A Low-Bandwidth Camera Sensor Platform with Applications in Smart Camera Networks

PHOEBUS CHEN, KIRAK HONG, NIKHIL NAIKAL, S. SHANKAR SASTRY, DOUG TYGAR, POSU YAN, ALLEN Y. YANG
University of California, Berkeley, CA 94720, USA
LUNG-CHUNG CHANG, LEON LIN, SIMON WANG
Industrial Technology Research Institute, Hsinchu, Taiwan 310, ROC
EDGAR LOBATÓN
North Carolina State University, Raleigh, NC 27695, USA
SONGHWAI OH
Seoul National University, Seoul, Korea
PARVEZ AHAMMAD
Howard Hughes Medical Institute, Ashburn, VA 20147, USA

Smart camera networks have recently emerged as a new class of sensor network infrastructure that is capable of supporting high-power in-network signal processing and enabling a wide range of applications. In this paper, we provide an exposition of our effort to build a low-bandwidth wireless camera network platform, called CITRIC, and its applications in smart camera networks. The platform integrates a camera, a microphone, 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 wireless camera mote. With reasonably low power consumption and extensive algorithmic libraries running on a decent operating system that is easy to program, CITRIC is ideal for research and applications in distributed image and video processing. Its capabilities of in-network image processing also reduce communication requirements, which has been high in other existing camera networks with centralized processing. Furthermore, the mote easily integrates with other low-bandwidth sensor networks via the IEEE 802.15.4 protocol. To justify the utility of CITRIC, we present several representative applications. In particular, concrete research results will be demonstrated in two areas, namely, distributed coverage hole identification and distributed object recognition.

Categories and Subject Descriptors: I.2.9 [**Artificial Intelligence**]: Robotics—*Sensors*; I.4.8 [**Artificial Intelligence**]: Image Processing and Computer Vision—*Sensor fusion*

General Terms: Wireless Sensor Networks, Camera Sensor Architecture

Corresponding Author: A. Yang, TRUST Center, Cory Hall, Berkeley, CA 94720. Email: `yang@eecs.berkeley.edu`.

This work is partially supported by ARO MURI W911NF-06-1-0076 and the UC Berkeley TRUST (Team for Research in Ubiquitous Secure Technology) Center.

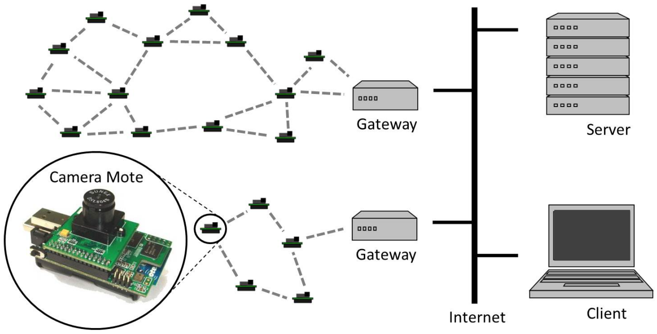


Fig. 1. An example of wireless camera network architecture.

1. INTRODUCTION

Traditionally, research in wireless sensor networks has been focused on low-bandwidth sensors, e.g., acoustic, infrared, and temperature, which limit the ability to identify complex, high-level physical phenomena. This limitation can be effectively addressed by integrating high-bandwidth sensors such as camera sensors. Recently, smart camera networks have emerged as a new class of sensor network infrastructure that is capable of supporting high-power in-network signal processing and enabling a wide range of applications, such as visual verification, object recognition, and tracking.

Figure 1 shows a typical network configuration for a surveillance system based on a camera network. The camera motes are networked with each other and possibly with other types of motes over standard wireless protocols such as IEEE 802.15.4 and Bluetooth. Some motes also communicate with gateway computers that are connected to the Internet. The motes would 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 supports different high-level applications.

In this paper, we provide a comprehensive review of our effort to design a wireless camera sensor platform, called CITRIC [Chen et al. 2008]. The platform is a wireless camera mote with a 1.3 megapixel camera, a PDA class processor, 64 MB RAM, and 16 MB FLASH. Given the existing technology constraints, our design has to make appropriate performance and usability tradeoffs for emerging smart camera network applications. These tradeoffs include computational power, power consumption, ease of programming, and ease of integrating with existing wireless sensor networks. The CITRIC platform enables a new set of in-network information processing techniques, and also provides a suitable infrastructure to support the development of high-level applications in the areas of sensor networks and computer vision.

The CITRIC platform has been successfully deployed as basic camera sensor infrastructure in several smart camera applications. Examples include recovery of camera network topology [Lobaton et al. 2009; Lobaton et al. 2010], distributed object recognition [Yang et al. 2009; Naikal et al. 2010], cooperative event detection and tracking [Wang et al. 2009; Wang et al. 2010], and traffic modeling and prediction [Shuai et al. 2010]. In the second part of the paper, we review these representative applications to validate our choice of performance and usability tradeoffs for the CITRIC platform. The reader is referred to the respective works for the implementation detail.

2. SMART CAMERA PLATFORMS: AN OVERVIEW

In the literature, the design of many existing camera motes consist of a camera-and-processor board and a networking mote, similar to the design of the CITRIC platform. Before we discuss the architecture of CITRIC in the next section, we first compare notable existing platforms in the literature. The comparison also justifies the need for a new smart camera platform such as CITRIC for the applications we are interested in. A good treatment on the baseline computation requirements for in-network image processing can be found in [Downes et al. 2006]. More detailed reviews of existing camera sensor platforms can be also found in [Akyildiz et al. 2008; Rinner et al. 2008].

Some platforms in the past focused on streaming video to a centralized server for processing, such as eCAM [Park and Chou 2006], 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 [Feng et al. 2005]. The latest version of the Panoptes platform consists of a Stargate "gateway mote," an 802.11b PCMCIA wireless card, and a USB camera. Panoptes targets applications where one would selectively stream video to conserve bandwidth. 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 [Rahimi et al. 2005], WiSN [Downes et al. 2006], and WiCa [Kleihorst et al. 2007] 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 [Kleihorst 2008].

The platform most similar to CITRIC is a prototype platform used by [Teixeira et al. 2006], which consists of an iMote2 [Memsic 2008] connected to a custombuilt 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.

Developed after iMote2 prototype camera mote and CITRIC, DSPcam [Kandhalu et al. 2009] is another surveillance-oriented camera sensor system. DSPcam uses a similar camera sensor from OmniVision as iMote2 and CITRIC, but uses a different processor (BF537) and has a smaller memory size. On the other hand, the platform integrates a WiPort communication module that supports a faster 802.11 b/g protocol. The software system is built on the uClinux OS and an open-source image processing library.

Finally, multi-tiered camera networks have also been proposed to use low-cost, low-power, and low-resolution camera motes to wake up higher-grade cameras to capture and process interesting images. One such notable multi-tier camera network system is SensEye [Kulkarni et al. 2005], which consists of 3 tiers of cameras.

Table I summarizes the key hardware specifications of the above smart camera systems.

3. CITRIC SMART CAMERA PLATFORM

3.1 Architecture

The CITRIC platform consists of a camera daughter board connected to a TelosB board [Polastre et al. 2005] (see Figure 2(a)), 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 cm × 5.8 cm processor board and a detachable image sensor board (see Figure 2(b)). The design of the camera board uses a small number of functional blocks to minimize size, power consumption, and manufacturing cost.

The two main choices for the onboard processor are field-programmable gate arrays (FPGAs) and 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-known image processing and computer vision algorithms have been efficiently coded in C/C++, such as the OpenCV library [Bradski et al. 2005]. Therefore, we chose to use a general-purpose processor running embedded Linux (as opposed to TinyOS) for the camera board for rapid prototyping and ease of programming and maintenance.

In the following, we discuss our choice of imaging and microphone sensors, mobile processor, power management, and wireless networking modules for the CITRIC architecture. More extensive justification for choosing these modules is given in [Chen et al. 2008].

