavelet App in Signal and Image Proc, SPIE*, 2004.

[5] Crossbow Inc. *Imote2: High-Performance Wireless Sensor Network Node Datasheet*, 2008.

[6] P. DiPaolo. NOR continues to battle NAND flash memory in the handset. `http://www.wirelessnetdesignline.com/showArticle.jhtml?articleID=165701488`, 2005.

[7] D. Donoho and M. Elad. Optimal sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization. *Proc of NAS of USA*, pages 2197–2202, March 2003.

[8] I. Downes, L. Rad, and H. Aghajan. Development of a mote for wireless image sensor networks. In *COGIS*, 2006.

[9] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *Pervasive Computing*, 1(1):59–69, 2002.

[10] W. Feng, E. Kaiser, W. Feng, and M. L. Baillif. Panoptes: scalable low-power video sensor networking technologies. *ACM Trans Multi Comp Comm Appl*, 1(2):151–167, 2005.

[11] H. Gharavi and S. Kumar. Special issue on sensor networks and applications. *Proc of IEEE*, 91(8):1151–1256, 2003.

[12] Intel Corp. *Intel PXA270 Processor Electrical, Mechanical and Thermal Specification Datasheet*, 2004.

[13] M. Kintner-Meyer and R. Conant. Opportunities of wireless sensors and controls for building operation. *Energy Engineering Journal*, 102(5):27–48, 2005.

[14] R. Kleihorst. Personal communication about Xetal-II WiCa platform, June 2008.

[15] R. Kleihorst, A. Abbo, B. Schueler, and A. Danilin. Camera mote with a high-performance parallel processor for real-time frame-based video processing. In *ICDSC*, pages 109–116, Sept. 2007.

[16] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu. SensEye: a multi-tier camera sensor network. In *MULTIMEDIA*, pages 229–238, 2005.

[17] A. LaMarca, W. Brunette, D. Koizumi, M. Lease, S. Sigurdsson, K. Sikorski, D. Fox, and G. Borriello. Making sensor networks practical with robots. In *PERVASIVE*, pages 152–166, 2002.

[18] M. Meingast, S. Oh, and S. Sastry. Automatic camera network localization using object image tracks. In *ICCV*, 2007.

[19] Moteiv Corp. *Tmote Sky Datasheet*, 2003.

[20] M. Nekovee. Ad hoc sensor networks on the road: the promises and challenges of vehicular ad hoc networks. In *Workshop on Ubiq Comp and e-Research*, 2005.

[21] NXP Semiconductor. PCF50606: How to connect to the Intel Bulverde application processor. `http://www.nxp.com/acrobat_download/literature/9397/75009763.pdf`.

[22] S. Oh, S. Russell, and S. Sastry. Markov chain Monte Carlo data association for general multiple-target tracking problems. In *CDC*, 2004.

[23] S. Oh, L. Schenato, P. Chen, and S. Sastry. Tracking and coordination of multiple agents using sensor networks: System design, algorithms and experiments. *Proc of IEEE*, 95:234–254, 2007.

[24] Omnivision Technologies Inc. *OV9655 Color CMOS SXGA (1.3MegaPixel) CAMERACHIP with OmniPixel Technology Datasheet*, 2006.

[25] C. Park and P. Chou. eCAM: ultra compact, high data-rate wireless sensor node with a miniature camera. In *SenSys*, pages 359–360, 2006.

[26] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *IPSN/SPOTS*, 2005.

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

[28] Silicon Lab. CP2102 single-chip USB to UART bridge. `http://www.silabs.com/public/documents/tpub_doc/dsheet/Microcontrollers/Interface/en/cp2102.pdf`, 2007.

[29] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Comm of ACM*, 47(6):34–40, 2004.

[30] T. Teixeira, D. Lymberopoulos, E. Culurciello, Y. Aloimonos, and A. Savvides. A lightweight camera sensor network operating on symbolic information. In *Proc of the First Workshop on Distributed Smart Cameras*, 2006.

[31] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *ICCV*, pages 255–261, 1999.

[32] UC Berkeley and TinyOS Open Source Community. TinyOS community forum. `http://www.tinyos.net/`.

[33] O. Vargas. Achieve minimum power consumption in mobile memory subsystems. `http://www.eetasia.com/ART_8800408762_499486_TA_673f1760.HTM`, 2006.

[34] A. Willig, K. Matheus, and A. Wolisz. Wireless technology in industrial networks. *Proc of IEEE*, 93(6):1130–1151, 2005.

[35] Wolfson Micro. Wolfson WM8950. `http://www.wolfsonmicro.com/uploads/documents/en/WM8950.pdf`, 2008.