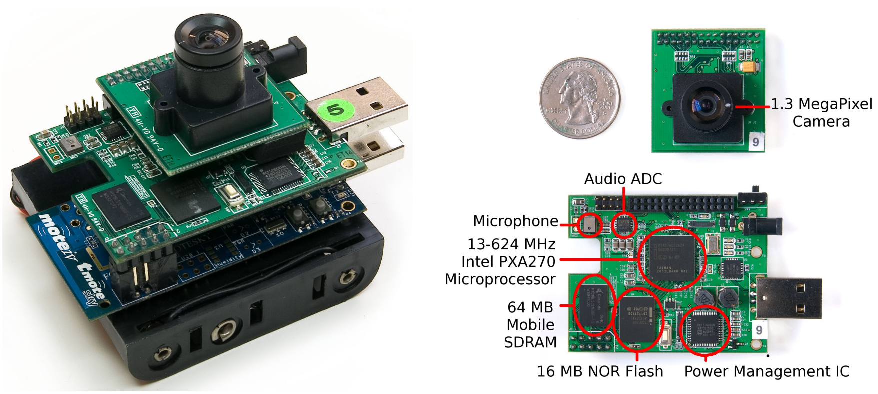
Table I. Comparison of existing wireless camera mote platforms with the new CITRIC mote platform.

Platform	Processor	RAM/ROM	Camera	Wireless
eCAM	OV528 Serial Bridge (JPEG Compression only)	-	COMedia C328-7640 OV7640 camera (640 × 480 @ 30 fps)	Eco node nRF24E1 radio (1 Mb/s, 10 m)
Panoptes	Intel XScale PXA255 (400 MHz, 32-bit CPU)	64/32 MB	Logitech 3000 USB (640 × 480 @ ≈ 13 fps) (160 × 120 @ ≈ 30 fps)	802.11 PCMCIA (11 Mb/s for 802.11b)
Cyclops	Atmel ATmega128L (7.3728 MHz, 8-bit CPU) Xilinx XC2C256 (16 MHz CPLD)	64/512 KB	ADCM-1700 (352 × 288 @ 10 fps)	Mica2 mote TR1000 radio (40 kbps)
WiSN	Atmel AT91SAM7S (48 MHz, 32-bit ARM7TDMI CPU)	64/256 KB	ADCM-1670 (352 × 288 @ 15 fps) ADNS-3060 (30 × 30 @ 100 fps)	CC2420 radio (802.15.4, 250 kbps)
WiCa	Xetal-II (84 MHz, 320 PE LPA + GCP)	1.75 MB/-	Philips (640 × 480 @ 30 fps)	Aquis Grain ZigBee CC2420 radio (802.15.4, 250 kbps)
iMote2 +Cam	Intel XScale PXA271 (up to 416 MHz, 32-bit CPU)	32/32 MB	OV7649 (640 × 480 @ 30 fps) (320 × 240 @ 60 fps)	CC2420 radio (802.15.4, 250 kbps)
DSPcam	Blackfin BF537 (600 MHz)	32/4 MB	OV9653 (1280 × 1024 @ 15 fps) (640 × 480 @ 30 fps)	WiPort (802.11 b/g)
CITRIC	Intel XScale PXA270 (up to 624 MHz, 32-bit CPU)	64/16 MB	OV9655 (1280 × 1024 @ 15 fps) (640 × 480 @ 30 fps)	TelosB mote CC2420 radio (802.15.4, 250 kbps)

3.2 Imaging and microphone sensors

The camera for the CITRIC 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 [Omnivision Technologies Incorporated 2006]. The typical active power consumption is 90 mW (15 fps @SXGA) and the standby current is less than $20\,\mu A$.

In order to run high-bandwidth, multi-modal sensing algorithms that utilize both audio and video sensor outputs, it is important to include a microphone on the camera daughter board rather than use a microphone attached to the TelosB wireless mote. This simplifies the operation of the entire system by dedicating the communication between the TelosB and the camera daughter board to data that needed to be transmitted over the wireless network. The microphone on the board is con-



(a) (b)

Fig. 2. CITRIC mote. (a) Assembled camera daughter board with TelosB. (b) Camera daughter board with major functional units outlined.

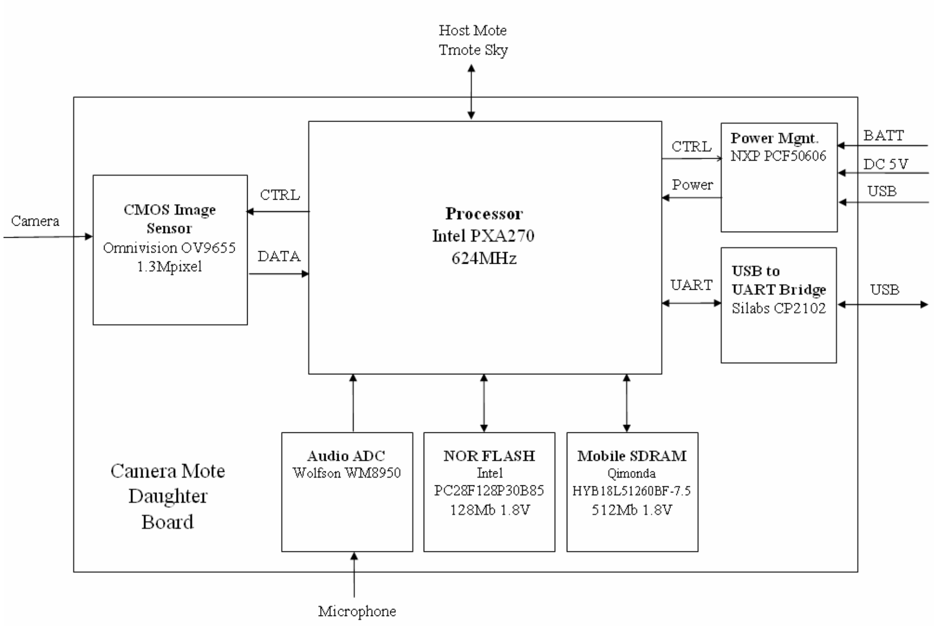


Fig. 3. Block diagram of major components of the CITRIC camera daughter board.

nected to the Wolfson WM8950 mono audio ADC [Wolfson Micro PLC 2008], which is 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.

3.3 Processor and Memory

The PXA270 [Marvell Corporation 2010] 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, which can be set in a program by sending specific bit sequences to an I/O memory address.

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 × 1.3 megapixel × 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.

3.4 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 [NXP Semiconductor 2003]. The entire camera mote, including the TelosB, is designed to be powered by either four AA batteries, a USB cable, or a 5 V DC power adapter cable.

3.5 Wireless Communications

Sensor data in a CITRIC system are designed to flow from the motes to a gateway over the IEEE 802.15.4 protocol, then from the gateway over an 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 the 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.

4. PERFORMANCE BENCHMARKS

4.1 Energy Consumption

The power consumption of the camera mote is determined by logging the current and voltage of the device when it is connected to four AA batteries (outputting $\approx 6\,\text{V}$). A Tektronix AM 503B Current Probe Amplifier is used to convert current to voltage,[1] and a National Instruments 9215 USB data logger is used to log both the voltage of the batteries and the voltage of the current probe.

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

We also measure 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 is 970 mW. Note that the power consumption may be reduced by enabling power management on the Tmote. Using four fully charged AA batteries each with a capacity of 2870 mAh, the average life span of the CITRIC mote continuously running background subtraction ranges from 5.7 hours at 520 MHz to 7.5 hours at 208 MHz.[2]

4.2 CPU Speed Benchmarks

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

(1) The *Add* benchmark adds two arrays.

(2) 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).

(3) The *Median Filter* benchmark performs smoothing by taking the median pixel value of a 3×3 pixel area at each pixel.

(4) The *Canny* benchmark implements the first stage of the Canny edge detection algorithm.

The benchmark results for *Add* and *Background Subtraction* are averaged over 1000 trials, while those for *Median Filter* and *Canny* are averaged over 100 trials.

Two sets of benchmarks are performed. Figure 4 shows the average run time for each function when the processor speed varies from 208 MHz to 520 MHz, while the image resolution is fixed at 512×512. Figure 5 shows the average run time in the

[1] The current probe device converts current to voltage through the use of inductance.

[2] In comparison, the battery life of a completely idle CITRIC lasts approximately 20 hours.

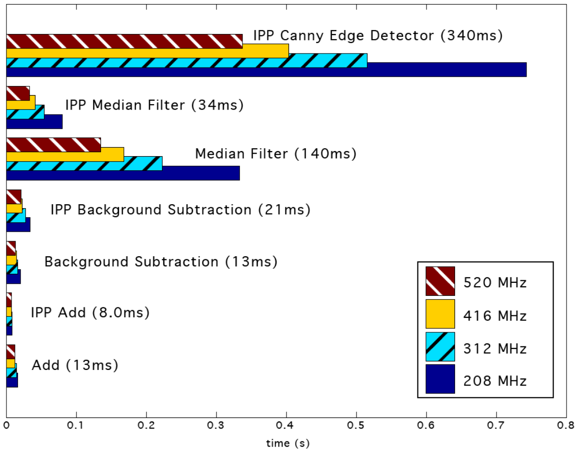


Fig. 4. Average run time of basic image processing functions on 512 × 512 images. The fastest run time at 520 MHz is shown in parentheses.

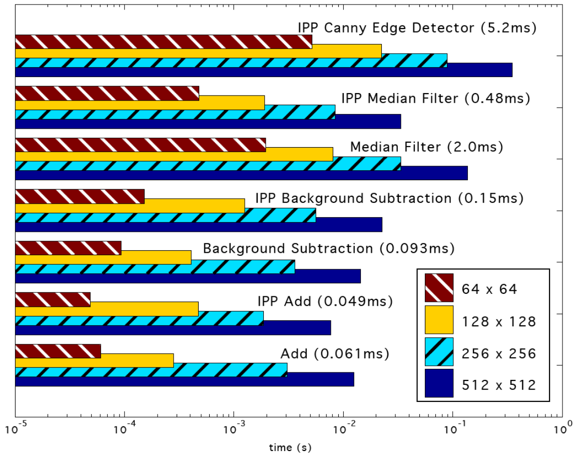


Fig. 5. Average run time (in log scale) of basic image processing functions on various image resolutions. The speed of the CPU is fixed at 520 MHz. The fastest run time (on 64 × 64 images) is shown in parentheses.

logarithmic scale when the image resolution varies from 512 × 512 to 64 × 64, while the CPU speed is fixed at 520 MHz.

First, 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 in 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 achieve optimal performance.

Second, the non-linear performance curve for different CPU frequencies in Figure 4 can be attributed to the constant speed of memory access (the bus speed is 208 MHz regardless of the processor speed).

Finally, in Figure 5, we can see that the change in the average run time of each image processing function with respect to different image resolutions largely depends on the complexity of the algorithm. For simple calculations such as image addition and background subtraction, the computational complexity (and hence the average run time) grows proportionally to the total number of pixels in the image. For more expensive algorithms such as median filter and edge detector, reducing the image resolution can still significantly improve the real-time performance. For example, the fastest run time to execute Canny edge detector on a 64 × 64 downsampled image is 5.2 ms.

4.3 Multihop Route Communication Performance

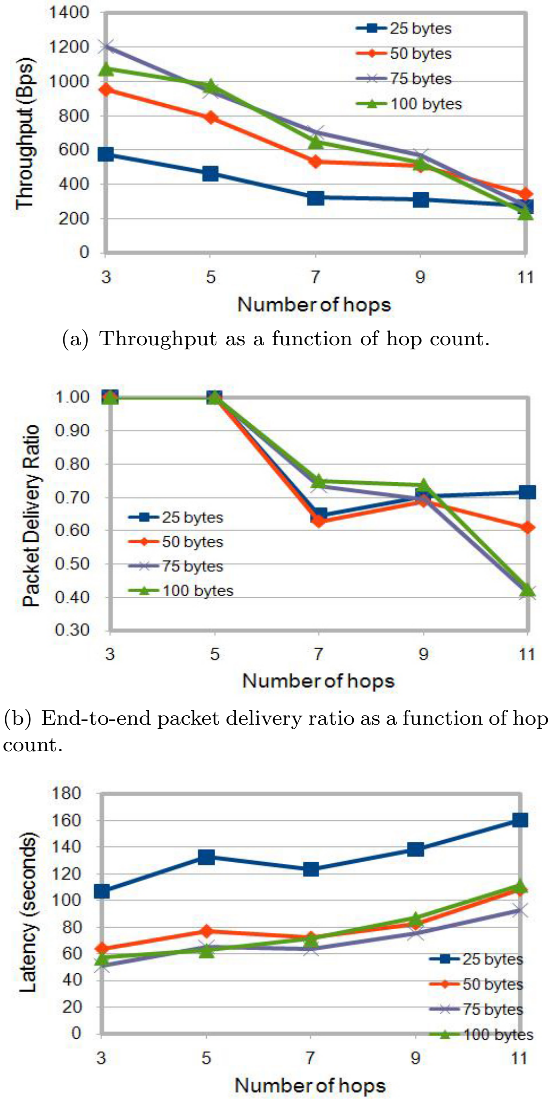
Although the CITRIC mote is designed for camera sensor networks that mainly perform in-network processing and consume very little communication bandwidth, some applications occasionally may transmit bursts of images over wireless channels using multihop routing. This bursty, heavy traffic may result in congestion and unpredictable behavior. In this experiment, we study the end-to-end performance of communicating over a multihop route with a complete communication stack that includes the popular WSN routing protocol, collection tree protocol (CTP) [Gnawali et al. 2009], under bursty, heavy traffic loads [Hong et al. 2009]. CTP is a tree-based, address-free, multihop routing protocol implemented in TinyOS. We choose to evaluate the communication overhead introduced by CTP because of its widespread use in WSNs and because its source code is readily available. The experiments below are meant to help practitioners find the optimal network configurations for different application scenarios.

In order to study the behavior of the network with a fixed number of hops, we first need to modify the routing engine of CTP such that the path from a source CITRIC mote to the gateway is fixed. As such, we are not studying routing instability (the routing algorithm changing paths). Along the path, a line of TelosB motes relays images from the source to the gateway. Each node is placed within the interference range of all the other nodes to test the performance of CTP under the worst network congestion conditions. The image data to be transmitted from the source node is 60 kByte, which is equivalent to the file size of a typical 640×480 JPEG color image that captures a natural scene.

We conduct two experiments to measure the throughput, end-to-end packet delivery ratio, and latency under different packet payload sizes and packet generation rates. Here, the packet generation period (1/rate) is the time the source CITRIC mote waits between sending successive packets containing the parts of an image. Each experiment include trials for 3, 5, 7, 9, and 11-hop topologies.

In the first experiment, we measure the network performance under 25, 50, 75, and 100 byte packet payloads.[3] The results are shown in Figure 6.

[3] The maximum packet size of an IEEE 802.15.4 packet is 127 bytes, and each packet has a header of 25 bytes consisting of the CTP, IEEE 802.15.4, and CITRIC mote specific header fields.



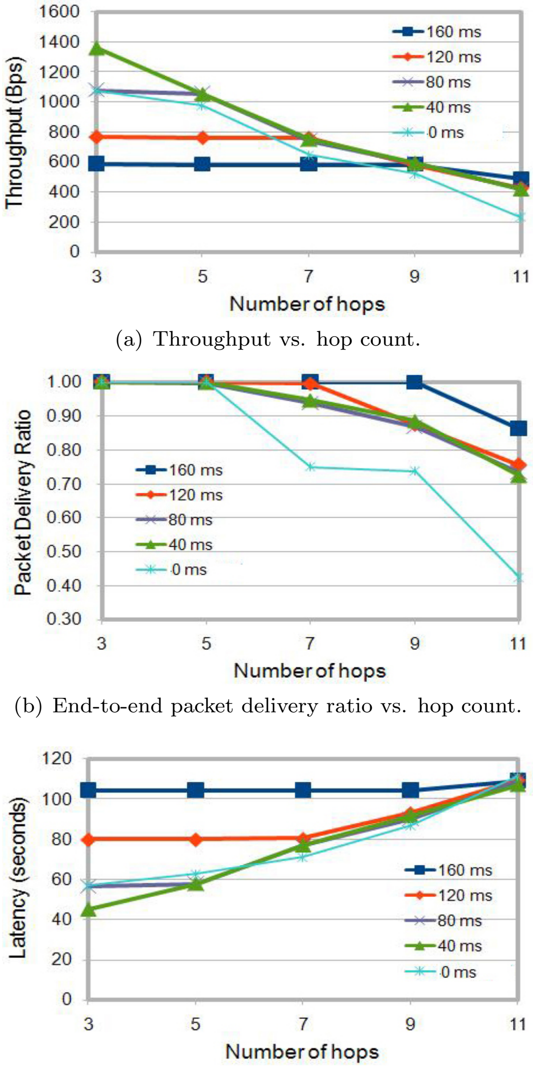
(a) Throughput as a function of hop count.

(b) End-to-end packet delivery ratio as a function of hop count.

(c) Latency to send a 60 kB image as a function of hop count.

Fig. 6. Network performance with different payload sizes while using the maximum packet generation rate (0 ms delay between generating/sending successive packets at the source).

As expected, the throughput and end-to-end packet delivery ratio decreases with increasing hop counts because of increased self-interference. In addition, experiments with a payload size of 100 bytes shows slightly lower throughput than the other smaller payload sizes when the number of hops increases. This is because a larger payload will take longer to transmit and have a higher chance of collision with other packets (CTP runs on a CSMA MAC layer).



(a) Throughput vs. hop count.

(b) End-to-end packet delivery ratio vs. hop count.

(c) Latency to send a 60 kB image vs. hop count.

Fig. 7. Network performance with different packet generation periods while using a 100-byte packet payload (0 ms means no delay between generating/sending successive packets at the source).

In the second experiment, we measure network performance when the packet generation period is 0,[4] 40, 80, 120, and 160 ms. Packet generation intervals over 160 ms are not tested because we rarely observe retransmissions at 160 ms intervals. The results are shown in Figure 7.

Note that using a packet generation period of 0 may not necessarily yield the best throughput and packet delivery ratio in a multihop network. For instance, a packet

[4] There are always packet ready for transmission when the package generation period is 0.

generation period of 40 ms performs better than other packet generation periods in many of the experiment runs. Also, the throughput and packet delivery ratio for different packet generation periods start to drop at different hop counts. This shows that the packet generation period can be optimized based on the network density.

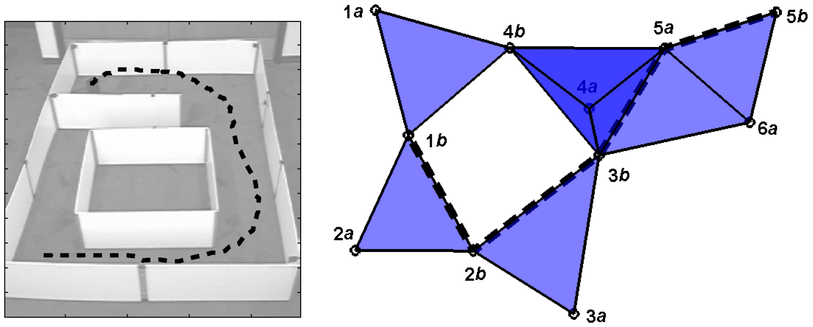
5. DISTRIBUTED COVERAGE HOLE IDENTIFICATION

In the next two sections, we present two applications of CITRIC camera networks, which demonstrate the utility of the computation and communication capabilities of these systems. The first application is recovering the coverage of a distributed camera network, which is defined as the set of locations in the physical space in which an agent is visible by at least one of the cameras. Extracting topological information (e.g., number of holes) about this coverage can enable surveillance and optimal message routing in the network without the need for calibration of the cameras. Holes in the coverage may be caused by lack of coverage or object occlusion in the scene. Identifying holes in a camera network helps us determine where new sensors need to be placed, and it also helps extract geometric information about the environment, such as the existence of circular corridors, without relying on exact localization. These two features are essential for ad-hoc camera networks.

Recovering network coverage and topology from sensor data has been a classical problem in sensor networks. Many prior works [Meguerdichian et al. 2001; de Silva and Ghrist 2007] have focused on estimating the static *communication graph* of a sensor network. In this approach, each sensor node would be assumed to occupy a radially symmetric broadcast region and a symmetric sensing region. However, for camera sensor networks, it is more relevant to consider recovering the *visual coverage* of the network, because the common field of view of a camera sensor is not radially symmetrical and is susceptible to various image nuisances (e.g., occlusion and false detection). For example, computing the *camera adjacency graph* (CAG) has been the topic of several recent studies, which are based on either *structure-from-motion* techniques [Cheng et al. 2007; Turcot and Lowe 2009] or object tracking algorithms [Khan and Shah 2003; Rahimi et al. 2004; Calderara et al. 2005]. One of the main constraints of the CAG approach is that, due to the computational complexity of extracting and matching robust image features across multiple images, the algorithm/system typically requires considerable power and communication bandwidth, and therefore is implemented in a centralized fashion.

5.1 A Topological Approach

Our objective of distributed coverage hole identification is the distributed aggregation of observations from the nodes in order to localize holes in the coverage of the camera network. An approach for solving this problem requires the estimation of the overlap between cameras in order to extract some topological invariants [Lobaton et al. 2009; Lobaton et al. 2010] without localizing the cameras. In Figure 8, we illustrate a graphical representation of the coverage of a CITRIC camera network. The representation is in a distributed fashion based on the CN-Complex, a concept that will be introduced below. As our focus is to identify holes in the coverage of a camera network, we can assume that all the nodes are synchronized, a communication network has been established, and they are capable of detecting



(a)

(b)

Fig. 8. A graphical representation of the *CN*-Complex, built using CITRIC camera network in Figure 9. (a) The physical layout with a sample path for the target. (b) The *CN*-Complex where the mapping of the path is determined by specifying the sequence of cameras for which the target is visible. Each vertex represents a single decomposed region; edges correspond to the pairwise intersection between regions; triangles correspond to the three-way intersections, etc. Coordinates for the vertices are assigned arbitrarily and are irrelevant in the identification of holes using the homology of the complex.

agents in their field of view.[5]

The distributed algorithm to find coverage holes is illustrated in Figure 9, in which the CITRIC cameras are used and no calibration information is assumed (i.e., camera locations and the scene are unknown). The paper [Lobaton et al. 2009] first introduced a simplicial representation called the ***CN*-Complex**, which captures accurate topological information about the coverage of the network. The *CN*-Complex is built by following two simple steps:

Step 1. Decomposing the coverage of each camera utilizing the observed occlusions of agents moving through the scene, and

Step 2. Determining the overlap between the resulting regions.

The *CN*-Complex is the nerve complex of the decomposed regions. In simple words, the nerve of a collection of regions is a list where each element corresponds to the sets of regions with a non-empty intersection. The *CN*-Complex can be used to identify holes in the coverage by computing the *homology* from the simplicial complex.

Using the *CN*-Complex model, the coverage hole of the test bed shown in Figure 9 can be established as follows (see [Lobaton et al. 2009] for more details). Each mote observes a remote controlled car moving through the environment. Occluding contours are detected in the image domain of each camera as the target appears and disappears behind the obstacles. Figure 9(b) illustrates how these contours can be used to decompose the image domain of camera 5, which in turn specifies a decomposition of the coverage of the camera (Figure 9(c)). Figure 9(d) illustrates

[5] Synchronization can be accomplished by methods such as the *Flooding Time Synchronization Protocol* (FTSP) [Maroti et al. 2004] for TinyOS. Agents can be detected using simple computer vision algorithms such as background subtraction for which some benchmarks have been provided for the CITRIC platform in Section 4.2.

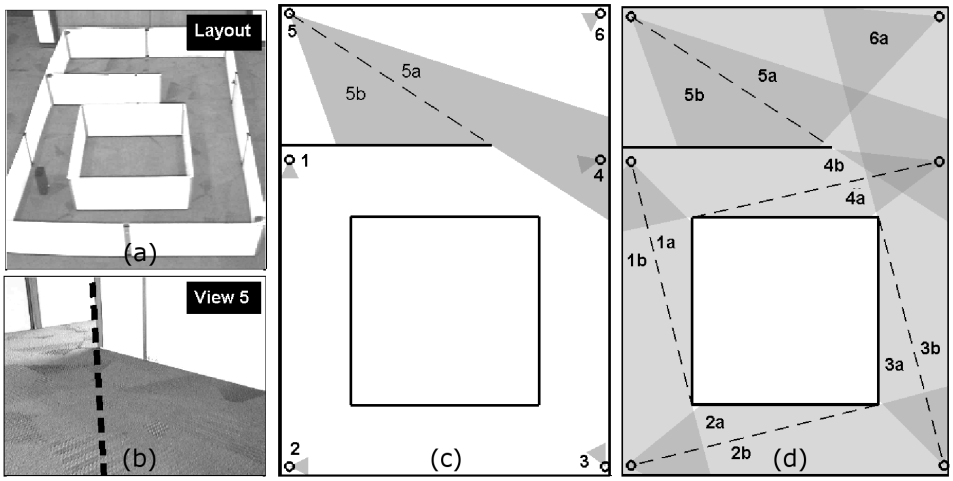


Fig. 9. A network setup consisting of six CITRIC motes. (a) The testbed in which the network is deployed and where a remote controlled car is used as a target. (b) Occluding contours detected in the view of camera 5 are used to decomposed its image domain. (c) This decomposition of the image domain leads to a decomposition of the coverage of camera 5 into regions $5a$ and $5b$. (d) The decomposed regions for all cameras are shown.

all of the decomposed regions (e.g., the coverage of camera 1 is decomposed into region $1a$ and $1b$). Note that this process is performed locally without transmitting any information or using calibration of the cameras. The detections of the target is done using the background subtraction function described in Section 4.2 followed by a median filtering step. As observed in Figure 4, these computations can be done at about 8 frames per second using 512 × 512 images. The detection of occluding contours is accomplished by detecting the edges along which a target disappears and then fitting lines.

Next, the overlap between the resulting regions is obtained by having each mote transmit an appearance or disappearance message to the network every time that an agent enters or leaves a region. Given that the motes have already synchronized their clocks, this is sufficient information to determine the overlap between the decomposed regions. The overlap information is captured in the collection of sets of overlapping regions, the nerve. The following collection determines all overlaps in the network:

$$\{[1a\ 1b\ 4b],\ [1b\ 2a\ 2b],\ [2b\ 3a\ 3b],\ [3b\ 4a\ 4b\ 5a],\ [3b\ 5a\ 6a],\ [5a\ 5b\ 6a]\}\,. \qquad (1)$$

This collection specifies an abstract simplicial complex that is depicted visually in Figure 8(b). This is a visual representation displayed to a user logging into the CITRIC network. Note that this representation captures the fact that there is a hole in the coverage corresponding to a circular corridor. In [Lobaton et al. 2009], it is proven that the decomposition step using the occluding contours in the images is essential to guarantee that the complex accurately captures topological information of the network coverage.

During the construction process using the CITRIC platform, the amount of information transmitted is minimal (a few transmission per minute) since the only observations transmitted were due to the agent moving in and out of the view of cameras. Note that direct streaming of video from multiple motes to a centralized location could not be supported by our network due to bandwidth and battery power limitations.

Finally, a distributed version of the *CN*-Complex that uses observations from multiple unidentifiable agents has been proposed without localization information of the cameras or the objects in the scene by exploiting temporal correlation of the detections of unidentified agents [Lobaton et al. 2010]. Imperfect foreground regions can be used to identify occluding contours in the image domain of each camera, and then the detection results of agents entering or leaving each of these regions can be aggregated in a distributed fashion in order to discover overlaps between camera regions. This process requires the storage of a few arrays of size equal to an image which could be easily accommodated by the CITRIC platform thanks to its 64 MB of SDRAM. In [Lobaton et al. 2010], the *CN*-Complex is also used for tracking unidentified agents.

5.2 Discussion

This particular application illustrates a family of distributed algorithms that require real-time processing of images with low-bandwidth communication requirements. The following are the key hardware requirements for the implementation of these algorithms:

Video Processing. The video from the cameras need to be processed in real-time in order to detect agents in the field of view of each node. Since the only events broadcasted by the nodes are the entrance/exit of agents within each camera view, the camera resolutions and frame rates need to be sufficient to identify these events. Each node needs to have enough memory and computational power in order to achieve this goal. A frame rate of 5 fps with an image resolution of 512 × 512 pixels was sufficient for the experiments outlined in this section. However, higher temporal and special resolution would be required for more complex scenarios.

Communication. For this application, the bandwidth requirements were minimal. Latency of packets is not a concern as long as enough agent detections can be stored in memory, which is the case for our platform. In order to further reduce the number of messages transmitted from each node, it is possible to aggregate detection events and transmit them as a batch of observations.

Power. As stated in Section 4.1, we expect a lifetime of about 5 hours for each node running the algorithm presented in this section, which requires the continuous detection of agents using background subtraction. However, as the probabilities of overlaps between cameras can be learned, they can be used to predict the transitions of agents tracked in the network. Under these assumptions, the camera daughter board in each node could be placed in *Idle* mode whenever there are no agents present nearby. This adaptation procedure can significantly extend the life of the network.

6. DISTRIBUTED OBJECT RECOGNITION

Object recognition is one of the most active topics in computer vision. As human perception is known to excel in recognizing complex objects with little effort in images, it is reasonable to expect that a well designed computer vision system may also perform the same functionality. Traditionally, this problem has been studied with respect to only single camera sensors. The development of smart camera sensors such as CITRIC has led to growing interests to be able to recognize object images in a distributed fashion. One of the advantages is that the multi-view information can effectively compensate many well-known visual nuisances in the shared field of view, such as object occlusion and pose variation, and hence boost the recognition accuracy. In this section, we discuss several novel algorithmic techniques that enable such distributed object recognition systems.

6.1 Literature Review

First, we shall briefly review existing methods to recognize object images on single cameras. In the literature, there exist two dominant approaches, namely, appearance-based methods and feature-based methods. Appearance-based methods classify query images based on their raw pixel values, edges, color histograms, or image gradients. The solutions usually appeal to only a small class of object categories such as human faces and vehicles. One of the challenges for appearance-based methods is that with varying illumination, camera vantage point, and object surface texture and shape, it is quite impossible to measure and train all appearances of the objects in question. Therefore, in our discussion, we will only focus on feature-based methods.

One influential theory in human vision explains the object recognition function on the basis of decomposing objects into constituent parts (i.e., distinctive image patches) [Oram and Perrett 1994; Agarwal and Roth 2002]. This approach in computer vision is generally referred to as *bag-of-words* (BoW) methods [Nistér and Stewénius 2006]. Local invariant features such as SIFT-type[6] features (e.g., SIFT [Lowe 1999], SURF [Bay et al. 2008], or CHoG [Chandrasekhar et al. 2009]) are first extracted from images. The vector representation of these features are also called *codewords*. Each codeword can be shared among multiple object classes. Hence, the codewords from all object categories can be clustered into a *vocabulary* (or codebook). The size of a typical vocabulary ranges from thousands to hundreds of thousands. One popular representation of the object features computes the frequencies of the instances of the codewords in an image, which is called a *histogram* [Nistér and Stewénius 2006; Chen et al. 2009] (as an example shown in Figure 10). As a result, the histogram becomes a compact representation of the object(s) that appear in the image.

[6] SIFT stands for *scale-invariant feature transform*. In computer vision, certain 2-D image features are considered viewpoint-invariant, such as corner points, T-junctions, and local extremal illumination regions, as their relative pixel values compared to the neighboring pixels are invariant to moderate camera viewpoint changes.

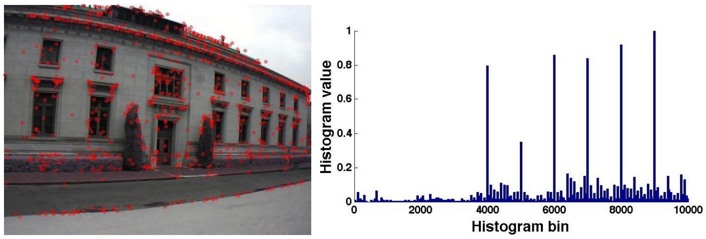


Fig. 10. Left: CHoG feature points detected in an image of a building. Right: The corresponding histogram vector based on a 10,000-D codebook.

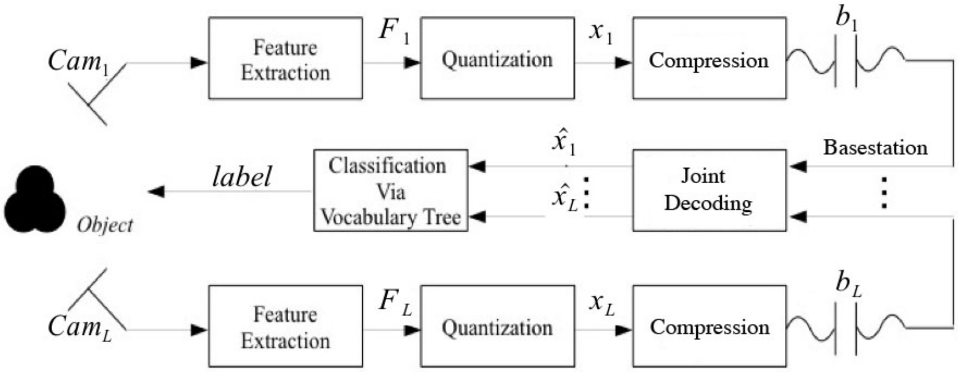


Fig. 11. The flow diagram of a distributed object recognition system consisting of multiple smart camera sensors and a base station computer.

6.2 Distributed Object Recognition with Camera Sensor Networks

Object recognition solutions have been demonstrated on several smart camera platforms, such as SensEye [Kulkarni et al. 2005] and Panoptes [Feng et al. 2005]. Using the CITRIC platform, we have recently studied more sophisticated distributed object recognition systems with bandwidth constraints [Yang et al. 2009; Naikal et al. 2010]. Figure 11 illustrates the flow diagram of a typical distributed object recognition system. The key idea is that the light-weight module that extracts and encodes high-dimensional image features should be implemented on the smart camera and can be deployed in a distributed fashion, while the second module that decodes the compressed features over the network and performs classification should be implemented on a base station computer.

To execute this design strategy, we note that the implementation of feature extraction on CITRIC is straightforward, as existing public computer vision code (e.g., SIFT and SURF) can be simply recompiled on the embedded Linux system. However, when the sensor measurements are received at the base station, it becomes more involved to achieve higher the recognition rate using multiple camera views compared to single cameras. More specifically, one needs to find good answers to the following problems:

(1) When a common object appears in a shared field of view, shared object features

and their correspondence need to be identified across multiple camera views.

(2) Once the local measurements from the camera sensors are transmitted to a network computer, the classification algorithms should be able to harness the multi-view information about the objects to boost the recognition accuracy.

In the rest of the section, we will highlight several approaches to these two problems.

The most important factor that one needs to consider in distributed object recognition is whether the sensors are allowed to exchange viewpoint information during the recognition. If information exchange between sensors is permitted, the power consumption to support such communication will increase significantly compared to the alternative scenario. This will be a major disadvantage in wireless sensor networks. Nevertheless, if the object of interest is a rigid body, its feature correspondence in multiple views can be established by standard *structure-from-motion* techniques [Turcot and Lowe 2009]. Using random sample consensus (RANSAC) algorithms [Fischler and Bolles 1981], the existing methods would iteratively sample a small set of image features that have similar vector representation in a pair of images to establish a hypothesis of a rigid-body motion. The motion model that achieves the highest consensus among the correspondences in the two views is declared as the optimal solution. In [Ferrari et al. 2004], the authors further proposed to improve the robustness of the matching process by building a *group of aggregated matches* (GAM) to cover larger image regions than individual features, and subsequently establishing a dense two-view correspondences using a sorted list of GAMs with the largest sizes.

In addition to the excessive power consumption involved in information exchange between sensors, the computational complexity of the structure-from-motion algorithms typically grows exponentially with the number of camera views. Namely, estimating consensus rigid-body models in three or four images is significantly more expensive than in a pair of images. Furthermore, if BoW methods are used for classification, the location information of the image features often does not play a significant role. Therefore, complementary to the above multi-view geometric algorithms, our works [Yang et al. 2009; Naikal et al. 2010] have studied signal processing algorithms to simultaneously compress high-dimensional visual histograms from multiple views and identify shared object features without resorting to prior training data, camera calibration, or inter-sensor communication.

Our proposal has been motivated by the theory of *compressive sensing* [Donoho 2006; Candès and Tao 2006] and its recent extension in sensor networks [Baron et al. 2006]. The key observation on the visual histograms that describe the object feature instances in individual images is that the representation is often *spare* on a large vocabulary, more specifically, all coefficients are nonnegative and most of them are (approximately) zero (as shown in Figure 10), as only a small number of features may be exhibited in a single image. Furthermore, since SIFT-type features are robust to some degree of camera rotation and translation, images from different vantage points may share a subset of the same features, thus yielding histograms with similar nonzero coefficient values. This phenomenon is called *joint sparsity* (JS).

Denote L as the number of the camera sensors that observe the same object in 3-D, and let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_L \in \mathbb{R}^D$ be the corresponding SIFT histogram vectors.

The problem of encoding multiple-view object images can be formulated as follows. For the high-dimensional histogram vectors extracted from the L images, define a JS model as

$$\begin{aligned} \boldsymbol{x}_1 &= \boldsymbol{x}_c + \boldsymbol{z}_1, \\ &\vdots \\ \boldsymbol{x}_L &= \boldsymbol{x}_c + \boldsymbol{z}_L, \end{aligned} \tag{2}$$

where $\boldsymbol{x}_c$ called the *common* component represents the JS pattern, and each $\boldsymbol{z}_i$ represents an *innovation* pertained to the ith view. Furthermore, both $\boldsymbol{x}_c$ and $\boldsymbol{z}_i$ are also sparse.

On each camera sensor i, a linear sampling function is sought using a random matrix $A_i \in \mathbb{R}^{d \times n}$:

$$f_i : \boldsymbol{x}_i \in \mathbb{R}^n \mapsto \boldsymbol{b}_i \doteq A_i \boldsymbol{x}_i \in \mathbb{R}^d. \tag{3}$$

At the base station, upon receiving $\boldsymbol{b}_1, \dots, \boldsymbol{b}_L$ compressed features, the multi-view JS model can be modeled in a single linear system:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_L \end{bmatrix} &= \begin{bmatrix} A_1 & A_1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \\ A_L & 0 & \cdots & 0 & A_L \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_c \\ \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_L \end{bmatrix} \\ \Leftrightarrow \qquad \boldsymbol{b}' &= A' \boldsymbol{x}' \in \mathbb{R}^{dL}. \end{aligned} \tag{4}$$

This underdetermined linear system of equations with respect to the unknowns $\boldsymbol{x}_c, \boldsymbol{z}_1, \dots, \boldsymbol{z}_L$ can be then solved via ℓ_1-minimization [Naikal et al. 2010]:

$$\min \|\boldsymbol{x}'\|_1 \quad \text{subj. to} \quad \boldsymbol{b}' = A' \boldsymbol{x}', \tag{5}$$

The equation (5) can be formulated as a linear programming problem, and has stable and efficient numerical solvers [Yang et al. 2010]. Using the JS model, one can flexibly choose different sampling rates (i.e., d/n) determined by the sampling matrix A_i for individual camera channels, and the subset of shared image features together with the innovations are simultaneously decoded without any prior information about the objects and camera locations.

6.3 Experiments

In the following, we present experimental results to validate the performance of the JS model under network bandwidth constraints. To aid peer evaluation, we have released a public multiple-view image database called the Berkeley Multiview Wireless (BMW) database in [Naikal et al. 2010].[7] The BMW database consists of multiple-view images of 20 landmark buildings on the campus of University of California, Berkeley, collected by the CITRIC cameras. For each building, 16 different vantage points have been selected to measure the 3-D appearance of the building.

Using the BMW database, we have benchmarked the performance of the distributed object recognition system shown in Figure 11 (more detail presented in [Naikal et al. 2010]). We design two testing scenarios to evaluate the performance

[7] The database can be accessed online at: `http://www.eecs.berkeley.edu/~yang/software/CITRIC/`.

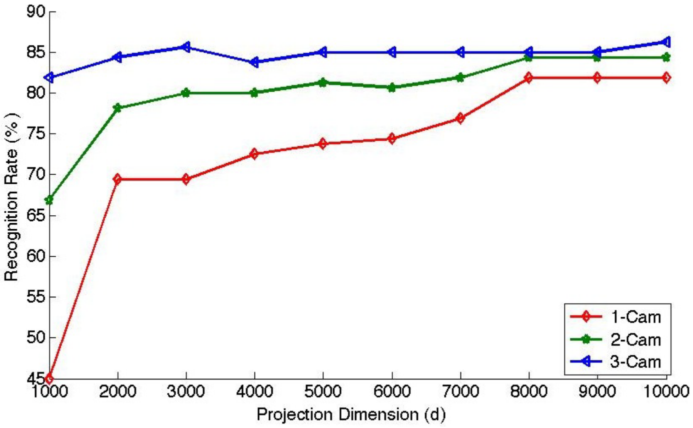


Fig. 12. Comparison of the CHoG recognition rates (in color) in the small-baseline scenario with different random projection dimensions.

of the distributed recognition scheme, namely, the small-baseline and the large-baseline scenarios. In the small-baseline scenario, images captured concurrently from multiple cameras at one vantage point are used to determine the object category. In the large-baseline scenario, images captured from one to three different vantage points are randomly chosen from the same testing category for recognition. In all the experiments, the query histograms are projected from 10,000-D space via random projection to lower projection dimensions ranging from 1000 to 9000 using the encoding function (3).

Figure 12 shows the recognition rates for one to three cameras in the small-baseline scenario. With small projection dimensions close to 1000, the recognition rates using two or three cameras improves significantly compared to the single-view recognition rates. For instance, at $d = 1000$, the recognition rate from a single camera is about 45%. The rate is boosted to 68% with two cameras and 82% with three cameras. It is also important to note that the improved recognition rates using the multiple-view information are also higher than merely increasing the projection dimension (i.e., the bandwidth) in the single-camera scenario. For instance, The recognition rate for 2-Cam at $d = 2000$ is higher than the rate for 1-Cam at $d = 4000$.

The large-baseline performance is evaluated using the same procedure as in the small-baseline experiments. Figure 13 shows the recognition rates versus the random projection dimension. Clearly, the recognition rates using a single camera does not change from the small-baseline scenario. As shown in the plot, the recognition rates at the low projection dimension of 1000 are lower than those of the small-baseline scenario for the 2 and 3-Cam cases. However, as the projection dimension increases, the multiple-view recognition rates reach about 95% and begin to plateau. Such rates are never achieved even without lossy compression in the single view case.

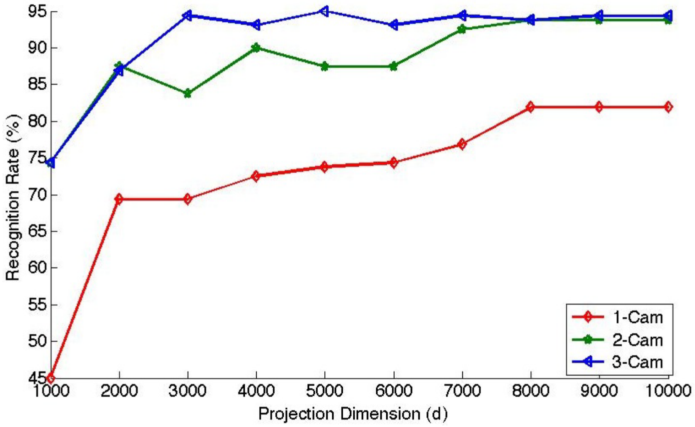


Fig. 13. Comparison of the CHoG recognition rates (in color) in the large-baseline scenario with different random projection dimensions.

7. OTHER APPLICATIONS

In addition to the elementary image processing functions and the two applications we have discussed so far, we here summarize a few more classes of camera network applications that could utilize the CITRIC camera platform.

(1) *Cooperative object detection and tracking.* Object/event detection and tracking has been a classical problem in computer vision and sensor networks [Javed et al. 2003; Dick and Brooks 2004; Gilbert and Bowden 2006; Song and Roy-Chowdhury 2008]. Traditional solutions have been based on a centralized scheme where the system relies on a back-end layer of computer servers to process the video sequences. Utilizing the computational power of CITRIC, the authors of [Wang et al. 2009; Wang et al. 2010] have studied how to perform cooperative object detection and tracking based on only peer-to-peer communication, and therefore completely forgo the centralized server model. The system is designed to locally detect certain well-defined primitive events on individual CITRIC sensors. When an event is detected on a camera sensor, it will inform its neighboring sensors about the event. Over time, more complex events can be recognized by aggregating several primitive local events in the perimeter of the camera network.

(2) *Traffic modeling and prediction.* Beyond the low-level object detection and tracking, there has been a consistent interest in modeling and predicting traffic patterns using information obtained from camera sensor networks. The goals of the application typically include path discovery [Kettnaker and Zabih 1999; Melo et al. 2006], traffic statistics prediction [Guitton et al. 2007; Tubaishat et al. 2009], and accident detection [Kamijo et al. 2000; Bramberger et al. 2006], to name a few. A concrete example using CITRIC motes is presented in [Shuai et al. 2010]. Without assuming camera calibration and reconstruction of global object tracks, the work utilizes a Bayesian framework to predict the traveling time of moving objects, estimate the object paths probabilistically, and perform

object association between multiple camera views in the region of interest.

(3) *Human behavior interpretation.* Another important topic in camera sensor networks is interpreting more complex human behavior, activities, and intent. The intent of humans under surveillance must be considered in the context of the surrounding environment. Therefore, analyzing the relations between humans and their surrounding 3-D structures has been the main focus of several studies (see [Sankaranarayanan et al. 2009] and the references therein). A recent work [Tron et al. 2008] also studied robust estimation of object poses using a camera network. When an object is observed by multiple cameras, its 3-D pose would be determined by a distributed averaging consensus algorithm directly on the manifold of 3-D rigid-body transformations. Finally, systems based on *grammatical inference* have been proposed to classify more complex human behavior and intent (see [Lymberopoulos et al. 2008; Geyik and Szymanski 2009] and the references therein). For instance, [Lymberopoulos et al. 2008] proposes a system called BScope to interpret human activity patterns in an assisted-living application.

(4) *Large-scale, multi-purpose camera sensor networks.* Eventually, any real-world deployment of a large-scale camera sensor network would prefer the system to support a diverse list of applications, which may or may not be well-defined at the time of deployment. In [Kulkarni et al. 2005], the authors advocate a heterogeneous, multi-tier sensor network to reconcile the traditionally conflicting design goals of latency and energy-efficiency. In VideoWeb [Nguyen et al. 2009], the authors argue that it is still a challenge to crease a robust wireless network to support multiple high-bandwidth video cameras at their peak performance; while another challenge is to make the entire camera system reconfigurable to implement a variety of real-world, real-time surveillance applications. Despite these pioneering works, it is fair to say that many open problems abound in this topic.

8. CONCLUSION AND FUTURE WORK

We have presented a comprehensive review of the architecture of CITRIC, a wireless camera mote system for low-bandwidth networks. The system enables the captured images to be processed locally on the camera board so that only compressed, low-dimensional features or other small pieces of data need to be transmitted over 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 WSNs. In this paper, we have also discussed several convincing examples to demonstrate the utility of the platform in smart camera networks.

For future work, we plan to continue 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. Finally, we are interested in integration of CITRIC and other heterogeneous sensor modalities to create a truly large-scale, multi-purpose sensor network for urban surveillance applications.

Acknowledgements

The authors would like to thank Colby Boyer, Dr. Marci Meingast, and Dr. Chuohao Yeo, whose research has contributed to part of the CITRIC project.

REFERENCES

Agarwal, S. and Roth, D. 2002. Learning a sparse representation for object detection. In *European Conference on Computer Vision*.

Akyildiz, I., Melodia, T., and Chowdhury, K. 2008. Wireless multimedia sensor networks: applications and testbeds. *Proceedings of the IEEE 96,* 10, 1–18.

Baron, D., Wakin, M., Duarte, M., Sarvotham, S., and Baraniuk, R. 2006. Distributed compressed sensing. Tech. Rep. TREE-0612.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. 2008. SURF: Speeded up robust features. *Computer Vision and Image Understanding 110,* 3, 346–359.

Bradski, G., Kaehler, A., and Pisarevsky, V. 2005. Learning-based computer vision with Intel's Open Source Computer Vision Library. *Intel Technology Journal 9,* 2, 119–130.

Bramberger, M., Doblander, A., Maier, A., Rinner, B., and Schwabach, H. 2006. Distributed embedded smart cameras for surveillance applications. *Computer 39,* 2, 68–75.

Calderara, S., Vezzani, R., Prati, A., and Cucchiara, R. 2005. Entry edge of field of view for multi-camera tracking in distributed video surveillance. In *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance*.

Candès, E. and Tao, T. 2006. Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory 52,* 12, 5406–5425.

Chandrasekhar, V., Takacs, G., Chen, D., Tsai, S., Grzeszczuk, R., and Girod, B. 2009. CHoG: Compressed histogram of gradients a low bit-rate feature descriptor. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*.

Chen, D., Tsai, S., Chandrasekhar, V., Takacs, G., Singh, J., and Girod, B. 2009. Tree histogram coding for mobile image matching. In *Proceedings of the IEEE Data Compression Conference*.

Chen, P. W.-C., Ahammad, P., Boyer, C., Huang, S.-I., Lin, L., Lobaton, E. J., Meingast, M. L., Oh, S., Wang, S., Yan, P., Yang, A., Yeo, C., Chang, L.-C., Tygar, J. D., and Sastry, S. S. 2008. CITRIC: A low-bandwidth wireless camera network platform. In *Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*.

Cheng, Z., Devarajan, D., and Radke, R. 2007. Determining Vision Graphs for Distributed Camera Networks Using Feature Digests. *EURASIP Journal on Applied Signal Processing 2007,* 1.

de Silva, V. and Ghrist, R. 2007. Homological Sensor Networks. *Notices of the AMS 54,* 1, 10–17.

Dick, A. and Brooks, M. 2004. A stochastic approach to tracking objects across multiple cameras. In *Proceedings of the Australian Joint Conference on Artificial Intelligence*.

Donoho, D. 2006. For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Math 59,* 6, 797–829.

Downes, I., Rad, L., and Aghajan, H. 2006. Development of a mote for wireless image sensor networks. In *Cognitive Systems with Interactive Sensors (COGIS)*. Paris, France.

Feng, W., Kaiser, E., Feng, W., and Baillif, M. L. 2005. Panoptes: scalable low-power video sensor networking technologies. *ACM Trans. Multimedia Comput. Commun. Appl. 1,* 2, 151–167.

Ferrari, V., Tuytelaars, T., and Gool, L. V. 2004. Integrating multiple model views for object recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

Fischler, M. and Bolles, R. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 381–395.

Geyik, S. and Szymanski, B. 2009. Event recognition in sensor networks by means of grammatical inference. In *Proceedings of the IEEE International Conference on Computer Communications*.

Gilbert, A. and Bowden, R. 2006. Tracking objects across cameras by incrementally learning inter-camear colour calibration and patterns of activity. In *Proceedings of European Conference on Computer Vision*.

Guitton, A., Skordylis, A., and Trigoni, N. 2007. Correlation-based data dissemination in traffic monitoring sensor networks. In *Proceedings of IEEE Wireless Communications and Networking Conference*.

Hong, K., Yan, P., Chen, P., Oh, S., and Sastry, S. 2009. Poster abstract: Multihop routing in camera sensor networks - an experimental study. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*.

Javed, O., Rasheed, Z., Shafique, K., and Shah, M. 2003. Tracking across multiple cameras with disjoint views. In *Proceedings of International Conference on Computer Vision*.

Kamijo, S., Matsushita, Y., Ikeuchi, K., and Sakauchi, M. 2000. Traffic monitoring and accident detection at intersections. *IEEE Transactions on Intelligent Transportation Systems 1,* 2, 108–118.

Kandhalu, A., Rowe, A., and Rajkumar, R. 2009. DSPcam: A camera sensor system for surveillance networks. In *Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*. 1 –7.

Kettnaker, V. and Zabih, R. 1999. Bayesian multi-camera surveillance. In *Proceedings of International Conference on Computer Vision*.

Khan, S. and Shah, M. 2003. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25,* 10, 1355–1360.

Kleihorst, R. 2008. Personal communication about Xetal-II WiCa platform.

Kleihorst, R., Abbo, A., Schueler, B., and Danilin, A. 2007. Camera mote with a high-performance parallel processor for real-time frame-based video processing. In *First ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*. Vienna, Austria, 109–116.

Kulkarni, P., Ganesan, D., Shenoy, P., and Lu, Q. 2005. SensEye: a multi-tier camera sensor network. In *Proceedings of the 13th annual ACM International Conference on Multimedia*. 229–238.

Lobaton, E., Ahammad, P., and Sastry, S. 2009. Algebraic approach to recovering topological information in distributed camera networks. In *International Conference on Information Processing in Sensor Networks*.

Lobaton, E., Vasudevan, R., Sastry, S., and Bajcsy, R. 2010. A distributed topological camera network representation for tracking applications. *IEEE Transactions on Image Processing*.

Lowe, D. 1999. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*.

Lymberopoulos, D., Teixeira, T., and Savvides, A. 2008. Macroscopic human behavior interpretation using distributed imager and other sensors. *Proceedings of the IEEE 96,* 10, 1657–1677.

Maroti, M., Kusy, B., Simon, G., and Ledeczi, A. 2004. The flooding time synchronization protocol. In *Proceedings of the International Conference on Embedded Networked Sensor Systems*.

Marvell Corporation. 2010. Marvell pxa270 processor electrical, mechanical and thermal specification datasheet. `http://www.marvell.com/products/processors/applications/pxa_family/pxa_27x_emts.pdf`.

Meguerdichian, S., Koushanfar, F., Potkonjak, M., and Srivastava, M. 2001. Coverage problems in wireless ad-hoc sensor network. In *IEEE INFOCOM*.

Melo, J., Naftel, A., Bernardino, A., and Santos-Victor, J. 2006. Ieee transactions on intelligent transportation systems. *7,* 2, 188–200.

Memsic. 2008. Imote2: High-performance wireless sensor network node datasheet. `http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html`.

Naikal, N., Yang, A., and Sastry, S. 2010. Towards an efficient distributed object recognition system in wireless smart camera networks. In *Proceedings of the 13th International Conference on Information Fusion*. Edinburgh, UK.

Nguyen, H., Bhanu, B., Patel, A., and Diaz, R. 2009. VideoWeb: Design of a wireless camera network for real-time monitoring of activities. In *Proceedings of the International Conference on Distributed Smart Cameras*.

Nistér, D. and Stewénius, H. 2006. Scalable recognition with a vocabulary tree. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

NXP Semiconductor. 2003. PCF50606: How to connect to the Intel Bulverde application processor. http://www.nxp.com/acrobat_download/literature/9397/75009763.pdf. Application Note AN03606-01.

Omnivision Technologies Incorporated. 2006. OV9655 color CMOS SXGA (1.3MegaPixel) CAMERACHIP with OmniPixel Technology datasheet. http://www.ovt.com. CMOS image sensor manufactured by Omnivision Technologies Inc.

Oram, M. and Perrett, D. 1994. Modeling visual recognition from neurobiological constraints. *Neural Networks 7*, 945–972.

Park, C. and Chou, P. 2006. eCAM: ultra compact, high data-rate wireless sensor node with a miniature camera. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. ACM, New York, NY, USA, 359–360.

Polastre, J., Szewczyk, R., and Culler, D. 2005. Telos: enabling ultra-low power wireless research. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS)*. Los Angeles, USA, 364–369.

Rahimi, A., Dunagan, B., and Darrell, T. 2004. Simultaneous Calibration and Tracking with a Network of Non-Overlapping Sensors. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*.

Rahimi, M., Baer, R., Iroezi, O., Garcia, J., Warrior, J., Estrin, D., and Srivastava, M. 2005. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems*. San Diego, USA, 192–204.

Rinner, B., Winkler, T., Schriebl, W., Quaritsch, M., and Wolf, W. 2008. The evolution from single to pervasive smart cameras. In *International Conference on Distributed Smart Cameras*.

Sankaranarayanan, A., Patro, R., Turaga, P., Varshney, A., and Chellappa, R. 2009. Modeling and vidualization of human activities for multicamera networks. *EURASIP Journal on Image and Video Processing*.

Shuai, Z., Oh, S., and Yang, M. 2010. Traffic modeling and prediction using camera sensor networks. In *Proceedings of the International Conference on Distributed Smart Cameras*.

Song, B. and Roy-Chowdhury, A. 2008. Robust tracking in a camera network: a multi-objective optimization framework. *IEEE Journal of Selected Topics in Signal Processing 2,* 4, 582–596.

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

Tron, R., Vidal, R., and Terzis, A. 2008. Distributed pose averaging in camera networks via consensus on $se(d)$. In *Proceedings of the International Conference on Distributed Smart Cameras*.

Tubaishat, M., Zhuang, P., Qi, Q., and Shang, Y. 2009. Wireless sensor networks in intelligent transportation systems. *Wireless communications and mobile computing 9,* 3, 287–302.

Turcot, P. and Lowe, D. 2009. Better matching with fewer features: The selection of useful features in large database recognition problems. In *ICCV Workshop on Emergent Issues in Large Amounts of Visual Data*.

Wang, Y., Casares, M., and Velipasalar, S. 2010. Cooperative object tracking and composite event detection with wireless embedded smart cameras. *IEEE Transactions on Image Processing 19,* 10, 2614–2633.

Wang, Y., Velipasalar, S., and Casares, M. 2009. Detection of composit events spanning multiple camera views with wireless embedded smart cameras. In *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras*. 1–8.

Wolfson Micro PLC. 2008. Wolfson WM8950. `http://www.wolfsonmicro.com/uploads/documents/en/WM8950.pdf`.

Yang, A., Ganesh, A., Ma, Y., and Sastry, S. 2010. Fast ℓ_1-minimization algorithms and an application in robust face recognition: A review. In *International Conference on Image Processing*.

Yang, A., Maji, S., Christoudias, C., Darrell, T., Malik, J., and Sastry, S. 2009. Multiple-view object recognition in band-limited distributed camera networks. In *Proceedings of the International Conference on Distributed Smart Cameras*. Como, Italy.